



BEAMER 文档类

用户手册 V3.24

```
\begin{frame}
  \frametitle{There Is No Largest Prime Number}
  \framesubtitle{The proof uses \textit{reductio ad absurdum}.}
  \begin{theorem}
    There is no largest prime number.
  \end{theorem}
  \begin{proof}
    \begin{enumerate}
      \item<1-| alert@1> Suppose  $p$  were the largest prime number.
      \item<2-> Let  $q$  be the product of the first  $p$  numbers.
      \item<3-> Then  $q+1$  is not divisible by any of them.
      \item<1-> But  $q + 1$  is greater than  $p$ , thus divisible by some prime
        number not in the first  $p$  numbers.\qedhere
    \end{enumerate}
  \end{proof}
\end{frame}
```

There Is No Largest Prime Number
With an introduction to a new proof technique

Euklid of Alexandria
Department of Mathematics
University of Alexandria

27th International Symposium on Prime Numbers, -280

1 Results
Proof of the Main Theorem

Results

There Is No Largest Prime Number
With an introduction to a new proof technique

Euklid of Alexandria
Department of Mathematics
University of Alexandria

27th International Symposium on Prime Numbers, -280

1 Results
• Proof of the Main Theorem

BEAMER 文档类用户手册 V3.24

<http://bitbucket.org/rivanvx/beamer>

Till Tantau、Joseph Wright、Vedran Miletic [著]

黄旭华 [译]

木有出版社

Für alle, die die Schönheit von Wissenschaft anderen zeigen wollen.

Copyright 2003–2007 by Till Tantau

Copyright 2010,2011 by Joseph Wright and Vedran Miletic

如遵循自由软件基金会发布的《GNU自由文档许可证》V1.3 或更新的版本，将可以拷贝、分发和/或修改该文档，不包括不可变章节（no Invariant Sections）、封面文本（Front-Cover Texts）、封底文本（Back-Cover Texts）。该许可证的一份拷贝包含在《GNU自由文档许可证》这一节中。

如遵循自由软件基金会发布的《GNU通用公共许可证》V2 或更新的版本，将可以拷贝、分发和/或修改该宏包的代码，该许可证的一份拷贝包含在《GNU通用公共许可证》这一节中。

如遵循《L^AT_EX 项目公共许可证》V1.3c 或更新的版本，可以分发和/或修改该文档及代码，该许可证的一份拷贝包含在《L^AT_EX 项目公共许可证》这一节中。

翻译及使用说明：

- 1、译者是在遵循上述许可证的情况下进行翻译的。该译本的使用请遵循上述许可证。
- 2、一小部分翻译来自网络。译者不能保证所有翻译的正确性，由此带来的问题译者不承担任何责任。
- 3、在译本中添加了一些源文档没有的图标、脚注。
- 4、该译本的编译是在安装了 C_TE_X 套装的 Window 7 平台中进行的，在其它平台中未测试过。
- 5、请将该译本放在 C_TE_X 目录下，否则无法打开该译本中的某些链接。
- 6、译者将继续对源文档的更新版本进行翻译。

联系译者：

黄旭华 江西省赣南医学院第一附属医院神经科二区

E-mail: huxhu_828@hotmail.com; QQ : 453148436

目 录

1 介绍	1
1.1 主要特点	1
1.2 历史	1
1.3 感谢	2
1.4 如何阅读该手册	2
1.5 获得帮助	4
I 开始	6
2 安装	7
2.1 版本和支撑	7
2.2 安装封装包	7
2.2.1 T _E X Live 和 MacT _E X	7
2.2.2 MiK _T E _X 和 proT _E Xt	7
2.2.3 Debian 和 Ubuntu	7
2.2.4 Fedora	8
2.3 在 texmf 目录树中安装	8
2.4 更新安装	9
2.5 检测安装	9
2.6 和其它宏包及类文件的兼容	9
3 指导：欧几里德教授的演示稿	15
3.1 问题说明	15
3.2 解答模板	15
3.3 标题部分	15
3.4 封面帧	16
3.5 创建演示稿的 PDF 文件	16
3.6 目录	16
3.7 节和小节	17
3.8 创建一个简单的帧	17
3.9 创建一个简单的叠层	18
3.10 使用叠层规则	18

3.11	构造一个帧	20
3.12	添加参考文献	21
3.13	抄录文本	21
3.14	更改外观的方式 I: 主题	23
3.15	更改外观的方式 II: 颜色和字体	23
4	创建 Beamer 演示稿的工作流程	24
4.1	第一步: 建立文件	24
4.2	第二步: 组建演示稿	24
4.3	第三步: 创建一个 PDF 或 PostScript 文件	25
4.3.1	创建 PDF 文件	25
4.3.2	创建 PostScript 文件	26
4.3.3	加快编译速度的方法	26
4.4	第四步: 创建帧	27
4.5	第五步: 测试演示稿	27
4.6	第六步: 创建讲义	27
4.6.1	创建讲义	27
4.6.2	打印讲义	27
5	创建演示稿的准则	29
5.1	组建演示稿	29
5.1.1	知晓时间限制	29
5.1.2	全局结构	29
5.1.3	帧结构	31
5.1.4	交互元素	33
5.2	使用图形	34
5.3	使用动画和过渡	34
5.4	选择恰当的主题	35
5.5	选择恰当的颜色	35
5.6	选择恰当的字体和字体属性	36
5.6.1	字体的尺寸	36
5.6.2	字族	37
5.6.3	字体形状: 斜体和小型大写	39
5.6.4	字体的粗细	40
6	解答模板	41
7	许可证和版权	42
7.1	必需遵守哪些许可证?	42
7.2	GNU 通用公共许可证, V2	42
7.2.1	前言	42
7.2.2	复制、分发与修改的条款与条件	43
7.2.3	无担保声明	45

7.3	Gnu 自由文档许可证, V1.3, 2008 年 11 月 3 日	45
7.3.1	前言	45
7.3.2	适应范围和约定	46
7.3.3	原样复制	47
7.3.4	大量复制	47
7.3.5	修改	47
7.3.6	合并文档	48
7.3.7	文档集	49
7.3.8	独立作品的聚合体	49
7.3.9	翻译	49
7.3.10	终止	49
7.3.11	本协议的未来修订版本	50
7.3.12	重新授权	50
7.3.13	补充: 如何使用本许可证	50
7.4	L ^A T _E X 项目公共许可证	51
7.4.1	导言	51
7.4.2	定义	51
7.4.3	分发和修改的条件	52
7.4.4	无担保声明	53
7.4.5	作品的维护	53
7.4.6	该许可证允许下能否和如何分发	54
7.4.7	选择这个或其它许可证	54
7.4.8	不进行分发时的修改的建议	55
7.4.9	如何使用该许可证	55
7.4.10	非替代的衍生作品	55
7.4.11	重要的建议	56

II 创建演示稿 57

8	创建帧	58
8.1	帧环境	58
8.2	帧的组成部分	64
8.2.1	顶部导航区和底部导航区	65
8.2.2	侧栏	68
8.2.3	导航条	71
8.2.4	导航符	75
8.2.5	徽标	77
8.2.6	帧标题	78
8.2.7	背景	80
8.3	帧尺寸和页边距	81
8.4	一帧的幻灯片数量的限制	83

9 创建叠层	84
9.1 暂停命令	84
9.2 叠层规则的一般概念	85
9.3 能跟叠层规则的命令	86
9.4 用于叠层规则的环境	91
9.5 动态改变文本或图像	92
9.6 高级叠层规则	94
9.6.1 定义命令和环境叠层规则 -已知的	94
9.6.2 模式规则	97
9.6.3 行为规则	98
9.6.4 递增的规则	100
10 组建演示稿: 静态的全局结构	103
10.1 添加封面	103
10.2 添加节和小节	106
10.3 添加部分	110
10.4 将教程分解成讲课	112
10.5 添加目录	113
10.6 添加参考书目	116
10.7 添加附录	119
11 组建演示稿: 交互的全局结构	120
11.1 添加超级链接和按钮	120
11.2 在后面的某点重复某帧	125
11.3 添加预期的缩放	126
12 组建演示稿: 局部结构	129
12.1 常规、排序、解说	129
12.2 高亮显示	137
12.3 块环境	140
12.4 定理环境	143
12.5 加框的文本和盒子文本	149
12.6 图和表	152
12.7 将帧分成多栏	153
12.8 文本和图形的绝对定位	155
12.9 抄录文本和脆弱文本	155
12.10摘要	156
12.11诗歌、引用（首行再缩进）、引用（首行不缩进）	157
12.12脚注	159
13 图形	161
13.1 包含外部图形文件与直接嵌入图形	161
13.2 包含 .eps 或 .ps 格式的图形文件	162

13.3 包含 .pdf、.jpg、.jpeg 或 .png 格式的图形文件	162
13.4 包含 .mps 格式的图形文件	162
13.5 包含 .mmp 格式的图形文件	163
14 动画、声音、幻灯片过渡	164
14.1 动画	164
14.1.1 包含外部的动画文件	164
14.1.2 快速演替幻灯片时形成动画	167
14.1.3 包含存放在多个图形文件中的外部动画	169
14.2 声音	170
14.3 幻灯片过渡	173
III 更改外观的方法	176
15 主题	177
15.1 五类主题	177
15.2 无导航条的演示主题	178
15.3 顶部带有树形导航条的演示主题	184
15.4 侧边带有目录的演示主题	185
15.5 带有微帧导航的演示主题	188
15.6 带有节和小节列表的演示主题	192
15.7 能兼容的演示主题	194
16 内部主题、外部主题、模板	195
16.1 内部主题	195
16.2 外部主题	198
16.3 改变演示稿不同元素的模板	204
16.3.1 Beamer 的模板管理概述	204
16.3.2 使用 Beamer 的模板	206
16.3.3 设置 Beamer 的模板	207
17 颜色	212
17.1 颜色主题	212
17.1.1 默认和专用的颜色主题	212
17.1.2 完整的颜色主题	214
17.1.3 内部颜色主题	219
17.1.4 外部颜色主题	221
17.2 改变演示稿不同元素的颜色	222
17.2.1 Beamer 的颜色管理概述	223
17.2.2 使用 Beamer 的颜色	223
17.2.3 设置 Beamer 的颜色	225
17.3 数学文本的颜色	227
17.4 调色板	228

17.5 杂色	229
17.6 透明效果	231
18 字体	234
18.1 字体主题	234
18.2 不通过字体主题而改变字体	237
18.2.1 选择字体的尺寸为常规	238
18.2.2 选择字族	238
18.2.3 选择字体的编码	239
18.3 改变演示稿不同元素的字体	239
18.3.1 Beamer 的字体管理概述	240
18.3.2 使用 Beamer 的字体	240
18.3.3 设置 Beamer 字体	240
IV 创建配套的材料	243
19 为自己添加笔记	244
19.1 指定笔记的内容	244
19.2 指定多个笔记的内容	246
19.3 指定显示哪个笔记和帧	247
20 创建胶片	249
21 创建讲义和演讲记录	251
21.1 使用讲义模式创建讲义	251
21.2 使用论文模式创建讲义	251
21.2.1 开始论文模式	251
21.2.2 工作流程	254
21.2.3 在论文版本中包含来自演示稿的幻灯片	255
21.3 模式的相关细节	256
22 多屏显示的好处	260
22.1 在第二个屏幕上显示笔记	260
22.2 在第二个屏幕上显示第二模式的材料	260
22.3 在第二个屏幕上显示以前的幻灯片	262
V 如何做	263
23 如何分段显示	264
23.1 分段显示排序列表	264
23.2 高亮显示排序列表中的当前条目	264
23.3 改变排序列表中的前导符	265
23.4 分段显示标记的公式	266

23.5 逐行显示表格	267
23.6 逐列显示表格	267
24 如何导入基于其它宏包和文档类的演示稿	268
24.1 Prosper、HA-Prosper、Powerdot	268
24.2 Seminar	274
24.3 FoilT _E X	278
24.4 T _E XPower	281
25 翻译字串	284
25.1 介绍	284
25.1.1 该宏包的概述	284
25.1.2 如何看懂这一节	284
25.1.3 贡献	284
25.1.4 安装	284
25.2 基本概念	285
25.2.1 关键词	285
25.2.2 语言名	285
25.2.3 语言路径	285
25.2.4 字典	286
25.3 用法	286
25.3.1 基本用法	286
25.3.2 提供翻译	286
25.3.3 创建和使用字典	288
25.3.4 创建用户字典	290
25.3.5 翻译关键词	291
25.3.6 语言路径和语言替代	291
25.3.7 宏包加载过程	292
索引	293

1 介绍

BEAMER 是 L^AT_EX 的一个文档类 (class)，它可用于创建用投影机 (projector)¹ 放映的演示稿 (presentations)，也可用于制作幻灯片 (slides)。用 BEAMER 创建演示稿不同于用 WYSIWYG (所见即所得) 的程序如 OpenOffice 的 Impress、Apple 的 Keynote、KOffice 的 KPresenter 或 Microsoft 的 PowerPoint 等创建演示稿，用 BEAMER 创建的演示稿类似于其它的 L^AT_EX 文档：有导言 (Preamble) 和正文 (body)，正文包含 `\sections` 和 `\subsections`，不同的幻灯片 (slides) (在 BEAMER 中放在 *frames* 里) 放在不同的环境 (environments) 中，后者由常规列表 (`itemize`) (或称项目式列表) 环境和排序列表 (`enumerate`) (或称列举式列表) 环境等构成。这种方法显而易见的缺点是在使用 BEAMER 时，我们必需先熟悉 L^AT_EX。优点是如果我们熟悉 L^AT_EX，则不仅写论文 (papers)，在创建演示稿时，均可应用我们所掌握的 L^AT_EX 知识。

1.1 主要特点

BEAMER 的特点 (feature) 很多 (不幸的是隐错即 bug 同样多)。在我看来，最重要的特点是：

- 在 BEAMER 中可以使用 `pdflatex`、`latex+dvips`、`lualatex`、`xelatex` 等编译。但不支持 `latex+dvipdfm` (打补丁后支持!)
- 可以在 BEAMER 中可直接调用 L^AT_EX 的标准命令。比如：`\tableofcontents` 仍用于生成目录 (Table of Contents)、`\section` 仍用于创建一个节、`itemize` 环境仍用于创建列表。
- 方便的创建叠层 (overlays) 和动态效果 (dynamic effects)。
- 主题 (Theme) 允许我们改变演示稿的外观以符合我们的意愿 (purposes)。
- 主题大都来自作者的演讲实践，它们致力于更好地表现内容，而不是为了让页面更好看，所以在提供的主题中不会出现用以描绘绿色草地的图片上显示有绿色正文文本这样的现象。
- 演示稿的布局、颜色、字体都可以实现全局控制，只不过这需要我们有控制 BEAMER 细节的能力。
- 特定的样式 (style) 文件允许我们在 L^AT_EX 的其它文档类 (class) 如 `article` 或 `book` 中直接使用 L^AT_EX 源文件。这使的从演讲记录 (lecture notes) 创建演示稿，或从演示稿创建演讲记录都变得极其容易。
- 最终的输出是典型的 PDF 文件。这种格式文件的阅读器存在于各种平台。用记忆棒带着我们的演示稿参加会议，无需当心相应版本的演示稿程序是否安装。而且，我们的演示稿不会走样，它和在我们的电脑中显示是一样的。

1.2 历史

提尔·坦图 (Till Tantau) 教授主要在其空闲时间创作了 BEAMER。很多人通过电子邮件给予帮助，在改进、改正、修订、新主题等方面提了很多建议 (到目前，他收到了 1000 多封关于 BEAMER 的邮件)。事实上，大部分的进展 (development) 归功于性能需求 (feature requests) 和错误报告 (bug reports)。没有这些反馈，BEAMER 仍然是其最初的样子：一个小的个人的宏指令 (macros) 集，它只是使使用 `seminar` 文档类 (class)

¹投影机的英文名称有三个：第一个是 Projector，它比较确切的名称，严谨的单位比如科研机构或者学校均用 Projector 表示投影机；第二个是 BEAMER；第三个是 BARCO，BARCO 确实是做投影机市场的著名生产厂商，很多公司也直接就把投影机叫做 BARCO。

变得更容易。提尔·坦图教授在 2003 年 2 月为准备其博士学位论文答辩创作了 BEAMER 第一版。一个月后，应一些同行的要求，把它放到了 CTAN² 上。此后，事情不知怎的变得有点失控。

2007 年后没有进行维护，在 2010 年 4 月提尔·坦图教授将维护工作交给约瑟夫·赖特 (Joseph Wright) 和维德兰·米莱迪克 (Vedran Miletić)，他们维护 BEAMER 至今：改进代码、修复隐错 (bugs)、添加新功能、帮助用户。

1.3 感谢

提尔·坦图 (Till Tantau)：“从哪儿开始说起呢？，BEAMER 的发展，不仅在我，更应归功于那些好心的人给我的反馈。许多功能已能实现，是因为一些人需要它，这些功能很好，也容易实现。另一些好心人在主题、用户手册、文档类的功能、后台执行 (*internals of the implementation*)、标准的 $L^A T^E X$ 功能、普通的传记上给予了有用的反馈。这些好心的人包括 (这里排名不分顺序，许多人我甚至忘掉了他们的真实姓名，而他们应该出现在这个名单中)：Carsten (总务)、Birgit (除我外，第一个使用 BEAMER)、Tux (感谢他们的批评指正)、Rolf Niepraschk (指导我正确地编写 $L^A T^E X$)、Claudio Beccari (编写了部分字体编码相关文档)、Thomas Baumann (负责 *emacs* 的资料)、Stefan Müller (鼓励)、Uwe Kern (负责 XCOLOR)、Hendri Adriaens (负责 HA-PROSPER)、Ohura Makoto (校样)。对主题作出贡献的好心人有：Paul Gomme、Manuel Carro 和 Marlon Régis Schmitz。”

约瑟夫·赖特 (Joseph Wright)：“感谢 Till Tantau 教授为创作 BEAMER 所作的巨大贡献。真诚地感谢 Vedran Miletić 为 BEAMER 的发展所起的领导作用。”

维德兰·米莱迪克 (Vedran Miletić)：“首先，我要感谢 Karl Berry 和 Sanda Bujačić 对我的鼓励，如果没有他们的鼓励，除了是一个 $L^A T^E X$ 用户外，我什么都不是。我也要感谢我的同事 Ana Meštrović，他高兴地期望使用 BEAMER；Ivona Franković 和 Marina Rajnović，我的信息系的学生，他们是首个听说 $L^A T^E X$ 、BEAMER 和如何准备 *class* 材料的。我还要感谢 Heiko Oberdiek (因为 HYPERREF 宏包)、Johannes Braams (因为 BABEL 宏包) 和 Philipp Lehman (因为 BIBLATEX 宏包)。最重要的是，我要感谢 Till Tantau 教授，他开发了 BEAMER。感谢 Joseph Wright 教授，他发展了 SIUNITX，并帮我将 BEAMER 发展得更远。”

1.4 如何阅读该手册

应该从第一部分开始。如果我们没有安装 BEAMER 软件包，请先阅读第 7 页第 2 节。BEAMER 新手接着阅读第 15 页第 3 节。当我们准备用 BEAMER 制作第一个真正的演示稿时，先阅读第 4 节，在这里会讨论一些技巧和工作流程。如果我们是第一次创建演示稿，第 5 节对我们有用，这个准则 (guidelines) 会告诉我们该做什么，不该做什么。最后，必需浏览一下第 6 节，这里有现成的解答模板 (Solution Template)，或许还有我们想要的相应语言的解答模板。这些解答模板可用于制作演讲稿 (talks)。

手册的第二部分详细阐述了 BEAMER 定义的命令，也阐述了跟制作演示稿有关的技术问题 (象如何插入图形和动画)。

手册的第三部分阐述如何通过使用主题，或通过指定演示稿特定元素的颜色、字体来改变演示稿的外观，演示稿的特定元素如一个排序式列表 (enumeration) 中的序号 (numbers) 的字体。

手册的第四部分阐述讲义 (Handout) 和演讲记录 (lecture notes)，也称作“配套材料 (support material)”。我们也许经常在演讲中或之后为我们的听众创建某种配套材料，这部分将阐述如何用创建演示稿的资源创建配套材料。

²CTAN: Comprehensive $T^E X$ Archive Network, 全面的 $T^E X$ 归档网络、全面的 $T^E X$ 文档网络。是一个权威的 $T^E X$ 相关资源库，几乎所有的 $T^E X$ 相关软件、文档都可以在这里找到最新的版本。

手册的最后一部分的内容是“如何做 (howtos)”，阐述如何用 BEAMER 完成一件事。

手册也包括“全局 (public)”命令、环境 (environments)、用 BEAMER 文档类定义的概念 (concepts) 的阐述。看下面的例子，一般规定，**红色文字**是下定义 (defined)，**绿色文字**是可选的 (optional)，**置于左侧的蓝色文字**表明特定模式 (special mode)。

```
\somebeamercommand[optional arguments]{first argument}{second argument}
```

即：

```
\somebeamercommand[可选参数]{第一参数}{第二参数}
```

这里有关于命令 `\somebeamercommand` 功能的阐述。绿色的参数是可选的。该命令的例子带有两个参数。

举例：`\somebeamercommand[opt]{my arg}{xxx}`

```
\begin{somebeamerenvironment}[optional arguments]{first argument}  
  environment contents  
\end{somebeamerenvironment}
```

即：

```
\begin{somebeamerenvironment}[可选参数]{第一参数}  
  environment contents  
\end{somebeamerenvironment}
```

这里有关于环境 `somebeamerenvironment` 功能的阐述，和命令一样，绿色的是可选参数。

举例：

```
\begin{somebeamerenvironment}{Argument}  
  一些文本。  
\end{somebeamerenvironment}
```

Beamer-Template/-Color/-Font **some beamer element**

即：

Beamer-Template/-Color/-Font **某个 beamer 元素**

这里有关于模板、颜色和/或字体等某个 BEAMER 元素 (**some beamer element**) 的阐述。“BEAMER元素”的概念将在第 16 节进行详细的阐述。简单地说，一个元素 (*element*) 就是以某种特定方式排版的演示稿的一部分。例如：帧标题 (Frame Title)、作者姓名 (author’s name)、或脚注符号 (footnote sign)。大部分的元素有模板 (*template*)，再看看第 16 节吧，还可以看看 BEAMER-color 和 BEAMER-font。

对每一元素，应该声明它是 **some beamer element** 的模板 (template)，还是 BEAMER-color，和/或是 BEAMER-font。在进行排版时，这三种元素必需存在且被使用，也就是说，当插入模板时，必需先安装 BEAMER-color 和 BEAMER-font。然而，有时模板没有与之相应的颜色 (color) 或字体 (font) (如 parent 模板)。而且已有的 BEAMER-colors 和 BEAMER-font 没有基本的 (underlying) 模板。

使用和更改模板³将在第 16.3 节进行阐述。要点是：要更改一个模板，我们可以声明：

```
\setbeamertemplate{some beamer element}{我们对该模板的定义}
```

³语法分别是：`\setbeamertemplate{some beamer element}{我们对该模板的定义}`；`\setbeamercolor{some beamer element}{我们对颜色的定义}`；`\setbeamerfont{some beamer element}{我们对字体的定义}`

不幸的是，要为某些模板提供好的定义，这条命令太微不足道了。幸运的是，模板常带有预定义的选项 (*predefined option*)。这些选项用如下方式显示：

- `[square]` 用小方形 (small square) 渲染模板 (render the template)。
- `[circle]{\radius}` 用指定半径 (radius) 的圆渲染模板 (render the template)。

我们可以象这样放置预定义的选项：

```
\setbeamertemplate{some beamer element}[square]
% 现在应用了一个方形 (squares)
```

```
\setbeamertemplate{some beamer element}[circle]{3pt}
% 现在应用了一个圆 (circle)
```

BEAMER-colors 将在第 17 节进行阐述。要点是：要改变前景色 (foreground) (如改成红色) 的命令是：

```
\setbeamercolor{some beamer element}{fg=red}
```

要改变背景色 (Background) (如改成黑色) 的命令是：

```
\setbeamercolor{some beamer element}{bg=black}
```

也可用 `fg=red,bg=black` 同时改变前景色和背景色。不能以背景“为荣 (honoured)”，因为很难正确地显示彩色的背景，而且背景会增加模板的负担 (前景色则自动应用)。BEAMER-字体 (BEAMER-fonts) 将在第 18 节进行阐述。改变字体大小 (如改成 large) 的命令是：

```
\setbeamerfont{some beamer element}{size=\large}
```

除大小 (size) 外，可以象 `series=\bfseries` 这样设置系列 (series)，象 `shape=\itshape` 这样改变形状 (shape)，象 `family=\sffamily` 这样改变字族 (family)，而且这些可以联用。为命令添加一个星号 (star) 可以“复原 (reset)”字体。

PRESENTATION 象本段一样，一些段落的前面有蓝色的 PRESENTATION，意味着这些段落只适合用 L^AT_EX 或 pdfL^AT_EX 排版演示稿 (presentation)。

ARTICLE 与此相反，一些段落的前面有蓝色的 ARTICLE，表示在 论文模式 (article mode) 下的一些行为。该模式常用于从演示稿 (presentation) 制作演讲记录 (lecture notes) (二者可共存于同一文件中)。

LYX 一些段落的前面有蓝色的 LYX，表明这些特定的行为只适用于用 L^AT_EX 制作演示稿。

1.5 获得帮助

可以从以下几方面获得关于 BEAMER 的帮助：

1. 最起码也要阅读与我们的问题相关的部分。
2. Perhaps someone has already reported a similar problem and someone has found a solution. 如果这不能解决问题，可以试着看看 bitbucket⁴ 的 BEAMER 开发页面 (查看该文档的标题)。也许已有别人报告了相似的问题并有了答案。

⁴BitBucket 是一家源代码托管网站，采用 Mercurial 做为分布式版本控制系统，主要特点是：无限制的磁盘空间可供使用、Bug 跟踪、项目 Wiki、API 支持、灵活的权限控制、可自定义域名、RSS 修改记录输出、自定义下载等。

3. 如果我们在这里没有找到答案，或者确定找到了 BEAMER 的一个隐错 (bug)，请到 [via bitbucket.org/rivanvx/beamer/issues](https://bitbucket.org/rivanvx/beamer/issues) 这里报告。
4. 在我们归档隐错报告之前，特别是归档与安装相关的隐错报告可以确定这是不是真正的隐错 (bug)。尤其要看当我们 \TeX 文件时产生的 `.log` 文件。该 `.log` 文件会显示从正确的目录中加载的所有正确文件。几乎所有的安装问题均可从 `.log` 文件中找到答案。

如果可能，在报告隐错之前，用最新版本的 BEAMER 和 \TeX Live 检测安装，这可以帮助隔离来自于可能影响 BEAMER 的宏包的隐错。
5. 作为最后的报告，我们可以写邮件给作者。作者乐意接受邮件。正因为这样，作者不能保证我们的邮件会得到及时回复甚至根本就不回复。如果我们写邮件给链接到 bitbucket 开发页面的 BEAMER 邮件列表，我们的问题将有更多的机会被建立（自然，当作者有时间时将会阅读该邮件列表并回答问题）。

部分 I

开始

从这儿上路吧。先看看如何安装文档类。但愿这很简单，幸运的是所有的文档类都已正确地安装好了！在这儿我们还会知道在使用其它宏包（packages）时应该知道的一些事情。

接下来，一个简短的指导（tutorial）会告诉我们一个典型的演示稿所拥有的主要特征。跟随该指导，我们会发现制作演示稿的“可能的工作流程（workflow）”。遵循这个工作流程，以后的麻烦会少点。

第 I 部分包含准则（guidelines）。遵循这些准则，我们就能制作一份好的演示稿（但不能保证）。这个准则尽可能是非专业的，也就是说，它适应于制作一般的演示稿，跟我们是否用 BEAMER 制作过演示稿无关。

这部分的最后有一个和 BEAMER 相关的解答模板（solutions templates）的摘要。利用解答模板，我们可以马上开始制作演示稿。

2 安装

据我们的设备和需要不同，有不同的安装 BEAMER 文档类的方法。同时必需安装下述的宏包（packages）。安装之前，请先复习第 7 节的许可证，BEAMER 文档类是在它的允许下分发的（distributed）。

幸运的是，很可能我们的系统已经预装（preinstalled）了 BEAMER，因此可以跳过这一节。

2.1 版本和支撑

这本使用手册是 BEAMER 文档类 V3.24 的一部分。BEAMER 需要最新版本的几个标准宏包及下述版本的两个宏包（新版本可用但不是必需的）才能运行：

- `pgf.sty` V1.00。
- `xcolor.sty` V2.00。

如果我们使用 `pdflatex` 或 `lyx`（它们是非必须的），需要：

- `lyx` V1.3.3。其它版本也能运行。
- `pdflatex` V0.14 或更高版本。更早的版本不能运行。

2.2 安装封装包

我无功于 BEAMER 封装包（Pre-bundled Packages）的制作及管理，幸运的是，其它朋友做了这样的好事。我给不出安装这些封装包的详细指令，但我能告诉大家在哪儿可以找到它们，那些好心的朋友是如何教会我安装它们的。如果在安装中有什么问题，我们可以先看看下面。

2.2.1 T_EX Live 和 MacT_EX

在 T_EX Live 中，使用 `tlmgr` 工具（`tlmgr tool`）安装名为 `beamer`、`pgf`、`xcolor` 的宏包。如果我们有最新版本的 T_EX Live，而且是完全安装，就已包含了 BEAMER。

2.2.2 MiK_TE_X 和 proT_EXt

对于 MiK_TE_X 和 proT_EXt，请使用更新向导（update wizard）或宏包管理器（package manager）安装最新版本的名为 `beamer`、`pgf`、`xcolor` 的宏包。

2.2.3 Debian 和 Ubuntu

命令“`aptitude install latex-beamer`”很有用。如果需要，将自动安装 `pgf` 和 `latex-xcolor` 宏包。一会儿，下述的宏包就安装好了：

<http://packages.debian.org/latex-beamer>

<http://packages.debian.org/pgf>

<http://packages.debian.org/latex-xcolor>

Debian 5.0 “lenny”已包含了 T_EX Live 2007，Debian 6.0 “squeeze”已包含了 T_EX Live 2009。允许我们手动安装新版的 BEAMER（到本地目录，请参考下面），而无需更新其它的 L^AT_EX 宏包。

Ubuntu 8.04、9.04 和 9.10 包含了 T_EX Live 2007，Ubuntu 10.04 包含了 T_EX Live 2009。

2.2.4 Fedora

Fedora 9、10、11、12 和 13 包含了 \TeX Live 2007，它带有 BEAMER。可以运行命令“`yum install texlive-texmf-latex`”来安装。与 Debian 一样，可以手动安装新版的 BEAMER 到我们的本地目录（如下所述）。

因德里赫·诺维（Jindrich Novy）提供了用于 Fedora 12 和 13 的 \TeX Live 2010 rpm 包，其下载地址是：<http://fedoraproject.org/wiki/Features/TeXLive>。一旦 \TeX Live 2010 开始分发，Fedora 14 将包含它。

2.3 在 texmf 目录树中安装

无论如何，我们都不想封装包，一个“好”的办法是把 BEAMER 安装在一个所谓的 `texmf` 目录树（`texmf tree`）中。做法如下：从 <http://bitbucket.org/rivanvx/beamer> 获得 BEAMER 宏包的最新版的源代码（以 `.tar.gz` 或 `.zip` 结尾）（也许我们早已做完了这一步）。接下来，需要 PGF 宏包和 XCOLOR 宏包，它们必需单独安装（请参考其安装说明）。

宏包包含一组文件，`beamer.cls` 是这些文件中的一个，也是最重要的一个。现要，我们必需把这些文件放到一个合适的 `texmf` 目录树（`texmf tree`）中。

当我们要 \TeX 使用某个文档类（`class`）或宏包（`package`）时， \TeX 常常在所谓的 `texmf` 目录树中查找需要的文件。这些树只是一个包含这些文件的巨大目录。默认情况下， \TeX 在下面三个 `texmf` 目录树中查找文件：

- 根 `texmf` 目录树（`root texmf tree`），它常常位于：
`/usr/share/texmf/`，`/usr/local/texlive/texmf/`，
`c:\texmf\`，或 `c:\texlive\texmf\`。
- 本地 `texmf` 目录树（`local texmf tree`），它常常位于：
`/usr/local/share/texmf/`，`/usr/local/texlive/texmf-local/`，
`c:\localtexmf\`，或 `c:\texlive\texmf-local\`。
- 个人 `texmf` 目录树（`personal texmf tree`），它常常位于我们的 `home` 目录即：
`~/texmf/` 或 `~/Library/texmf/`。

将宏包安装在本地目录树（`local tree`）或个人目录树（`personal tree`）中，这依据我们是否具有对本地目录树的写权限（`write access`）。将宏包安装在根目录树（`root tree`）中可能带来麻烦，因为升级 \TeX 系统时会替换整个根目录树

不管在我们选择的哪一个 `texmf` 目录中，都会创建 `sub-sub-sub` 目录 `texmf/tex/latex/beamer` 并将宏包的全部文件放在该目录中

最后，我们必需刷新（`rebuild`） \TeX 的文件名数据库（`filename database`）。这可以通过运行命令 `texhash` 或 `mktexlsr`（它们是等效的）来实现。在 $\text{MiK}\TeX$ 宏包管理器和 \TeX Live `tlmgr`，有个菜单项⁵可以实现。

LYX BEAMER 文档类（`beamer class`）和 LYX 一起使用时，完成上面的步骤后，还必需让 LYX 知道 `beamer.layout` 这个文件，该文件不是 BEAMER 宏包的一部分，因为它是由 LYX 开发团队更新和管理。这意味着在最新版本的 LYX 中，该文件已安装而无需再劳了。

⁵在安装了 CTeX 套装的 Windows 7 中，该菜单项是：开始 → 所有程序 → CTeX → MiKTeX → Maintenance (Admin) → Settings (Admin) → General → Refresh FNDB。

想知道更详细的标准的宏包的安装过程，可以到 <http://www.ctan.org/installationadvice/> 这里去请教。然而，应注意 BEAMER 宏包中没有 `.ins` 文件（只需跳过那部分）。

2.4 更新安装

更新以前的版本，只需用新版本的文件替换 `texmf/tex/latex/beamer` 目录中的所有旧文件。最简单和方法是先删除旧版本的文件，然后继续进行上述的安装过程。

注意，如果安装了两个版本，一个安装在 `texmf` 目录，另一个安装在 `texmf-local` 目录， \TeX 更有可能选择 `texmf-local` 目录。这允许我们在没有管理员权限时手动更新宏包。

2.5 检测安装

要检测安装，请复制 `beamer/solutions/generic-talks` 目录中的 `generic-ornate-15min-45min.en.tex`⁶ 文件到我们制作演示稿常用的地方。然后对该文件运行几次 `pdflatex` 命令，并检测生成的 PDF 文件是否正确。如果正确，则安装是正确的。

LYX 要检测 LyX 的安装，可以以目录 `beamer/solutions/generic-talks` 中的 `generic-ornate-15min-45min.en.lyx`⁷ 为模板创建一个新文件。

2.6 和其它宏包及类文件的兼容

某些宏包（packages）或文档类（classes）和 `beamer` 文档类联用时，需要额外的选项或保护措施（precautions）。

```
\usepackage{AlDraTeX}
```

必需像抄录文本（verbatim text）那样处理由 `AlDraTeX`⁸ 创建的图形（Graphics）。原因是带有 `catcodes`⁹ 和空隔（spaces）的 `DraTeX` 字段非常象抄录文本。因此，要插入一张图片，要么给帧添加 `fragile` 选项，要么用 `\defverbatim` 命令创建一个盒子（box）用以容纳该图片。

```
\usepackage{alltt}
```

必需象抄录文本（verbatim text）那样处理处在 `alltt` 环境中的文本。所以给帧加上 `fragile` 选项以容纳这个环境或使用 `\defverbatim`。

```
\usepackage{amsthm}
```

会自动加载 `amsthm` 宏包，因为 BEAMER 用它排版定理（theorems）。该宏包在 `article` 模式中是必需的。如果我们不希望加载这个宏包，或该宏包与文档类不兼容时，我们可以用文档类的 `noamsthm` 选项以抑制自动加载该宏包。详情请参阅第 12.4 节。

```
\usepackage[french]{babel}
```

使用 `french` 样式时，某些和 BEAMER 文档类功能相冲突的功能（features）将被关闭。例如，按照主题（theme）的要求产生排序列表（enumerations）。

⁶在安装了 CTEX 套装的 Windows 平台中，该文件位于如 `D:\CTEX\MiKTeX\doc\latex\beamer\solutions\generic-talks` 中。

⁷在安装了 CTEX 套装的 Windows 平台中，该文件位于如 `D:\CTEX\MiKTeX\doc\latex\beamer\solutions\generic-talks` 中。

⁸`DraTeX` 是一个低级系统，它包含的命令用于创建简单的形状如直线、椭圆、弧线、文本，并将这些简单的形状联结成复杂的结构。

`AlDraTeX` 是 `DraTeX` 的扩展，它是一个高级系统，用于指定普通的实体如图表、曲线图。

⁹`catcode` 是验证码的一种，其中有猫咪形状的字母。

```
\usepackage[spanish]{babel}
```

PRESENTATION 使用 `spanish` 样式时，某些和 BEAMER 文档类功能相冲突的功能（features）将被关闭。特别是尖括号 `<` 和 `>` 将失效。

ARTICLE 在 `article` 模式中，要使尖括号 `<` 和 `>` 起作用，必需传递 `activespeccharacters` 选项给 `beamerbasearticle` 宏包。这将导致与叠层规则（Overlay Specifications）相关的问题。

```
\usepackage{color}
```

PRESENTATION `beamer.cls` 会自动加载 `color` 宏包。这使得在文档的导言（Preamble）区很难用普通的方式传递选项给 `color` 宏包。要传递一个选项列表 *(list of options)* 给 `color` 宏包可以使用下面的文档类（class）选项：

```
\documentclass[color=<list of options>]{beamer}
```

将 *(list of options)* 传递给 `color` 宏包。如果 *(list of options)* 包含多个选项，必需用花括弧（curly bracket）将它们围住。

ARTICLE 如果 `beamerarticle` 加载了 `noxcolor` 选项，则不会自动加载 `color` 宏包。

```
\usepackage{colortbl}
```

PRESENTATION 在新版的 `xcolor.sty` 中，如果我们想使用 `colortbl` 宏包，则必需传递 `table` 选项给 `xcolor.sty`。具体操作请参阅下面 `xcolor` 说明。

```
\usepackage{CJK}
```

PRESENTATION 使用亚洲字体时需要加载 `CJK` 宏包，并使用文档类（class）的 `CJK` 选项。请参阅 `beamerexample4.tex` 这个例子。

```
\usepackage{deluxetable}
```

PRESENTATION `deluxetable` 宏包产生标题的功能将失效。代之以 `caption` 模板（caption template）。

```
\usepackage{DraTex}
```

请参考 `AlDraTex`。

```
\usepackage{enumerate}
```

ARTICLE 在 `presentation` 模式中，会自动加载该宏包，但在 `article` 模式中不会自动加载。想在 `article` 模式中使用它的功能（features），必需“手工”加载该宏包。

```
\documentclass{foils}
```

如果我们想用 BEAMER 模仿 `foils` 文档类，请参阅第 24.3 节。

```
\usepackage[T1,EU1,EU2]{fontenc}
```

¹⁰ 我们只有使用像 `times` 字体或者 `lmodern` 字体这样以 T1 编码的轮廓字体（outline fonts）的时候才能用 T1 选项。标准安装下的标准计算机现代字体（Computer Modern fonts）（默认使用的字体，最初由

¹⁰这段由 C_T_EX 论坛管理员 Neals 翻译。

Donald Knuth 即唐纳德·克努特设计)并非 T1 编码的。如果使用该选项的话将会让我们的幻灯片在像 Acrobat、xpdf、evince 或 okular 这样的 PDF 阅读器下的渲染效果极其拙劣。如果我们真的想使用 T1 编码的计算机现代字体,就要调用 `lmodern` 宏包。请参阅第 18.2.3 节。这适应于 `latex+dvips` 和 `pdflatex`: 对于 `xelatex`, 使用 EU1 选项; 对于 `lualatex`, 使用 EU2 选项。注意 `xelatex` 和 `luatex` 支持 OpenType 字体, 字体编码的机理和 `pdflatex` 不同。详细内容请参阅第 18.2.3 节。

```
\usepackage{fourier}
```

这个宏包切换至 T1 编码, 它不重定义所有字体, 以致轮廓字体 (outline fonts) (nonbitmapped fonts 即非点阵字体) 被默认使用。例如, 无衬线字体 (sans-serif) 文本和打字机 (typewriter) 字体文本不被替换。如使用轮廓字体, 在引入 `fourier` 宏包之前, 写上 `\usepackage{lmodern}`。

```
\usepackage{HA-prosper}
```

这个宏包不能和 `BEAMER` 一起使用。但可以用 `beamerprosper` 宏包代替, 请参考第 24.1 节。

```
\usepackage{hyperref}
```

PRESENTATION `hyperref` 宏包由 `beamer.cls` 自动加载且建立某些选项。为传递另外的选项给 `hyperref` 宏包或使某些选项失效, 我们要使用下面的文档类选项:

```
\documentclass[hyperref=(list of options)]{beamer}
```

将 *(list of options)* (选项列表) 传递给 `hyperref` 宏包。

举例: `\documentclass[hyperref={bookmarks=false}]{beamer}`

二者择一地, 我们也可以使用 `\hypersetup` 命令。

ARTICLE 在 `article` 版本中, 如果要使用 `hyperref` 宏包, 必需手工加载, 我们也可以传递选项 `hyperref` 给 `beamerarticle`。它不会自动加载。

```
\usepackage[utf8,utf8x]{inputenc}
```

PRESENTATION 在使用 Unicode 时, 下面的一些文档类选项可用:

```
\documentclass[ucs]{beamer}
```

加载 `ucs` 宏包并传递正确的 Unicode 选项给 `hyperref`。同样的, 它预加载 (preload) Unicode 编码页 0 和 1。

```
\documentclass[utf8x]{beamer}
```

和 `ucs` 选项相同, 将输入编码设置成 `utf8x`。我们也可以使用 `ucs` 选项并在导言 (Preamble) 区声明 `\usepackage[utf8x]{inputenc}`。在大部分 `TeX` 系统中, 会自动加载 `ucs` 宏包。

如果我们在节 (section) 或小节 (subsection) 的标题中使用最先的两个编码页 (它包含拉丁字母和扩展的拉丁字母) 以外的 Unicode 字符, 我们必需使用命令 `\PreloadUnicodePage{<code page>}` 让 `ucs` 先加载这些编码页。如果得到一条“请插入到导言 (Please insert into preamble)”这样的消息, 那么没有加载字符。由 Unicode 给出的字符的编码页的字符数被分成 256。

```
\documentclass[utf8]{beamer}
```

该选项将输入编码设置成 `utf8`。它要使用 *without* `ucs`。它也要在导言区声明 `\usepackage[utf8]{inputenc}`。

注意：这些选项不适应于 `lualatex` 和 `xelatex`，因为 `lualatex` 和 `xelatex` 无需其它宏包天生就支持 Unicode。大多数情况下使用这些选项实际上会降低输出质量，因此使用它们时需小心。如果我们想使文档能用多个驱动编译，请参阅 `iftex`、`ifxetex` 和 `iflualatex` 宏包的文档。

ARTICLE 传递选项 `utf8` 给 `beamerarticle` 的效果与在导言区中声明 `\usepackage[utf8]{inputenc}` 的效果相同。此外，如使用 `lualatex` 或 `xelatex` 时需小心。

```
\usepackage{listings}
```

PRESENTATION 注意必需象处置 `verbatim` 环境一样，处置 `lstlisting` 环境。当使用包含彩色的 `lstlisting` 的 `\defverbatim` 时，请使用 `\defverbatim` 的 `colored` 选项。

举例：

```
\usepackage{listings}

\begin{document}
\defverbatim[colored]\mycode{%
  \begin{lstlisting}[frame=single, emph={cout}, emphstyle={\color{blue}}]
    cout << "Hello world!";
  \end{lstlisting}
}

\begin{frame}
  \mycode
\end{frame}
\end{document}
```

```
\usepackage{msc}
```

PRESENTATION 这个宏包在后台使用 `pstricks`，适应于 `pstricks` 的也适应于 `msc`。

```
\usepackage{musixtex}
```

当我们使用 `MusiXTEX` 排版乐谱时，我们必需激活 ϵ -`TEX`。大多数现代分发版（distribution）在 `pdflatex` 和 `latex` 中能激活它。然而，如果我们的分发版较旧，必需用 `pdfelatex` 或 `elatex` 编译文档，而不能用 `pdflatex` 或 `latex` 编译。

在 `music` 环境中，`\pause` 被重新定义以适应 `MusiXTEX` 的规定（在整体四分之一的时候休息一下）。在这个环境中，我们可以使用 `\beamerpause` 命令创建叠层。

```
\usepackage{pdfpages}
```

象 `\includepdf` 这样的命令仅工作在帧之外，它们“自己”产生页面。我们也可以声明 `\setbeamercolor{background canvas}{bg=}`。当背景（甚至白色背景）会打印出来盖住我们想包含的图像时，可以使用该命令。

举例：

```
\begin{document}
\begin{frame}
```

```

\titlepage
\end{frame}

{
\setbeamercolor{background canvas}{bg=}
\includepdf{somepdfimages.pdf}
}

\begin{frame}
A normal frame.
\end{frame}
\end{document}

```

```
\usepackage{<professional font package>}
```

PRESENTATION 如果我们使用 `professional font` 宏包，在如何排版变量方面，BEAMER 的内部重定义将和 `font` 宏包的高级排版方式相冲突。既然如此，我们必需使用文档类的 `professionalfont` 选项，以抑制 (suppress) 任何字体替代 (font substitution)。详细内容请参阅第 18.2.2 节

```
\documentclass{prosper}
```

如果我们想用 BEAMER (部分) 模仿 `prosper` 文档类，请参阅第 24.1 节。

```
\usepackage{pstricks}
```

Y 我们必需添加 `xcolor=pst` 选项让 `xcolor` 知道我们正在使用 `pstricks`。

```
\documentclass{seminar}
```

如果我们想用 BEAMER 模仿 `seminar` 文档类，请参阅第 24.2 节。

```
\usepackage{texpower}
```

不可以和 BEAMER 一起用，但可以用 `beamertexpower` 宏包代替，请参阅第 24.4 节。

```
\usepackage{textpos}
```

PRESENTATION 除非我们安装了不同的背景模板 (Background template)，否则 BEAMER 会自动安装白色背景 (white Background)。鉴于此，在使用 `textpos` 时，必需使用 `overlay` 选项，以便在任何事物之前放置盒子 (boxes)。二选一，我们也可以安装空的背景模板，但某些情况下，会导致在旧版本的 Acrobat Reader 中显示不正常。

```
\usepackage{ucs}
```

请参考 `\usepackage[utf8,utf8x]{inputenc}`。

```
\usepackage{xcolor}
```

PRESENTATION `xcolor` 宏包由 `beamer.cls` 自动加载，`color` 宏包同样如此。

```
\documentclass[xcolor=<list of options>]{beamer}
```

传递 `<list of options>` (选项列表) 选项给 `xcolor` 包。

当 BEAMER 和 `pstricks` 宏包一起使用时，记住传递 `xcolor=pst` 选项给 BEAMER（以后给 `xcolor`）。

ARTICLE 如果加载了带有 `noxcolor` 选项的 `beamerarticle`，则不会自动加载 `color` 宏包。

3 指导: 欧几里德教授的演示稿

该节提供一个简短的指导, 集中讲述我们开始使用 BEAMER 时可能会用到的 BEAMER 的功能。这个指导将忽略具体的细节, 这些细节将在以后阐述。

3.1 问题说明

我们希望帮助亚历山大大学的欧几里德教授创建一个演示稿 (presentation) 以演示其最近的发现: 质数有无限多个! 他的关于该发现的论文已被第 27 届国际质数 - 280 研讨会 (ISPN'80) 接受。教授想用 BEAMER 文档类 (class) 创建演示稿, 并在研讨会上演示。他的演讲及问答将耗时 20 分钟。

3.2 解答模板

欧几里德教授要做的第一件事就是寻找一个演示稿的解答模板 (Solution Template)。浏览了第 6 节后, 他发现 `beamer/solutions/conference-talks/conference-ornate-20min.en.tex`¹¹ ([点击查看](#)) 这个文件比较合适。教授在其论文的目录下创建了 `presentation` 子目录, 将解答模板文件拷入其中, 并将解答模板文件 `conference-ornate-20min.en.tex` 重命名为 `main.tex`。

如果欧几里德教授使用 LyX 编辑器, 他会执行“从模板创建 (New from template)”菜单命令并选择 `beamer/solutions/conference-talks/conference-ornate-20min.en.lyx`¹² 模板文件。在编辑器中打开该文件后, 发现开始的代码是 `\documentclass{beamer}`, 他感到很惊奇, 接下来的一行是 `\mode <presentation>`。欧几里德教授看不懂。发现文件里充满了他看不懂的东西后, 欧几里德教授决定先忽略它们

3.3 标题部分

接下来的事情是确定在哪里放置命令 `\title`。很自然, 可以用命令 `\title{There Is No Largest Prime Number}` 代替它, 因为它是论文的标题。他知道 `\title` 命令带有可选的选项“short”参数, 该参数放在方括号中, 当容纳标题的空间比较小时, 可以选用该参数。

接下来教授象下面那样调整 `\author` 和 `\date` 字段:

```
\author{Euclid of Alexandria}
\date[ISPN '80]{27th International Symposium of Prime Numbers}
```

对于日期 (date), 他觉得有点长, 所以用其简写形式 (ISPN'80)。进一步考虑后, 教授决定添加邮箱地址, 并象下面那样替换 `\author` 字段:

```
\author[Euclid]{Euclid of Alexandria \\ \texttt{euclid@alexandria.edu}}
```

不知何故, 教授竟然发现 BEAMER 没有“`\email`”命令。他决定写封邮件给 BEAMER 的作者, 希望他建立这条命令, 这稍稍推迟了演示稿的完成。

这里有两个字段 `\subtitle` 和 `\institute`。教授不知道, 但可以猜测其意义。教授对它们作了调整。(因为作者及大学都只有一个, 他没有使用 `\and` 和 `\inst` 命令, 前者用于分开多位作者, 后者用于分开多个大学。)

LYX 在 LyX 中, 教授只编辑了前几行, 这几行由诸如 Author、Title 或 Date 这些字段组成。他删除了可选的 short 字段。

¹¹ 在安装了 CTEX 套装的 Windows 平台中, 该文件位于如 `D:\CTEX\MiKTeX\doc\latex\beamer\solutions\conference-talks` 中。

¹² 在安装了 CTEX 套装的 Windows 平台中, 该文件位于如 `D:\CTEX\MiKTeX\doc\latex\beamer\solutions\conference-talks` 中。

3.4 封面帧

接下来的事情是在哪里创建第一个“帧”呢，在 `\begin{document}` 后面：

```
\begin{frame}
  \titlepage
\end{frame}
```

在 BEAMER 中，一个演示稿由一系列的帧（frame）组成，而每一帧又依次由一系列幻灯片（slide）组成；当幻灯片不止一张时，称它们为叠层（overlays）。通常，位于 `\begin{frame}` 和 `\end{frame}` 之间的内容会放在一张幻灯片中，不会分页显示。因此欧几里得教授合乎逻辑地推断，第一个帧由封面（title page）“填充”。

LYX 封面帧（The Title Page Frame）由 LyX 自动创建。其它帧由 `BeginFrame` 开始，由 `EndFrame` 结束，或者，自动从下一帧、小节、节开始。

3.5 创建演示稿的 PDF 文件

迫切想知道第一页象什么样子，教授用 `pdflatex` 编译（两次）`main.tex` 文件。也可以使用 `latex` 编译（两次），接着用 `dvips` 编译，然后用 `ps2pdf` 或 `lualatex` 编译（两次），或 `xelatex` 编译（两次）。最后用 Acrobat Reader 或 `xpdf`、`evince`、`okular` 浏览生成的 `main.pdf`。事实上，第一页包含了迄今为止欧几里得教授提供的所有信息。彩色的标题、圆形的拐角、还有阴影，给人的印象是多么的深刻。但他怀疑是否应该保持这样，他决定以后解决这个问题。

欧几里得教授欣喜地发现，顶部导航条（Navigation Bar）的节（section）或小节（subsection）的超链接是可以点击的。同样，底部的导航符看起来也是可以点击的。玩弄了一会儿，他发现单击导航符的向左或向右的小箭头会跳到前或后一幻灯片/帧/小节/节。单击导航符最左边或最右边的小箭头会跳到幻灯片/帧/小节/节的开始或结尾。他决定不写邮件给 BEAMER 的作者了，因为他认为大的导航符会让人分心。

LYX 选择查看 → PDF 浏览演示稿。在慢的电脑上这将要一会儿。请参考第 4.3.3 节以获得加快编译速度的方法。

3.6 目录

下一帧包含了一个目录（Table of Contents）：

```
\begin{frame}
  \frametitle{Outline}
  \tableofcontents
\end{frame}
```

而且，这个帧还有单独的标题（即 Outline）。该帧的注释说明，可以添加 `[pausesections]` 选项。试一下，把上面的帧改成：

```
\begin{frame}
  \frametitle{Outline}
  \tableofcontents[pausesections]
\end{frame}
```

用 `pdfLATEX` 重新编译演示稿后，教授发现在演示稿中有两张“目录”幻灯片。在第一张中，只显示第一节，在第二张中，会显示第一节和第二节（细看源文件后，欧几里得教授发现命令 `\section` 控制这些节）。`pausesections` 选项的效果看起来是让人讲完第一节后再显示第二节。当然，教授可以按向下或向右的按键（`down-` or `right-key`），以显示整个目录并且演讲第二节的内容。

3.7 节和小节

接下来教授看见的两条命令是：

```
\section{Motivation}
\subsection{The Basic Problem That We Studied}
```

这些命令位于帧之外。教授假设在调用点处（at the point of invocation）它们没有直接的作用，它们仅创建目录的条目（entries）。看来，用一个“Motivation”节是合理的，但他改变了 `\subsection` 的标题。

当他查看演示稿时，注意到他的假设不很正确：看来每一个 `\subsection` 命令都向演示稿插入一个包含目录的帧。循原路返回后他发现：`\AtBeginSubsection` 命令插入一个仅含有当前小节（the current subsection）的帧，当前小节在每一节的开始高亮显示。如果欧几里得教授不喜欢这样，他只要在每一个小节开始消失处删除整个 `\AtBeginSubsection` 和目录

`\section` 和 `\subsection` 命令带有可选的 `short` 参数。每当需要使用短的节名或短的小节名时，就可以加上 `short` 参数。虽然这与 BEAMER 处理 `\title` 的可选参数的方式是一致的，但与 L^AT_EX 处理节的可选参数的常规方式不同，在 L^AT_EX 的节中，可选参数指定什么将在目录中显示，而主要参数指定什么将在所有地方显示；在 BEAMER 中，这些情况正好相反。

3.8 创建一个简单的帧

欧几里得教授象下面那样修改下一个帧，这是演示稿的第一个“真正”的帧：

```
\begin{frame}
  \frametitle{What Are Prime Numbers?}
  A prime number is a number that has exactly two divisors.
\end{frame}
```

这样产生（yield）了预期结果（desired result）。在定义（质数）时进行强调（emphasis）看来是个好主意。欧几里得教授试着用命令 `\emph`，但发现强调力度不够。像使用 `\emph` 一样，BEAMER 提供了命令 `\alert`，它的参数默认用亮红色（bright red）显示。

LYX `\alert` 命令必需在笨拙的（awkward）T_EX-模式（TEX-mode）中使用。它很容易使文本变成红色。

接下来，欧几里得教授决定用 `definition` 环境包围所下的定义，以使它更醒目。

```
\begin{frame}
  \frametitle{What Are Prime Numbers?}
  \begin{definition}
    A \alert{prime number} is a number that has exactly two divisors.
  \end{definition}
\end{frame}
```

其它有用的环境如 `theorem`、`lemma`、`proof`、`corollary`、`example` 已经由 BEAMER 预定义好了。正如在 `amsmath` 中一样，它们均带有放置于括号中的可选参数。事实上，BEAMER 会自动加载 `amsmath`。

因为举例子是个好主意，欧几里得教授决定添加一个例子（example）：

```
\begin{frame}
  \frametitle{What Are Prime Numbers?}
  \begin{definition}
    A \alert{prime number} is a number that has exactly two divisors.
  \end{definition}
\end{frame}
```

```

\end{definition}
\begin{example}
  \begin{itemize}
    \item 2 is prime (two divisors: 1 and 2).
    \item 3 is prime (two divisors: 1 and 3).
    \item 4 is not prime (\alert{three} divisors: 1, 2, and 4).
  \end{itemize}
\end{example}
\end{frame}

```

3.9 创建一个简单的叠层

该帧看起来已经很漂亮了，而且是彩色的。然而，欧几里得教授想让三个条目（item）依次显示，而不是同时显示，为达到这个目的，他把 `\pause` 命令放到第一和第二个条目后面：

```

\begin{itemize}
\item 2 is prime (two divisors: 1 and 2).
  \pause
\item 3 is prime (two divisors: 1 and 3).
  \pause
\item 4 is not prime (\alert{three} divisors: 1, 2, and 4).
\end{itemize}

```

逐步显示后，他希望观众将注意力集中在他正在演讲的条目（item）上。进一步考虑后，他再一次删除了 `\pause`，在象上面简单的例子中，应用暂停（pause）是不明智的。实际上，欧几里得教授发现，好的演示稿只有在特定的情况下才会利用这种显示（uncovering）机制（mechanism）。

LYX 我们可以使用暂停风格（Pause style）添加暂停。

欧几里得教授发现在定义（definition）和例子（example）之间也可添加 `\pause`。这样看来，`\pauses` 似乎是优于（transcede）环境的，这非常有用。经过试验后，他发现 `\pause` 只在 `align` 环境中不能应用。于是他立即写邮件给 BEAMER 的作者告诉他的发现，从客气的回信中他知道了执行 `align` 将导致（does）一些问题（wicked things），并且没有解决办法（fix）。而且被告知在用户手册（user’s guide）的最后一部分，讲述了变通方法（workaround）。

3.10 使用叠层规则

下一帧放在一个“Results”节中以显示他的主要论点。欧几里得教授希望他的帧有更复杂的叠层行为（overlay behavior）：在一个有四个要点（points）的列表（enumeration）中，他希望逐个显示要点，并且显示第一个要点的同时显示第四个要点。这样做的目的是展示（illustrate）他的新的证明方法（proof method），即反证法（proof by contradiction），反证法就是以错误的前提（assumption）开始，经过一系列不重要的中间步骤，最后导致矛盾的结果。为达到这个目的，教授使用了叠层规则（*Overlay Specifications*）：

```

\begin{frame}
  \frametitle{There Is No Largest Prime Number}
  \framesubtitle{The proof uses \textit{reductio ad absurdum}.}

  \begin{theorem}

```

```

    There is no largest prime number.
\end{theorem}
\begin{proof}
  \begin{enumerate}
    \item<1-> Suppose  $p$  were the largest prime number.
    \item<2-> Let  $q$  be the product of the first  $p$  numbers.
    \item<3-> Then  $q + 1$  is not divisible by any of them.
    \item<1-> But  $q + 1$  is greater than  $1$ , thus divisible by some prime
      number not in the first  $p$  numbers.\qedhere
  \end{enumerate}
\end{proof}
\uncover<4->{The proof used \textit{reductio ad absurdum}.}
\end{frame}

```

叠层规则 (Overlay Specifications) 放在尖括号 (pointed bracket) 中。规则 <1-> 的意义是“从幻灯片 1 开始”。因此，第一个和第四个条目 (item) 将在帧的第一张幻灯片上同时显示，但另外两个标签条目 (item) 不显示。更确切地说，第二个要点从第二张幻灯片向前显示。BEAMER 会自动计算出每一帧所需的幻灯片的张数。因此，叠层规则 (Overlay Specifications) 是数字序列 (list) 或数字区间 (range)，这个区间是离散的 (leave open)。例如：-3,5-6,8- 的含义是“在所有的幻灯片中，除外幻灯片 4 和 7”。

LYX 要添加叠层规则，首先必须进入 T_EX-模式 (T_EX-mode) (按下 T_EX 图标) 并键入 <1->。这个 T_EX-文本 (T_EX-text) 必需放在条目 (item) 的开始处。

\qedhere 命令用于在排序列表 (enumeration) 内的行末放置 QED 符号 (symbol)。通常，会在的证明环境 (proof environment) 末尾自动插入 QED 记号，但在空行中这会很难看

除 \item 命令外，\uncover 命令也可以带有叠层规则。并在由叠层规则指定的幻灯片上显示它的参数 (argument)。在其它的幻灯片上，参数是隐藏的 (但仍占空间)。\only 命令是类似的，欧几里得教授也可以试试：

```
\only<4->{The proof used \textit{reductio ad absurdum}.}
```

在没有用叠层规则的幻灯片中，\only 命令仅仅“丢弃它的参数 (argument)”，并且参数不占空间。这将导致前三张幻灯片和第四张幻灯片文本高度的不同。如果文本是垂直居中，这将导致文本“不稳定 (wobble)”，因此，必需应用 \uncover 命令。然而，有时我们希望幻灯片中的一此东西“真正消失”，那么 \only 命令就很有用。欧几里得教授也可以应用类 (class) 选项 t，它使帧中的文本顶部垂直对齐。这样，不同高度的文本将不会导致不稳定。像下面这样，在帧环境中使用可选参数 [t]，同样可以实现单一帧中的垂直对齐 (Vertical flushing)。

```

\begin{frame}[t]
  \frametitle{There Is No Largest Prime Number}
  ...
\end{frame}

```

反之亦然，如果带有 t 类选项，那帧可以用 [c] 选项实现垂直居中。

事实证明，某些环境包括 theorem 环境和上述的 proof 环境也可接受叠层规则。接受了叠层规则的 theorem 环境或 proof 环境只在指定的幻灯片上显示。

3.11 构造一个帧

在下一个帧中，欧几里得教授希望对比质数相关的已解决（Solved）和未解决（Open）的问题。因为不存在类似于定理（theorem）环境的“已解决的问题（Solved problem）”环境，教授决定使用块（block）环境，在该环境中允许使用任意的标题：

```
\begin{frame}
  \frametitle{What's Still To Do?}
  \begin{block}{Answered Questions}
    How many primes are there?
  \end{block}
  \begin{block}{Open Questions}
    Is every even number the sum of two primes?
  \end{block}
\end{frame}
```

他可以在导言（Preamble）区放置下面的命令来定义自己的和定理环境相似的环境（theorem-like environment）：

```
\newtheorem{answeredquestions}[theorem]{Answered Questions}
\newtheorem{openquestions}[theorem]{Open Questions}
```

可选参数 [theorem] 保证这些环境与其它的环境以同样的方式被编号（number）。因为这些编号（number）不总是显示，给出这些编号与否无关紧要，但给出这些编号是一个很好的习惯（practice），也许哪一天欧几里得教授需要这些编号。

在二择一（alternative）时可能会嵌套 itemize：

```
\begin{frame}
  \frametitle{What's Still To Do?}
  \begin{itemize}
    \item Answered Questions
      \begin{itemize}
        \item How many primes are there?
      \end{itemize}
    \item Open Questions
      \begin{itemize}
        \item Is every even number the sum of two primes?
      \end{itemize}
    \end{itemize}
\end{frame}
```

考虑许久后，欧几里得教授决定把“已回答的问题（Answered Questions）”放在左边，把“未回答的问题（Open Questions）”放在右边，这样可以产生更强的视觉对比。为实现这一点，他使用了 columns 环境。在 columns 环境中，用 \column 命令产生新的一栏。

```
\begin{frame}
  \frametitle{What's Still To Do?}
  \begin{columns}
```

```

\column{.5\textwidth}
\begin{block}{Answered Questions}
  How many primes are there?
\end{block}

\column{.5\textwidth}
\begin{block}{Open Questions}
  Is every even number the sum of two primes?
\end{block}
\end{columns}
\end{frame}

```

试验后，教授不满意结果，因为左右两边块（block）的高度不同。他想左右两边块如果能垂直顶部对齐的话效果会更好。于是他在 `columns` 环境中加入了 `[t]` 选项。

欧几里得教授发现在第一栏的末尾放置 `\pause` 命令，将允许他只“显示（uncover）”帧的第二张幻灯片的第二栏。

3.12 添加参考文献

欧几里得教授想为未解决问题（open questions）列表添加一条引文（citation），因为这些问题是由他的好友克里斯琴（Christian）提出。教授不敢确定在他的演讲中使用参考书目（bibliography）是否是个好主意，但他义无反顾地这样做。

最后，欧几里得教授在参考书目中添加了条目（entry），且幸运地发现在已解决的文件（solution file）中已经有了。他不喜欢在附录（appendix）中添加参考书目，于是他删除了 `\appendix` 命令。同时，他注意到了 `<presentation>` 叠层规则（Overlay Specifications）并对它有点陌生，但它们之间互不影响。希望它们是有用的。他的参考书目如下所示：

```

\begin{thebibliography}{10}
\bibitem{Goldbach1742}[Goldbach, 1742]
  Christian Goldbach.
  \newblock A problem we should try to solve before the ISPN '43 deadline,
  \newblock \emph{Letter to Leonhard Euler}, 1742.
\end{thebibliography}

```

然后教授就可以添加一条引文：

```

\begin{block}{Open Questions}
  Is every even number the sum of two primes?
  \cite{Goldbach1742}
\end{block}

```

3.13 抄录文本

在另一帧中，欧几里得教授想显示一个由他朋友埃拉托色尼（Eratosthenes）寄给他的运算法则（algorithm）列表（据说是在他整理其收藏时发现的）。教授通常使用 `verbatim` 环境，有时也使用与 `lstlisting` 环境相似的环境去排版列表（listings）。在 BEAMER 中，同样可以使用它们，但必需为帧加上 `fragile` 选项：


```

\begin{frame}[fragile]
  \frametitle{An Algorithm For Finding Primes Numbers.}

\begin{verbatim}
int main (void)
{
  std::vector<bool> is_prime (100, true);
  for (int i = 2; i < 100; i++)
    if (is_prime[i])
      {
        std::cout << i << " ";
        for (int j = i; j < 100; is_prime [j] = false, j+=i);
      }
  return 0;
}
\end{verbatim}

\begin{uncoverenv}<2>
  Note the use of \verb|std::|.
\end{uncoverenv}
\end{frame}

```

进一步考虑后，欧几里得教授想逐步显示（uncover）部分运算法则，并且强调某几行或某行的一部分。他可以使用宏包如 `alltt` 来实现，事实上，由 BEAMER 定义的 `{semiverbatim}` 环境更有用：除 `\`、`{`、和 `}` 保留（可以通过键入 `\\`、`\{`、和 `\}` 来排版）它们的含义外，`semiverbatim` 环境象 `{verbatim}` 一样起作用。教授可以象下面那样排版他的运算法则：

```

\begin{frame}[fragile]
  \frametitle{An Algorithm For Finding Primes Numbers.}

\begin{semiverbatim}
\uncover<1->{\alert<0>{int main (void)}}
\uncover<1->{\alert<0>{\{}}
\uncover<1->{\alert<1>{ \alert<4>{std::}vector<bool> is_prime (100, true);}}
\uncover<1->{\alert<1>{ for (int i = 2; i < 100; i++)}}
\uncover<2->{\alert<2>{ if (is_prime[i])}}
\uncover<2->{\alert<0>{ \{}}
\uncover<3->{\alert<3>{ \alert<4>{std::}cout << i << " " ;}}
\uncover<3->{\alert<3>{ for (int j = i; j < 100;}}
\uncover<3->{\alert<3>{ is_prime [j] = false, j+=i);}}
\uncover<2->{\alert<0>{ \{}}
\uncover<1->{\alert<0>{ return 0;}}
\uncover<1->{\alert<0>{\{}}
\end{semiverbatim}

\visible<4->{Note the use of \alert{\texttt{std::}}.}
\end{frame}

```

`\visible` 命令和 `\uncover` 命令几乎有相同的作用。如果使用命令 `\setbeamercovered{transparent}` 使文本“transparent”隐藏，则会出现差异，在没有叠层规则的幻灯片中 `\visible` 命令仍使文本完全“不可见”。欧几里得教授感到命名约定（convention）有点奇怪，但无法彻查这个问题。

3.14 更改外观的方式 I：主题

演讲的内容准备好了，欧几里得教授决定使他的其演示稿拥有另外一个外观。他回到开始处并找到这行：

```
\settheme{Warsaw}
```

代入（substituting）其它城市的名称（他注意到，这些城市有理论计算科学的场所或协会，在这儿，经常由一个人拿着论文参加或进行演讲）后欧几里得教授可以改变演示稿的外观。他决定选择某个适当简单的主题（theme），因为他的演讲不太短，因此选定的主题必需能显示一些导航信息。

他停留在法兰克福（Frankfurt）主题上，发现黑 - 白对比太明显，于是添加：

```
\usecolortheme{seahorse}
```

```
\usecolortheme{rose}
```

结果看起来更柔和些。

欧几里得教授发现标题的字体不够正统（在亚历山大大学，正统的字体是最近的 chic），于是他添加

```
\usefonttheme[onlylarge]{structuresmallcapsserif}
```

欧几里得教授发现导航条（Navigation Bar）的字体太小而很难阅读。于是添加下面的命令：

```
\usefonttheme[onlysmall]{structurebold}
```

3.15 更改外观的方式 II：颜色和字体

为了使演讲尽可能完美，他决定标题字体毫无疑问用衬线斜体字（serif italics）。只改变标题字体，欧几里得教授使用了下面的命令¹³：

```
\setbeamerfont{title}{shape=\itshape,family=\rmfamily}
```

他发现字体仍然相当大（这是他喜欢的），令众人惊讶的是他并没有指定字体的大小却出现了这种情况。原因是调用了 `\setbeamerfont` 命令并且字体已经被字体主题（font theme）设置成了 `\large`。使用带星号的 `\setbeamerfont` 命令会“重置（resets）”字体。

欧几里得教授决定把标题的颜色改为有活力的红色（dashing red），也许可以加点黑色。他使用了下面的命令：

```
\setbeamercolor{title}{fg=red!80!black}
```

试验了下面的命令后，欧几里得教授欣喜地发现指定背景（Background）色也是有效的：

```
\setbeamercolor{title}{fg=red!80!black,bg=red!20!white}
```

最终，欧几里得教授对他的演示稿感到很满意，在研讨会上进行了顺利的演讲，并结交了许多新朋友。教授也写了邮件给 BEAMER 的作者，告诉他在使用 BEAMER 时失败或出现的一系列问题。让他感到失望的是，他得知这些问题必需等到 ISPN'79 才能解决，而且他知道 BEAMER 的作者需要一些时间进行研究，另外，未给他的演示稿提什么建议。

¹³请参阅第 18.3.3 节

4 创建 Beamer 演示稿的工作流程

本节呈现 (present) 制作 BEAMER 演示稿的可能的 workflow, 在制作演示稿的同时也可能会制作讲义 (handout)。象什么程序调用了哪个参数这样的技术问题在这节也将得到论述 (address)。

4.1 第一步: 建立文件

PRESENTATION 建议为每一个演示稿建立一个文件夹。即使我们的演示稿通常只有一个文件, $\text{T}_{\text{E}}\text{X}$ 会产生众多的额外文件以致于容易和其它的文件混淆。该文件夹的名称应按 ISO 格式以我们的演讲日期开始 (如在圣诞节的演讲, 该文件夹的名称以 2003-12-25 开始), 后跟能提示演讲内容的文字。该文件夹的名称以演讲日期开始这种命名方式, 可以使我们的众多演示稿文件夹在一个目录下排列得更有条理。如果我们为每一演示稿建立一个 extra 目录, 则可以把主文件命名为 `main.tex`。

为我们的演讲制作一个初始的 `main.tex` 文件, 从 `beamer/solutions` 目录中按需要复制一个现有的文件。其中有一系列可能的 BEAMER 解决方案, 这些解决方案包含演示稿的 $\text{T}_{\text{E}}\text{X}$ - 文件的模板 (templates)。

如果希望我们的演讲 (talk) 与某些不同的、非演示稿 (non-presentation) 论文 (article) 版的文本放在同一文件中, 建议建立一个更加详细的文件结构图 (scheme)。详细内容请参考第 21.2.2 节。

LYX 我们可以打开一个新文件然后选择 `beamer` 作为文档类, 也可以选择“从模板新建 (New from template)”然后从目录 `beamer/solutions` 能中选择模板。

4.2 第二步: 组建演示稿

下一步是用 `\section` 和 `\subsection` 填充演示稿以形成初步的纲要 (preliminary outline)。在第 5.1 节有线索提示如何形成一个好的纲要。

在主文件 (main file) 中放置 `\section` 和 `\subsection` 命令, 主文件几乎都是空的。在我们还没有目录的第一个工作版 (first working version) 时不要创建任何帧。这时主文件看起来象下面这样:

```
\documentclass{beamer}
% This is the file main.tex

\usetheme{Berlin}

\title{Example Presentation Created with the Beamer Package}
\author{Till Tantau}
\date{\today}

\begin{document}

\begin{frame}
  \titlepage
\end{frame}

\section*{Outline}
\begin{frame}
  \tableofcontents
```

```

\end{frame}

\section{Introduction}
\subsection{Overview of the Beamer Class}
\subsection{Overview of Similar Classes}

\section{Usage}
\subsection{...}
\subsection{...}

\section{Examples}
\subsection{...}
\subsection{...}

\begin{frame}
\end{frame} % to enforce entries in the table of contents

\end{document}

```

末尾的空帧（最后会被删除）确保了节（sections）和小节（subsections）事实上是目录（Table of Contents）的一部分。这个帧是必需的，因为跟在文档最后一页的 `\section` 或 `\subsection` 命令不起作用。

4.3 第三步：创建一个 PDF 或 PostScript 文件

PRESENTATION 一旦建立了初始结构，我们必需为我们的（没有内容的）演讲（talk）创建 PDF 或 PostScript 文件以确保它们能正确运转。这个 PDF 或 PostScript 文件仅包含标题页（title page）和目录（Table of Contents）。

LYX 使用“查看（View）”检测演示稿编译是否正确。我们不必把目录放在一个帧中，但是标题页（title page）会自动产生。

4.3.1 创建 PDF 文件

PRESENTATION 对 `main.tex` 运行至少两次 `pdflatex` 才能生成相应的 PDF 文件。我们必需运行两次 `pdflatex`，这样 \TeX 才能产生目录。（通常需要运行多次才能生成各种辅助文件。）在下面的例子中，大于号（greater-than-sign）是提示（prompt）。

```

> pdflatex main.tex
... 许多输出 ...
> pdflatex main.tex
... 许多输出 ...

```

二选一，在上面的命令中我们可以使用 `lualatex` 或 `xelatex` 代替 `pdflatex`。

接下来，我们可以使用 Acrobat Reader、`xpdf`、`evince` 或 `okular` 查看生成的演示稿。

```

> acroread main.pdf

```

LYX 选择“查看 pdf”查看我们的演示稿。

4.3.2 创建 PostScript 文件

PRESENTATION 要创建 PostScript 文件，首先必需确定 HYPERREF 宏包（由 BEAMER 文档类自动加载）使用了 `dvips` 选项或其它兼容的（compatible）选项，详细内容请参阅 HYPERREF 宏包的说明文档。究竟使用了什么选项依赖我们本地的 `hyperref.cfg` 文件的内容。我们可以通过传递 `dvips` 选项或其它兼容的选项给 BEAMER 文档类（写成这样：`\documentclass[dvips]{beamer}`）以强制它使用这个选项，这个选项最终传递给了 HYPERREF 宏包。

我们可以运行两次 `latex`，然后运行 `dvips`。

```
> latex main.tex
... 许多输出 ...
> latex main.tex
... 许多输出 ...
> dvips -P pdf main.dvi
```

选项（`-P pdf`）告诉 `dvips` 使用 Type 1 轮廓字体（outline fonts）代替通常的 Type 3 点阵（bitmap）字体。如果存在问题，我们可以忽略这个选项。

我们可以使用下面的命令把 PostScript 文件转换成 pdf 文件

```
> ps2pdf main.ps main.pdf
```

LYX 使用“查看（View）Postscript”查看该 Postscript 文件

4.3.3 加快编译速度的方法

在创建演示稿的过程中，用 \TeX 快速地编译我们的 `.tex` 文件并使演示稿只包含重要的信息是有益的，特别是当我们的电脑很慢时。既然如此，我们可以想办法提高编译的速度¹⁴，首先我们可用 `draft` 文档类选项。

```
\documentclass[draft]{beamer}
```

用灰色矩形（它们的尺寸经过计算）替换顶部导航区（headline）、底部导航区（footline）、和侧栏（sidebar）。许多其它的宏包包括 `pgf` 和 `hyperref` 当给定 `draft` 选项时也会得到“提速”。

其次，我们可以使用下面的命令：

```
\includeonlyframes{<frame label list>}
```

这条命令的行为（behave）有点象 `\includeonly` 命令：只包括列表（list）中提及的帧，其它的帧将被抑制。然而，节（section）和小节（subsection）命令仍得到执行，因此我们仍然会有正确的导航条（Navigation Bar）。标记（labeling）或声明（say）当前的帧为 `current`，然后声明 `\includeonlyframes{current}`，这样，我们就可以快速地编译单个帧。

帧标签列表 `<frame label list>` 是个由逗号分开的列表（没有空格），这个列表由添加了标签的帧的名称组成。要标记一个帧，必需传递 `label=<name>` 选项给 `frame` 命令或 `frame` 环境。

举例：

```
\includeonlyframes{example1,example3}
```

¹⁴使一段不编译的方法是：`\iffalse` 不要编译的部分 `\fi`

```

\frame[label=example1]
{This frame will be included. }

\frame[label=example2]
{This frame will not be included. }

\frame{This frame will not be included.}

\againframe{example1} % 将被包括进去

```

4.4 第四步：创建帧

一旦目录看起来令人满意，就可以添加帧（`frame`）环境，为我们的演讲稿创建帧。在第 5.1.3 节，我们将会知道有关在帧上应该放置些什么东西的准则（`guidelines`）。

LYX 要创建帧，请使用样式“`BeginFrame`”。该样式已在导航条（`line`）上给定了帧标题（`Frame Title`）。上一帧会在下一帧开始时自动结束，也会在一个节（`section`）和小节（`subsection`）命令处自动结束，帧还会在样式“`EndFrame`”的空导航条（`empty line`）处自动结束。注意演示稿的最后一帧必需以“`EndFrame`”结束，在附录（`appendix`）前的最后一帧也必需以这种方式结束。

4.5 第五步：测试演示稿

经常测试演示稿。为此，必需在安静的环境中朗读（`vocalize`）或默读（`subvocalize`）我们的演讲（`talk`）。通常会发现我们的演讲太长了。删除演讲稿的一部分以适合分配好的时段（`allotted time slot`）。不要加快演讲的速度压缩（`squeeze`）我们的演讲以适合给定的时间，如果这样的话，注定会失去观众。

不要试图马上就能创建“完美的（`perfect`）”演示稿。宁可测试再测试演讲，必要时修改演示稿。

4.6 第六步：创建讲义

4.6.1 创建讲义

一旦我们的演讲（`talk`）完成了，如果适当的话，就可以创建一份讲义（`handout`）。这可以按第 21.1 节所述的方法加上文档类的讲义（`handout`）选项来实现。通常我们希望在一個页面上（`on one page`）放置几张讲义的幻灯片（`handout slides`），参考下面的方法很容易实现这一点。

也许我们希望为演讲（`talk`）创建一个论文版本（`article version`）。演示稿的“论文版本”是用 `article` 或 `llncs` 文档类或类似的文档类排版的 $\text{T}_\text{E}\text{X}$ 文本。BEAMER 文档类具有这样的功能，它可以使论文版本（`article version`）和演示稿版本（`presentation version`）共存（`coexist`）于同一文件并共享代码。而且，我们可以将演示稿的幻灯片以图像（`figures`）的形式包含在论文版本中。如何建立论文版本（`article version`）请参阅第 21.2 节。

LYX $\text{L}_\text{Y}\text{X}$ 中创建论文版本。我们可以试试看，但不建议这样做。

4.6.2 打印讲义

打印演示稿的最简便的方法是使用 Acrobat Reader，并在打印对话框中激活“放大页面以适合纸张大小”选项。必需激活这个选项，因为幻灯片默认大小是 128mm×96mm（而 A4 纸的大小为 210mm×297mm）。

对于 PostScript 文件，以及在一个页面上打印多张幻灯片，上述简便的方法（即使用 Acrobat Reader，并在打印对话框中激活“放大页面以适合纸张大小”选项）将不起作用。这种情况下，可以使用 `pgfpages` 宏包，这个宏包可以直接使用 `pdflatex`、`lualatex`、`xelatex` 和 `latex + dvips`。注意：`pgfpages` 宏包会撤消（*destroys*）超链接（*hyperlinks*）。这归因于 PDF-规范中的基本原理（*fundamental*）上的瑕疵（*flaws*），并且该瑕疵或许不能改变。

`pgfpages` 宏包可以对页面作出各种各样的设置。打印 BEAMER 幻灯片时最重要的设置可以通过下面的命令实现：

```
\usepackage{pgfpages}
\pgfpagesuselayout[resize to][a4paper,border shrink=5mm,landscape]
```

该命令的含义是：“重设所有页面为横向（*landscape*）A4，不管它们的原始尺寸，均收缩 5mm，即在页面的周围有点空隙”。自然而然，我们可以用 `letterpaper` 或其它标准的纸张尺寸代替 `a4paper`。更多的选项和更详细的内容请参阅 `pgfpages` 宏包的说明文档。

接下来也许我们希望把多张幻灯片放在一个页面中，这可以用下面的命令实现：

```
\usepackage{pgfpages}
\pgfpagesuselayout{2 on 1}[a4paper,border shrink=5mm]
```

该命令的含义是：“在一个页面中放置两张讲义的幻灯片，并且调整大小以适合 A4 纸”。注意：这时我们无需将最终的页面设为横向（*landscape*），毕竟，这不是处在横向模式（*mode*）中。

可以使用 `4 on 1` 代替 `2 on 1`，但必需再使用 `landscape` 一次，同样可用 `8 on 1` 甚至 `16 on 1` 代替 `2 on 1`，以获得更好（但字迹难以辨认）的概貌（*overview*）。

如果我们在一个页面中（*on one page*）放置几张幻灯片，并且这些幻灯片的背景是白色的，那么，在导言区（*Preamble*）放置下面的命令将会有好处：

```
\mode<handout>{\setbeamercolor{background canvas}{bg=black!5}}
```

这会使幻灯片的讲义版本（*Handout version*）拥有非常亮的灰色背景。如果在一个页面中放置几张幻灯片，将很容易分辨（*discern*）幻灯片的边界。

5 创建演示稿的准则

在这节，我们草拟 (sketch) 了创建演示稿时应坚持 (stick to) 的准则 (guidelines)。这些准则来自于经验 (experience)、常识 (common sense)、或其它人或书本上的建议 (recommendation)。这些准则的确不是戒律 (commandment)，但如果不遵循它们，将会导致灾难 (catastrophe)。印刷术 (typography) 的核心规则 (central rule) 同样适应于创建演示稿：规则可以被打破但不可以被忽略。

5.1 组建演示稿

5.1.1 知晓时间限制

当我们开始创建时，真正当心的第一件事是我们的演示稿应该花多长时间。据不同的场合，可能从 2 分钟到 2 小时不等。

- 对于帧的数量，一个简单的规则是：每分钟最多一帧。
- 大多数情况下，我们的演示稿时间比我们想要的短。
- 不要在规定的时间内挤进更多的内容。不管细节看起来如何重要，忽略细节抓住主要内容比既不抓主要内容又不抓细节要好。

在多数情形中，快速判断 (appraisal) 我们有多少时间，这样我们就知道不能去提及某些细节。清楚这一点，可以节省一些时间，在这时间内准备的幻灯片后来无论如何又必需删除。

5.1.2 全局结构

创建有时间限制的演示稿的“全局结构”，按照下面这几条进行：

- 为在有限的时间内要讲到的内容列一清单。
- 为上述内容清单分成节 (Section) 和小节 (Subsection)。
- 对于长的演讲 (如 90 分钟的演讲)，我们必需用 \part 命令把它分成独立的几部分 (如“演讲前的概述部分”和“主体部分”)。注意每部分 (part) 均有各自的目录 (Table of Contents)。
- 不要害怕在后来的演讲中改变结构。

部分、节、小节

- 每部分 (part) 不要超过四节 (section) 也不要少于两节。

即使是四节也太多，除非它们有共同的简易模式。五节甚至更多对观众来说很难记住。毕竟当我们演示目录时，观众将难以真正抓住不同节的要点 (importance) 及其关联性 (relevance)，从而在到达该节时忘记它们。

- 理想的目录应该是透过其本身就可理解。特别是观众在听我们的演讲之前就能够理解它们。
- 保持节和小节 (subsections) 标题含义明了 (self-explaining)。
- 节和小节必需符合逻辑结构 (logical pattern)。

- 先说明一下我们的演讲是干什么的。（不是每个人都知道这一点。无知的观众规律表明：重要的观众知道的东西比我们认为每个人都应该知道的东西更少，即使我们考虑了这条规律）。
- 然后说明我们或其它人关于主题（subject matter）的发现。
- 结束演讲时总结一下，简短地重复我们演讲的主要内容。观众在演讲开始和结束时注意力更集中。这个简短的总结（summary）是我们让观众接受信息的“第二次机会”。
- 也可以用 \appendix 命令添加附录（appendix）。在这里可以放置我们不想演讲的任何东西，当被提问时它们可以派上用场。
- 不要使用小小节（subsubsections），它们会带来不幸（evil）。

提供一份摘要 在论文（papers）中，摘要（abstract）提供了整篇论文的简短概述（summary），它只有 100 字左右。目的是帮助读者决定是否阅读整篇论文。

- 因为观众可能在显示第一张幻灯片后溜走，对于演示稿（presentation），我们无需提供摘要。
- 然而，如果能（succinct）简洁有趣地陈述（nice statement）我们的演讲，则可以包含一份摘要。
- 确保所包含的摘要只是简短的信息（message），而不是长的原文（text）。
- 千万不要把论文（paper）的摘要用于演示稿（presentation），除非这个摘要是“ $We\ show\ P = NP$ ”或“ $We\ show\ P \neq NP$ ”
- 如果摘要是上述二者之一，请仔细检查（double-check）哪个是对的。

已编号的定理和定义 全局性地构建（structuring）（数学）论文（articles）和书籍（books）的一个常用方法是给定理（theorems）和定义（definitions）编流水号。不幸的是，对于演示稿，情况更复杂，我不主张在演示稿中给定理和定义编流水号。观众没有机会记住这些数字。决不要说这样的话：“现在，据我以前说过的定理 2.5，我们将 ...”，而应该将“定理 2.5”替换成“Kummer’s 定理”。如果定理 2.5 是模糊不清的（obscure），没有它自己的名字（不象 Kummer’s 定理、主定理或第二主定理、关键引理），当我们再次提及它们时，观众无论如何也记不起它们。

在我们看来，在演示稿中，给定理编号有意义的（make sense）唯一情形是在演讲（lecture）中，学生可以阅读到与演讲相对应的演讲记录（lecture notes），在演讲稿中，定理编上了同样的流水号。

如果给定理和定义编号，请编流水号。如果有一条定理（theorem）、一条引理（lemma）、一条定义（definition），给它们编号成定理 1、引理 2、定义 3。一些人更喜欢把它们编号成 1。我强烈反对这样做。这样做实际上查询不到任何东西，因为定理 2 可能出现在定义 10 之后或相反（the other way round）。论文（Papers）、更糟糕的是书籍（books）如果出现定理 1 和定义 1 这种编号将是一种痛苦。

- 别让它人遭受痛苦

参考书目 也许我们希望在演讲（talk）结束时提供一份参考书目（bibliography），这样大家可以“更深入地阅读”。在演示稿中添加参考书目请紧记以下几条：

- 演示稿中的参考书目不宜过长，应只包含少数几个参考文献（references）。（这条仅适应于演讲即 talk 本身，不适应于讲义即 Handout。）

- 如果我们提供的参考文献在一张幻灯片中容纳不下，可以明确告诉我们的是这些参考文献不会被人记住。
- 提供参考文献仅为了“更深入地阅读（further reading）”，不要象论文（paper）那样列出所有的东西。
- 除非应别人要求，否则不应提供一长列我们自己的其它大作（great papers）。
- 使用 `\cite` 命令会使人糊涂，因为观众很少有机会记住交叉引用（citations），如果引用了参考书目，应象 “[Tantau, 2003]” 这样引用作者全名和年份，而不应该象 “[2,4]” 或 “[Tan01,NT02]” 这样。
- 如果我们很谦虚（modest），引用我们自己时，可以象 “[Nickelsen and T., 2003]” 或 “[Nickelsen and T, 2003]” 这样缩写（abbreviate）我们的名字。然而这样会让观众感到迷惑，因为他们不能立即明了谁是“T.”，所以我推荐使用全名。

5.1.3 帧结构

正如整个演示稿一样，必需创建每一个帧（frame）。只填满了长文本的帧很难跟随（follow）。我们的任务就是组织每一帧的目录（contents），这样，观众一看就知道哪些内容是重要的，哪些是细节，这些内容的关系如何等等。

帧标题

- 在每一帧中放置标题（title）。它说明了这个帧的内容，从而大家无需了解幻灯片上的所有具体内容。
- 标题应能反映内容，而不应该是不可理解含义模糊的摘要（cryptic summary），除非别人已了解了整张幻灯片。例如，标题“The Poset”会让每一个人迷惑，这张幻灯片是讲什么的呢。而标题“偏序集（Partially Ordered Sets, Posets）定义的回顾”或“因子型矩阵（Genotype Matrix）栏的偏序”则含义更明确。
- 理想的是，连续帧的标题连贯起来就能“反映一件事情（tell a story）”。
- 在英语中，帧标题（Frame Title）中的单词除象“a”或“the”这样的单词外（因为在标题中），要么全部大写，要么全部小写，不要大小混写；坚持前后一致性原则。这条原则同样适用于块（block）的标题。例如，不要使用象“A short Review of Turing machines.”这样书写的标题。而应用象“A Short Review of Turing Machines.”或“A short review of Turing machines.”（Turing 仍然用大写字母开头是因为它是人的名称即图灵）这样书写的标题。
- 英语的全文标题要大写，不管其它的内容是否大写。
- 在德语和其它语言中，有许多大写单词，它们常使用恰当的（correct）大写-/小写（upper/lowercase）字母。不要大写所有的东西，除非它经常是大写的。

一个帧可以容纳多少东西？

- 一个帧的内容宁少勿多。通常是 20 至 40 个单词，最多 80 个。
- 不要假设每一位观众在演讲的主题（subject matter）方面都是专家。即使他们是专家，几年前我们考虑的事情，也许他们现在才听到。我们必需留有时间快速地提醒（reminder）什么是“语义复杂性类（semantical complexity class）”或什么是“ ω -完全偏序（ ω -complete partial ordering）”。一些特定的名词要给出解释。

- 不要把整个演讲过程中都不会讲到的东西放在幻灯片中，不要给人留下我们的演讲主题很复杂这样的印象。但我们也可以演讲幻灯片中没有的内容。
- 尽量保持简单。通常，我们的观众看一张幻灯片的时间不会超过 50 秒。他们没有时间思索（puzzle through）长长的句子或复杂的公式。
- Lance Forthnow 认为（claims）：PowerPoint 使用者更适合演讲。他的理由是：因为 PowerPoint 在排版数学方面很整脚，因而它使用很少的数学内容，这样使演讲容易听懂。

在我看来，这有一定的道理。TeX 极强的数学排版能力毫不费力地引诱我们使用更多的数学公式。例如，无需输入 “Since $|\{x \in \{0,1\}^* \mid x \sqsubseteq y\}| < \infty$ ”，只需输入 “Since y has only finitely many prefixes, we have...”。

也许我们会惊讶数学文本怎么能用普通的英文再现。很自然，如果我们演讲的正是数学论点，那就使用 TeX 排版我们的中心内容吧。

构造一个帧

- 象 block、theorem、proof、example 等等一样，使用块（Block）环境。
- 对排序式（enumerations，或称列举）和常规（itemize，或称项目）文本比对普通文本（plain text）更喜爱。
- 定义多个事物时，使用叙述式列表（description，或称解说列表）。
- 不使用超过两个级别的“子常规列表（subitemizing）”。BEAMER 虽然支持三个级别，但我们不要应用那三个级别。通常，我们甚至不要使第二个级别。而应用好的图形代替。
- 不要无限制使用常规列表（itemize，或称项目式列表）或排序式列表（enumerate，或称列举式列表）。
- 不要分段显示（piecewise uncover）列表。
- 强调（Emphasis）是创建结构（creating structure）的重点。使用 `\alert` 命令高亮显示（highlight）重要的内容。它们可以是一个单词或整个句子。然而，不能滥用高亮显示，因为滥用会减弱高亮显示的效果。
- 使用分栏（columns）。
- 决不使用脚注（footnotes）。它们会中断阅读流程。不管脚注的内容是多么的重要以至于要放在正常的文本中，或者脚注的内容不重要以至于可以忽略（特别在演示稿中）。
- 使用 `quote` 或 `quotation` 排版引文（quoted text）。
- 除长的参考书目（bibliography）外，不要使用 `allowframebreaks` 选项。
- 不要使用长的参考书目（bibliographies）。

编写正文

- 使用短句。
- 更喜欢短语而不喜欢长句。例如，用“Left: A Turing machine. Right: A finite automaton.”代替“The figure on the left shows a Turing machine, the figure on the right shows a finite automaton.”。如能把它放入常规列表（itemize）或解说式列表（description）中则更好。
- 正确地添加标点：短语后面不加标点，完整的句子中间或结尾添加句号（complete punctuatio）。
- 不要为了“在一帧内放更多的内容”而选用小字体。永远不要使用糟糕的 `shrink` 选项。
- 不使用连字符连接单词。如果确实需要，用命令 \- “手工”连接单词。
- 用命令 \\ “手工”断行而不要依赖自动断行。并且在逻辑停顿处断开。例如，在句子“the tape alphabet is larger than the input alphabet”中，正确的断行是在“is”之前和第二个“the”之前。错误的断行是在“alphabet”之前或“larger”之前。
- 图形中的文本和数字必需和普通的文本大小相同。中轴上（on axes）字迹模糊的数字通常毁坏图表和它的信息。

5.1.4 交互元素

以完全线性方式演示幻灯片的过程中，理想（ideally）的演示方法大概（presumably）是每按一次下一页按钮（page-down-key）就演示一张幻灯片。然而，存在不同的原因使我们背离（deviate）这个线性顺序，这些原因包括：

- 我们的演示稿包含“不同的细节层次（levels of detail）”，它可以或不可以跳读或展开（skip or expand），这取决于观众的反应。
- 我们被提问，希望引导到辅助的（supplementary）幻灯片上。
- 在演示复杂的图片时，我们必需“放大（zoom out）”不同的部分以显示细节。
- 我们被问及和以前幻灯片有关的问题，不得不找到以前的幻灯片。

我们不可能事先准备好应付上述最后一种情况（即：我们被问及和以前幻灯片有关的问题，不得不找到以前的幻灯片），在这种情况下，我们可以用导航条和导航符（navigation bars and symbols）查找到相应的幻灯片，请参考第 8.2.3 节。

对于前三种情况，则可以事先准备好“计划好的便道（detours）”或“计划好的捷径（short cuts）”。

- 我们可以添加“跳过按钮（skip buttons）”。当按下该按钮时，将跳过（jump over）演讲的定义明确的（well-defined）部分。和连续按下向前按钮（the forward key）相比，按下跳过按钮有两个优点：第一，可以在正确的位置停下来；第二，按钮上的标签会给观众直观的反馈，它可以告诉观众什么会被跳过。例如，当我们按下带有“跳过证明（Skip proof）”标签的跳过按钮时，没有人会迷惑自己已经跳过了什么内容。
- 我们可以在演讲中添加附录（appendix）。附录应“完全独立（perfectly separated）”。一旦“进入”附录部分（大概通过 hyperjumping 即超链接进入），它的结构将一目了然。可以把所有我们在演讲过程中不打算讲的帧（但是，它们被问及时却可以很快被找到）放到附录部分。

- 我们可以添加“前往按钮 (goto buttons)”和“返回按钮 (return buttons)”。按下前往按钮将跳到演示稿的特定部分，在这里，会显示额外的细节。并且有返回按钮返回到前往按钮所在的地方。
- 在 BEAMER 中，我们可以使用 `\againframe` 命令“继续”先前在某个地方开始的帧，在这个帧中，特定的细节被隐藏 (suppressed)。我们可以在后面的地方使用 `\againframe` 命令，例如仅在附录部分显示附加的幻灯片。
- 在 BEAMER 中，我们可以使用 `\framezoom` 命令创建链接以放大 (zoom out) 复杂幻灯片的局部。

5.2 使用图形

Graphics often convey concepts or ideas much more efficiently than text: A picture can say more than a thousand words. (Although, sometimes a word can say more than a thousand pictures.) 图形 (graphics) 比文字 (text) 能更高效地传递概念或想法 (concepts or ideas): 一幅图形胜过 1000 句话。(然而，有时一句话胜过 1000 幅图形)。

- 只要可能，在每张幻灯片中放置 (至少) 一幅图形。这可以极大地直观地 (visualizations) 帮助观众。
- 通常，把图形放置在文字的左边。(使用 `columns` 环境)。在由左向右 (left-to-right) 的阅读习惯中，我们首先看左边。
- 图形和文本应有相同的印刷参数 (typographic parameters): 在图形中使用和主体文本 (main text) (相同尺寸的) 相同字体。在图形中的小圆点 (small dot) 应和文本中的小圆点有相同的尺寸。行宽 (line width) 应和用于创造字体轮廓的笔划宽度 (stroke width) 相同。例如，一个 11pt 非加粗的计算机现代字体 (Computer Modern font) 具有 0.4pt 的笔划宽度。
- 点阵图 (bitmap graphics) 如照片，比文本具有更丰富的色彩，矢量图 (vector graphics) 象主体文本 (main text) 一样，遵循同样的“颜色规律 (color logic)” [如: 黑色 = 普通的线条 (normal lines)、红色 = 高亮部分 (highlighted parts)、绿色 = 例子 (examples)、蓝色 = 结构 (structure)]。
- 象正文 (text) 一样，我们应该解释清楚图形上显示的所有东西。未解释清楚的细节会使观众迷惑，我们遗漏的东西是否重要。从论文 (paper) 或其它地方输入图形时要小心。它们拥有比我们所能解释的更多的细节，我们必需简单从事 (即解释得简单些)。
- 有时图形的复杂性 (complexity) 是人为的 (intentional)，我们甘愿花更多的时间对图形进行更详细的解释。既然如此 (In this case)，也许我们经常陷入这样的难题中：图形的细节很难对观众讲清楚。在这种情况下，我们可以使用 `\framezoom` 命令对图形的局部进行放大，这样可以让观众对放大的图形局部迅速产生兴趣，参考第 11.3 节。

5.3 使用动画和过渡

- 使用动画 (animation) 演示系统的动态部分 (dynamics of systems)、运算法则 (algorithms) 等。
- 不要用动画吸引观众的眼球。这样会把观众的注意力从幻灯片的主题上转移。不管旋转飞舞的定理看起来是多么地可爱，也不管我们多么强烈地感到观众需要动画，其实大多数观众会认为我们在取笑他们 (making fun of them)。

- 除非有充足的使用理由否则不要使用分散特效 (distracting special effects) 如“溶解 (dissolving)”幻灯片。如果要使用,也只是偶尔 (sparsely) 使用一下。在某些情形中使用特效是有益的:例如,在一张幻灯片上显示一个小孩,应用溶解特效后,取而代之显示一个成人。这样,溶解特效使观众直观地感受到一个小孩“慢慢长大”成成人。

5.4 选择恰当的主题

BEAMER 提供了许多不同的主题 (themes)。在选择它们时,请紧记以下几点:

- 不同的主题适合不同的场合。不要过分依赖某个喜欢的主题;据不同的场合选取不同的主题。
- 一个长的演讲 (talk) 和一个短的演讲需要差不多相同 (more likely) 的导航提示 (navigational hints)。为学生作 90 分钟的演讲,应该选择能在侧边栏高亮显示当前话题的主题,这样,每个人都知道“我们正在讲什么”;作 10 分钟的介绍性讲话 (introductory speech),选择能显示目录的主题可能是愚蠢的 (silly)。
- 在观众不了解我们的情况下 (如在讨论会上) 选择能显示作者和联系方法的主题是可行的。如果大家都知道我们是谁,在每张幻灯片上显示我们的大名显得有点虚荣 (vanity)。
- 首先选择一个布局 (layout) 和我们的演讲 (talk) 相称的演示稿的主题。
- 接下来可以通过安装不同的颜色主题 (color theme) 来改变色彩 (colors)。这样会彻底地改变演示稿的外观。象 Berkeley (伯克利) 这样的“彩色”主题看起来没有 seahorse (海马) 主题和 lily (百合) 主题浮华 (flashy)。
- 同样可以通过安装不同的字体主题 (font theme) 来改变字体 (fonts)。

5.5 选择恰当的颜色

- 偶尔使用一下颜色。精制的主题丰富多彩 (蓝色 = 结构、红色 = 提示、绿色 = 举例)。在有很好的理由时,可以为一些内容如代码 (code)、数学文本 (math text) 等定制颜色。
- 在白背景上使用亮色 (bright colors) 须小心,特别是绿色。在我们的显示器上看来不错,而在演示 (presentation) 时看起来可能很糟糕,这归因于显示器 (monitors)、beamers、打印机重现颜色的方式不同。在亮色背景上用纯色 (pure colors) 时,应在纯色上添加更多的黑色。
- 对比鲜明。白底黑字,至少也应该是暗的内容放在亮的背景上。千万不能做“在不太亮的绿色背景上使用亮绿色文字”这样的蠢事。
- 渐变的背景 (Background shadings) 会降低文字的易读性而不能增加信息。不要仅仅因为“好看些”而用渐变背景。
- 反色效果 (Inverse video) (黑背景亮文本) 在一个比较亮的环境下放映时 (presentations) 可能会出问题,因为只有演示稿很小的一部分被 beamer 显示 (light up)。反色效果在打印输出和幻灯片输出时均很困难。

5.6 选择恰当的字体和字体属性¹⁵

文本 (Text) 和字体 (fonts) 随处可见。想想最近的一次在我们周围 10 米范围内没有文字是什么时候。很可能, 这在我们的生活中从未发生过! (只要我们穿衣甚至是泳衣, 就有许多的文字和我们接触。) 文字的历史和文明的历史一样长。目前 (these days), 有了好几万文字, 一部分产生于最初的几百年。

为演示稿选择正确的字体决不是微不足道的, 选择错了, 将会“很难看”, 更糟糕的是会让观众看不懂我们的幻灯片。这份用户手册 (user's guide) 在排印书籍 (book) 时, 可能效果不会很好, 但在当前章节, 我们会发现许多有益的提示 (hints), 它们可以帮助我们为 BEAMER 演示稿设置字体, 从而让演示稿更好看。一种字体有众多的属性如权重 (weight)、字族 (family)、尺寸 (size)。所有这些属性对字体在演示稿中的使用均有影响。下面将概述上述属性及不同选择的优缺点。

5.6.1 字体的尺寸

也许字体最明显的属性就是它的尺寸 (size), 字体通常用“点 (points)”来度量其大小。一点 (point) 有多大依我们所问的对象不同而不同。T_EX 规定 1 点 (point) = 1/72.27 英寸 (inch), 1 英寸 (inch) = 2.54 厘米 (cm)。另一方面, PostScript 和 Adobe 规定 1 点 (point) = 1/72 英寸 (inch) (T_EX 称这之为一个大点)。美国和欧洲的点 (points) 有所不同。1 点 (point) 的大小一旦确定下来, 字体的大小为“11pt”就意味着文字中字母的“高度”是 11pt。然而, 这个“高度 (height)”起源于 (stem from) 字母铸成铅字 (cast in lead) 的时候, 它参照了铅字母 (lead letters) 的垂直高度。因此, 该高度无需与字母 x 或 M 的实际高度相关联。来自 Adobe 的与来自 UTC 的字体为罗马 (Times) 字体大小为 11pt 的字母 x 的高度是不同的, 与来自 Adobe 的字体为瑞士 (Helvetica) 字体大小为 11pt 的字母 x 的高度也是不同的。

总之, 字体尺寸和字母的实际尺寸无关。目前, 约定“正常阅读”的字体尺寸是 10pt 或 11pt。字体被设计成这样的尺寸是为了便于阅读。

在演不稿中, 标准的 (classical) 字体尺寸明显没有意义。没人能看清投影中实际尺寸为 11pt 的文字。投影中的文字高度必需是几厘米高。这样, 通过普通的方法指定演示稿中的“字体尺寸”实际上没有意义 (make sense)。如果幻灯片充满了一行行的文本, 我们必需考虑一张幻灯片的文本的行数。据观众从投影中分散注意力的程度和投影的大小, 在每张幻灯片中放置 10 至 20 行文本是可行的。行数越少, 可读性越好。

在 BEAMER 中, 默认的字体尺寸某种程度是很难“非常”适合幻灯片的。而且, 必需确保在糟糕的条件下如在大的场所 (large room) 和小的投射区 (projection area) 我们的幻灯片仍然是可读的。然而, 我们也许希望放大或缩小一点字体使之更适合演讲环境。

一旦普通文本 (the normal text) 的尺寸确定下来了, 基于这个尺寸的所有其它尺寸也就确定下来了。正是这个原因, L^AT_EX 有象 `\large` 或 `\small` 这样的命令。这些命令规定的实际尺寸是以普通文本的尺寸为基准。

在演示稿中, 也许我们想让顶部导航区 (headlines)、底部导航区 (footlines) 或侧栏 (sidebars) 中的文本使用很小字体, 因为这些地方的文本不很重要 (vital), 观众只在空闲的时候才会阅读它们。当然, 它们也必需足够大才能看得清楚而无需使用望远镜。在正常的演示稿放映环境中, 观众应能看清 `\tiny` 命令指定的文本。

然而, 使用小字体 (small fonts) 时要小心。很多 PostScript 字体在使用小尺寸 (small sizes) 时会按比例缩小。当字体以小于其普通尺寸被使用时, 字符 (characters) 会被比缩放产生的笔触略粗的“笔触”描边。正

¹⁵L^AT_EX 的每种字体有 5 种属性: 编码 (encoding)、字族 (family)、序列 (series)、形状 (shape)、尺寸 (size)。编码就是将字符按照某种规定的方式编制成相应的代码信息, 如 OT1、T1、GB 编码等。字族就是某一类型字体集合的名称, 如 Adobe Times 字族、仿宋体字族等; 字族就是通常所说的字体。序列是指字体笔画的粗细 (或称为权重) 和宽窄程度。形状是指字体的外有形态如倾斜、直立等。尺寸是指字体大小, 以点数表示, 如 11pt、20pt 等。

是这个原因，高质量的多重母版字体（multiple master fonts, MM 字体）¹⁶ 或计算机现代字体（the Computer Modern fonts）使用不同的字体用于小字符（small characters）和普通字符（normal characters）。然而，在使用普通的瑞士（Helvetica）或罗马（Times）字体时，字符会被按比例缩小（scaled down）。在暗的背景上使用亮的字体时会产生类似的问题。当以高分辨率（resolution）打印在纸上时，暗背景亮文本（light-on-dark text）会被暗背景“溢出（overflowed）”，而演示稿中的暗背景亮文本这种效果更明显，它使文本不可读。

可以加粗小文本（small text），从而对抗（counter）这两种负面效果。

另一方向，在标题（titles）中使用大文本（larger text），然而，使用大字体并不能产生预期的效果。原因是，帧标题（Frame Title）以大字母（large letters）打印，并不意味着它首先被阅读。看看我们喜欢的杂志封面就清楚这一点，很可能，杂志的名称用最大字体（largest font）排版，但我们从不会首先注意到封面上的主题（topics）。同样（Likewise），对于目录，我们首先注意到的是目录的条目（entries），而不是“目录”这个单词。很可能，我们连拼写错误（spelling mistake）都没有发现（spot）（我的一位朋友在他的论文封面上拼写错了他自己的名字，直到一年后才发现）。其实，一页顶部的大文本（larger text）表明“不重要，因为我知道我所期望的是什么”。这样看来，帧标题不使用大（large）字体，而使用加粗或斜体（bold or italics）的常规尺寸（normal size）字体排版。

5.6.2 字族

字体的另一重要属性（central property）是它的族（family）。字族的例子是 Times 或 Helvetica 或 Futura。正如名字所示，许多不同的字体可能属于同一族，例如，Times 可以有不同尺寸（sizes），可以有粗体版本（bold version），可以有斜体版本（italics version）等等。令人迷惑的是，像 Times 这样的字族常叫做“字体 Times（font Times）”。

字族有两大类：衬线字体（serif fonts）和无衬线字体（sans-serif fonts）。无衬线字体是指字体的字母没有（来源于法语 sans，含义是“没有”）衬线的字体。衬线是指组成字母的笔画（strokes）末端带有的小钩（hooks）。我们现在阅读的字体就是衬线字体。对比一下，这个句子的文本所用的字体就是无衬线的（By comparison, this text is in a sans-serif font）。演示稿中的无衬线字体易于（大家都这样认为）阅读。在低分辨率中的渲染中，衬线会降低字体的清晰度。然而，在高分辨率的投影仪中衬线文本和无衬线文本一样易读。以衬线字体排版的演示稿给人以保守的（conservative）印象，这可能是我们想要的。

在我们的系统中很大可能已预装（preinstalled）了大量不同的字族（如上所述常称为字体）。T_EX（和 BEAMER）默认使用的字体是计算机现代字体（Computer Modern font）。该字族（字体）是唐纳德·克努特（Donald Knuth）为 T_EX 程序设计的原始字族（original font family）。它是一个成熟的（mature）字族，它可用于我们所希望的所有东西：扩展的数学希腊字母表（extensive mathematical alphabets）、轮廓 PostScript 版本（outline PostScript versions）、真正的小型大写字母（real small caps）、真正的旧体数字（real oldstyle numbers）、特殊设计的小写和大写字母等等。

然而，以下的原因使我们使用计算机现代字体（字族）之外的字族：

- 看久了计算机现代字体会感到厌烦，如果使用其它字族（不是 Times !）则可以给我们一个清新的外观。
- 其它字族，特别是 Times 和 Helvetica，有时可以渲染（rendered）得更好，因为它们看起来具有更好的内部小字还原技术（internal hinting）。

¹⁶Multiple Master fonts: 即多重母版字体，简称 MM 字体，是 Type 1 字体的延伸，特点是一个字体内包含两个或多个字体设计，有一个或多个变化轴心，可模仿多个字款，当缺字款时便可用 MM 字体代替之，但始终不是相同字体，字形会有区别，而且经常出现输出问题。不是所有软件或系统支持 MM 字体。

- 计算机现代字体的无衬线版本设计得不如衬线版本好。其实，无衬线版本就是带不同设计参数的衬线版本，而不是一个独立的设计
- 计算机现代字体比像 Times 这样俭省的 (economic) 字体 (这解释了为什么 Times 受那么多人的喜欢，这些人需要使完美的论文只占 12 页) 占用更多的空间。Times 就是为了俭省 (新闻报纸公司印刷 Times 需要的是稳健的而非空间俭省的字体) 而特别设计的。

下面是一个计算机现代字体的供选方案：

- Latin Modern 是计算机现代字体的衍生物 (derivate)，它提供了更多的字符，但不认为它是真正的替代物 (alternative)。虽然它的被推荐程度超过了计算机现代字体。
- Helvetica 是常用的替代物。然而，看久了 Helvetica 字体也会感到厌烦 (因为它也是到处使用的字体)。Helvetica 字体有很大的 x 高度 (指字母 x 的高度，和字母 M 的高度类似)。通常认为，大的 x 高度对于像英语这样的很少使用大写字母的语言来说是有益的，但对于像德语这样的常用大写字母的语言来说不是很有益。(然而，对于这一点我们没有足够的信心。) 注意：Helvetica 的 x 高度和 Times 的 x 高度有很大的不同，如果在一行中混合使用两者则看起来会有点怪。然而，用于加载 Times 和 Helvetica 的宏包提供了混合两者的选项。
- Futura 是一款漂亮的字体，非常适合演示稿。它厚重的字母即使在缩放 (scaling)、翻转 (inversion)、低对比 (low contrast) 时都显得很健壮 (robust)。不幸的是，虽然在我们的系统中很可能已安装了该字体，但是要在 $\text{T}_\text{E}\text{X}$ 中调用该字体却是一个复杂的过程。然而，如使用现代的排版引擎如 `luatex` 和 `xetex` 则上述字体调用过程则变得很简单。
- Times 也是计算机现代字体的一个可能的替代物。其主要的缺点 (disadvantage) 就是它是一个衬线字体，它需要一台高分辨率的投影机。该字体很常用，而且很好用。
- Bitstream Vera 的衍生物 DejaVu，也是一款很好且免费的计算机现代字体的替代字体。它的 TrueType 版本来自 OpenOffice.org，在 $\text{T}_\text{E}\text{X}$ 中使用该 TrueType 字体是一个很复杂的过程，幸好名为 `arev` 的 $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ 宏包提供了使用该字体的 Type 1 版本 (名为 Bera) 的简便途径。它有无衬线与衬线版本，`arev` 宏包两者均提供。

普通文本不宜使用的字族包括：

- 所有的等宽字体 (monospaced fonts) (如 Courier)。
- 手写字体 (Script fonts) (有点像手迹)。它们的描边 (stroke) 宽度正适合演示稿。
- 更雅致的 (delicate) 衬线字体如 Stempel 和 Garamond (Garamond 是真正的漂亮字体)。
- 哥特 (Gothic) 字体。只有一小部分观众能流畅地阅读它。

有一款流行的 (popular) 字体即微软的 Comic Sans，它有点特别。一方面，有的网站游说禁止使用该字体。事实上，该字体的主要麻烦是它不便于阅读，使用该字体时部分的数学排版将很难看。另一方面，使用该字体的幻灯片给人一种“用手写”的感觉，看起来很很自然。使用该字体前请三思，但不要害怕。

最重要的印刷 (typography) 原则之一是在文本中使用尽可能少的字族。但是，印刷很少规定在一个页面中使用的字族不应超过两个。然而当排版数学文本 (mathematical text) 时，常常需要使用不同的字族。例如，使用哥特 (Gothic) 字母表示 (denote) 向量 (vectors) 在过去是个惯例。而且，常用等宽 (monospace) 字体

排版程序文本 (program texts)。如果我们的观众习惯于用特定的字族排版的特定的文本，请使用该字族，而不要理会印刷上是怎么说的。

印刷上的惯例是标题使用无衬线字体，而普通文本（看看我们喜爱的杂志）使用衬线字体。我们也可以使用两种不同的无衬线字体或两种不同的衬线字体，但必须确保字体看起来“足够地不同”。如果它们略有不同，则页面看起来会“有点怪”，但观众却不知道原因。例如，不要混用 Arial 和 Helvetica（它们是完全相同的），也不要混用计算机现代字体和 Baskerville（它们很相似）。Gills Sans 和 Helvetica 的联用是危险的却是可能的。Futura 和 Optima 的联用当然是可以的，至少就字体而言它们不相同。

5.6.3 字体形状：斜体和小型大写

L^AT_EX 介绍了字体形状 (*shape*) 的概念。真正重要的是斜体 (*italic*) 和小型大写 (*small caps*)。斜体字体就是字体略向右侧倾斜, *like this*。有关斜体的一些情况如下：

- 斜体常用在小说 (novels) 中以表示重点。然而，特别是对于无衬线字体，斜体还“不够强壮”以致于会丢失演示稿中的重点。在演示稿中表示重点，使用不同的颜色或粗体文本则更合适。
- 如果仔细看，我们会发现斜体文本不光是倾斜的，而且还使用了不同的字母（请比较 a 和 *a*）。然而，对于衬线文本是这样，对于无衬线文本却不是这样。只是使用了倾斜 (slanted) 的而没有使用不同字母的文本叫做“倾斜 (slanted)”的文本，而不是“斜体 (italic)”文本。有时，单词“倾斜的 (oblique)”也只用于 slanted (倾斜)，但有时也用于 italics (斜体)，因此最好别用 oblique (倾斜的)。使用倾斜的 (slanted) 衬线文本是非常招排版者讨厌的，并被认为是“低级的计算机排版”。然而，大家包括唐纳德·克努特 (Donald Knuth) 都会在他们的书籍中使用倾斜的文本。

在演示稿中，如果我们要不辞辛劳地使用衬线字体，那么请使用斜体 (italic) 文本，而不要使用倾斜的 (slanted) 文本。

- 用于衬线斜体的字母比普通的衬线文本的字母没有改变多少。用于衬线斜体的字母是基于手写字母 (handwritten letters) 的，正因如此，衬线斜体给人以手写体的感觉，手写体很能让演示稿更具有“人情味 (personal touch)”（使用 Times 斜体没有这种功效，因为我们已看过它千百遍了）。然而，它比普通文本更难阅读，因此，普通文本使用衬线斜体时不要超过一行。

T_EX 支持的第二种字体形状是小型大写 (small caps)。使用小型大写字母给人以保守甚至刻板的印象，下面是一些告诫的话：

- 小型大写 (small capitals) 不同于全部大写 (all-uppercase)。小型大写的文本不会改变普通的大写字母，小型大写使用大写字母的更小版本排版小写字母。因此，单词“German”用小型大写字母排版成 GERMAN，而用全部大写字母则排版成 GERMAN。
- 小型大写看起来是“假装的 (faked)”或“真正的”小型大写。假装的小型大写由缩小普通的大写字母产生，这将导致字母看起来太细 (thin)。真正的小型大写由大写字母的小版本设计而成，这些大写字母的轮廓宽度和普通文相同。
- 计算机现代字体和 PostScript 字体的专业版本带有 (come with) 真正的小型大写（虽然计算机现代字体的小型大写由于难以理解的原因大了一点，但观众不会注意到这一点）。“普通的 (Simple)” PostScript 字体如独特的 (out-of-the-box) Helvetica 或 Times 只能带有假装的小型大写。
- 以小型大写排版的文本比普通文本更难阅读。原因是我们是通过单词的“形状 (shape)”来阅读的，例如，单词的形状是通过看 (seing) 一个普通的字母、一个上升的字母、一个普通的字母、一个下降的字母

母、一个普通的字母来识别的。人们发现像“shepe”这样的拼写错误比“spape”更难。小型大写破坏了单词的形状，因为 SHAPE、SHEPE、SPAPE 均有相同的形状，因此，很难将它们区分开来。观众阅读小型大写的文本要比普通文本慢。顺便提一下，这就是法律免责声明（legal disclaimers）常用大写字母（uppercase letters）书写的原因：不是让它们显示得更重要，而是要让它们更难阅读。

5.6.4 字体的粗细

字体的“权重（weight）”指的是字母的浓度（thickness）。通常，字体是以常规（regular）或粗体（bold）显示的。还有半粗体（semibold）、极粗体（ultrabold）（或黑体）、细（thin）、极细（ultrathin）等版本。

在印刷中，用粗体字体表示重点是招人讨厌的，特别是在普通文本中（普通文本中的粗体单词被认为是“垃圾”）。对于演示稿，这个不使用粗体文本的规则并不真正适应。在演示稿的幻灯片中，常常只有少量的文本，而有大量的用于吸引观众注意力的元素。使用传统的斜体字表示重点常被看漏（overlooked）。因此，在演示稿中使用粗体文本看来是个好的选择。然而，一个更好的选择是使用亮色（bright color）如红色来吸引注意力。

如前所述，我们可以为小文本（small text）使用粗体文本，除非使用的字体是像 Futura 或 DejaVu 这样的特别健壮（robust）的字体。

6 解答模板

在 `beamer/solutions` 目录的子目录中有不同语言的解答模板 (*Solution Template*)。解答模板是 $\text{T}_\text{E}\text{X}$ -文本, 它用于“解答”某一特定的问题。这些问题如: “我要为会议准备一个 20 分钟的演讲稿”、“我要制作一幻灯片用于介绍下一位演讲人”、“我要制作一张分段显示的表格”等等。为解决这一类的问题而设计的解答模板由模板和例子混合而成, 其中的例子可以解决特定的问题。只需拷贝解答模板文件 (或其中一部分) 并按需自由调整即可解决我们的问题。

BEAMER 解答模板的收集工作才刚刚开始, 目前的解答模板数量很少。我们希望将来有更多的解答模板, 当然也鼓励 BEAMER 文档类的用户将他们开发的解答模板奉献给我们。鼓励用户们在将解答模板翻译成除英语和德语以外的语言的工作中多帮忙。如果您创作或翻译了一个解答模板, 请尽管将它奉献给我们 (务必包含它的使用说明和对现有模板的修改)

下面的解答模板列表据演讲所耗时间分类。一如既往, 解答模板存放于 `beamer/solutions` 目录中。

解答模板 `short-talks/speaker_introduction-ornate-2min`

- 介绍另一位演讲者。
- 演讲耗时 2 分钟
- 风格华丽。

PRESENTATION $\text{T}_\text{E}\text{X}$ -版本有 `de`、`en`、`fr` 语版。

L^AT_EX

LYX $\text{L}_\text{Y}\text{X}$ -版本有 `de` 和 `en` 语版。

解答模板 `generic-talks/generic-ornate-15min-45min`

- 普通的解答模板, 适应于所有演讲主题。
- 演讲耗时 15 分钟 ~ 45 分钟。
- 风格华丽。

PRESENTATION $\text{T}_\text{E}\text{X}$ -版本有 `de`、`en`、`fr` 语版。

L^AT_EX

LYX $\text{L}_\text{Y}\text{X}$ -版本有 `de` 和 `en` 语版。

解答模板 `conference-talks/conference-ornate-20min`

- 适应于会议/讨论会 (`conference/colloquium`)。
- 演讲耗时 20 分钟。
- 风格华丽。

PRESENTATION $\text{T}_\text{E}\text{X}$ -版本有 `de`、`en`、`fr` 语版。

L^AT_EX

LYX $\text{L}_\text{Y}\text{X}$ -版本有 `de` 和 `en` 语版。

7 许可证和版权

7.1 必需遵守哪些许可证？

分发 (distribute) BEAMER 宏包不同的部分遵守不同的许可证 (licenses):

1. 宏包的代码 (*code*) 是双重许可 (dual-license) 的。这意味着当使用 BEAMER 宏包时您可以选择其中一个许可证。这两个许可证是：
 - (a) 自由软件基金会 (Free Software Foundation, FSF) 颁布的 GNU 通用公共许可证 (GNU General Public License, GNU GPL) V2 或更新的版本。
 - (b) L^AT_EX 项目公共许可证 (L^AT_EX Project Public License, LPPL) V1.3c 或更新的版本。
2. 宏包的文档 (*documentation*) 也是双重许可的, 同样, 您可以选择其中一个许可证。这两个许可证是：
 - (a) 自由软件基金会颁布的 GNU 自由文档许可证 (GNU Free Documentation License) V1.3 或更新的版本。
 - (b) L^AT_EX 项目公共许可证 (L^AT_EX Project Public License, LPPL) V1.3c 或更新的版本。

“宏包的文档 (documentation of the package)”指的是 BEAMER 宏包 `doc` 子目录中的所有文件。在 `doc/licenses/manifest-documentation.txt` 这个文件中可以发现所有这些文件的清单。其它目录中的所有文件是“宏包的代码 (code of the package)”的一部分。在 `doc/licenses/manifest-code.txt` 这个文件中可以发现所有这些文件的清单。

这一节的剩余部分提供这些许可证。下面是这些许可证的具体内容, 在 `doc/licenses` 目录中有这些许可证的纯文本 (plain text) 文件的版本。

7.2 GNU 通用公共许可证, V2

7.2.1 前言

大多数软件许可证 (licenses) 的用意在于剥夺您共享和修改软件的自由。相反的, GNU 通用公共许可证 (GNU General Public License, GNU GPL) 力图保证您共享和修改自由软件的自由 — 保证自由软件对所有使用者都是自由的。GNU GPL 适用于大多数自由软件基金会 (Free Software Foundation, FSF) 的软件, 以及任何经作者授权使用的其他软件。[有些自由软件基金会软件受 GNU 函数库通用许可证 (GNU Library General Public License, GNU LGPL) 的保护]。您也可以将它用到您的程序中。

当我们谈到自由软件 (free software) 时, 我们谈的是自由 (freedom) 而不是价格 (price)。我们把 GNU 通用公共许可证设计成您的保障, 确保您拥有发布自由软件的自由 (您可以自由决定是否要对此项服务收费); 确保您能收到程序源码或者在您需要时能得到它; 确保您能修改软件或将它的一部分用于新的自由软件 (free programs); 而且还确保您知道您拥有这些权利。

为了保护您的权利, 我们需要作出规定: 禁止任何人剥夺您的权利, 或者要求您放弃这些权利。如果您修改了自由软件或者发布了软件的副本, 这些规定就转化为您的责任。

例如, 如果您发布这样一个程序的副本, 不管是免费的还是收费的, 您必须将您具有的一切权利给予您的接受者; 您必须确认他们能收到或得到源码; 并且必须向他们展示这些条款的内容, 使他们知道他们有这样的权利。

我们采取两项措施来保护您的权利: (1) 用版权来保护软件; 以及 (2) 提供您许可证, 赋予您复制、发布和/或修改这些软件的法律许可 (legal permission)。

同样，为了保护每个作者和我们自己，我们需要清楚地让每个人明白，自由软件没有担保（no warranty）。如果由于某人修改了软件，并继续加以传播，我们需要它的接受者明白：他们所得到的并不是原来的自由软件。由其他人引入的任何问题，不应损害原作者的声誉（reputations）。

最后，由于任何自由软件不断受到软件专利的威胁，故我们希望避免这样的风险，即如果自由软件的再发布者以个人名义获得专利许可证，也就等同将软件变为私有。为防止这一点，我们必须明确声明：任何专利必须以允许每个人自由使用为前提，否则就不应授予专利。

下面是有关复制、发布和修改的确切的条款（terms）和条件（conditions）。

7.2.2 复制、分发与修改的条款与条件

0. 凡是版权所有者在其程序和作品中声明其程序和作品可以在 GNU GPL 条款的约束下发布，这样的程序或作品都受到本许可证约束。下面提到的“程序（Program）”指的是任何这样的程序或作品。而“程序的衍生作品（work based on the Program）”指的是这样的程序或者版权法认定下的衍生作品，也就是说包含此程序或程序的一部分的套件，可以是原封不动的，或经过修改的，和/或翻译成其他语言的（程序）。[在下文中，“修改（modification）”一词的涵义一律包含翻译作品。] 每个许可证（license）接受人用“您（you）”来称呼。

本许可证条款不适用于复制、发布和修改以外的行为。这些行为超出这些条款的范围。执行本程序（running the Program）的行为不受条款的限制。而程序的输出（the output from the Program）只有在其内容构成本程序的衍生作品（并非只是因为该输出由本程序所产生）时，这一条款才适用。至于程序的输出内容是否构成本程序的衍生作品，则取决于程序具体的用途。

1. 只要您在每一程序副本上明显和恰当地宣告版权声明和无担保（warranty）的声明，并原封不动保持此许可证的声明和无担保的声明，并将此许可证连同程序一起给其他每位程序接受者，您就可以用任何媒体复制和发布您收到的程序的源码。

您可以据转让副本的实际行动收取一定费用。您也可以自由决定是否以提供担保来换取一定的费用。

2. 您可以修改程序的一个或几个副本或程序的任何部分，以此形成基于这些程序的衍生作品。只要您同时满足下面的所有条件，您就可以按前面第一款的要求复制和发布这一经过修改的程序或作品：
 - (a) 您必须在修改过的文件上附加明显的说明：叙明您修改过这些文件，以及修改的日期。
 - (b) 您必须让您发布或出版的作品，包括本程序的全部或一部分，或内含本程序的全部或部分所衍生的作品，允许第三方在此许可证条款下使用，并且不得因为此项授权行为而收费。
 - (c) 如果修改的程序在执行时以互动（interactively）方式读取命令，您必须使它在开始进入最常使用的方式时列印（print）或显示（display）这样的声明：适当的版权声明和无担保的声明（或者您提供担保的声明）；使用者可以按此许可证条款重新发布程序的声明，并告诉使用者如何看到这一许可证的副本。（例外的情况：如果原始程序以互动方式工作，但它通常并不列印这样的声明，那么您基于此程序的作品也就不需要列印声明）。

这些要求适用于整个修改过的作品。如果能够确定作品的一部分并非是本程序的衍生产品，且可以合理地单独考虑并将它与原作品分开的话，则当您将它作为独立的作品发布时，它不受此许可证和其条款的约束。但是当您将这部分与基于本程序的作品一同发布时，则整个套件将受到本许可证条款约束，因为本许可证对于其他许可证持有人的授权扩大到整个产品，也就是套件的每个部分，不管它是谁写的。

因此，本条款的意图不在于剥夺您拥有对完全由您自身完成作品的权利，而在于履行权利来控制基于本程式的集体作品或衍生作品的发布。

此外，将与本程序无关的作品和本程序（或本程序的衍生作品）一起放在贮存媒体或发布媒体的同一卷上，并不使其他作品置于此许可证的约束范围之内。

3. 只要您遵守前面的第 1、2 款，并同时满足下列三条中的任一条，您就可以以目标码或可执行形式复制或发布程序（或符合第 2 款，本程序的衍生作品）：
 - (a) 在通常用作软件交换（software interchange）的媒体上，和目标码一起附上机器可读的（machine-readable）完整的本程序源码。这些源码的发布应符合上面第 1、2 款的要求。或者，
 - (b) 在通常用作软件交换的媒体上，和目标码一起，附上书面报价，提供替第三方复制源码的服务。该书面报价有效期不得少于 3 年，费用不得超过完成原程序发布的实际成本，源码的发布应符合上面的第 1、2 款的要求。或者：
 - (c) 和目标码一起，附上您收到的发布源码的报价信息。（这一条款只适用于非商业性发布，而且您只收到程序的目标码或可执行码，和按 b 款要求提供的报价。）

作品的源码（source code）指的是对作品进行修改最优先择取的形式。对可执行的作品而言，完整的源码套件包括：所有模组（modules）的所有原始程序，加上有关的介面的定义，加上控制可执行作品的安装和编译的脚本（script）。至于那些通常伴随着执行本程序所需的作业系统元件（如编译器、核心等）而发布的软件（不论是源码或可执行码），则不在本许可证要求以程序源码形式伴随发布之列，除非它是本程序的一部分。

如果可执行码或目标码是以指定复制地点的方式来发布，那么在同一地点提供等价的源码复制服务也可以算作源码的发布，然而第三方并不需因此而负有必与目标码一起复制源码的义务。

4. 除了本许可证明白声明的方式之外，您不能复制、修改、转发许可证和发布程序。任何试图用其他方式复制、修改、转发许可证和发布程序是无效的，而且将自动结束许可证赋予您的权利。然而，对那些从您那里按许可证条款得到副本和权利的人们，只要他们继续全面履行条款，许可证赋予他们的权利仍然有效。
5. 您没有在许可证上签字，因而您没有必要一定接受此许可证。然而，没有任何其他东西赋予您修改和发布程序及其衍生作品的权利。如果您不接受许可证，这些行为是法律禁止的。因此，如果您修改或发布程序（或本程序的衍生作品），您就表明您接受这一许可证以及它的所有有关复制、发布和修改程序或基于程序的作品的条款和条件。
6. 每当您重新发布程序（或任何程序的衍生作品）时，接受者自动从原始许可证颁发者那里接到受这些条款和条件支配的复制、发布或修改本程序的许可。您不可以增加任何条款来进一步限制本许可证赋予他们的权利。您也没有强求第三方履行许可证条款的义务。
7. 如果因法院判决或因违反专利的指控或任何其他原因（不限于专利问题），使得强加于您的条件（不管是法院判决、协议或其他）和许可证的条件有冲突时，他们也不能令您背离许可证的条款。在您不能同时满足本许可证规定的义务及其他相关的义务来发布程序时，结果是您只能根本不发程序。例如，如果某一专利许可证不允许所有直接或间接从您那里接受副本的人们，在不付专利费的情况下重新发布程序，唯一能同时满足两方面要求的办法是停止发布程序。

如果本条款的任一部分在特定的环境下无效或无法实施，本条其余部分仍应适用，并将这部分条款作为整体用于其他环境。

本条款的目的不在于引诱您侵犯专利或其他财产权的主张，或争论这种主张的有效性。本条款的主要目的在于保护自由软件发布系统的完整性。它是通过公共许可证的应用来实现的。许多人已依赖同是出自此系

统的应用程序，经由此系统发布大量自由软体而做出慷慨的奉献。作者/捐献者有权决定他/她是否通过任何其他系统发布软体，许可证接受者不能强迫作者/捐献者做某种特定的选择。

我们相信许可证其他部分已涵盖本节所述状况，本节目的只在更明确说明许可证其余部分可能产生的结果

8. 如果由于专利或者由于有版权的介面问题使程序在某些国家的发布和使用受到限制，则以本许可证发布程序的原始作者可以增加发布地区的限制条款，将这些国家明确排除在外，并在这些国家以外的地区发布程序。在这种情况下，这些限制条款如同写入本许可证一样，成为许可证的条款。
9. 自由软体基金会可能随时出版通用公共许可证的修改版和/或新版。新版和当前的版本在精神上保持一致，但在细节上可能有所不同，以便处理新的问题与状况。

每一版本都有不同的版本号（version number）。如果程序指定可适用的许可证版本号以及“任何更新的版本”，您有权选择遵循指定的版本或自由软体基金会以后出版的新版本。如果程序未指定许可证版本，您可选择自由软体基金会已经出版的任何版本。

10. 如果您愿意将程序的一部分结合到其他自由程序中，而它们的发布条件不同，请写信给作者，要求准予使用。如果是自由软件基金会加以版权保护的软件，请写信给自由软件基金会，我们有时会作为例外的情况处理。我们的决定受两个主要目标的指导，这两个主要目标是：我们的自由软件的衍生作品继续保持自由状态，以及从整体上促进软件的共享和重复利用。

7.2.3 无担保声明

10. 由于准予免费使用本程序，因此在法律许可范围内，对本程序并不负担担保责任。除非另有书面说明，版权所有者和/或其他提供程序的人们“一样”不提供任何类型的担保，不论是明确的，还是隐含的，包括但不限于可销售和适合特定用途的隐含保证。全部的风险，如程序的质量和性能问题都由您来承担。如果程序出现缺陷，您应当承担所有必要的服务、修复和改正的费用。
11. 非经法律要求或书面同意，在任何情况下，任何版权所有者或任何按许可证条款修改和/或发布程序的人们都不对您的损失负有任何责任。包括由于使用或不能使用程序引起的任何一般的、特殊的、偶然发生的或重大的损失（包括但不限于数据的损失，或者数据变得不精确，或者您或第三方的持续的损失，或者程序不能和其他程序相兼容）。即使版权所有者和其他人已被告知这种损失的可能性也不例外。

7.3 Gnu 自由文档许可证, V1.3, 2008 年 11 月 3 日

Copyright ©2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

允许每个人复制和发布本许可证文件的完整副本，但不允许对它进行任何修改。

7.3.1 前言

本许可证着重于保证手册（manual）、教程（textbook）或其他功能的有用的文档（document）的“自由（free）”：以确保任何人不管在商业领域还是非商业领域都可以复制和（修改或没有修改并）重新发布这些文档的自由。其次，本许可证保护文档作者和发布者由于他们的工作获得的信誉不会因他人对其文档的修改而受损。

本许可证是一种“著佐权（Copyleft）”的，其含义为：文档的衍生作品必须与原文档同样是自由的。本文档是 GNU 通用公共许可证（为自由软件设计的一种 copyleft 协议）的补充。

我们设计的该许可证供自由软件的手册使用，因为自由软件需要自由文档：一个自由的程序附带的手册当然要享有与该软件同等的自由。不过本许可证并不仅限于供程序的手册使用，所有的文档作品都可以使用本协议。我们鼓励那些指导性性质或参考书性质的文档作品使用本协议。

7.3.2 适应范围和约定

本许可证适用于任何媒体上的任何手册或其它文档，文档版权所有人只需在文档中声明使用本许可证的条款即可发布。该声明允许了在后文中的规定下，即可在任何时间、地点、无版权地使用该作品。以下的“**文档 (Document)**”都是指此类（使用 GFDL 发布的）手册（manual）或作品（work）。公众的任何一员都是本许可证的许可对象，这里将用“您（you）”来称呼。如果您复制、修改或发布了这些（某种意义上也可以说是在版权法保护下的）文档，就表明您已经接受了本许可证。

文档的“**修订版 (Modified Version)**”是指任何包含全部或部分文档的作品，无论是原始拷贝还是经过加工修改和/或是翻译成了其它的语言。

“**附属章节 (Secondary Section)**”是取名为附录（appendix）或前言（front-matter）的文档部分，专门描述有关文档主题的文档出版者或作者的相关信息（或对一些相关情况进行说明）并且包含文档主题不会直接提及的内容。（如果文档是数学教材的一部分，附属章节或许连一点数学都不会提到。）这些相关信息可以是与主题有关的历史关联、相关问题或者相关的法律、商业、哲理、道德或政治立场。

“**固定章节 (Invariant Sections)**”是某些指定了标题的附属章节，固定章节做为文档的一部分，也象声明提到的那样，在本许可证的保护下发布。如果某章节不符合上面关于附属章节的定义，就不能称之为固定章节。文档可以没有固定章节。如果看不出文档中有什么固定章节，那就是没有了。

“**封皮文本 (Cover Texts)**”是那些简短的小段文字列，就象封面或封底的文字。封皮文本做为文档的一部分，也象声明提到的那样，在本许可证的保护下发布。封面文本（Front-Cover Text）一般最多 5 个词语，封底文本（Back-Cover Text）一般最多 25 个词语。

文档的“**透明 (Transparent, 此处理解为兼容、开放)**”副本是指可以用计算机处理的副本，表现为其格式符合一般通用规范，这样就适于直接用通用文本编辑器（text editors）或（对由像素构成的图片使用的）通用绘画程序（paint programs）或那些被广泛应用的（用来绘图的）图像编辑器（drawing editor）修改文档，并且其格式适应于输入文本格式器或可转换为适应于输入文本格式器的兼容格式。如果某副本（不论有没有标记）为反对或防止读者的后续修订而采用其它格式，那么就是不兼容（not Transparent）的。如果图片格式的使用受限于任何实质性专利条文，那么该图片格式就是不兼容（封闭）的。文档副本的不“Transparent”，我们称为“Opaque（不透明，此处理解为不兼容、封闭）”。

文档副本可用的兼容格式包括没有标记的纯 ASCII 码、Texinfo 格式、LaTeX 格式、使用公用 DTD 的 SGML 格式或 XML 格式，还有符合标准的简单 HTML 格式、PostScript 或 PDF 这些便于编辑的图像格式。图片适用的兼容格式包括 PNG、XCF 和 JPG。不兼容格式包括用专有文档处理器（proprietary word processors）才能阅读编辑的专有格式、使用非公认的文件格式定义和/或使用非公认的处理工具处理的 SGML 格式或 XML 格式，还有那些机器生成的 HTML 及字处理器生成的 PostScript 或 PDF 等格式。

“**扉页或标题页 (Title Page)**”是指刊印成册的书（book）的扉页本身及附加下列有必要明显地保留的页面：本许可证要求在扉页出现的材料。象那些版式中没有扉页的作品，“扉页”是指在作品正文之前最显著的标题附近的文字。

“**出版商 (publisher)**”是指将文档的拷贝发布给公众的任何个人或单位。

章节“**特殊标题 XYZ (Entitled XYZ)**”表示文档的一个特定的子单元，其标题就是 XYZ 或包含 XYZ，其后面插入的文本将 XYZ 翻译为其他语言。[这里 XYZ 代表下面提及的特定章节的名字，比如“致谢 (Acknowledgements)”、“献给 (Dedications)”、“签名 (Endorsements)”、“历史 (History)”] 对这些章节

“保护标题 (Preserve the Title)”就是依据这个定义保持这样一个“Entitled XYZ”章节。

文档可能会在文档遵照本许可证的声明后面包含免责声明 (Warranty Disclaimer)。这些免责声明被认为是包含在本许可证中的，但这仅视为拒绝担保：免责声明中任何其它的暗示都是无用的并对本许可证的含义没有影响。

7.3.3 原样复制

您可以以任何媒质拷贝并分发文档，无论是否处于商业目的，只要保证本许可证、版权声明和宣称本许可证应用于文档的声明都在所有复制品中被完整地、无任何附加条件地给出即可。您不能使用任何技术手段阻碍或控制您制作或发布的拷贝的阅读或再次复制。不过您可以在复制品的交易中得到报酬。如果您发布足够多的拷贝，您必须遵循下面第三节中的条件。

您也可以在和上面相同的条件下出租拷贝和向公众放映拷贝。

7.3.4 大量复制

如果您发行文档的印刷版的拷贝（或是有印制封皮的其他媒质的拷贝）多于 100 份，而文档的许可证声明中要求封皮文本，您必须将它清晰明显地置于封皮之上，封面文本在封面上，封底文本在封底上。封面和封底上还必须标明您是这些拷贝的发行者。封面必须以同等显著、可见地完整展现标题的所有文字。您可以在标题上加入其他的材料。改动仅限于封皮的复制，只要保持文档的标题不变并满足这些条件，可以在其他方面被视为是原样复制 (verbatim copying)。

如果需要加上的文本对于封面或封底过多而无不清晰明了，您应该在封皮上列出前面（在合理的前提下尽量多）的文本，把其它的放在邻近的页面上。

如果您出版或分发了超过 100 份文档的不兼容（即不透明）的拷贝，您必须在每个不兼容拷贝中包含一份机器可读的兼容拷贝，或是在每个不兼容拷贝中给出一个计算机网络地址，通过这个地址，使用计算机网络的公众可以使用标准的网络协议没有任何附加条件的下载一个文档的完整的兼容拷贝。如果您选择后者，您必须在开始大量分发非兼容拷贝的时候采用相当谨慎的步骤，保证兼容拷贝在其所给出的位置在（直接或通过代理和零售商）分发最后一次该版本的非兼容拷贝的时间之后一年之内始终是有效的。

大量发布拷贝之前，请您（但不是必须）与文档的作者联系，以便可以得到文档的更新版本。

7.3.5 修改

在上述第 2、3 节的条件下，您可以复制与分发文档的修改后的版本 (Modified Version)，只要严格的按照本许可证发布修改后的文档，在文档的位置填入修改后的版本，也就是许可任何得到这个修改版的拷贝的人分发或修改这个修改后的版本。另外，在修改版中，您需要做到如下几点：

- A. 在标题页或在封面上使用，如果有与先前版本不同的文件，应该被列在文件的历史章节不同的标题。如果版本的原始出版者允许，您可以使用与某一个先前版本相同的标题。
- B. 在修改版本的标题页上列出担负作者权的一个或多个人或实体作为作者，并且列出至少五位文件的主要作者。如果少于五位，则列出全部的主要作者，除非他们免除了您这个要求。
- C. 在标题页陈述修改版本的出版者的名称作为出版者。
- D. 保存文件的所有版权声明。
- E. 为您的修改增加一个与其它版权声明相邻的适当的版权声明。

- F. 在版权声明后面，以授权附录所显示的形式，包括一个给予公众在本授权条款下使用修改版本的许可声明。
- G. 在那个许可声明中保存固定章节（Invariant Sections）和文件许可声明中必要封面文字的全部列表。
- H. 包括一个未被改变的本许可证的副本。
 - I. 保存标题为历史的“章节”和其标题，并且增加一项至少陈述如同在标题页中所给的修改版本的标题、年份、新作者和出版者。如果在文件中没有标题为历史的章节，则制作出一个陈述如同在它的标题页中所给的文件的标题、年份、新作者和出版者，然后增加一项描述修改版本如前面句子所陈述的情形。
 - J. 如果有的话，保存在文件中为了给公众存取文件的兼容拷贝，而给予的网络位址，以及同样地在文件中为了它所根据的先前版本，而给予的网络位址。这些可以放在“历史（History）”章节中。您可以省略一个在文件本身之前，已经至少出版了四年的作品的网络位址，或是如果它所参照的那个版本的原始出版者给予允许的情形下也可以省略它。
- K. 在任何标题为“感谢（Acknowledgements）”或“贡献（Dedications）”的章节，保存章节的标题，并且在那章节保存到那时候为止，每一个贡献者的感谢和/或贡献的所有声色。
- L. 保存文件的所有固定章节（Invariant Sections），于其文字以及标题皆不得变更。章节号码或其同等物并不被认为是章节标题的一部份。
- M. 删除任何标题为“背书（Endorsements）”的章节。这样子的章节不可以被包括在修改版本中。
- N. 不要重新命名任何现存的章节，而使其标题为“背书（Endorsements）”，或造成与任何固定章节（Invariant Sections）相冲突的标题。
- O. 保存任何的担保放弃。

如果修改版本加入了新的符合次要章节定义的引言（front-matter）或附录（appendices）章节，并且不含从原文档复制的内容（material），您可以按照您的意愿将它标记为不可变（invariant）。如果需要这样做，就把它们的标题加入修改版本的许可声明的不可变章节列表之中。这些标题必须和其他章节的标题相区分。

您可以加入一个命名为特殊标题“签名（Endorsements）”的章节，只要它只包含对您的修改版本由不同的各方给出的签名—例如书评（statements of peer review）或是声明文本已经被一个组织认定为一个标准的权威定义

您可以加入一个最多 5 个字的段落（passage）作为封面文本（Front-Cover Text）和一个最多 25 个字的段落作为封底文本（Back-Cover Text），把它们加入到修改版本的封皮文本列表的末端。一个实体（entity）可以加入 [或通过排列（arrangement）制作] 一段封面或封底文本。如果原档已经为该封皮（封面或封底）包含了封皮文本，由您或您所代表的实体先前加入或排列的文本，您不能再新加入一个，但您可以在原来的发行者的显示的许可下替换掉原来的那个。

作者和发行者不能通过本许可证授权公众使用他们的名字推荐或暗示认可任何一个修改版本。

7.3.6 合并文档

遵照第 4 节所说的修改版本的规定，您可以将文档和其他文档合并（combine）并以本许可证发布，只要您在合并结果中包含原文档的所有不可变章节，对它们不加以任何改动，并在合并结果的许可声明中将它们全部列为不可变章节，而且维持原作者的免责声明不变。

合并的作品仅需要包含一份本许可证，多个相同的不可变章节可以由一个来取代。如果有多个名称相同、内容不同的不可变章节，通过在章节的名字后面用包含在括号中的文本加以原作者、发行者的名字（如果有的话）来加以区别，或通过唯一的编号加以区别。并对合并作品的许可声明中的不可变章节列表中的章节标题做相同的修改。

在合并过程中，您必须合并不同原始文档中任何以特殊标题“版本历史”命名的章节，从而形成新的特殊标题“版本历史”的章节；类似地，还要合并特殊标题“致谢”和“献给”命名的章节。您必须删除所有以特殊标题“签名（Endorsements）”命名的章节。

7.3.7 文档集

您可以制作一个文档和其他文档的合集（collection），在本许可证下发布，并在合集中将不同文档中的多个本许可证的拷贝以一个单独的拷贝来代替，只要您在文档的其他方面遵循本许可证的逐字地拷贝的条款即可。

您可以从一个这样的合集中提取一个单独的文档，并将它在本许可证下单独发布，只要您想这个提取出的文档中加入一份本许可证的拷贝，并在文档的其他方面遵循本许可证的逐字地拷贝的原则。

7.3.8 独立作品的聚合体

文档或文档的派生品和它的与之相分离的独立文档或作品编辑在一起，在一个大包中或大的发布介质上，如果其结果著作权对编辑作品的使用者的权利的限制没有超出原来的独立作品的许可范围，称为文档的聚合体“（aggregate）”。当以本许可证发布的文档被包含在一个聚合体中的时候，本许可证不施加于聚合体中的本来不是该文档的派生作品的其他作品。

如果第 3 节中的封皮文本的需求适用于文档的拷贝，那么如果文档在聚合体中所占的比重小于全文的一半，文档的封皮文本可以被放置在聚合体内包含文档的部分的封皮上，或是电子文档中的等效部分。否则，它必须位于整个聚合体的印刷的封皮上。

7.3.9 翻译

翻译被认为是一种修改（modification），所以您可以按照第 4 节的规定发布文档的翻译版本。如果要将文档的不可变章节用翻译版取代，需要得到著作权人的授权，但您可以将部分或全部不可变章节的翻译版附加在原始版本的后面。您可以包含一个本许可证和所有许可证声明、免责声明的翻译版本，只要您同时包含他们的原始英文版本即可。当翻译版本和英文版发生冲突的时候，原始版本有效（prevail）。

在文档的特殊章节“致谢”、“献给”、“版本历史”章节，第 4 节的保持标题的要求恰恰是要更换实际的标题的。

7.3.10 终止

除非确实遵从本许可证，否则您不可以对遵从本许可证发布的文档进行复制、修改、附加许可证或发布。任何其它的试图复制、修改、附加许可、发布本文档的行为都是无效的，并自动终止本许可证所授予您的权利。然而其他从您这里依照本许可证得到的拷贝或权力的人（或组织）得到的许可证都不会终止，只要他们仍然完全遵照本许可证。

terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation. 然而，如果你终止所有违反本授权的行为，特定版权所有人会暂时恢复你的授权直到此版权所有者明确并最终地终止你的授权。或者特定版权所有人永久地恢复你的授权如果此版权所有人在停止违反授权后的 60 天内没有通过合理的方式通知你违反授权。

此外，此版权所有用一些合理的方式通知你违反了授权规定，也是你第一次从此版权所有收到违反授权的通知，并且你在收到通知后的 30 天内终止了这种行为，那么此版权所有会永久地恢复你的授权。

你的权利在此章节的终止并不代表终止得到你的拷贝和权利的当事人的授权。如果你的权利被终止而没有被永久性地恢复，那么你将没有任何权利去使用此章节的全部或部分资料和资料的拷贝。

7.3.11 本协议的未来修订版本

未来的某天，自由软件基金会（FSF）可能会发布 GNU 自由文档许可证的修订版本。这些版本将会和现在的版本体现类似的精神，但可能在解决某些问题和利害关系的细节上有所不同。参见 <http://www.gnu.org/copyleft/>。

本许可证的每个版本都有一个唯一的版本号。如果文档指定服从一个特定的本协议版本“或任何之后的版本（or any later version）”，您可以选择遵循指定版本或自由软件基金会的任何更新的已经发布的版本（不是草案）的条款和条件来遵循。如果文档没有指定本许可证的版本，那么您可以选择遵循任何自由软件基金会曾经发布的版本（不是草案）。

7.3.12 重新授权

“Massive Multiauthor Collaboration 网站”（或“MMC 网站”）是指任何发布有著作权作品的网站服务器，也为任何人提供卓越的设施去编辑一些作品。任何人都可编辑的一种公众维基（wiki）就是这种服务器的一个例子。包含在这个网站的“MMC”是指任何一套在 MMC 网站上发布的具有著作权的作品。

“CC-BY-SA”是指 Creative Commons Attribution-Share Alike 3.0 授权，它是被“知识共享组织”颁布的，“知识共享组织”是一家非赢利性的，在圣弗朗西斯科、加利福尼亚具有重要的商业地位的组织。而且未来的“copyleft”版本的授权也是被同一个组织发布的。

“合并（Incorporate）”是指以整体或作为另一个文本的部分发布或重新发布一个文本。

MMC 有“重新授权的资格”，如果它是在 MMC 下授权；或者所有的作品首次发布并非在 MMC 下授权，后来以整体或部分合并到 MMC 下，它们（1）没有封面文本（cover texts）或不变的章节，并且（2）在 2008 年 11 月 1 日之前合并。

那么 MMC 网站的操作者会在 2009 年 8 月 1 日之前的任何时间在同一网站重新发布包含在这一网站的 MMC 是经过 CC-BY-SA 授权的，只要那个 MMC 有资格重新授权。

7.3.13 补充：如何使用本许可证

要使用本许可证发布您写的文档，请在文档中包含本许可证的一个副本，并在紧接着扉页之后加入如下版权声明与许可声明：

Copyright ©YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

如果您有不可变章节、封面文本和封底文本，请将“with ... Texts.”一行替换为：

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

如果您有不可变章节而没有封皮文本或上述三项的其他组合，将两个版本结合使用以满足实际情况。

如果您的文档包含有意义的程序示例代码，我们建议您同时将代码按照您的选择以自由软件许可证发布，比如 GNU 通用公共许可证。以授权它们作为自由软件被使用。

7.4 L^AT_EX 项目公共许可证

LPPL Version 1.3c 2008-05-04

Copyright 1999, 2002–2008 L^AT_EX3 Project

允许每个人复制和发布本授权文件的完整副本，但不允许对它进行任何修改。

7.4.1 引言

L^AT_EX 项目公共许可证 (The L^AT_EX Project Public License, LPPL) 是分发 (distribute) L^AT_EX 内核和基本的 L^AT_EX 宏包时必须遵守的许可证。

您可以将该许可证用于您的维护版权 (hold the copyright) 和分发的的工作。该许可证很适合您的 T_EX 相关的工作 (如 L^AT_EX 宏包)，但它是写于这样一种情况之下，即您的工作与 T_EX 毫不相干，您也可以使用该许可证。

本节的内容是“是否以及如何在这个许可证下做分发工作”，下面给出一些说明、例子和建议以方便那些正在考虑在这个许可证下分发他们作品的作者。

这个许可证在某个作品可能被分发和修订，以及在那个作品的修订版的情况下可能被分发。

我们相信，L^AT_EX3 项目下述的条件给您制作和发布您作品修订版本的自由，同时与您期望的技术规范一致，保持您作品的可用性、完整性、可靠性。如果您不知道如何实现您的目标，同时满足这些条件，请在分发的 L^AT_EX 中阅读文档“`cfgguide.tex`”和“`modguide.tex`”来获得建议。

7.4.2 定义

这个许可证文档会使用下列术语：

作品 (Work) 指在这个许可证许可下进行的任何分发工作。

衍生作品 (Derived Work) 指衍生于作品的任何合法的其它作品。

修正 (Modification) 指产生一个合法衍生作品的任何过程，例如，一个文件的产生过程，包括与这个文件相关的原始文件或者像这样文件的一个重要部分无论是直接引用或修改和/或翻译成另一种语言。

修改 (Modify) 指实施产生合法衍生作品的任何过程。

分发 (Distribution) 指从一个人复制作品的全部或部分给另一个人。分发包括 (但不限于) 制作作品的电子元件 (electronic components)，该电子元件适合文件传协议 (file transfer protocols) 如 FTP 或 HTTP，或适合共享文件系统 (shared file systems) 如升阳公司的网络文件系统 (Sun's Network File System) (NFS)。

编译作品 (Compiled Work) 指将作品处理成可以直接在计算机系统里可用的格式的过程。这种处理可能包括作品提供的安装工具 (installation facilities) 的使用、作品的转变 (transformations of the Work)、作品元件 (components of the Work) 的复制, 或其他处理。请注意, 任何对由作品提供的安装工具的修改 (modification) 构成了对作品的修改。

当前维护者 (Current Maintainer) 指在作品里被提名的一个或者几个人。如果没有这样明确的提名则是合法的“版权持有人 (Copyright Holder)”

基本解释程序 (Base Interpreter) 指一个软件 (program) 或程序 (process), 它对运行或解释作品的一部分或全部通常是必需的。

一个基本解释程序可能依赖外部组件 (external components), 但这些外部组件不认为是基本解释程序的一部分, 规定 (provided that) 无论何时在交互使用 (used interactively) 外部组件时都必须明确标识 (clearly identifies) 每一个外部组件。否则, 将许可证应用于作品时, 必须明确指明 (explicitly specified) 唯一可用的基本解释程序是“L^AT_EX-Format”或程序 (program) 执行“T_EX”语言的文件属于“L^AT_EX-Format”。

7.4.3 分发和修改的条件

1. 除分发和/或修改作品的行为之外, 其它行为没有涵盖在许可证之中。特别地, 作品运转 (run) 的行为不受限制, 并且没有任何关于提供支持这项作品的要求。
2. 您可以分发一个完整的、未经修改的您收到的作品的副本。只有部分作品的分发是需要修改的, 但没有权利分发这样的符合该条款 (clause) 中术语 (term) 的衍生作品。
3. 在符合上面第 2 条款时, 您可以分发一件从一个完整的、未经修改的作品副本产生的编译作品 (Compiled Work)。只要该编译作品是直接从中产生的, 并按照这样的分发方式, 收件人 (recipient) 便可以在他们的系统中完全正确地安装编译作品。
4. 如果您是作品当前的维护者 (Current Maintainer), 则可以没有限制地修改作品, 从而创建一个衍生作品。您也可以没有限制地分发衍生作品, 包括衍生作品中产生的编译作品 (Compiled Work)。当前维护者用这种方式分发的衍生作品被认为是作品的新版本。
5. 如果您不是作品当前维护者, 则可以修改您的作品副本, 从而在作品的基础上创建一个衍生作品, 并编译此衍生作品从而这个衍生作品的基础上创造一个编译作品。
6. 除了版权说明 (copyright notice) 中明确声明的部分, 只要对作品的每个部分符合下列条件时, 即使您不是作品的当前维护者, 您也可以分发生成作品。只有当前维护者可以对作品的一个组成部分增加这样的声明 (statement)。
 - (a) 如果衍生作品的一个部分可以直接替换作品中一个基本解释程序使用的部分, 然后, 当交互式地使用基本解释程序时, 作品这个部分被用户识别的任何地方, 衍生作品的替代部分清楚地被用户识别为这个部分的一个修改版本 (modified version)。
 - (b) 衍生作品的每个部分都包含针对该部分详述性质变化的突出声明 (prominent notices), 或对另一个文件的突出参考条目 (prominent reference), 这个文件是作为衍生作品的一部分分发的, 并且包含一个完整准确的更改日志。

- (c) 衍生作品的信息中没有针对该衍生作品收件人的任何暗示，这些人包括（但不限于）作品的初始版本（original version）作者、提供任何支持包括（但不限于）错误报告和处理的人，除非这些人已经明确表示为衍生作品提供了这种支持。
- (d) 您可以至少分发以下衍生作品中的一种：
 - i. 一个完整的、未经修改的作品副本；如果修改部分（modified component）的分发是通过提供权限（offering access）从一个指定的地点（designated place）复制修改的部分来实现，那么，会提供相同的权限（equivalent access）用于从相同或相似的符合该条件的地点复制作品，即使第三方（third parties）不会随着修改部分被迫复制作品。
 - ii. 足以获得一个完整的、未经修改的作品副本的信息。
- 7. 如果您不是作品的当前维护者，只要衍生作品是分发给编译作品的全部收件人，并且满足上述第 6 条的关于衍生作品的条件，您就可以分发来自衍生作品的编译作品。
- 8. 上述条件的目的不是禁止，并且因此不适用以任何方式修改任何部分，以便使其成为作品的那个部分相同的更新版本，正如上述第 4 条当前维护者分发的那样。
- 9. 以可选的方式（alternative format）分发作品或衍生作品，不是放宽（relax）或者废弃（nullify）该许可证的任何部分（section），因为它涉及（pertain to）应用过程（process）的结果，在可选的方式中，作品或衍生作品（全部或部分）由适应于该方式的应用过程产生。
- 10. (a) 衍生作品可能在不同的许可证许可下进行分发，关于作品，如果那个许可证本身尊重上述第 6 条中的条件，虽然它并没有必须尊重本许可证的其他条件。
(b) 就作品的变化而言，如果派生作品在不同的许可证许可下进行分发，衍生作品必须提供足够的文件作为它的一部分来允许每一个收件人，兑现上述第 6 条中的限制。
- 11. 本许可证对与该作品无关的作品没有限制，也对以任何方式聚集这些作品的工作没有任何限制。
- 12. 本许可证没有任何企图或用于破坏适应的法律。

7.4.4 无担保声明

作品没有保修。除非另有书面说明，版权持有人提供作品“按照原样（as is）”，没有任何形式的担保，明示或暗示，包括但不限于，针对特定用途的适销性和适用性的暗示保证。作为作品的质量和性能的全部风险与您同在。作品应证明有缺陷，否则您将承担所有必要的维修、修理、或改正的费用。

除非适用法律要求，或版权持有人书面同意，或该作品任何署名作者，分发和/或修改作品的其他人员，同意上述或任何其他方，负责对您损害赔偿，否则将不赔偿。损害包括任何一般，特殊，附带或间接损害所产生的任何无法使用作品的情况，（包括但不限于数据丢失，所呈现的数据不准确，或持续受损失人作为工作的任何失败与任何其他程序）的结果，即使版权持有人或者作者，或者对方已被告知此类损害的可能性。

7.4.5 作品的维护

工作中有“作者维持（author-maintained）”这么一个状态，如果版权所有人明确地、突出地说要注意作品的主要版权，那么作品只能通过版权所有人或者是“作者维持”这么一个简单的状态去维持。

如果当前维护者表明他们愿意接受来自作品中的错误（例如，通过提供一个有效地电子邮箱），则作品处于“维持”状态。但并不要求当前维护者对这些错误有深刻理解或者采取措施。

如果没有当前维护者，或者是有人声明要做作品的当前维护者，然后在 6 个月的时间内，他并没有通过指定的手段使得他能胜任，且没有其他显著的有效维护的迹象 (signs of active maintenance)，则作品从“维护”状态进入“未维护”状态。

通过与任何一位当前维护者达成一致，您就可以取代他的角色成为一名维护者。

如果作品处于未维护状态，您也可以成为作品的当前维护者，只需通过以下几个步骤：

1. 通过互联网或类似的搜索手段做出一个合理的打算去跟随当前维护者（和版权持有人，如果两者不是同一人）。
2. 如果这个搜索是成功的，然后询问作品是否仍处于维护状态。
 - (a) 如果它正处于维护状态，然后向当前维护者要求更新他们一个月内的通讯数据 (communication data)。
 - (b) 如果搜索不成功，或当前维护者没采取行动以恢复有效维护 (active maintenance)，那么相关的社区宣布您接管维护。（如果作品是一个 L^AT_EX 作品，可以这样做，例如，张贴到 `comp.text.tex`。）
3.
 - (a) 如果当前维护者做到了，并同意作品的维护交给您，那么公布后将立即生效。
 - (b) 如果当前维护者做不到，并且版权持有人同意把作品的维护交给您，那么公布后将立即生效。
4. 如果您做一个像在上面 2b 所述的那样的“意向性声明 (intention announcement)”，三个月后您的意向不被当前维护者反驳 (challenge)，也不被版权持有人或其它人反驳，那么，您可以改编 (arrange for) 作品，从而您成了 (新的) 当前维护者。
5. 如果在 3b 或 4 的条件下 3 个月内完成了某个更改 (a change)，以前的未达当前维护者 (unreachable Current Maintainer) 再一次成为可达当前维护者 (reachable Current Maintainer)，那么，那个当前维护者必须成为或保持为上述的当前维护者，要求提供他们一个月内更新的通讯数据 (communication data)。

当前维护者的变化，本身并不改变在 LPPL 许可证下这项工作的分发的的事实。

如果您成为作品的当前维护者，您应该立即提供，在作品中，突出自己的地位和毫不含糊的声明，您也应向有关社区宣布新的状态，像上面在 2b 所述。

7.4.6 该许可证允许下能否和如何分发

本节为那些正在考虑在这个许可证下分发他们作品的作者提供了重要指示、例子和建议。这些作者在本节中以“您”的形式出现。

7.4.7 选择这个或其它许可证

如查您想或需要在您作品的任何一部分使用与该许可证显著不同的分发条件 (distribution conditions)，那么，请勿在您作品的任何地方提及 (refer to) 该许可证，而应在不同的许可证下分发您的作品。您可以将该许可证的文本作为您的许可证的模板，但您的许可证不应该提及 LPPL，否则会给人您的作品在 LPPL 下分发的印象。

基本 L^AT_EX 分发中的“`modguide.tex`”文档解释了这个许可证条件 (conditions of this license) 后面的动机。它解释说，例如，为什么 GNU 通用公共许可证 (GPL) 下分发 L^AT_EX 被认为是不恰当的。即使您的作品与 L^AT_EX 无关，而在“`modguide.tex`”里的讨论仍可能是相关的，鼓励那些想在某些许可证下分发自己作品的作者去读一下。

7.4.8 不进行分发时的修改的建议

不去修改作品的任何部分是明智的，即使是您自己用的，而且分发修改部分没有满足上述条件时也不要修改。尽管您可能希望永远不会分发这样的修改，但这还是常常发生 – 您也许忘记曾经修改过，或者其他人进入您修改过的版本，您这样分发并且违反了本许可证的条件，从而可能引起法律纠纷，更糟糕的是，引发社会问题。因此，通常在最有利的情况下保持您的作品副本与公开发布的一样。许多作品在没有改变任何得到许可的部分的情况下提供一些方式来控制那种行为。

7.4.9 如何使用该许可证

要使用此许可证，可以在您作品的每一个组成部分里放置一个明确的版权声明，包括您的姓名和作品的撰写和/或最后大幅修改的年份。还包括一个受该许可证约束的分发和/或修改该组件的声明。

下面是一个这样的注意和声明的例子：

```
%% pig.dtx
%% Copyright 2005 M. Y. Name
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3
% of this license or (at your option) any later version.
% The latest version of this license is in
% http://www.latex-project.org/lppl.txt
% and version 1.3 or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is M. Y. Name.
%
% This work consists of the files pig.dtx and pig.ins
% and the derived file pig.sty.
```

在一个文件中给出这样的一个注意和声明，将适应该许可证的条件，“Work”适应三个文件即“pig.dtx”、“pig.dtx”、“pig.sty”（最后这个文件产生于用“pig.ins”产生的“pig.dtx”），“Base Interpreter”适应任何“L^AT_EX-Format”，“Copyright Holder”和“Current Maintainer”适应“M. Y. Name”。

如果您不希望 LPPL 的维护（Maintenance）这一节适应于您的作品，请将上述的“维护（maintained）”更改为“作者维护（author-maintained）”。然而，我们建议您使用“维护（maintained）”，因为添加维护（Maintenance）这一节就是为了确保您的作品对社会（community）有用，即使您自己都不再维护并支持您的作品。

7.4.10 非替代的衍生作品

LPPL 所指定的若干条款（clauses）为用户社区（user community）提供可靠性和稳定性。因此，他们关心这种情形，即衍生作品是用作初始作品（original Work）的（合适的或不合适的）替代品。如果是这种情形并非如此（例如，如果一个完全不同的任务中重复使用几行代码），第 6b 和 6d 的条款不适应。

7.4.11 重要的建议

定义是什么构成了作品 LPPL 要求作品的分发需包含作品的全部文件。因此，您提供给许可用以证判断哪些文件构成作品的方式是非常重要的，例如，通过明确列出每个文件的版权声明（copyright notice）或在那里使用诸如这样的一行字：

```
% This work consists of all files listed in manifest.txt.
```

在没有一个明确的清单的情况下，许可证不可能判定哪些文件组成了作品，在这种情况下，许可证将有权作出合理的猜测哪些文件组成了作品。

部分 II

创建演示稿

这部分将阐述用于创建演示稿 (presentations) 的所有命令。首先看看用于创建帧 (*frames*) 的命令和环境, 以及演示稿的基本构造块 (building blocks); 然后是如何创建叠层 (overlays)。

接下来的三节的内容与组建 (*structuring*) 演示稿的命令和方法有关。依次是: 静态的全局结构 (*static Global Structure*)、交互的全局结构 (*nteractive Global Structure*)、局部结构 (*local structure*)。

最后两节的内容与插图 (graphics) 和动画 (Animation) 有关。这些内容不仅适用于 BEAMER, 也适用于其它宏包。

8 创建帧

8.1 帧环境

一个演示稿 (presentation) 由一系列帧 (frame) 组成, 而每一帧又由一系列幻灯片 (slide) 组成。我们可使用 `\frame` 或 `frame` 环境创建帧, 二者作用相同。`\frame` 命令带有一个参数即帧的内容。未经叠层规则 (overlay specifications) 标记的所有文本将在帧的所有幻灯片中显示。叠层规则将在后面详细介绍。目前, 暂且将叠层规则定义为放置于尖括号中的数字或数字范围组成的一个列表, 该列表放置于某一命令之后, 如 `\uncover<1,2>{Text}`。如果某个帧包含叠层规则, 则该帧包含多张幻灯片; 如果某个帧不包含叠层规则, 则该帧只有一张幻灯片。

```
\begin{frame}<<overlay specification>>[<<default overlay specification>>][<options>]{<title>}{<subtitle>}
  <environment contents>
\end{frame}
```

即:

```
\begin{frame}<<叠层规则>>[<<默认的叠层规则>>][<选项>]{<标题>}{<子标题>}
  <environment contents>
\end{frame}
```

`<overlay specification>` 控制着显示帧的哪张幻灯片。如果忽略, 则会自动计算尖括号中的数字或数字范围。`<environment contents>` 可以是普通的 L^AT_EX 文本, 但不可以包含 `\verb` 命令或 `verbatim` 环境¹⁷, 也不可以包含能改变字符代码的任何环境, 否则必需加上 `fragile` 选项。

可选的选项 `<title>` 会被左侧的大括号 (opening brace) 检测到, 换言之, 如果帧的第一个东西是左大括号, 那么 `<title>` 选项会被帧标题 (frame title) 霸占。同理, 可选的选项 `<subtitle>` 也会被左侧的大括号检测到, 也就时说, 被 `<title>` 后面左大括号检测到。而 `title` 和 `subtitle` 可由 `\frametitle` 和 `\framesubtitle` 命令给出。

当普通的 L^AT_EX 命令 `\frame` 放在帧内时, 其意义和平常一样。在帧内外, 命令 `\frame` 均可用, 命令 `\framelatex` 也一样。

举例:

```
\begin{frame}{A title}
  一些内容。
\end{frame}
% 相同的效果:
\begin{frame}
  \frametitle{A title}
  一些内容。
\end{frame}
```

举例:

¹⁷解决 beamer 下 frame 环境不能使用 `\verb` 命令或 `verbatim` 环境的三个办法: ①为 frame 环境添加 `fragile` 选项; ②为 frame 环境添加 `containsverbatim` 选项; ③在 frame 环境中使用 `semiverbatim` 环境。

```
\begin{frame}<beamer>{Outline} % 帧只显示在 beamer 模式中
  \tableofcontent[current]
\end{frame}
```

通常，完整的 $\langle environment contents \rangle$ 放在一张幻灯片中。如果文本太多在一张幻灯片中容不下时，文本将尽可能地被挤压，并发出警告，文本将向底部溢出。我们可以使用 `allowframebreaks` 选项以使 $\langle frame text \rangle$ 分放在几张幻灯片中，但这时不可以使用叠层 (overlays)。详细内容请参考 `allowframebreaks` 选项。

$\langle default overlay specification \rangle$ 是可选的选项。它的“被检测”规则如下：如果方括号中的第一个可选参数由 `<` 开始，则该参数是一个 $\langle default overlay specification \rangle$ ，否则就是普通的 $\langle options \rangle$ 参数。因此 `\begin{frame}[<+>][plain]` 是合法的，`\begin{frame}[plain]` 也是合法的。

$\langle default overlay specification \rangle$ 的效果如下：帧内的每一条命令或每一个环境（包括 `\item`、`actionenv` 环境、`\action` 命令，和所有的 block 环境），均接受一个行为规则 (Action Specification)，请参考第 98 页第 9.6.3 节，但不接受 $\langle default overlay specification \rangle$ 作为其规则。通过提供递增的规则 (Incremental Specification) 如 `<+>`，请参考第 100 页第 9.6.4 节，实际上使全部块 (blocks) 和全部 enumerations 分段显示 [块内使用行为规则 (Action Specification)]。

举例：在这帧，定理 (theorem) 从第一张幻灯片开始显示，证明 (proof) 从第二张幻灯片开始显示，开始的两个条目 (itemize points) 相继显示，最后一个条目与第一个条目一起显示。该帧总共包含四张幻灯片。

```
\begin{frame}[<+>]
  \begin{theorem}
    $A = B$.
  \end{theorem}
  \begin{proof}
    \begin{itemize}
      \item Clearly, $A = C$.
      \item As shown earlier, $C = B$.
      \item<3-> Thus $A = B$.
    \end{itemize}
  \end{proof}
\end{frame}
```

可能会给出下列 $\langle options \rangle$ ：

- `allowdisplaybreaks=\langle break desirability \rangle` 将 AMSTeX 命令 `\allowdisplaybreaks[\langle break desirability \rangle]` 发送给当前帧。 $\langle break desirability \rangle$ 的取值可以从 0（意味着公式不被断开）到 4（为默认值，意味着公式无条件地在任何地方被断开）。 $\langle break desirability \rangle$ 只有和 `allowsframebreaks`¹⁸ 一起时才有意义。
- `allowframebreaks=\langle fraction \rangle` 给定该选项时，如果文本在一张幻灯片中容纳不下，帧将被自动分成多个页面¹⁹。详细地情况是，当给定该选项时，会发生下面的事情：
 1. 不支持叠层。
 2. 用 `\note` 命令创建的帧的任何笔记 (notes) 将被插入到帧的第一页之后。

¹⁸注意这三个命令的区别：`\allowframebreaks` 的作用是将帧分成多个页面；`\shrink` 的作用是缩小内容以充满一张幻灯片；`\squeeze` 的作用是挤压垂直空间。

¹⁹一个帧就像一本图书，一张幻灯片就像图书的一页。因此，帧的一个页面其实就是帧的一张幻灯片。

3. 帧的任何脚注 (footnotes) 将被插入到帧的最后一页。
4. 如果存在帧标题 (frame title), 那么该帧的每一页将拥有该帧标题, 插入的记号 (note) 表明这一页是帧的哪一页。默认情况下, 该特定记号是一个罗马数字。然而可以通过下述模板改变该记号。

Beamer-Template/-Color/-Font `frametitle continuation`

即:

Beamer-Template/-Color/-Font 帧标题连续

设定 `allowframebreaks` 选项后, 在帧标题的末尾插入该模板的文本。

下面的模板选项是预定好的:

- `[default]` 安装罗马数字作为模板文本。该罗马数字指明了帧的当前页。
- `[roman]` 罗马数字的默认字体为 `Alias`。
- `[from second]` `[<text>]` 安装该模板后, 将从帧的第二页开始插入 `<text>`。默认情况下, 插入的 `<text>` 是 `\insertcontinuationtext`, 它依次是 `(cont.)`。

下面的插入物 (inserts) 是有效的:

- `\insertcontinuationcount` 从帧的第二页开始, 在帧标题的末尾插入帧的当前页的阿拉伯数字序号²⁰。
- `\insertcontinuationcountroman` 从帧的第二页开始, 在帧标题的末尾插入帧的当前页的 (大写的) 罗马数字序号。
- `\insertcontinuationtext` 从帧的第二页开始, 在帧标题的末尾只插入文本 `(cont.)` 或其翻译 [如德语中的 `(Forts.)`]

如果要将帧分成多个页面, 默认情况下, 除最后一页外, 其它页面只会填满 95% 的内容。因此, 根据帧的垂直放置 (`vertical placement`) 选项会在顶部和/或底部会留下一些空白。这可以获得更好地视觉外观, 而填满 100% 只会显得很拥挤。当然, 我们可以改变可选的参数 `<fraction>` (因子) 来更改变这个百分比, 当 `<fraction>` 为 1 时表示 100%, 当 `<fraction>` 为 0.5 时表示 50%。这个百分比包含帧标题 (frame title)。因此, 要将帧分成“大致的两半”, 我们必须将 `<fraction>` 设置成 0.6。

常规 \TeX 分页的细节也适应于该选项。例如, 如果我们想自动断开一个方程式, 请使用 `\allowdisplaybreaks` 命令。我们可以插入 `\break`、`\nobreak`、`\penalty` 命令以控制中断 (breaks) 在哪里出现。`\pagebreak` 和 `\nopagebreak` 命令同样有效, 包括它们的选项。如果我们不想在帧中及论文模式 (`article mode`) 中应用分页, 可以添加像 `<presentation>` 这样的模式规则 (mode specification) 以使这些命令只适应于演示稿模式 (`presentation modes`)。`\framebreak` 是 `\pagebreak<presentation>` 的简写形式 (shorthand), `\noframebreak` 是 `\nopagebreak<presentation>` 的简写形式。

使用该选项是有害的 (*evil*)。在 (好的) 演示稿中, 我们会精心准备每一张幻灯片, 会再三考虑是否将一张幻灯片的内容放到不同的幻灯片中。使用 `allowframebreaks` 选项, 会让我们的演示稿变得很糟糕、没完没了, 越来越象“投影在墙上的论文”。然而, 该选项也有它的作用。最引人注目的作用是它可以方便地自动分开参考书目 (bibliographies) 或长的方程式 (equations)。

举例:

```
\begin{frame}[allowframebreaks]{References}
```

²⁰实现的语句是: `\setbeamertemplate{frametitle continuation}[from second][\insertcontinuationcount]`

```

\begin{thebibliography}{XX}

\bibitem...
\bibitem...
...
\bibitem...
\end{thebibliography}
\end{frame}

```

举例：

```

\begin{frame}[allowframebreaks,allowdisplaybreaks]{A Long Equation}
\begin{align}
\zeta(2) &= 1 + 1/4 + 1/9 + \cdots \\
&= \dots \\
&= \pi^2/6.
\end{align}
\end{frame}

```

- **b**, **c**, **t** 使帧以底/中/顶垂直对齐。它会废除 (overrides) 全局放置策略 (global placement policy), 全局放置策略由文档类的 **t** 选项和 **c** 选项控制着。
- **fragile=singleslide** 告诉 BEAMER 帧内容 (frame contents) 是“脆弱的 (fragile)”。这意味着帧包含的文本不会“像平常一样被显示 (interpreted as usual)”。例如, 这适应于抄录文本 (verbatim text), 显而易见, 抄录文本和普通文本的显示是不同的。

如果帧包含脆性文本 (fragile text), 在排版帧时将使用不同的内部机制 (mechanisms) 以确保在帧内重置字符代码 (reset character codes)。转换成另一内部机制的代价是, 我们不能使用叠层, 或者必需写入一个外部文件 (external file) 并将它读回 (read back) (这往往是不称心的)。

更详细的, 当该选项给定给标准的 (pdf)LaTeX 时, 会发生下面的情况: 扫描帧的内容并将其写入一个特定的名为 $\langle jobname \rangle.vrb$ 的文件, 或者如果给帧指定了标签 (label), 则该特定的文件名为 $\langle jobname \rangle.\langle current frame number \rangle.vrb$ 。然后, 再一次启动该帧, 并读回该特定文件的内容。从这以后, 在读取文件的基础上, 就可以修改字符代码 (character codes) 了, 这样我们就可以使用抄录文本 (verbatim text) 和叠层了。

常用下面的规则 (rule) 判断帧的结束: 包含 $\end{\langle frame environment name \rangle}$ 的一行首次出现时结束该帧。 $\langle environment name \rangle$ 是一般的 (normally) **frame**, 但可以用 **environment** 选项改变它。这个规则是必需的, 因为帧的内容聚集在一起时不能被解释 (interpreted)。

也可以添加可选的信息 **=singleslide**。这会告诉 BEAMER 该帧只包含一张幻灯片。既然如此, 帧的内容就不会写入特定的文件, 而是直接被解释, 这是“更快 (faster) 和更干净 (cleaner)”的。

- **environment= $\langle frame environment name \rangle$** 该选项只有和 **fragile** 选项 (但不能用作 **fragile=singleslide**, 只能用作平常的 **fragile**) 连用时才是有益的。当聚集帧内容时, $\langle frame environment name \rangle$ (帧环境名) 用于判断扫描的结束。通常, 到达 \end{frame} 时帧结束。然而, 如果我们在另一环境中使用了 \begin{frame} , 我们必需使用该选项:

举例：


```

\newenvironment{slide}[1]
  {\begin{frame}[fragile,environment=slide]
   \frametitle{#1}}
  {\end{frame}}

```

```

\begin{slide}{My title}
  Text.
\end{slide}

```

如果在上述的例子中我们指定了选项 `environment=slide`, T_EX 将“迷失”幻灯片的结束处, 因为当帧的内容聚集在一起时是不能被解释的。

- **label=*<name>*** 使帧的内容存贮在名为 *<name>* 的文件中, 以后用命令 `\againframe` 从该文件中读回帧的内容。而且, 在帧的每一张幻灯片上创建名为 *<name><slide number>* 的标签。在第一张幻灯片, 会创建名为 *<name>* 的标签 (因此标签 *<name>* 和 *<name><1>* 指向同一张幻灯片)。注意这些标签可用作超链接的目标²¹。

该选项可以和 `fragile` 选项一起使用。

- **plain** 该选项会抑制 (suppress) (即不显示) 顶部导航区 (headline)、底部导航区 (footline) 和侧栏 (sidebar)。这有利于创建一个简单的帧, 使该帧带有不同的顶边和底部导航区。当某帧含有一幅填满该帧大的图像时, 该选项也很有用。

举例: 某帧含有一幅填满该帧大的图像:

```

\begin{frame}[plain]
  \begin{centering}%
    \pgfimage[height=\paperheight]{somebigimagefile}%
  \par%
  \end{centering}%
\end{frame}

```

举例: 封面的顶部和底部导航区被两幅图像替代。

```

\setbeamertemplate{title page}
{
  \pgfuseimage{toptitle}
  \vskip0pt plus 1filll

  \begin{centering}
    {\usebeamerfont{title}\usebeamerfont[fg]{title}\inserttitle}

    \insertdate

```

²¹例如: 在一个帧标题名为“中暑”、帧标签名为“ZS”的帧中, 有一个“返回”按钮, 点击该按钮后将返回到帧标签名为“WL”的另一个帧中, 实现语句是:

```

\begin{frame}[label=ZS]{中暑}
  帧内容
  \hyperlink{WL}{\beamerreturnbutton{返回}}
\end{frame}

```

```

\end{centering}

\vskip0pt plus 1filll
\pgfuseimage{bottomtitle}
}
\begin{frame}[plain]
\titlename
\end{frame}

```

- `shrink=<minimum shrink percentage>` 该选项会使太长的帧文本（frame text）缩小以适合帧。BEAMER 首先会排版全部帧。然后查看帧文本（除外帧标题）的垂直尺寸，如果这个垂直尺寸大于文本的高度减去帧标题的高度的差，BEAMER 会计算出一个缩小因子（shrink factor），并据该缩小因子按比例缩小帧文本，这样帧文本便可填满整个帧。使用该选项会自动应用 `squeeze` 选项。

因为缩小发生在所有的东西已排版完毕后，已缩小的帧文本在水平方向将不会填满整个帧。因此，我们可以指 `<minimum shrink percentage>` 如 20。如果指定了该百分比，帧文本将按大于或等于该百分比进行缩小。因为 BEAMER 知道这一点，会按比例增大水平宽度以使缩小的文本再一次填满全部帧。然而，如果该百分比不是足够大，则文本会尽可能缩小，我们会得到一条提示信息。

使用该选项的最佳情形是识别已填满的帧，但是它的文本又必需绝对地容纳在单一的帧内。开始可以指定 `shrink=5`，然后指定 `shrink=10` 等，直到不再有提示信息出现（或直到结果看起来让人满意，从而忽略该提示信息）。

使用该选项是非常不幸的（*very evil*）。因为它会改变幻灯片的字体尺寸，而这将导致印刷恶梦。通过重组（restructure）和简化（simplify）帧可以避免使用该选项，从而使演示稿更好。

举例：

```

\begin{frame}[shrink=5]
Some evil endless slide that is 5\% too large.
\end{frame}

```

- `squeeze` 将文本中的垂直间隙（vertical spaces）尽可能地挤压在一起。当前，该选项仅能将排序（enumerations）或常规（itemizations）列表中的垂直间隙挤压至零。

使用该选项不是好事，也不是坏事。

LYX 使用样式“BeginFrame”开始帧，使用样式“EndFrame”结束帧。上一帧也会在下一帧开始处自动结束。一帧也会在一个新节（new section）或节（section）开始处自动结束（不是文档的结束处!）。

LYX 在 \TeX -模式中，将选项和叠层规则作为帧标题（frame title）的第一件东西，从而将选项和叠层规则传递给帧。

LYX 出于便利，已包含了样式“BeginPlainFrame”。它将 `plain` 选项传递给帧。要传递更多的选项给普通的帧（plain frame），我们必须使用普通的样式“BeginFrame”，并指定全部选项（包括 `plain`）。

LYX 原因是 \LaTeX 使用不同的内部机制排版抄录文本，这对 BEAMER 来说，处理变得更容易。

ARTICLE 在 `article` 模式中，帧环境不会为初始帧（original frame）产生任何可视的参照（reference）（不产生帧）。当然啦，帧文本会插入到普通的文本中。要改变这种状况，我们可以修改模板 `frame begin` 和 `frame end`，方法见下面。在 `article` 模式中，要抑制帧，例如，我们可以指定 `<presentation>` 作为叠层规则。

Beamer-Template `frame begin`

即:

Beamer-Template 帧开始

在论文 (article) 模式中, 将该模板的文本插入到每一帧的开始处 (也只能在这里)。我们可以用它在帧的起始处开始一个 `minipage` 环境, 或插入一个水平条 (horizontal bar) 或任何东西 (whatever)。

Beamer-Template `frame end`

即:

Beamer-Template 帧结束

在 `article` 模式中, 将该模板的文本插入到每一帧的结束处。

我们可以在其它环境中使用帧环境, 像这样:

```
\newenvironment{slide}{\begin{frame}}{\end{frame}}
```

或像这样:

```
\newenvironment{myframe}[1]
  {\begin{frame}[fragile,environment=myframe]\frametitle{#1}}
  {\end{frame}}
```

然而, 真实的机制有几分敏感 (sensitive), 因为“聚集”帧内容不容易, 所以不要将事情想象得太好。通常, 环境的开始可以相当的任意 (pretty arbitrary), 但必须用 `\end{frame}` 结束, 而且不能包含任何 `\end{xxx}`。复杂一点的事情看起来会失败。如果我们要包含 `\end{xxx}`, 请定义一新命令包含该要素 (stuff), 如下所示:

```
\newenvironment{itemizeframe}
  {\begin{frame}\startitemizeframe}
  {\stopitemizeframe\end{frame}}

\newcommand\startitemizeframe{\begin{bfseries}\begin{itemize}}
\newcommand\stopitemizeframe{\end{itemize}\end{bfseries}}

\begin{itemizeframe}
\item First item
\end{itemizeframe}
```

8.2 帧的组成部分

每一帧由以下几个组成部分:

1. 一个顶部导航区 (headline) 和一个底部导航区 (footline)。
2. 一个左侧栏和一个右侧栏 (sidebar)。
3. 导航条 (navigation bar)。

4. 导航符 (navigation symbol)。
5. 一个徽标 (logo)。
6. 一个帧标题 (frame title)。
7. 背景 (background)。
8. 一些帧内容。

一个帧无需具备上述所有组成部分。通常，前面的三个组成部分由我们使用的主题自动建立。

8.2.1 顶部导航区和底部导航区

顶部导航区 (headline) 是帧的顶部区域。如果它不是空的，在这里可以显示一些信息，在演讲中帮助观众定位。同样，底部导航区 (footline) 是帧底部的区域。

BEAMER 使用标准的 L^AT_EX 机制排版顶部导航区和底部导航区，它们由特定的 headline 模板和 footline 模板排版。

顶部导航区和底部导航区尺寸的决定因素：宽度就是纸 (paper) 宽度；高度由 `\begin{document}` 命令之后的顶部导航区和底部导航区尝试性排版决定。顶部导航区和底部导航区的头部 (head) 那一点是“冻结的 (frozen)”并用于整个文档的各处，即使顶部导航区和底部导航区的高度在后来有所变化。

顶部导航区和底部导航区的外观由下列模板决定：

Beamer-Template/-Color/-Font headline

即：

Beamer-Template/-Color/-Font 顶部导航区

该模板用于排版顶部导航区。一开始就会安装 `BEAMER-color` 和 `-font headline`。默认情况下不使用 `BEAMER-color` 的背景 (background)，也就是说，在顶部导航区和底部导航区的后面不会绘制背景矩形 (background rectangle) (在以后的顶部导航区和底部导航区画布介绍中这一点会改变)。

顶部导航区的宽度是整个纸 (paper) 的宽度，高度由上述的机制自动决定。在垂直模式 (vertical mode) 中排版顶部导航区，关闭了行内跳跃 (interline skip) 并将段落跳跃 (paragraph skip) 设为 0。

在该模板中，命令 `\` 改变了，代之以插入一个逗号 (comma)。

举例：

```
\setbeamertemplate{headline}
{%
  \begin{beamercolorbox}{section in head/foot}
    \vskip2pt\insertnavigation{\paperwidth}\vskip2pt
  \end{beamercolorbox}%
}
```

下面的模板选项是预定好的：

- `[default]` 默认的顶部导航区是空的。在早期版本的 BEAMER 文档类中，要获得默认的顶部导航区，必需使用 `compatibility` 主题。
- `[infolines theme]` 如果加载了 `infolines` 外部主题²²，则该选项可用 (且被使用)。顶部导航区会显示当前的节 (section) 和小节 (subsection)。

²²实现方法：在导言区添加命令：`\useoutertheme{infolines}`

- `[miniframes theme]` 如果加载了 `miniframes` 外部主题，则该选项可用（且被使用）。顶部导航区会显示节（section），要节下面带有小的可点击的微帧（mini frames）。
- `[sidebar theme]` 如果加载了 `sidebar` 外部主题，并且顶部高度（head height）（和 `sidebar` 主题的选项）不是零，则该选项可用（且被使用）。这时，顶部导航区放置了背景色的帧标题（frametitle），左或右的侧栏放置了徽标（Logo）。
- `[smoothtree theme]` 如果加载了 `smoothtree` 外部主题，则该选项可用（且被使用）。在顶部导航区中显示一个“平滑的（smoothed）”导航树（navigation tree）。
- `[smoothbars theme]` 如果加载了 `smoothbars` 外部主题，则该选项可用（且被使用）。在顶部导航区中显示一个“平滑的（smoothed）”微帧（miniframes）。
- `[tree]` 如果加载了 `tree` 外部主题，则该选项可用（且被使用）。在顶部导航区中显示一个导航树（navigational tree）。
- `[split theme]` 如果加载了 `split` 外部主题，则该选项可用（且被使用）。顶部导航区被分成左右两部分，左部分显示节（sections），右部分显示子节（sections）。
- `[text line]{<text>}` 顶部导航区排版成普通的文本行，该文本行的内容是 `<text>`。左右两侧的页边距（margin）是一样的。`<text>` 排版在 `\hbox` 中，而顶部导航区以垂直模式（vertical mode）排版。

在该模板中，可以使用众多的插入物（inserts）²³：

- `\insertnavigation{<width>}` 在模板中插入一个宽度为 `<width>` 的水平导航条（navigation bar）。在这里罗列了节（sections），在它的下面是该节内的每一帧的微帧（mini frames）。
- `\insertpagenumber` 在模板中插入当前的页码（page number）²⁴。
- `\insertsection` 在模板中插入当前的节。
- `\insertsectionnavigation{<width>}` 插入一个包含所有节的垂直导航条（navigation bar），当前节高亮显示。
- `\insertsectionnavigationhorizontal{<width>}{<left insert>}{<right insert>}` 插入一个包含所有节的水平导航条（navigation bar），当前节高亮显示。在节的左边插入 `<left insert>`，在节的右边插入 `<right insert>`。通过插入一个三元的 fill（即一个 `fillll`），可以将导航条冲洗（flush）到左或右边。

举例：

```
\insertsectionnavigationhorizontal{.5\textwidth}{\hskip0pt plus1filll}{}
```

- `\insertshortauthor[<options>]` 在模板中插入作者的简写形式。文本将打印在一长行中，由 `\\` 命令产生的换行符被抑制。可能会给出下面的 `<options>`：
 - `width=<width>` 将文本放入一个给定尺寸的多行的微页（minipage）中。默认抑制换行符。
 - `center` 在由 `width` 选项创建的微页（minipage）内居中文本，而不是左对齐。
 - `respectlinebreaks` 使由 `\\` 命令产生的换行符有效。

举例：`\insertauthor[width={3cm},center,respectlinebreaks]`

²³在安装了 CTEX 套装（安装在 D 区）的 Windows 7 系统中，这些插入物的具体放置位置及放置方法可以在 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\outer\beamerouterthemeinfolines.sty` 文件中找到。例如，在该文件中就有“`\insertframenumbers / \inserttotalframenumbers`”这样的语句。

²⁴一张幻灯片就像图书的一页，其序号就叫页码。

- `\insertshortdate[options]` 在模板中插入日期的简写形式。可能会同时给出 `\insertshortauthor`。
- `\insertshortinstitute[options]` 在模板中插入大学的简写形式。可能会同时给出 `\insertshortauthor`。
- `\insertshortpart[options]` 在模板中插入系的简写形式。可能会同时给出 `\insertshortauthor`。
- `\insertshorttitle[options]` 在模板中插入文档标题 (document title) 的简写形式。可能会同时给出 `\insertshortauthor`。
- `\insertshortsubtitle[options]` 在模板中插入文档副标题 (document subtitle) 的简写形式。可能会同时给出 `\insertshortauthor`。
- `\insertsubsection` 在模板中插入当前的小节 (subsection)。
- `\insertsubsubsection` 在模板中插入当前的小小节 (subsection)。
- `\insertsubsectionnavigation{width}` 插入一个包含当前节的全部小节的导航条 (navigation bar), 当前的小节高亮显示。
- `\insertsubsectionnavigationhorizontal{width}{left insert}{right insert}` 请参考 `\insertsectionnavigation`。
- `\insertverticalnavigation{width}` 在模板中插入给定 *width* 的垂直导航条。该导航条显示一个小的目录。使用模板 `section in head/foot` 和 `subsection in head/foot` 排版该个性化的行 (lines)。
- `\insertframenumbers` 在模板中插入当前帧 (不是幻灯片) 的序号。
- `\inserttotalframenumbers` 在模板中插入帧 (不是幻灯片) 的总序号。该总序号只有在对文档第二次运行 $\text{T}_{\text{E}}\text{X}$ 时才是正确的。
- `\insertframestartpage` 插入当前帧的第一页的页码。
- `\insertframeendpage` 插入当前帧的最后一页的页码。
- `\insertsubsectionstartpage` 插入当前小节的第一页的页码。
- `\insertsubsectionendpage` 插入当前小节的最后一页的页码。
- `\insertsectionstartpage` 插入当前节的第一页的页码。
- `\insertsectionendpage` 插入当前节的最后一页的页码。
- `\insertpartstartpage` 插入当前部分 (part) 的第一页的页码。
- `\insertpartendpage` 插入当前部分 (part) 的最后一页的页码。
- `\insertpresentationstartpage` 插入演示稿的第一页的页码。
- `\insertpresentationendpage` 插入演示稿 (除外附录) 的最后一页的页码。
- `\insertappendixstartpage` 插入附录的第一页的页码。如果没有附录, 该页码就是文档的最后一页的页码。
- `\insertappendixendpage` 插入附录的最后一页的页码。如果没有附录, 该页码就是文档的最后一页的页码。
- `\insertdocumentstartpage` 插入 1。
- `\insertdocumentendpage` 插入文档 (包括附录) 的最后一页的页码。

即：

Beamer-Template/-Color/-Font 底部导航区

该模板的所作所为和顶部导航区是一样的。注意，有时比较烦人，BEAMER 会在帧文本和底部导航区之间添加 4pt 的间隙。

下面的模板选项是预定好的：

- `[default]` 默认为空的底部导航区。注意默认情况下导航符 (navigational symbols) 不是底部导航区的一部分。相反，它恰恰是 (看不见的) 右侧栏 (sidebar) 的一部分。
- `[infolines theme]` 如果加载了 `infolines` 外部主题，则该选项可用 (且被使用)。在底部导航区会显示作者姓名和演讲标题之类的东西。
- `[miniframes theme]` 如果加载了 `miniframes` 外部主题，则该选项可用 (且被使用)。当加载了 `miniframes` 主题，依据使用的选项不同，在底部导航区会显示不同的东西。
- `[page number]` 在底部导航区显示当前页码。
- `[frame number]` 在底部导航区显示当前帧码。
- `[split]` 如果加载了 `split` 外部主题，则该选项可用 (且被使用)。底部导航区 (就象顶部导航区一样) 被分成左右两部分，左部分作者姓名，右部分显示演讲的标题。
- `[text line]{<text>}` 底部导航区排版成普通的文本行，该文本行的内容是 `<text>`。左右两侧的页边距 (margin) 是一样的。`<text>` 排版在 `\hbox` 中，而顶部导航区以垂直模式 (vertical mode) 排版。在这样的行 (line) 中使用 `\strut` 命令也许是个好主意。

在底部导航区中可以使用和顶部导航区相同的插入物 (inserts)。

Beamer-Color/-Font page number in head/foot

即：

Beamer-Color/-Font 顶部/底部的页码

BEAMER-color 和 -font 用于在底部导航区中排版页码 (page number) 或帧码 (frame number)²⁵。

8.2.2 侧栏

侧栏 (sidebar) 是从顶部导航区的底部延伸到底部导航区的顶部的垂直区域。侧栏可以在左边，也可以在右边 (或两侧均有)。侧栏可以显示目录 (table of contents)，但添加侧栏也可以仅仅是为了好看。

安装一个 sidebar 模板时，我们必须明确地指定侧栏的水平尺寸，这可以通过使用带有 `sidebar left width` 选项或 `sidebar right width` 选项的 `\setbeamersize` 命令来实现。而侧栏的垂直尺寸可以被自动地算出。每一侧栏拥有它自己的背景画布 (background canvas)，使用侧栏画布 (sidebar canvas) 模板可以对背景画布进行设置。

要添加一给定尺寸如 1cm 的侧栏，将使主文本 (main text) 变窄 1cm。当装入侧栏时，侧栏内侧与文本外侧间的距离不会改变，该距离由带有 `text margin left` 选项的 `\setbeamersize` 命令指定。右侧的页边距 (margin) 也是如此。

²⁵一张幻灯片就像图书的一页，其序号就叫页码；一个帧环境产生一个帧，其序号就叫帧码，一个帧可以包含一张或多张幻灯片。

侧栏的内部排版机制是侧栏显示为顶部导航区 (headline) 的一部分。BEAMER 文档类记录了六个尺度 (dimensions)，每一幻灯片的三个尺度即：变量 `\beamer@leftsidebar` 和 `\beamer@rightmargin` 储存了侧栏的 (水平) 尺寸；变量 `\beamer@leftmargin` 和 `\beamer@rightmargin` 储存了侧栏与文本间的间距；宏指令 (macros) `\Gm@lmargin` 和 `\Gm@rmargin` 储存了纸 (paper) 边缘和文本边缘之间的间距。这样，`\beamer@leftsidebar` 和 `\beamer@leftmargin` 的总和就是 `\Gm@lmargin`。这样，如果想紧靠左侧栏放置某些文本，我们必需写入 `\hskip-\beamer@leftmargin`。

Beamer-Template/-Color/-Font sidebar left

即：

Beamer-Template/-Color/-Font 左侧栏

Color/font 父模板：sidebar

该模板用于排版左侧栏。如上所述，用下面的命令设置左侧栏的尺寸：

```
\setbeamersize{sidebar width left=2cm}
```

如果侧栏太大，BEAMER 不会自动裁剪侧栏。

在排版侧栏时，会将它放入一个 `\vbox` 中。我们必需随即设置好像 `\hsize` 或 `\parskip` 这样的东西。

下面的模板选项是预定好的：

- **[default]** 安装一空的模板。
- **[sidebar theme]** 如果加载了带 `left` 选项的 `sidebar` 外部主题，则该选项是可用的。这种情况下，会自动选定该选项。并在侧栏中显示一个微目录 (mini table of contents)。

Beamer-Template/-Color/-Font sidebar right

即：

Beamer-Template/-Color/-Font 右侧栏

Color/font 父模板：sidebar

该模板的工作方式和上述的左侧栏相同。

下面的模板选项是预定好的：

- **[default]** 默认的右侧栏的宽度为零。然而，会显示导航符 (navigational symbols)，还会在侧栏的底部显示徽标 (如果安装了的话)，并向左突入到文本中。
- **[sidebar theme]** 如果加载了带 `right` 选项的 `sidebar` 外部主题，则该选项是可用的。这种情况下，会自动选定该选项。并在侧栏中显示一个微目录 (mini table of contents)。

Beamer-Template sidebar canvas left

即：

Beamer-Template 左侧栏画布

像所有的背景画布 (background canvas) 一样，在侧栏实际的文本之后绘出该左侧栏画布。该模板通常会插入侧栏大小的矩形，该矩形的高度太高不会显示错误警告。当调用该模板时，将安装 `BEAMER-color sidebar left`。

下面的模板选项是预定好的：

- **[default]** 用 `sidebar.bg` 颜色的大矩形作为侧栏画布。然而，如果侧栏的背景是空的，则不会显示任何东西，即画布是“透明的 (transparent)”。
- **[vertical shading]** [*⟨color options⟩*] 安装垂直的阴影背景 (shaded Background)。将会给定下列 *⟨color options⟩* :
 - **top=⟨color⟩** 指定侧栏顶部的颜色。默认使用 25% 的 `BEAMER-color palette primary` 的前景色。
 - **bottom=⟨color⟩** 指定侧栏底部 (更精确地讲，侧栏顶部的下面的页面高度) 的颜色。默认情况下，在调用该命令时会使用普通文本 (`normal text`) 的背景。
 - **middle=⟨color⟩** 指定侧栏中部的颜色。因此，如给定该选项，阴影的颜色由底部的颜色改变成该颜色，然后改变成顶部的颜色。
 - **midpoint=⟨factor⟩** 指定在页面的哪一点使用中间色 (middle color)。因子 0 指定给页面底部，因子 1 指定给页面顶部。默认情况下，页面中间的因子为 0.5。

注意，在这里使用的颜色必须是“真正的”`LATEX` 颜色。在使用该命令之前，必需调用 `\usebeamercolor` 命令。

还应注意，在使用该选项之前，必须设定侧栏的宽度。

举例：流行但不实用的阴影：

```
{\usebeamercolor{palette primary}}
\setbeamertemplate{sidebar canvas}[vertical shading]
[top=palette primary.bg,middle=white,bottom=palette primary.bg]
```

- **[horizontal shading]** [*⟨color options⟩*] 安装水平的阴影背景 (shaded Background)。将会给定下列 *⟨color options⟩* :
 - **left=⟨color⟩** 指定侧栏左侧的颜色。
 - **right=⟨color⟩** 指定侧栏右侧的颜色。
 - **middle=⟨color⟩** 指定侧栏中部的颜色。
 - **midpoint=⟨factor⟩** 指定在侧栏的哪一点使用中间色 (middle color)。因子 0 指定给侧栏底部，因子 1 指定给侧栏顶部。默认情况下，侧栏中间的因子为 0.5。

举例：添加两个“柱子”

```
\setbeamersize{sidebar width left=0.5cm,sidebar width right=0.5cm}

{\usebeamercolor{sidebar}}

\setbeamertemplate{sidebar canvas left}[horizontal shading]
[left=white,middle=sidebar.bg,right=white]
\setbeamertemplate{sidebar canvas right}[horizontal shading]
[left=white,middle=sidebar.bg,right=white]
```

Beamer-Template `sidebar canvas right`

即：

Beamer-Template 右侧栏画布

该模板的工作方式和上述的左侧栏画布 (`sidebar canvas left`) 相同。

8.2.3 导航条

许多主题安装了顶部导航区 (headline) 或侧栏 (sidebar), 在此会显示一个导航条 (navigation bar)。虽然这些导航条会占用一些地方, 因为两个原因他们是有用的:

- 导航条可以为观众提供视觉反馈, 让观众知道我们的演讲隐藏了多少, 我们的演讲要指向哪里。没有这样的反馈, 观众将对我们当前所讲感到迷惑, 不管我们后来是否进行更详细的解释。
- 我们可以点击导航条的任何部分。这样可以直接“跳转”到点击所指向的目标处。这对于跳过演讲的某些部分很有用; 这对于在“问答”时, 我们跳回到某人所问的特定帧时是很有用的。

使用下面的选项可以“压缩”一些导航条:

```
\documentclass[compress]{beamer}
```

使所有导航条尽可能地小, 例如, 单个节的所有微帧会互相横靠地显示在导航条中。通常, 不同小节的微帧以不同的行显示。此外, 节和小节的导航被压缩成一行。

一些主题使用 `\insertnavigation` 命令在顶部导航区中插入导航条 (navigation bar)。在该导航条中, 会显示一些小图标 (称之为“微帧”), 这些小图标用于呈现演示稿的帧。当我们点击这样的小图标时, 会发生下面的事情:

- 如果我们点击 (小图标) 除当前帧以外的其它帧, 将跳到我们点击帧的第一张幻灯片处。
- 如果我们点击了当前帧但又不是在该帧的最后一张幻灯片处, 将跳到该帧的最后一张幻灯片处。
- 如果我们点击了当前帧而且又是在该帧的最后一张幻灯片处, 将跳到该帧的第一张幻灯片处。

按以上的规则, 我们可以:

- 点击小图标一次后会从其它某处跳到帧的开始处。
- 点击小图标两次后会从其它某处跳到帧的结束处。
- 点击小图一次后跳过当前帧的其余 (幻灯片)。

我们尝试跳转到一个已访问过的帧并自动跳到该帧的最后一张幻灯片处。然而, 结果却让人迷惑。双击后常将我们带到幻灯片的结束处, 不管我们从哪里“来”。

Parent Beamer-Template `mini frames`

即:

Parent Beamer-Template `微帧`

这个父模板具有子模板微帧 (mini frame) 和当前小节的微帧 (mini frame in current subsection)。

举例: `\setbeamertemplate{mini frames}[box]`

下面的模板选项是预定好的:

- `[default]` 微帧显示为小圆圈 (circles)。
- `[box]` 微帧显示为小矩形 (rectangles)。

- `[tick]` 微帧显示为小直条 (vertical bars)。

Beamer-Template/-Color/-Font `mini frame`

即:

Beamer-Template/-Color/-Font 微帧

该模板用于在导航条呈现 (render) 当前帧的微帧 (mini frame)。

模板的宽度被忽略。当显示多栏微帧 (multiple mini frames) 时, 它们的位置由 `BEAMER-sizes mini frame size` 和 `mini frame offset` 算出。请参考 `\setbeamersize` 命令以了解如何更改它们。

Beamer-Template `mini frame in current subsection`

即:

Beamer-Template 当前小节的微帧

该模板用于呈现当前小节的帧 (不是当前帧) 的微帧。在调用该模板之前, 先安装了 `BEAMER-color/-font mini frame`。

Beamer-Template `mini frame in other subsection`

即:

Beamer-Template 其它小节的微帧

该模板用于呈现除当前小节以外的小节的帧的微帧。

下面的模板选项是预定好的:

- `[default]` `[<percentage>]` 默认情况下, 该模板显示当前小节的微帧 (`mini frame in current subsection`), 只可惜颜色首先更改为 `fg!<percentage>!bg`。默认值是 50%。

举例: 当前小节以外的帧要呈现为极端地“阴影”, 我们可以使用:

```
\setbeamertemplate{mini frame in other subsection}[default][20]
```

举例: 除当前小节以外的小节的全部微帧要相同地显示, 可以使用:

```
\setbeamertemplate{mini frame in other subsection}[default][100]
```

一些主题会在导航条中显示节和/或小节。点击导航条上的节或小节, 将跳到相应的节或小节处。点击以 `\tableofcontents[currentsection]` 开始的节是很有用的, 因为这样做可以跳到不同的小节处。

Beamer-Template/-Color/-Font `section in head/foot`

即:

Beamer-Template/-Color/-Font 顶部/底部中的节

该模板用于呈现节的条目 (entry), 如果它们出现在顶部导航区或底部导航区。`BEAMER-color` 的背景 (background) 通常用作整个“区域”的背景, 该“区域”是指顶部导航区中显示节的条目的区域。我们不能经常使用该模板, 因为只有当在顶部导航区排版一个节列表时, 才能正确地设置 (setup) 插入的 `\insertsectionhead`。

默认的模板只插入节名 (section name)。该模板的下列插入物 (inserts) 很有用:

- `\insertsectionhead` 插入排版在导航条 (navigation bar) 中的节的名称。
- `\insertsectionheadnumber` 插入排版在导航条中的节的总数。
- `\insertpartheadnumber` 插入排版在导航条中的当前节或小节的总的部分数 (the number of the part)。

Beamer-Template `section in head/foot shaded`

即:

Beamer-Template 顶部/底部中的带阴影的节

该模板取代 `section in head/foot` 用于排版当前带阴影的节。该阴影常用于除当前节外的所有节。

注意, 该模板没有它自己的颜色和字体。当调用该模板时, 会设置 `BEAMER-font` 和 `color`、`section in head/foot`。然后, 在该模板的起始处, 我们可以改变当前的颜色或开始一个 `colormixin` 环境。

下面的模板选项是预定好的:

- `[default][<percentage>]` 默认模板将当前颜色更改为 `fg!<percentage>!bg`。这将导致当前颜色变得“被冲洗过 (washed out)”或“有阴影 (shaded)”。默认的百分比是 50。

举例: 我们可用下面的命令使带阴影的条目变得很“亮 (light)”:

```
\setbeamertheme{section in head/foot shaded}[default][20]
```

Beamer-Template/-Color/-Font `section in sidebar`

即:

Beamer-Template/-Color/-Font 侧栏中的节

该模板用于呈现出现在侧栏中的节的条目, 特别是作为微目录 (mini table of contents) 的一部分显示在这里。`BEAMER-color` 的背景 (background) 用作条目的背景。就像 `section in head/foot` 一样, 我们不能经常使用该模板, 必需使用 `\insertsectionhead` 插入要排版的节的名称。

仅此一次 (For once), 不会为该模板安装默认。

下面的模板选项是预定好的:

- `[sidebar theme]` 只有加载了 `sidebar` 外部主题后, 该模板才会插入一个 `BEAMER-color` 的前景色和背景色的条, 在该条中显示节名。该条的宽度和整个侧栏的宽度相同。

可以使用和 `section in head/foot` 相同的插入物 (inserts)。

Beamer-Template/-Color `section in sidebar shaded`

即:

Beamer-Template/-Color 侧栏中带阴影的节

该模板取代 `section in sidebar` 用于排版当前带阴影的节。该阴影常用于除当前节外的所有节。

和 `section in head/foot shaded` 不同, 该模板有它自己的 `BEAMER-color`。下面的模板选项是预定义好的:

下面的模板选项是预定好的:

- `[sidebar theme]` 除使用不同的 `BEAMER-color` 外，其它和无阴影的版本（nonshaded version）相同。

Beamer-Template `/-Color/-Font subsection in head/foot`

即：

Beamer-Template `/-Color/-Font` 顶部/底部中的小节

该模板的所作所为极像 `section in head/foot`，只不过是用于小节。

- `\insertsubsectionhead` works like `\insertsectionhead`.
- `\insertsubsectionheadnumber` works like `\insertsectionheadnumber`.

Beamer-Template `subsection in head/foot shaded`

即：

Beamer-Template 顶部/底部中的带阴影的小节

该模板的所作所为极像 `section in head/foot shaded`，只不过是用于小节。

下面的模板选项是预定好的：

- `[default]` `[<percentage>]` 所作所为像用于节（section）的相关选项。

举例：

```
\setbeamertemplate{section in head/foot shaded}[default][20]
\setbeamertemplate{subsection in head/foot shaded}[default][20]
```

Beamer-Template `/-Color/-Font subsection in sidebar`

即：

Beamer-Template `/-Color/-Font` 侧栏中的小节

该模板的所作所为极像 `section in sidebar`，只不过是用于小节。

Beamer-Template `subsection in sidebar shaded`

即：

Beamer-Template 侧栏中的带阴影的小节

该模板的所作所为极像 `section in sidebar shaded`，只不过是用于小节。

Beamer-Template `/-Color/-Font subsubsection in head/foot`

即：

Beamer-Template `/-Color/-Font` 顶部/底部中的小小节

该模板的所作所为极像 `section in head/foot`，只不过是用于小小节（subsubsections）。当前，它不用于默认的主题。

- `\insertsubsubsectionhead` works like `\insertsectionhead`.
- `\insertsubsubsectionheadnumber` works like `\insertsectionheadnumber`.

Beamer-Template `subsubsection in head/foot shaded`

即:

Beamer-Template 顶部/底部中的带阴影的小小节

该模板的所作所为极像 `section in head/foot shaded`，只不过是用于小小节。

下面的模板选项是预定好的:

- `[default]` [`<percentage>`] 所作所为像用于节的相关选项。

Beamer-Template/`-Color/-Font` `subsubsection in sidebar`

即:

Beamer-Template/`-Color/-Font` 侧栏中的小小节

该模板的所作所为极像 `section in sidebar`，只不过是用于小小节。

Beamer-Template `subsubsection in sidebar shaded`


即:









Beamer-Template 侧栏中的带阴影的小小节


该模板的所作所为极像 `section in sidebar shaded`，只不过是用于小小节。

点击导航条 (navigation bar) (并非所有的主题都会显示) 中的文档标题 (document title) 后, 将跳到演示稿的第一张幻灯片 (常常是封面) 处, 除非我们已经位于第一张幻灯片处。在第一张幻灯片中, 点击文档标题后, 会跳到演示稿的结束处。双击导航条中的文档标题, 会跳到演示稿的结束处。

8.2.4 导航符










导航符 (navigation symbols) 是默认显示在每一张幻灯片中的小图标即: .

1. 一个幻灯片图标 : 用单个的矩形表示。在它的左右两侧, 是向左和向右的箭头。
2. 一个帧图标 : 用三个“叠起来的”幻灯片图标表示。在它的左右两侧, 是向左和向右的箭头。
3. 一个小节图标 : 用在目录中高亮显示的小节条目表示。在它的左右两侧, 是向左和向右的箭头。
4. 一个节图标 : 它显示为在目录中高亮显示的节条目 (以及所有的小节)。在它的左右两侧, 是向左和向右的箭头。
5. 一个演示稿图标 : 它显示为一个完全高亮的目录。
6. 一个附录图标 : 它显示为一个完全高亮的只有一个节的目录。(仅当存在附录时才会显示该图示。)
7. 返回图标  和前进图标 : 显示为圆形箭头。

8. 一个“搜索”或“查找”图标 : 显示为一个侦探用的放大镜。

点击小图标左边的箭头, 会跳到 (最后一张幻灯片的) 前一张幻灯片、前一帧、前一小节。点击小图标右边的箭头, 会跳到 (第一张幻灯片的) 下一张幻灯片、下一帧、下一小节或下一节。

点击不同的小图标有不同的效应:

1. 如果浏览器支持, 点击幻灯片图标  后会弹出一个窗口, 在此我们可以输入我们想跳转到的幻灯片的序号。
2. 点击帧图标  的左侧将跳转到帧的第一张幻灯片处, 点击帧图标的右侧将跳转到帧的最后一张幻灯片外 (这对于 skipping 叠层很有用)。
3. 点击小节图标  的左侧将跳转到小节的第一张幻灯片处, 点击小节图标的右侧将跳转到小节的最后一张幻灯片处。
4. 点击节图标  的左侧将跳转到节的第一张幻灯片处, 点击节图标的右侧将跳转到节的最后一张幻灯片处。
5. 点击演示稿图标  的左侧将跳转到演示稿的第一张幻灯片处, 点击演示稿图标的右侧将跳转到演示稿的最后一张幻灯片处。然而, 这不包括附录。
6. 点击附录图标  的左侧将跳转到附录的第一张幻灯片处, 点击附录图标的右侧将跳转到附录的最后一张幻灯片处。
7. 如果浏览器支持, 点击返回图标  将跳转到前一访问过的幻灯片处; 点击前进图标 , 将跳转到下一幻灯片处。
8. 如果浏览器支持, 点击“搜索”或“查找”图标  后会弹出一个窗口, 在此可以输入要查找的字串 (string), 如果查找到了, 浏览器会跳转到找到的字串处。

通过调整导航符 (navigation symbols) 模板, 我们可以减少小图标的显示及改变它们的布局。

Beamer-Template/-Color/-Font navigation symbols

即:

Beamer-Template/-Color/-Font 导航符

在显示导航符的地方由主题 (themes) 以“三星模式 (three-star-mode)”调用该模板。“三星模式”是指使用 `\usebeamertemplate***` 命令。

注意, 虽然看起来这些导航符是底部导航区 (footline) 的一部分, 其实它们是可见的右侧栏 (right sidebar) 的一部分。

下面的模板选项是预定好的:

- `[default]` 让导航符水平显示。
- `[horizontal]` 这是默认 (default) 的别名, 即和默认等效。
- `[vertical]` 让导航符垂直显示²⁶。










²⁶相应的语句是: `\setbeamertemplate{navigation symbols}[vertical]`

- `[only frame symbol]` 只为导航帧 (navigating frames) 显示导航符。

举例：下面的命令将抑制（即不显示）导航符：

```
\setbeamertemplate{navigation symbols}{}
```

在该模板中，下面的插入物 (inserts) ²⁷ 是可用的：

- `\insertslidenavigationsymbol` 插入幻灯片图标，也就是说，幻灯片图标 （一个矩形）及左右两侧的箭头带有超链接。
- `\insertframenavigationsymbol` 插入帧图标 .
- `\insertsubsectionnavigationsymbol` 插入小节图标 .
- `\insertsectionnavigationsymbol` 插入节图标 .
- `\insertdocnavigationsymbol` 插入演示稿图标  和（如果需要）附录图标 .
- `\insertbackfindforwardnavigationsymbol` 插入返回图标 、查找图标 、前进图标 .

8.2.5 徽标

要安装一个徽标 (logo)，请使下面的命令：

```
\logo{<logo text>}
```

即：

```
\logo{<徽标文本>}
```

`<logo text>` 通常是一个用于插入图形的命令，但它可以是任何文本。插入徽标的位置由当前的主题判定，目前我们不能直接指定这个位置。

举例：

```
\pgfdeclareimage[height=0.5cm]{logo}{tu-logo}
\logo{\pgfuseimage{logo}}
```

举例：

```
\logo{\includegraphics[height=0.5cm]{logo.pdf}}
```

目前，该命令的作用只是用于设置徽标 (logo) 模板。然而，在将来可能有更复杂的作用。

ARTICLE 该命令无效。

Beamer-Template/-Color/-Font logo


即：

Beamer-Template/-Color/-Font 徽标

该模板用于呈现徽标。

下面的插入物可用于在某处插入一个徽标：

- `\insertlogo` 在当前的位置插入徽标。该命令和 `\usebeamertemplate*{logo}` 等效。

²⁷例如：在用 BEAMER 做成的幻灯片中，要显示“这就是返回、查找、前进图标 ”，则在源文件中其相应的语句是“这就是返回、查找、前进图标 `\insertbackfindforwardnavigationsymbol`”。

8.2.6 帧标题

帧标题 (frame title) 非常显眼地显示在帧的顶部, 它可以用下面的命令指定:

```
\frametitle<<overlay specification>>[<short frame title>]{<frame title text>}
```

即:

```
\frametitle<<叠层规则>>[<帧标题简写形式>]{<帧标题文本>}
```

如果帧标题是一个完整的句子, 我们可以用句号结束 $\langle frame\ title\ text \rangle$, 否则不可以用句号。常常不会显示 $\langle short\ frame\ title \rangle$, 但通过 `\insertshortframetitle` 命令, 它则变得可用。在论文 (article) 模式中, $\langle overlay\ specification \rangle$ 常用于抑制帧标题。

举例:

```
\begin{frame}
  \frametitle{A Frame Title is Important.}
  \framesubtitle{Subtitles are not so important.}

  Frame contents.
\end{frame}
```

如果在当前帧使用了 `allowframebreaks` 选项, 则会自动在 $\langle frame\ title\ text \rangle$ 的末尾根据 `frametitle continuation` 模板添加连续 (continuation) 的文本如 “(cont.)” 或类似的东西, $\langle frame\ title\ text \rangle$ 与连续文本之间用空白 (space) 分开。

PRESENTATION 当遇到 `\frametitle` 命令时不会立即排版帧标题。当然, 该命令的参数存贮在内部, 仅当读入完整的帧 (the complete frame) 时才会排版帧标题。这样我们就可以访问 $\langle frame\ title\ text \rangle$ 和 $\langle subframe\ title\ text \rangle$, $\langle subframe\ title\ text \rangle$ 由 `\framesubtitle` 命令引入。

ARTICLE 默认情况下, 在论文 (article) 模式中, 该命令创建一新段落, 其标题是 $\langle frame\ title\ text \rangle$ 。有时使用 $\langle overlay\ specification \rangle$ 可以非常容易地抑制帧标题。在论文 (article) 模式中如果我们想抑制所有的帧标题, 可以声明: `\setbeamertemplate<article>{\frametitle}{}`。

LYX 帧标题是跟在 “BeginFrame” 样式这一行后面的文本。

Beamer-Template/-Color/-Font frametitle

即:

Beamer-Template/-Color/-Font 帧标题

Color/font 父模板: `titlelike`

当排版帧标题和子标题 (subtitle) 时, 会调用该模板以及 `BEAMER-color` 和 `-font frametitle` 设置。当调用 `\frametitle` 或 `\framesubtitle` 命令时, 还不会调用该模板 (如下所述)。当然啦, 当读入完整的帧时, 则会调用该模板。直到这时, 帧标题 (frame title) 和帧子标题 (frame subtitle) 存贮在一个特别的地方。当调用该模板时, 就可以正确地创建帧标题和帧子标题。结果是, \TeX -盒子就神奇般地放在了帧顶部的后面。

下面的模板选项是预定好的:

- `[default]` [`<alignment>`] 用 `BEAMER-color frametitle` 和 `BEAMER-font frametitle` 排版帧。用 `color` 和 `font framesubtitle` 排版子标题，并将它放在下面。如果 `color frametitle` 有背景，拉伸了整个帧宽度的背景条（Background bar）将放在标题的后面。会忽略子标题的背景色。`<alignment>` 传递给了 `beamercolorbox` 环境。有用的选项是左对齐（`left`）、居中（`center`）、右对齐（`right`）。作为一种特殊情况，右对齐（`right`）选项会使帧标题的左边界比正常的更大一点，所以帧标题更多的是置于帧的中间。
- `[shadow theme]` 如果加载了 `shadow` 外部主题，则该选项可用，它在水平阴影（horizontal shading）的顶部绘制帧标题，水平阴影处于 `frametitle` 背景色和 `frametitle right` 之间。如果有子标题（`subtitle`），也会放在这个条上。在这个条下面，绘制“阴影（shadow）”。
- `[sidebar theme]` 如果加载了 `sidebar` 外部主题，并且顶部导航区（`headline`）的高度不是设为 `0pt`（可以使用 `sidebar` 主题的一个选项将它设为 `0pt`），则该选项可用。使用该选项后，帧标题将放在一个矩形区域中，该矩形区域是顶部导航区的一部分 [用一些“暗间隙（negative space）”将帧标题升高到该区域中]。不会使用颜色 `frametitle` 的背景，而这是顶部导航区模板（`headline template`）的工作。
- `[smoothbars theme]` 如果加载了 `smoothbars` 外部主题，则该选项可用。它在 `frametitle` 背景色的有色条中排版帧标题。条的顶部和底部与背景的上和下面平滑地熔合在一起。
- `[smoothtree theme]` 和 `smoothbars theme` 相似，只用于 `smoothtree` 主题。

下面的命令可用于该模板：

- `\insertframetitle` 产生帧标题。
- `\insertframesubtitle` 产生帧子标题。

`\framesubtitle<<overlay specification>>{<frame subtitle text>}`

即：

`\framesubtitle<<叠层规则>>{<帧子标题文本>}`

如果有，将在主标题（`main title`）的下面以更小的字体显示子标题（`subtitl`）。正如 `\frametitle` 命令，可以在帧的任意位置给出该命令，因为，实际上在所有其它的东西排版完成后才排版帧标题。

举例：

```
\begin{frame}
  \frametitle<presentation>{Frame Title Should Be in Uppercase.}
  \framesubtitle{Subtitles can be in lowercase if they are full sentences.}

  Frame contents.
\end{frame}
```

ARTICLE 默认情况下，在论文（`article`）模式中，任何时候都不会显示子标题。

Beamer-Color/-Font framesubtitle

即：

Beamer-Color/-Font 帧子标题

Color/font 父模板: `frametitle`

该元素 (element) 为子标题提供了颜色和字体, 但没有模板。 `frametitle` 模板也可用于排版子标题。

默认情况下, 幻灯片的所有材料 (material) 均垂直居中。可以使用下面的文档类选项改变这种对齐方式:

```
\documentclass[t]{beamer}
```

将幻灯片的文本 (垂直地) 放在幻灯片的顶部。这和垂直 “平齐 (flush)” 相类似。可以在个别帧中用 `c` 选项或 `b` 选项覆盖 `t` 选项。

```
\documentclass[c]{beamer}
```

将幻灯片的文本 (垂直地) 放在幻灯片的中央, 这是默认的。可以在个别帧中用 `t` 选项或 `b` 选项覆盖 `c` 选项。

8.2.7 背景

每一帧均有一背景 (*background*), 正如其名 -它位于 “所有东西之后”。背景是一令人惊讶的复杂的对象 (object): 在 BEAMER 中, 它由背景画布 (*background canvas*) 和主背景 (*main background*) 组成。

背景画布可以想象成是一个巨大的区域, 可以在其中放置任何东西 (主背景和其它东西)。默认情况下, 该画布是一个用帧填充的大的矩形, 帧的颜色就是 `BEAMER-color background canvas` 的背景。因为该颜色继承自普通文本 (`normal text`), 通过更改普通文本的颜色, 就可以改变画布的颜色。

举例: 下面的命令将背景色更改成亮红色:

```
\setbeamercolor{normal text}{bg=red!20}
```

画布不一定是单色的 (monochrome), 它可以有阴影或是透明的 (transparent)。如果我们想在其它文档中包含幻灯片, 将画布设置成透明是个好主意。

举例: 下面的命令将背景设置成透明:

```
\setbeamercolor{background canvas}{bg=}
```

Beamer-Template/-Color/-Font background canvas

即:

Beamer-Template/-Color/-Font 背景画布

Color 父模板: `normal text`

该模板插入在 “所有东西的后面”。该模板可以是某些用于生成高度为 `\paperheight` 宽度为 `\paperwidth` 的矩形的 \TeX 命令。

下面的模板选项是预定好的:

- `[default]` 安装一个颜色为背景色的矩形。如果背景是空的, 则画布是 “透明” 的。因为背景画布 (`background canvas`) 继承自普通文本 (`normal text`), 我们可以更改 `BEAMER-color normal text` 的背景从而改变默认画布的颜色。然而, 要使画布透明, 只能将画布的背景 (background of the canvas) 设为空 (`empty`), 将普通文本的背景设为白色 (`white`)。

- `[vertical shading]` `[<color options>]` 安装一垂直的阴影背景。小心使用：背景阴影（*background shadings*）常分散人的注意力！可能会给出下面的 `<color options>`：
 - `top=<color>` 指定页面顶部的颜色。默认情况下，使用 25% BEAMER-color palette primary 的前景色。
 - `bottom=<color>` 指定页面底部的颜色。默认情况下，在调用该命令时使用普通文本（normal text）的背景。
 - `middle=<color>` 指定页面中间的颜色。如果给定该选项，阴影将从底部的颜色变换到该颜色然后再变成顶部的颜色。
 - `midpoint=<factor>` 指定在页面的哪一点使用中间色（middle color）。页面底的因子为 0，页面顶部的因子为 1。默认情况下，页面中间的因子为 0.5。

主背景（main background）绘制于背景画布（background canvas）的上面。主背景可用于在每一帧上或大的背景图片上或其它任何东西上添加栅格（grid）。如果我们准备将一 PNG 图片用作背景图片（background image）²⁸，该图片必须带有 alpha 通道，这样可以避免在某些 PDF 浏览器中因透明带来的潜在的问题

Beamer-Template/-Color/-Font background

即：

Beamer-Template/-Color/-Font 背景

Color 父模板：background canvas

该模板插入在“所有东西的后面，但在背景画布的上面”。该模板用于无需填满整个背景的东西如图片、栅格或其它东西。当排版该模板时，将建立（setup）BEAMER-color 和 -font background。

下面的模板选项是预定好的：

- `[default]` 是空（empty）。
- `[grid]` `[<grid options>]` 在背景上放置栅格。可能会给出下面的 `<grid options>`：
 - `step=<dimension>` 指定栅格线间的距离，默认是 0.5cm。
 - `color=<color>` 指定栅格线的颜色，默认是 10% 的前景色。

8.3 帧尺寸和页边距

帧的尺寸（size of a frame）其实就是 BEAMER 演示稿的“页面尺寸（paper size）”，它是可变的。默认情况下，帧的尺寸是 128mm×96mm。该尺寸的长宽比（aspect ratio）是 4:3。演示稿程序（如 `acroread`、`xpdf`、`okular` 或 `evince`）全屏显示幻灯片时的尺寸就是 4:3。使用小的“页面尺寸”的好处是我们可以使用普通字

²⁸ BEAMER 设置某一帧的背景图片（background image）的方法有二：

方法①：在导言区添加 `\setbeamercolor{background canvas}{bg=}` 命令（必须加上这句，以防背景包括图片被其它东西覆盖），然后在 frame 环境中添加 `\ThisTileWallPaper{paperwidth}{paperheight}{bg.jpg}` 命令。其中 `bg.jpg` 为背景图片文件名。

方法②：其实这也是添加水印的一种方法。其中 `structure.png` 为背景图片文件名。

```
{
\usebackgroundtemplate{\includegraphics[width=\paperwidth,height=\paperheight]{structure.png}}
\begin{frame}{test for background image} see? \end{frame}
}
```

体 (normal fonts) 的真实尺寸 (natural sizes)。特别是, 插入一带有 11pt 标签的图形, 在演讲期间, 标签的大小是适当的。

要更改“页面尺寸”和长宽比, 可以使用下面的文档类选项:

```
\documentclass[aspectratio=1610]{beamer}
```

将长宽比设为 16:10, 帧尺寸设为 160mm×100mm。

```
\documentclass[aspectratio=169]{beamer}
```

将长宽比设为 16:9, 帧尺寸设为 160mm×90mm。

```
\documentclass[aspectratio=149]{beamer}
```

将长宽比设为 14:9, 帧尺寸设为 140mm×90mm。

```
\documentclass[aspectratio=54]{beamer}
```

将长宽比设为 5:4, 帧尺寸设为 125mm×100mm。

```
\documentclass[aspectratio=43]{beamer}
```

默认的长宽比和帧尺寸, 我们无需指定该选项。

```
\documentclass[aspectratio=32]{beamer}
```

将长宽比设为 3:2, 帧尺寸设为 135mm×90mm。

除外使用这些选项, 我们应该克制而尽量不使用 (refrain from) 更改“页面尺寸”。然而, 我们可以改变左右两侧的页边距 (margins), 默认情况下, 左右两侧的页边距为 1cm。要更改它们, 我们可以使用下面的命令:

```
\setbeamersize{<options>}
```

即:

```
\setbeamersize{<选项>}
```

可能会给出下面的 *<options>*:

- `text margin left=<TEX dimension>` 设定新的左侧页边距 (left margin)。它不包含左侧栏 (left sidebar)。因此, 左侧页边距就是左侧栏的右缘和文本的左缘之间的距离。
- `text margin right=<TEX dimension>` 设定新的右侧页边距。
- `sidebar width left=<TEX dimension>` 设定左侧栏 (left sidebar) 的尺寸。目前, 在为侧栏画布安装阴影之前给出该选项。
- `description width=<TEX dimension>` 设定描述标签 (description labels) 的默认宽度。请参考第 12.1 节。
- `description width of=<text>` 将描述标签的默认宽度设定为 *<text>* 的宽度, 请参考第 12.1 节。
- `description width of=<text>` sets the default width of description labels to the width of the *<text>*, see Section 12.1.

- `mini frame size=<TeX dimension>` 设定导航条 (navigation bar) 上微帧的尺寸。当两个微帧图标相互横靠地显示时, 它们的左末端 (left end points) 是 $\langle TeX dimension \rangle$ 相距甚远。
- `mini frame offset=<TeX dimension>` 设定一个额外的垂直偏移 (offset), 当微帧垂直排列时, 将它添加到微帧尺寸上。

ARTICLE 在论文 (article) 模式中, 该命令无效。

8.4 一帧的幻灯片数量的限制

一帧的幻灯片的数量会被自动地计算出来。如果帧的叠层规则 (overlay specification) 提到的最大数值是 4, 则引入 (introduce) 4 张幻灯片 (尽管事实上像 `<4->` 这样的叠层规则说可能超过 4 张幻灯片)。

我们也可以“手工”指定帧的幻灯片的数量。要变样做, 我们必须给 `\frame` 命令传递一个叠层规则。帧将包含由该参数中指定的幻灯片数。看看下面的例子:

```
\begin{frame}<1-2,4->
  This is slide number \only<1>{1}\only<2>{2}\only<3>{3}%
  \only<4>{4}\only<5>{5}.
\end{frame}
```

该命令将创建一个包含四张幻灯片的帧。第一张幻灯片包含文本 “This is slide number 1,”, 第二张幻灯片包含文本 “This is slide number 2,”, 第三张幻灯片包含文本 “This is slide number 4,”, 第四张幻灯片包含文本 “This is slide number 5.”。

一个有用的规则是 `<0>`, 它使帧根本就不包含幻灯片。例如, `\begin{frame}<handout:0>` 可以在讲义 (handout) 版本中抑制帧, 但在其它版本中会正常地显示帧。另一有用的规则是 `<beamer>`, 它可以使帧在 `beamer` 模式中正常地显示, 而在其它版本中被抑制。

9 创建叠层

9.1 暂停命令

暂停命令（`pause command`）提供了一个简易但不易变通（flexible）的分段显示帧的方法。如果在帧的某处放置 `\pause` 命令，只有 `\pause` 命令之前的文本会在该帧的第一张幻灯片中显示。第二张幻灯片中的内容一直显示到第二个 `\pause` 命令出现时，等等。我们也可以在环境内部使用 `\pause` 命令，其作用持续到环境以后。然而，在极端的情况下和在嵌套环境的深层放置 `\pause` 命令可能达不到预期的结果。

更多的关于在每一幻灯片中显示什么可以用叠层规则（overlay specifications）控制，请参阅下一节。然而，对于很多简单的情形，`\pause` 命令足以应对。

`\pause` 命令的作用持续到下一 `\pause`、`\onslide` 命令处，或持续到帧结束处。

```
\begin{frame}
  \begin{itemize}
    \item
      Shown from first slide on.
    \pause
    \item
      Shown from second slide on.
    \begin{itemize}
      \item
        Shown from second slide on.
      \pause
      \item
        Shown from third slide on.
    \end{itemize}
    \item
      Shown from third slide on.
    \pause
    \item
      Shown from fourth slide on.
    \end{itemize}

    Shown from fourth slide on.

    \begin{itemize}
      \onslide
      \item
        Shown from first slide on.
      \pause
      \item
        Shown from fifth slide on.
    \end{itemize}
  \end{frame}
```

`\pause` [*number*]

即：

`\pause`[(序号)]

该命令使跟随其后的文本从下一张幻灯片开始显示，或者，如果给定了可选的 $\langle number \rangle$ ，则从第 $\langle number \rangle$ 张幻灯片开始显示。如果给定了可选的 $\langle number \rangle$ ，会将计数器 `beamerpauses` 设置成该数字。该命令在内部使用 `\onslide` 命令。该命令在 `amsmath` 环境如 `align` 内部无效，因为这些环境会做一些坏事（wicked things）。

举例：

```
\begin{frame}
  \begin{itemize}
    \item
      A
    \pause
    \item
      B
    \pause
    \item
      C
  \end{itemize}
\end{frame}
```

ARTICLE 该命令会被忽略。

LYX 使用带有一空行的“Pause”风格插入一个暂停（pause）。

要“解除暂停（unpause）”某些文本，也就是说，要临时推迟暂停，可以使用 `\onslide` 命令，请参阅下面。

9.2 叠层规则的一般概念

大多数演示稿类（presentation classes）叠层所采用的方法和上述的 `\pause` 命令有几分相似。这些命令以确定的幻灯片序号作为输入，并以特定的方式影响跟随该命令的文本。例如，PROSPER的 `\FromSlide{2}` 命令使跟随其后的所有文本从第 2 张幻灯片开始显示。

BEAMER 文档类使用不同的方法（虽然前述的命令仍有用：`\onslide<2->` 和 `\FromSlide{2}` 具有相同的效果，只可惜 `\onslide` 超出了环境；同样，`\pause` 命令在内部映射一条带有叠层规则的命令）。一种办法就是将叠层规则添加到特定的命令。这些规则常常放置于尖括号（pointed brackets）内并“尽可能”跟在命令之后，虽然某些特定情形中，BEAMER 允许在稍后的地方给出叠层规则。最简单的情形是，叠层规则仅包含一个数字。后跟有叠层规则的命令只“影响（effect）”叠层规则指定的幻灯片。“影响”的具体含义因命令不同而不同。请仔细看看下面的例子：

```
\begin{frame}
  \textbf{This line is bold on all three slides.}
  \textbf<2>{This line is bold only on the second slide.}
  \textbf<3>{This line is bold only on the third slide.}
\end{frame}
```


对于 `\textbf` 命令，叠层规则只将指定幻灯片中的文本设置成粗体（黑体，boldface）。而其它幻灯片中的文本为普通字体（normal font）。

对于第二个例子，仔细看看后面的帧：

```
\begin{frame}
  \only<1>{This line is inserted only on slide 1.}
  \only<2>{This line is inserted only on slide 2.}
\end{frame}
```

由 BEAMER 引进的 `\only` 命令，通常只是将其参数插入到当前帧。然而，如果提供了叠层规则，则在没有提及的幻灯片中它会“扔掉（throws away）”其参数。

叠层规则只能写在特定的（而不每个）命令的后面。哪些命令后面能跟叠层规则以及会产生什么样的效果，将在下一节阐述。然而，重定义一个现有的命令非常容易，使得“叠层规则敏感（叠层规则能察觉，overlay specification aware）”也就变得很容易。请参阅第 9.3 节。

叠层规则（基本）的语法如下：叠层规则是用逗号分隔（comma-separated）的幻灯片列表和范围。范围的格式是：2-5，其含义是从第 2 张幻灯片到第 5 张幻灯片。可以省略范围的开始或结束，例如：3- 的含义是“第 3、4、5、…张幻灯片”；-5 的含义和 1-5 相同。一个复杂的例子：-3, 6-8, 10, 12-15，其含义是第 1、2、3、6、7、8、10、12、13、14、15 张幻灯片。

LYX 在 LyX 中也可以指定叠层规则。必须在 TeX-模式中给出（不同的是尖括号会被 LyX “避开”，但在所有版本中会出现这种情况）。例如，要给一个条目（item）添加一条叠层规则，只需插入像 `<3>` 这样的 TEX-模式文本作为该条目的第一个东西（first thing）。同样，在 TeX-模式中，我们可以在像 `theorem` 这样的环境开始处添加叠层规则。

9.3 能跟叠层规则的命令

commands documented here are *all* fragile even if the L^AT_EX 2_ε kernel versions are not. **重要：** 由于实现叠层规则方式的原因，这里的命令都是脆弱的，即使 L^AT_EX 2_ε 内核（kernel）不是脆弱的。

对于下面的命令²⁹，添加叠层规则后会忽略叠层规则未包含的幻灯片：`\textbf`、`\textit`、`\textsl`、`\textrm`、`\textsf`、`\color`、`\alert`、`\structure`。如果一条命令如 `\color` 带有多个参数，叠层规则必须直接跟在该命令的后面，如下例所示（但该规则存在例外的情况）：

```
\begin{frame}
  \color<2-3>[rgb]{1,0,0} This text is red on slides 2 and 3, otherwise black.
\end{frame}
```

对于下面的命令，叠层规则的效果有点特别：

```
\onslide<modifier><overlay specification>{<text>}
```

即：

```
\onslide<修饰符><叠层规则>{<文本>}
```

该命令的行为（behavior）据是否给定了可选的 `<text>` 参数（注意该可选的参数是放在普通的大括号中而不是放在方括号中）而不同。如果给定了 `<text>` 参数，则 `<modifier>` 可以是 `a +` 或 `a *`。

²⁹这些命令的作用是：`\textbf`：粗宽序列；`\textit`：斜体形状；`\textsl`：倾斜形状；`\textrm`：罗马体字族；`\textsf`：等线体字族；`\color`：颜色；`\alert`：高亮显示；`\structure`：标记一个东西为结构的一部分。

如果没有给定 $\langle text \rangle$ ，将发生下面的事情：跟在该命令之后的所有文本只会显示（显露）于指定的幻灯片中。在非指定（non-specified）的幻灯片中，文本仍占据了空间（space）。如果没有指定幻灯片，后面的文本会一直处于显示状态。在相同的 T_EX group 中，无需调用该命令，它的效力（effect）优先于（transcends）block groups。然而，该命令在一个 `overprint` 环境内具有不同的效力，请参阅 `overprint` 的说明。

如果 $\langle modifier \rangle$ 是 +，隐藏的（hidden）文本将不按 `covered` 处置，将按不可见（invisible）处理。不同之处与 `\uncover` 和 `\visible` 之间的不同相同。如果没有给定 $\langle text \rangle$ 参数则不能给出 `modifier *`。

举例：

```
\begin{frame}
  Shown on first slide.
  \onslide<2-3>
  Shown on second and third slide.
  \begin{itemize}
  \item
    Still shown on the second and third slide.
  \onslide+<4->
  \item
    Shown from slide 4 on.
  \end{itemize}
  Shown from slide 4 on.
  \onslide
  Shown on all slides.
\end{frame}
```

如果给定了 $\langle text \rangle$ 参数，`\onslide`（不带有 $\langle modifier \rangle$ ）会映射成（is mapped to）`\uncover`，`\onslide+` 会映射成 `\visible`，`\onslide*` 会映射成 `\only`。

举例：

```
\begin{frame}
  \onslide<1>{Same effect as the following command.}
  \uncover<1>{Same effect as the previous command.}

  \onslide+<2>{Same effect as the following command.}
  \visible<2>{Same effect as the previous command.}

  \onslide*<3>{Same effect as the following command.}
  \only<3>{Same effect as the previous command.}
\end{frame}
```

`\only<overlay specification>{<text>}<overlay specification>`

即：

`\only<叠层规则>{<文本>}<叠层规则>`

如果给定了 $\langle overlay specification \rangle$ （虽然只给定了一个）， $\langle text \rangle$ 将被插入到指定的幻灯片中。在其它幻灯片中则会扔掉（throw away）这些文本。特别之处，它不会占用空间。

举例：`\only<3->{Text inserted from slide 3 on.}`

因为可以在文本之后给出叠层规则，所以可以经常使用 `\only` 命令以一种简单的方式使其它命令变得叠层规则敏感（overlay specification-aware）：

举例：

```
\newcommand{\myblue}{\only{\color{blue}}}  
\begin{frame}  
  \myblue<2> This text is blue only on slide 2.  
\end{frame}
```

`\uncover<overlay specification>{<text>}`

即：

`\uncover<叠层规则>{<文本>}`

如果给定了 `<overlay specification>`，则 `<text>` 只会在指定的幻灯片中显示（show）[“揭开（uncovered）”]。在其它幻灯片，文本仍占据空间并被排版出来，但它不会显示或仿佛是透明（transparent）的。如何指定文本不可见（invisible）或透明请参考第 17.6 节。

举例：`\uncover<3->{Text shown from slide 3 on.}`

ARTICLE 该命令的效果和 `\only` 相同。

`\visible<overlay specification>{<text>}`

即：

`\visible<叠层规则>{<文本>}`

该命令的所作所为大部分和 `\uncover` 相同。不同之处是不会显示文本（show text），从不以透明的（transparent）方式显示，根本就不显示。因此，透明设置（transparency settings）对该命令无效

举例：`\visible<2->{Text shown from slide 2 on.}`

ARTICLE 该命令的效果和 `\only` 相同。

`\invisible<overlay specification>{<text>}`

即：

`\invisible<叠层规则>{<文本>}`

该命令的效果和 `\visible` 相反。

举例：`\invisible<-2>{Text shown from slide 3 on.}`

`\alt<overlay specification>{<default text>}{<alternative text><overlay specification>}`

即：

`\alt<⟨叠层规则⟩>{⟨默认文本⟩}{⟨提醒文本⟩}<⟨叠层规则⟩>`

只能给定一个 *⟨overlay specification⟩*。只在指定的幻灯片中显示 default text（默认的文本），在其它幻灯片中显示 alternative text（提醒文本）。常常必须给定叠层规则。

举例：`\alt<2>{On Slide 2}{Not on slide 2.}`

当在另一条命令内使用该命令时，在结束处再一次给定叠层规则是很有用的。

举例：这是 `\uncover` 的定义（definition）：

```
\newcommand{\uncover}{\alt{\@firstofone}{\makeinvisible}}
```

`\temporal<⟨overlay specification⟩>{⟨before slide text⟩}{⟨default text⟩}{⟨after slide text⟩}`

即：

`\temporal<⟨叠层规则⟩>{⟨幻灯片前文本⟩}{⟨默认文本⟩}{⟨幻灯片后文本⟩}`

该命令据当前幻灯片：是暂时处于指定的幻灯片之前、是指定的幻灯片之一、紧跟在指定的幻灯片之后这三种不同的情况而在三种不同的文本之间选择。如果 *⟨overlay specification⟩* 不是一个区间（interval）[也就是说，如果它有一个“洞（hole）”]，则该“洞（hole）”被认为是指定幻灯片之前的一部分。

举例：

```
\temporal<3-4>{Shown on 1, 2}{Shown on 3, 4}{Shown 5, 6, 7, ...}
\temporal<3,5>{Shown on 1, 2, 4}{Shown on 3, 5}{Shown 6, 7, 8, ...}
```

下面的例子是 `\temporal` 命令的适应情形：

举例：

```
\def\colorize<#1>{%
  \temporal<#1>{\color{red!50}}{\color{black}}{\color{black!50}}

\begin{frame}
  \begin{itemize}
    \colorize<1> \item First item.
    \colorize<2> \item Second item.
    \colorize<3> \item Third item.
    \colorize<4> \item Fourth item.
  \end{itemize}
\end{frame}
```

`\item<⟨alert specification⟩>[⟨item label⟩]<⟨alert specification⟩>`

即：

`\item<⟨提醒规则⟩>[⟨条目标签⟩]<⟨提醒规则⟩>`

PRESENTATION 可能只给出一个 *⟨alert specification⟩*。第 9.6.3 节阐述了 *⟨alert specification⟩* 的效果。

举例：

```
\begin{frame}
```

```

\begin{itemize}
\item<1-> First point, shown on all slides.
\item<2-> Second point, shown on slide 2 and later.
\item<2-> Third point, also shown on slide 2 and later.
\item<3-> Fourth point, shown on slide 3.
\end{itemize}
\end{frame}

\begin{frame}
\begin{enumerate}
\item<3-| alert@3>[0.] A zeroth point, shown at the very end.
\item<1-| alert@1> The first and main point.
\item<2-| alert@2> The second point.
\end{enumerate}
\end{frame}

```

ARTICLE 目前会完全忽略 $\langle action\ specification \rangle$ 。并且在条目 (item) 的开始处给出。

LYX 必须在 $\text{T}_\text{E}\text{X}$ -模式中给出 $\langle action\ specification \rangle$ ，并且在条目 (item) 的开始处给出。

相关的 `\bibitem` 命令也是叠层规则敏感的，其方式和 `\item` 相同。

$\langle label \rangle \langle overlay\ specification \rangle \{ \langle label\ name \rangle \}$

即：

$\langle label \rangle \langle \text{叠层规则} \rangle \{ \langle \text{标签名} \rangle \}$

如果给定了 $\langle overlay\ specification \rangle$ ，那么只在指定的幻灯片中插入标签 (label)。在多张幻灯片中插入一个标签将显示“多标签 (multiple labels)”警告。然而，如果没有给定叠层规则，则会自动将规则设为“1”，因此只在第一张幻灯片中插入标签。这是期望的行为，因为这对于插入了标签的幻灯片无影响，除非我们使用了 `\only` 命令，除非我们希望将那个标签作为超链接的目标 (hyperjump target)。然后必须指定一张幻灯片。

标签 (label) 可以用作超链接跳转的目标 (hyperjump target)。标签一个帧 (labelling a frame) 的便利方法是使用带 `label=\langle name \rangle` 选项的帧 (frame) 环境³⁰。然而，这会导致整个帧驻留在内存中直到编译结束，这可能带来问题。

举例：

```

\begin{frame}
\begin{align}
a \&= b + c \quad \langle label\{first\} \rangle \ \ \ % no specification needed
\end{align}
\end{frame}

```

³⁰例如：在一个帧标题名为“中暑”、帧标签名为 ZS 的帧中，有一个“返回”按钮，点击该按钮后将返回到帧标签名为 WL 的帧中，实现语句是：

```

\begin{frame}[label=ZS]{中暑}
帧内容
\hyperlink{WL}{\beamerreturnbutton{返回}}
\end{frame}。

```

```

    c &= d + e   \label{second}\% no specification needed
\end{align}

Blah blah, \uncover<2>{more blah blah.}

\only<3>{Specification is needed now.\label<3>{mylabel}}
\end{frame}

```

9.4 用于叠层规则的环境

环境 (environment) 也能装备叠层规则。对于大多数的预定义 (predefined) 环境, 请参阅第 12.3 节, 添加了叠层规则后使整个 (whole) 环境只在指定的幻灯片中显示。叠层规则对递增地显示 (showing things incrementally) 一些东西很有用, 如下面的例子所示。

```

\begin{frame}
  \frametitle{A Theorem on Infinite Sets}

  \begin{theorem}<1->
    There exists an infinite set.
  \end{theorem}

  \begin{proof}<3->
    This follows from the axiom of infinity.
  \end{proof}

  \begin{example}<2->
    The set of natural numbers is infinite.
  \end{example}
\end{frame}

```

在上面的例子中, 第一张幻灯片只包含了定理 (theorem), 第二张幻灯片则增加了一个例子 (example), 在第三张幻灯片还会显示证明 (proof)。

`\only`、`\alt`、`\visible`、`\uncover`、`\invisible` 这些基本的命令均有相对应的“环境版本 (environment versions)”即: `onlyenv`、`altenv`、`visibleenv`、`uncoverenv`、`invisibleenv`。除 `altenv`、`onlyenv` 外, 这些环境版本的所作所为与对应的基本命令的所作所为相同。

```

\begin{onlyenv}<<overlay specification>>
  <environment contents>
\end{onlyenv}

```

即:

```

\begin{onlyenv}<<叠层规则>>
  <environment contents>
\end{onlyenv}

```

如果给定了 `<overlay specification>`, 那么会在指定的幻灯片中插入环境的内容 (environment contents)。该环境和 `\only` 命令的不同之处是, 文本 (text) 实际上是先排入一个盒子内然后扔掉该盒子, 而 `\only` 命

令则立即扔掉其内容。如果文本是不“可排版的（”typesettable）”，`onlyenv` 环境就会出错，但 `\only` 不会。

举例：

```
\begin{frame}
  This line is always shown.
  \begin{onlyenv}<2>
    This line is inserted on slide 2.
  \end{onlyenv}
\end{frame}
```

```
\begin{altenv}<<overlay specification>>{<begin text>}{<end text>}{<alternate begin text>}{<alternate end text>}<<overlay
specification>>
  <environment contents>
\end{altenv}
```

即：

```
\begin{altenv}<<叠层规则>>{<开始文本>}{<结束文本>}{<备选的开始文本>}{<备选的结束文本>}<<叠层规则>>
  <environment contents>
\end{altenv}
```

只会给定一个 `<overlay specification>`。在指定的幻灯片中，会在该环境的开始处插入 `<begin text>`，在该环境的结束处插入 `<end text>`。而在其它的幻灯片中，则使用 `<alternate begin text>` 和 `<alternate end text>`。

举例：

```
\begin{frame}
  This
  \begin{altenv}<2>{({})}{[[]]}
    word
  \end{altenv}
  is in round brackets on slide 2 and in square brackets on slide 1.
\end{frame}
```

9.5 动态改变文本或图像

有时可能希望让帧的某部分在幻灯片之间动态地（dynamically）改变。在帧的每一张幻灯片中，在该区域（area）中显示不同的东西。要实现这样的动态改文本的（dynamically changing text）效果，可以使用一个由 `\only` 命令组成的列表，如下所示：

```
\only<1>{Initial text.}
\only<2>{Replaced by this on second slide.}
\only<3>{Replaced again by this on third slide.}
```

这种办法（approach）的麻烦之处是可能导致轻微（slight）但烦人的行高度（heights of the lines）的差异，这可能导致整个帧的幻灯片“摇晃不定（whobble）”。如果替换的文本（replacement text）占据了几行，则这个问题会严重得多。

为解决该问题，可以使用两个环境：`overlayarea` 和 `overprint`。第一个环境更有弹性（flexible），但不便使用。

```
\begin{overlayarea}{\langle area width \rangle}{\langle area height \rangle}
  \langle environment contents \rangle
\end{overlayarea}
```

即:

```
\begin{overlayarea}{\langle 区域宽度 \rangle}{\langle 区域高度 \rangle}
  \langle environment contents \rangle
\end{overlayarea}
```

该环境内的所有东西会放入一个指定尺寸的矩形 (rectangular) 区域中。该矩形区域在一个帧的所有幻灯片中具有相同的尺寸，而不管其实际的内容。

举例:

```
\begin{overlayarea}{\textwidth}{3cm}
  \only<1>{Some text for the first slide.\\Possibly several lines long.}
  \only<2>{Replacement on the second slide.}
\end{overlayarea}
```

LYX 使用 “OverlayArea” 风格插入一个叠层区域 (overlay area)。

```
\begin{overprint}[\langle area width \rangle]
  \langle environment contents \rangle
\end{overprint}
```

即:

```
\begin{overprint}[\langle 区域宽度 \rangle]
  \langle environment contents \rangle
\end{overprint}
```

$\langle area width \rangle$ 默认地使用文本宽度 (text width)。在该环境内，用 `\onslide` 命令指定显示在不同幻灯片中的不同的东西。`\onslide` 命令的用法和 `\item` 命令相似。该环境内的所有东西均被放入一个指定宽度的矩形区域内。该矩形区域内的高度和深度很大足以容纳环境内容。`\onslide` 命令的叠层规则必须是不相交的 (disjoint)。对讲义 (handout) 来说这是个问题，因为所有的叠层规则默认为 1。如果使用了讲义 (handout) 选项，我们可设定一个 `\onslide` 的叠层规则为 1，其它 `\onslide` 的叠层规则为 0，从而禁用 (disable) 几乎全部的 `\onslide`。

举例:

```
\begin{overprint}
  \onslide<1| handout:1>
    Some text for the first slide.\\
    Possibly several lines long.
  \onslide<2| handout:0>
    Replacement on the second slide. Supressed for handout.
\end{overprint}
```

LYX 使用 “Overprint” 风格插入一个 `overprint` 环境。必须在 T_EX-模式中插入 `\onslide` 命令。

动态改变 (dynamical changes) 的另一个用途是：我们拥有一组名为 `first.pdf`、`second.pdf`、`third.pdf` 的图片，它们分别显示于某个进程 (process) 的不同阶段。可以使用下面的代码创建一个帧，在该帧的不同幻灯片中分别显示这些图片：

```
\begin{frame}
  \frametitle{The Three Process Stages}

  \includegraphics<1>{first.pdf}
  \includegraphics<2>{second.pdf}
  \includegraphics<3>{third.pdf}
\end{frame}
```

上述的代码实际上是 BEAMER 使 `\includegraphics` 命令能感知叠层规则 (specificationaware)。只有每一个 `.pdf` 文件包含要显示的完整的图形时才能顺得地工作。然而，某些程序如 `xfig`，产生的图形序列 (series of graphics) 的每一文件只包含了附加的 (*additional*) 显示于下一幻灯片的图形元素 (graphic elements)。这样，第一幅图片不是在叠层 1 (overlay 1) 中显示，而是从叠层 1 开始显示，等等。虽然可以通过将叠层规则 `<1>` 改为 `<1->` 很容易地实现，但图形 (graphics) 必定是叠放在一起 (*on top of each other*) 显示。一个简易的实现方法是使用 $\text{T}_{\text{E}}\text{X}$ 的 `\llap` 命令，如下所示

```
\begin{frame}
  \frametitle{The Three Process Stages}

  \includegraphics<1->{first.pdf}%
  \llap{\includegraphics<2->{second.pdf}}%
  \llap{\includegraphics<3->{third.pdf}}
\end{frame}
```

或者像这样：

```
\begin{frame}
  \frametitle{The Three Process Stages}

  \includegraphics{first.pdf}%
  \pause%
  \llap{\includegraphics{second.pdf}}%
  \pause%
  \llap{\includegraphics{third.pdf}}
\end{frame}
```

一个更便利的方法是使用 `\multiinclude` 命令，详情请参阅第 14.1.3 节。

9.6 高级叠层规则

9.6.1 定义命令和环境叠层规则 - 已知的

本节阐述如何定义一个叠层规则敏感的 (overlay specification-aware) 新命令。本节还阐述如何正确建立随帧 (不是随幻灯片) 递增的计数器 (counters) (如公式编号)。除非我们要写自己的 BEAMER 文档类扩展 (extensions)，否则可以跳过这一节。

BEAMER 扩展了 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 标准命令 `\newcommand` 的语法：

`\newcommand<>{⟨command name⟩}[⟨argument number⟩][⟨default optional value⟩]{⟨text⟩}`

即:

`\newcommand<>{⟨命令名⟩}[⟨参数编号⟩][⟨默认可选的值⟩]{⟨文本⟩}`

声明名为 `⟨command name⟩` 的新命令。`⟨text⟩` 必须包含该命令的主体 (body) 并包含像 `#⟨number⟩` 这样的参数。这里的 `⟨number⟩` 必须介于 1 和 `⟨argument number⟩ + 1` 之间。额外的可允许的参数是叠层规则。

当使用 `⟨command name⟩` 时, 会扫描和 `⟨argument number⟩` 一样多的参数。在扫描参数时, 会查找叠层规则, 该叠层规则会在任意两个参数之间给出, 在第一个参数之前, 或在最后的参数之后。如果查找到了一个像 `<3>` 这样的叠层规则, 则会调用 (call) `⟨text⟩` 和参数 1 的设置 (set) 给普通参数 (normal arguments), 而参数序号 (argument number) `⟨argument number⟩ + 1` 则设为 `<3>` (包括尖括号)。如果没有查找到叠层规则, 则额外参数 (extra argument) 为空。

如果提供了 `⟨default optional value⟩`, 则 `⟨command name⟩` 的第一个参数是可选的。如果在方括号中没有指定可选参数, 则使用 `⟨default optional value⟩`。

举例: 下面的命令将在指定的幻灯片中以红色排版它的参数:

```
\newcommand<>{\makered}[1]{\color#2{red}#1}
```

举例: 这是 `\emph` 命令的 beamer 定义:

```
\newcommand<>{\emph}[1]{\only#2{\itshape}#1}
```

举例: 这是 `\transdissolve` 命令 (`\beamer@dotrans` 命令主要传递其参数给 `hyperref`) 的 BEAMER 定义:

```
\newcommand<>{\transdissolve}[1][\only#2{\beamer@dotrans[#1]{Dissolve}}]
```

`\renewcommand<>{⟨existing command name⟩}[⟨argument number⟩][⟨default optional value⟩]{⟨text⟩}`

即:

`\renewcommand<>{⟨现有的命令名⟩}[⟨参数编号⟩][⟨默认可选的值⟩]{⟨文本⟩}`

以和 `\newcommand<>` 相同的方式重新声明 (redeclare) 一个已有的命令。在 `⟨text⟩` 内, 我们仍可以使用 `\beameroriginal` 命令访问初始定义 (original definitions), 请参阅下面的例子

举例: 在 BEAMER 中使用该命令让 `\hyperlink` 变得叠层规则敏感 (overlay specification-aware):

```
\renewcommand<>{\hyperlink}[2]{\only#3{\beameroriginal{\hyperlink}{#1}{#2}}}
```

`\newenvironment<>{⟨environment name⟩}[⟨argument number⟩][⟨default optional value⟩]
{⟨begin text⟩}{⟨end text⟩}`

即:

`\newenvironment<>{⟨环境名⟩}[⟨参数编号⟩][⟨默认可选的值⟩]
{⟨开始文本⟩}{⟨结束文本⟩}`

声明一个叠层规则敏感的新环境。如果该环境有冲突 (encountered), 则使用和 `\newcommand<>` 相同的法则 (algorithm) 解析参数和叠层规则。

注意，一如既往，`<end text>` 不能包含任何像 `#1` 这样的参数。特别是，我们无权使用叠层规则。既然如此，在 `<begin text>` 内使用 `altenv` 环境是个好主意。

举例：声明我们自己的行为块（action block）：

```
\newenvironment<>{myboldblock}[1]{%
  \begin{actionenv}#2%
    \textbf{#1}
    \par}
{\par%
\end{actionenv}}

\begin{frame}
  \begin{myboldblock}<2>
    This theorem is shown only on the second slide.
  \end{myboldblock}
\end{frame}
```

举例：下面环境中的文本在非指定的幻灯片中显示为一般的粗体（bold）和斜体（italic）：

```
\newenvironment<>{boldornormal}
{\begin{altenv}#1
  {\begin{bfseries}}{\end{bfseries}}
  {}{}}
{\end{altenv}}
```

顺便说一下，因为 `altenv` 也在末尾接受其参数，使用下面的代码也能取得同样的效果：

```
\newenvironment{boldornormal}
{\begin{altenv}
  {\begin{bfseries}}{\end{bfseries}}
  {}{}}
{\end{altenv}}
```

```
\renewenvironment<>{<existing environment name>}[<argument number>][<default optional value>]
  {<begin text>}{<end text>}
```

即：

```
\renewenvironment<>{<现有的环境名>}[<参数编号>][<默认可选的值>]
  {<开始文本>}{<结束文本>}
```

重定义一个现有的环境。名为 `original<existing environment name>` 的初始环境仍有用。

举例：

```
\renewenvironment<>{verse}
{\begin{actionenv}#1\begin{originalverse}}
{\end{originalverse}\end{actionenv}}
```

下面的两条命令可用于确保在一个帧后来的（subsequent）幻灯片中自动重置（automatically reset）某个计数器（counter）。这对于等式计数（equation count）是很有用的。我们可能希望计数器随帧递增，但

不随叠层的幻灯片 (overlay slide) 而递增。对于等式计数器 (equation counters) 和脚注计数器 (footnote counters) (最好不用脚注), 已经调用了 (been invoked) 这些命令。

```
\resetcounteronoverlays{<counter name>}
```

即:

```
\resetcounteronoverlays{<计数名>}
```

如果我们已经调用了该命令, 则指定计数器的值在每一帧的所有幻灯片中是相同的。

举例: `\resetcounteronoverlays{equation}`

```
\resetcountonoverlays{<count register name>}
```

即:

```
\resetcountonoverlays{<计数注册名>}
```

和 `\resetcounteronoverlays` 相同, 不同之处是, 该命令必须和计数 (counts) 一起使用, 该计数是由 \TeX 原始的 `\newcount` 而不是 \LaTeX 的 `\definecounter` 创建。

举例:

```
\newcount\mycount
\resetcountonoverlays{\mycount}
```

9.6.2 模式规则

如果我们想用 BEAMER 的模式机制 (mode mechanism) 创建演示稿的不同版本, 则这节的内容很重要。如果我们对 BEAMER 模式不熟悉的话, 请跳过该节或先阅读第 21 节。

在特定的情况下, 我们可能希望不同的叠层规则适应于一个命令的不同模式。例如, 希望特定的文本只从演示稿的第三张幻灯片开始显示, 而观众手中的讲义 (handout) 没有第二张幻灯片, 所以对于讲义来说, 特定的文本只从第二张幻灯片开始显示。既然这样, 我们可以书写下面的代码:

```
\only<3|handout:2>{Some text}
```

后跟一个空隔 (space) 的竖条 (vertical bar) 将两个不同的规则 (specifications) 3 和 `handout:2` 分开。通过在冒号 (colon) 之前写入一个模式名 (mode name), 我们就可以指定跟随其后的规则只适应于该模式。如果没有给定模式, 只给定了 3, 则会自动添加模式 `beamer`。为此, 如果给我们的代码是 `\only<3>{Text}` 并处于 `handout` 模式, 则文本会显示于所有的幻灯片中, 因为没有为讲义 (handout) 指定限制 (restriction), 因为这里的 3 和 `beamer:3` 相同。

也可以给定只包含一个 (或多个, 之间用竖条分开的) 模式名的叠层规则:

```
\only<article>{This text is shown only in article mode.}
```

一个不包含任何幻灯片序号 (slide numbers) 的叠层规则叫做 (纯) 叠层规则。如果给定了模式规则, 则所有没有提及的模式将被抑制。因此, `<beamer:1->` 含意是 “在 `beamer` 模式中的所有幻灯片中及在其它模式中的所有幻灯片中, 因为没有为它们指定特别的东西”; 而 `<beamer>` 的含意是 “在 `beamer` 模式中的所有幻灯片中, 不在其它模式中的任何幻灯片中”。

模式规则也可用于帧外 (outside frames), 如下例所示:

```
\section<presentation>{This section exists only in the presentation modes}
\section<article>{This section exists only in the article mode}
```

演示稿模式 (Presentation modes) 包括: `beamer`、`trans`、`handout`。

我们也可以将纯模式规则和叠层规则混合在一起, 虽然这样可能导致混乱:

```
\only<article| beamer:1>{Riddle}
```

上述代码会在 `article` 模式中插入文本 `Riddle`, 在 `beamer` 模式中帧的第一张幻灯片中插入文本 `Riddle`, 但不会在 `handout` 模式或 `trans` 模式中插入文本 `Riddle`。(看看 `<beamer| beamer:1>`、`<beamer>`、`<beamer:1>` 之间的不同)

上述已够复杂了, 却还有另一个模式 `second`, 该模式的行为较特殊。用于该模式的规定 (rule) 是: 适应于 `beamer` 模式的叠层则也适应于 `second` 模式 (反过来则不行)。这样, 如果处于 `second` 模式中, `<second:2>` 规则的含意是“在第 2 张幻灯片中”, 而 `<beamer:2>` 的含义也是“在第 2 张幻灯片中”。要在 `beamer` 模式中而不是在 `second` 模式中排版幻灯片, 可以使用 `<second:0>`。

9.6.3 行为规则

本节介绍一个更高级的概念。在首次阅读该手册时可以跳过该节。

一些叠层规则敏感 (overlay specification-aware) 的命令不仅能处理 (handle) 普通的叠层规则, 而且可以处理所谓的行为规则 (*action specification*)。在一个行为规则中, 幻灯片序号和范围 (numbers and ranges) 列表之前缀有 `<action>@`, 这里的 `<action>` 是在指定幻灯片中执行的特定行为的名称:

```
\item<3-| alert@3> Shown from slide 3 on, alerted on slide 3.
```

在上述的例子中, 后跟有指定行为的 `\item` 命令, 将从第 3 张幻灯片开始显示 (uncover) 条目文本 (item text), 此外, 在第 3 张幻灯片提醒 (alert) 该条目。

不是所有的命令都能处理行为规则。目前, 只有 `\item` 命令 (不能处于 `article` 模式中)、`\action` 命令、`actionenv` 环境、块 (block) 环境 (如 `block` 或 `theorem`) 才能处理行为规则。

默认情况下, 下述的行为是可用的:

- `alert` 提醒 (alters) 条目或块。
- `uncover` 显示 (uncovers) 条目或块 (如果没有指定行为则这是默认的)。
- `only` 只在指定的幻灯片中插入所有条目或块。
- `visible` 只在指定的幻灯片中使文本可见 (visible) (`uncover` 和 `visible` 之间的不同之处与 `\uncover` 和 `\visible` 之间的不同之处相同)。
- `invisible` 在指定的幻灯片中使文本不可见 (invisible)。

该节的其余部分阐述如何添加自己的行为以及如何使命令变得叠层规则敏感。如果是首次阅读该手册则可以跳过该节。

我们可以轻而易举地添加自己的行为: 像 `<action>@<slide numbers>` 这样的行为规则只在 `\item` 的周围插入一个名为 `<action>env` 的环境, 或只插入带 `<slide numbers>` (作为叠层规则) 的 `\action` 命令的参数。因此, 通过定义一个名为 `<my action name>env` 的叠层规则敏感的环境, 我们就可以添加自己的行为:

```
\newenvironment{checkenv}{\only{\setbeamertemplate{itemize item}{X}}{}}
```

然后我们就可书写下面的代码:

```
\item<beamer:check@2> Text.
```

这样，在 `beamer` 模式中的第 2 张幻灯片上，`Text.` 前的 itemization 的标记 (symbol) 将改成 X。 `checkenv` 这个定义实际上是 `\only` 也接受在其参数后给定的一个叠层规则。

所有的行为机制 (action mechanism) 基于下面的环境：

```
\begin{actionenv}<<action specification>>
  <environment contents>
\end{actionenv}
```

即：

```
\begin{actionenv}<<行为规则>>
  <environment contents>
\end{actionenv}
```

该环境为当前模式提取了 (extract) `<action specification>` 的所有行为。对于每一个来自 `<action>@<slide numbers>` 的行为，该环境会插入下面的文本：在环境的开始处插入 `\begin{<action>env}<<slide number>>`，在环境的结束处插入 `\end{<action>env}`。如果有多个行为规则，则会打开 (和以合适的顺序关闭) 多个环境。一个没有行为的 `<overlay specification>` 会晋升为 `uncover@<overlay specification>`。

如果所谓的 `default overlay specification` 不为空 (empty)，则会使用它以防没有给定 `<action specification>`。默认的叠层规则通常为 `empty`，但可能会通过提供一个额外的可选参数给命令 `\frame` 或给环境 `itemize`、`enumerate`、`description` 来设置该默认的叠层规则 (请参阅相关的内容以获得细节)。而且，默认的行为规则也可以通过使用 `\beamerdefaultoverlayspecification` 命令来设置，请看下面。

举例：

```
\begin{frame}
  \begin{actionenv}<2-| alert@3-4,6>
    This text is shown the same way as the text below.
  \end{actionenv}

  \begin{uncoverenv}<2->
    \begin{alertenv}<3-4,6>
      This text is shown the same way as the text above.
    \end{alertenv}
  \end{uncoverenv}
\end{frame}
```

```
\action<<action specification>>{<text>}
```

即：

```
\action<<行为规则>>{<文本>}
```

在一个 `actionenv` 中放置 `<text>` 可以取得同样的效果。

举例： `\action<alert@2>{Could also have used \alert<2>{}}.`

```
\beamerdefaultoverlayspecification{<default overlay specification>}
```

即：

`\beamerdefaultoverlayspecification`{<默认的叠层规则>}

局部 (Locally) 将默认的叠层规则设为给定的值。该叠层规则将用在每一个 `actionenv` 环境和每一个 `\item` 中, 这些 `\item` 没有它们自己的叠层规则。该命令的主要用途是安装一个像 `<+>` 或 `<+| alert@+>` 这样的递增的 (incremental) 叠层规则, 请参阅第 9.6.4 节。

通常, 通过可选的参数自动安装默认的叠层规则给 `\frame`、`frame`、`itemize`、`enumerate`、`description`。如果我们想做些有趣的事情, 则可以使用该命令。

如果在帧外给出该命令, 则该命令将为所有后面的帧设置默认的叠层规则, 而这样的默认的叠层规则我们不能将它覆盖 (override)。

举例: `\beamerdefaultoverlayspecification<+>`

举例: `\beamerdefaultoverlayspecification`{} 清除默认的叠层规则。(实际上, 它安装默认的叠层规则 `<*>`, 它的含意是“总是”而且是“方便地”清除默认的叠层规则)

9.6.4 递增的规则

本节对于已经使用过大量叠层规则, 并且厌倦于三番五次地书写像 `<1->`、`<2->`、`<3->` 这样的东西的朋友来说是很重要的。如果是第一次阅读该手册, 则可以跳过该节。

通常, 我们想让叠层规则是下面的样子:

```
\begin{itemize}
\item<1-> Apple
\item<2-> Peach
\item<3-> Plum
\item<4-> Orange
\end{itemize}
```

如果我们想在开始处插入一种新水果 (fruit), 则会出现问题。因此我们必须调整叠层规则。而且, 如果在 `itemize` 之前添加了 `\pause` 命令, 我们也必不得不更新 (update) 叠层规则。

BEAMER 提供了特殊的语法 (syntax) 使上述的列表的叠层规则变得更“稳健 (robust)”。我们可以用下面列表的递增的叠层规则 (*incremental overlay specifications*) 代替上面列表的叠层规则:

```
\begin{itemize}
\item<+> Apple
\item<+> Peach
\item<+> Plum
\item<+> Orange
\end{itemize}
```

+ 号的作用如下: 在叠层规则的任何要放置数字 (number) 的地方我们都可能使用 + 号。如果遇到一个 + 号, 它会被 L^AT_EX 计数器 `beamerpauses` 的当前值代替, 在帧的开始处 `beamerpauses` 的值是 1。然后计数器 (counter) 按 1 递增, 然而对每一叠层规则来说, 它只会递增一次, 即使规则包含多个 + 号 (它们被同一个数字代替)。

在上述的例子中, 第一个叠层规则被 `<1->` 代替, 第二个叠层规则被 `<2->` 代替, 等等。现在我们就可以轻而易举地插入新条目 (entries) 而无需更改任何东西。我们也可以书写下面的语句:

```
\begin{itemize}
```

```

\item<+| alert@+> Apple
\item<+| alert@+> Peach
\item<+| alert@+> Plum
\item<+| alert@+> Orange
\end{itemize}

```

当当前条目显示 (uncovered) 时上述语句会提醒 (alert) 当前条目, 例如, 第一个规则 `<+| alert@+>` 被 `<1-| alert@1>` 替代, 第二个规则被 `<2-| alert@2>` 代替, 等等。因为 `itemize` 环境也允许我们指定默认的叠层规则 (请参考 `itemize` 环境的文档), 上述例子也可写成如下的更节约的 (economically) 形式:

```

\begin{itemize}[<+| alert@+>]
\item Apple
\item Peach
\item Plum
\item Orange
\end{itemize}

```

`\pause` 命令也会更新计数器 `beamerpauses`。我们可以使用普通的 L^AT_EX 命令 `\setcounter` 或 `\addtocounter` 更改这个计数器。

+ 号后可能跟有一个放在圆括号中的偏移量 (*offset*)。该偏移量会添加到 `beamerpauses` 的值中。因此, 如果 `beamerpauses` 的值是 2, 那么 `<+(1)->` 会扩大为 `<3->`, `<+(-1)->` 会扩大为 `<1-2>`。例如:

```

\begin{frame}
\frametitle{Method 1}
\begin{itemize}
\item<2-> Apple
\item<3-> Peach
\item<4-> Plum
\item<5-> Orange
\end{itemize}

```

上面的例子和上面的例子等效:

```

\begin{itemize}[<+(1)->]
\item Apple
\item Peach
\item Plum
\item Orange
\end{itemize}

```

还有一个特殊符号小圆点 (dot) 可用于叠层规则, 它的行为和 + 号相似。除计数器 `beamerpauses` 不会递增及 `beamerpauses` 的值减少了 1 之外, `<.->` 的行为和 `<+>` 相似。这样, 小圆点 (后面也可能跟有一个偏移量) 只扩大为计数器 `beamerpauses` 的当前值减 1, 而不论偏移量的大小。这个小圆点符号 (dot notation) 在下面的情形中很有用:

```

\begin{itemize}[<+>]
\item Apple
\item<.-> Peach

```



```
\item Plum
\item Orange
\end{itemize}
```

在上述的例子中，第二个条目和第一个条目会同时显示，因为第二个条目不会更新计数器。

在下面的例子中，每次条目显示时，指定的文本就会被提醒。当下一个条目显示时，上一次的提醒就会结束。

```
\begin{itemize}[<+>]
\item This is \alert<.>{important}.
\item We want to \alert<.>{highlight} this and \alert<.>{this}.
\item What is the \alert<.>{matrix}?
\end{itemize}
```

10 组建演示稿: 静态的全局结构

这一节罗列了用于组建演示稿的命令, 这些命令是“全局性的”, 如 `\section` 或 `\part`。这些命令用于创建静态结构 (*static structure*), 静态的含义是指最终的演示稿是以一张幻灯片接一张幻灯片按顺序显示这种方式呈现的。第 11 节将阐述哪些命令可用于创建交互结构 (*interactive structure*)。对于交互结构, 通常通过点击超链接 (hyperlinks) 与演示稿程序 (presentation program) 交互。

10.1 添加封面

我们可以使用 `\titlepage` 命令将一封面 (title page) 插入到帧。默认情况下, 将对封面的下列元素进行排列: 文档标题 (documenttitle)、作者姓名 [author(s)'s names]、合作者 (affiliation)、标题图片 (title graphic)、日期 (date)。

`\titlepage`

在当前帧插入封面的文本。

举例: `\frame{\titlepage}`

举例: `\frame[plain]{\titlepage}` 对于封面, 它会填满整个帧。

LYX 如果在我们的演示稿中使用了“Title”样式, 将会自动插入封面。

Beamer-Template/-Color/-Font title page

即:

Beamer-Template/-Color/-Font 封面

当使用 `\titlepage` 命令时, 调用该模板。

下面的模板选项是预定好的:

- **[default]** [*<alignment>*] 排版好的封面会显示标题 (title)、后跟作者、他或她的合作者、日期、标题图片 (titlegraphic)。如果缺失其中某项, 则不会显示。除标题图片外, 如果分别定义了 `BEAMER-color title`, `author`, `institute`, 或 `date` [即: `BEAMER-颜色标题 (title)`、作者 (`author`)、大学 (`institute`)、日期 (`date`)], 这些条目将使用文本色 (`textcolor`); 如果为这些条目定义了背景色 (Background color), 则在这些条目的后面绘制相应颜色的色带 (colored bar), 该色带会跨越文本 (即比文本更宽)。这些条目将应用相关的 `BEAMER-fonts`。

<alignment> 选项传递给 `beamercolorbox` 并可以使用, 例如, 可以在这里指定 `left` 以使页面左对齐。

下面的命令对于该模板是有用的:

- `\insertauthor` 在封面中插入作者姓名。
- `\insertdate` 插入日期。
- `\insertinstitute` 插入大学。
- `\inserttitle` 在封面中插入适合的文档标题。
- `\insertsubtitle` 在封面中插入适合的文档子标题。
- `\inserttitlegraphic` 在一模板中插入标题图片。

为和其它文档类兼容，在论文（`article`）模式中，还提供了下面的命令：

`\maketitle`

PRESENTATION 如用在帧内，与 `\titlepage` 等效。如用在帧外，与 `\frame{\titlepage}` 等效果；换言之，按需添加帧。

在调用 `\titlepage` 命令之前，我们必需指定所有要显示的元素。可用下面的命令实现这点：

`\title[<short title>]{<title>}`

即：

`\title[<标题简写形式>]{<标题>}`

<short title> 用在顶部导航区（headline）和底部导航区（footline）。在 *<title>* 内可以使用双反斜线符（double-backslash，即：`\\`）命令插入换行符（line breaks）。

举例：

```
\title{The Beamer Class}
\title[Short Version]{A Very Long Title\\Over Several Lines}
```

ARTICLE 在论文（`article`）模式中，会忽略简写形式。

`\subtitle[<short subtitle>]{<subtitle>}`

即：

`\subtitle[<子标题简写形式>]{<子标题>}`

默认情况下不使用 *<short subtitle>*，但插入 `\insertshortsubtitle` 后即可使用 *<short subtitle>*。子标题用更小的字体显示在标题的下面。

举例：

```
\title{The Beamer Class}
\subtitle{An easily paced introduction with many examples.}
```

ARTICLE 该命令在标题的后面用换行符（linebreak）和 `\normalsize` 附加子标题。是否添加子标题由我们决定。

`\author[<short author names>]{<author names>}`

即：

`\author[<作者姓名简写形式>]{<作者姓名>}`

多个作者的姓名之间用 `\and` 命令分隔。如果作者有不同的合作者，可以用带不同参数的 `\inst` 命令在作者的后面缀上合作者。

举例：`\author[Hemaspaandra et al.]{L. Hemaspaandra\inst{1} \and T. Tantau\inst{2}}`

ARTICLE 在论文（`article`）模式中，会忽略简写形式。

`\institute[<short institute>]{<institute>}`

即:

`\institute[⟨大学名的简写形式⟩]{⟨大学名⟩}`

多个大学之间用 `\and` 命令分隔，并在前面缀以带不同参数的 `\inst` 命令。

举例:

```
\institute[Universities of Rijeka and Berlin]{
  \inst{1}Department of Informatics\
  University of Rijeka
  \and
  \inst{2}Fakultät für Elektrotechnik und Informatik\
  Technical University of Berlin}
```

ARTICLE 在论文 (article) 模式中，会忽略简写形式。除文档类 (如 `llncs`) 定义了大学的全称外，也会忽略全称形式。

`\date[⟨short date⟩]{⟨date⟩}`

即:

`\date[⟨日期简写形式⟩]{⟨日期⟩}`

举例: `\date{\today}` or `\date[STACS 2003]{STACS Conference, 2003}`.

ARTICLE 在论文 (article) 模式中，会忽略简写形式。

`\titlegraphic{⟨text⟩}`

即:

`\titlegraphic{⟨文本⟩}`

`⟨text⟩` 用作标题图片 (title graphic)。通常，一个 `picture` 环境用作 `⟨text⟩`。

举例: `\titlegraphic{\pgfuseimage{titlegraphic}}`

ARTICLE 在论文 (article) 模式中，会忽略该命令。

`\subject{⟨text⟩}`

即:

`\subject{⟨文本⟩}`

用 `⟨text⟩` 作为 PDF 文档信息 (document info) 中的主题文本 (subject `⟨text⟩`)。目前，它没有其它作用。

`\keywords{⟨text⟩}`

即:

`\keywords{⟨文本⟩}`

用 `⟨text⟩` 作为 PDF 文档信息 (document info) 中的主关键词 (keywords `⟨text⟩`)。目前，它没有其它作用。

默认情况下，`\title` 命令和 `\author` 命令会将它们的参数插入到最终的 PDF 件的文档信息字段 (document information fields) 中。如果我们使用复杂的东西如盒子 (boxes) 作为这些命令的参数时，可能带来问题。既然这样，我们可能希望关闭这种自动生成这些条目的功能，可以用下面的文档类选项实现：

```
\documentclass[usepdftitle=false]{beamer}
```

抑制自动生成 PDF 文档信息中的标题和作者条目。

10.2 添加节和小节

我们可以用 `\section` 命令和 `\subsection` 命令组织文本。不像标准的 L^AT_EX，这些命令不会在我们使用它们的地方创建标题 (heading)。相反，它们会添加条目到目录 (Table of Contents) 和导航条 (Navigation Bar) 中。

要在目录中创建换行符 (这不是个好主意)，可以使用 `\breakhere` 命令。注意，标准的 `\\` 命令会失效 (不知道为什么；可以将它注释掉)。

```
\section<<mode specification>>[<short section name>]{<section name>}
```

即：

```
\section<<模式规则>>[<节名简写形式>]{<节名>}
```

开始一个节。不会创建标题 (heading)。在目录和导航条中会显示 `<section name>`，除非指定了 `<short section name>`。既然这样，可以在导航条上代之以 `<short section name>`。如果给定了 `<mode specification>`，则该命令只在指定的模式中有效。

举例：`\section[Summary]{Summary of Main Results}`

ARTICLE 在论文 (article) 模式中，`<mode specification>` 允许我们提供一个可替代的节命令。这是必需的，例如，如果 `<short section name>` 不适合目录：

举例：

```
\section<presentation>[Results]{Results on the Main Problem}
\section<article>{Results on the Main Problem}
```

Beamer-Template/-Color/-Font section in toc

即：

Beamer-Template/-Color/-Font 目录中的节

当排版节条目时会使用该模板。允许的 `<options>` 请参考父模板目录 (table of contents)。

下面的命令对该模板有用：

- `\inserttocsection` 插入当前节名的目录。
- `\inserttocsectionnumber` 插入当前节的序号 (在目录中)。

Beamer-Template/-Color/-Font section in toc shaded

即：

Beamer-Template/-Color/-Font 目录中的带阴影的节

如果节带有阴影，可使用该模板，因为它不是当前节。允许的 *(options)* 请参考父模板目录 (table of contents)。

```
\section<<mode specification>>*{<section name>}
```

即：

```
\section<<模式规则>>*{<节名>}
```

开始一个在目录中没有条目的节。不会创建标题 (heading)。在导航条中会显示 *(section name)*。

举例：`\section*{Outline}`

举例：`\section<beamer>*{Outline}`

```
\subsection<<mode specification>>[<short subsection name>]{<subsection name>}
```

即：

```
\subsection<<模式规则>>[<小节名简写形式>]{<小节名>}
```

该命令的所作所为与 `\section` 命令相同。

举例：`\subsection[Applications]{Applications to the Reduction of Pollution}`

Beamer-Template/-Color/-Font subsection in toc

即：

Beamer-Template/-Color/-Font 目录中的小节

像目录中的节 (section in toc)，它只用于小节 (subsection)。

除用于目录中的节 (section in toc) 模板的插入物 (inserts)，下面的命令对该模板有用：

- `\inserttocsubsection` 插入当前小节名的目录。
- `\inserttocsubsectionnumber` 插入当前小节的序号 (在目录中)。

Beamer-Template/-Color/-Font subsection in toc shaded

即：

Beamer-Template/-Color/-Font 目录中的带阴影的小节

该命令的所作所为与 `section in toc shaded` 相同，只用于小节。

```
\subsection<<mode specification>>*{<subsection name>}
```

即：

`\subsection<模式规则>*<小节名>`

开始一个在目录中没有条目的小节。不会创建标题 (heading)，除非 `<subsection name>` 是空的，否则在导航条中会显示 `<subsection name>`。既然如此，不会创建目录的条目，也不创建导航条的条目，但在该“空”小节中的帧会显示在导航条中。

举例：

```
\section{Summary}
```

```
\frame{This frame is not shown in the navigation bar}
```

```
\subsection*{} 
```

```
\frame{This frame is shown in the navigation bar, but no subsection entry is shown.}
```

```
\subsection*{A subsection}
```

```
\frame{Normal frame, shown in navigation bar. The subsection name is also shown in the navigation bar, but not in the table of contents.}
```

`\subsubsection<mode specification>[<short subsection name>]{<subsection name>`

即：

`\subsubsection<模式规则>[<小小节名简写形式>]{<小小节名>`

该命令的所作所为与 `\subsection` 相同。然而，对小小节的支持不如小节。例如，在目录中，小小节显示有和小节相同的阴影/隐藏 (shading/hiding) 参数。

我们不赞成在演示稿使用小小节。如果不使用小小节，我们的演示稿会更好。

举例：`\subsubsection[Applications]{Applications to the Reduction of Pollution}`

Beamer-Template/-Color/-Font subsection in toc

即：

Beamer-Template/-Color/-Font 目录中的小小节

像目录中的小节 (`subsection in toc`)，它只用于小节 (`subsection`)。

除用于目录中的小节 (`subsection in toc`) 模板的插入物 (`inserts`)，下面的命令对该模板有用：

- `\inserttocsubsubsection` 插入当前小小节名的目录。
- `\inserttocsubsubsectionnumber` 插入当前小小节的序号 (在目录中)。

Beamer-Template/-Color/-Font subsection in toc shaded

即：

Beamer-Template/-Color/-Font 目录中的带阴影的小小节

像目录中的带阴影的小节 (`subsection in toc shaded`)，它只用于小小节 (`subsubsections`)。

`\subsubsection<mode specification>*<subsection name>`

即:

`\subsubsection<模式规则>*<小小节名>`

开始一个在目录中没有条目的小小节。不会创建标题 (heading), 但 `<subsection name>` 会显示在一个可能的侧栏中。

我们常常可能想在一个节或小节后面直接显示某类型的帧, 例如, 我们希望在显示每一小节时, 在帧的目录中高亮显示当前小节。要实现这一点, 可以使用下面的命令:

`\AtBeginSection[<special star text>]{<text>}`

即:

`\AtBeginSection[<特定的开始文本>]{<文本>}`

给定的文本会插入到每一节的开始处。如果指定了 `<special star text>` 的参数, 该文本用于主要的节 (starred sections)。对该命令的不同调用不会“叠加 (add up)”给定的文本 (像 `\AtBeginDocument` 命令这样), 但会覆盖以前的文本。

举例:

```
\AtBeginSection[] % Do nothing for \section*
{
  \begin{frame}<beamer>
    \frametitle{Outline}
    \tableofcontents[currentsection]
  \end{frame}
}
```

ARTICLE 在论文 (article) 模式中, 该命令无效。

LYX 必须使用 T_EX-模式 (TEX-mode) 的文本插入该命令。

`\AtBeginSubsection[<special star text>]{<text>}`

即:

`\AtBeginSubsection[<特定的开始文本>]{<文本>}`

给定的文本会插入到每一小节的开始处。如果指定了 `<special star text>` 的参数, 该文本用于主要的小节 (starred subsections)。对该命令的不同调用不会“叠加 (add up)”给定的文本。

举例:

```
\AtBeginSubsection[] % Do nothing for \subsection*
{
  \begin{frame}<beamer>
    \frametitle{Outline}
    \tableofcontents[currentsection,currentsubsection]
  \end{frame}
}
```


`\AtBeginSubsubsection[<special star text>]{<text>}`

即:

`\AtBeginSubsubsection[<特定的开始文本>]{<文本>}`

像 `\AtBeginSubsection`，只用于小小节 (subsubsections)。

BEAMER 提供了 `\sectionpage` 和 `\subsectionpage` 命令，这些命令用节或小节序号 (section or subsection number) 和标题 (title) 以流行的方式填充帧。它们和下面描述的 `\partpage` 命令类似。

举例:

```
\section{A section}

\frame{\sectionpage}

\frame{Some text.}
```

如果我们象第 25.3.1 节那样激活 TRANSLATOR，我们将会获得 “Section”、“Subsection” 和其它的已翻译的字串。

10.3 添加部分

如果我们的演讲很长 (如专题讲座)，可以将它分成几个部分 (part)。每一个部分是相对独立的，有它自己的目录、导航条等等。在一个部分内，其它部分的节和小节不会显示。

可以使用 `\part` 命令新建一个部分，跟随其后的所有节和小节将 “安置 (local)” 在该部分中。像 `\section` 和 `\subsection` 命令一样，`\part` 命令不会生成任何帧和特定的文本。然而，使用 `\partpage` 命令将一些文本插入到一个帧 “宣告” 新部分的开始，以这种方式开始一个新部分是合理的。请参考 `beamerexample3.tex` 该例子。

`\part<<mode specification>>[<short part name>]{<part name>}`

即:

`\part<<模式规则>>[<部分名简写形式>]{<部分名>}`

开始一个部分。当使用 `\partpage` 命令时，会显示 *<part name>*。默认情况下，不会在任何地方显示 *<short part name>*，但可通过 `\insertshortpart` 命令访问它。

举例:

```
\begin{document}
  \frame{\titlepage}

  \section*{Outlines}
  \subsection{Part I: Review of Previous Lecture}
  \frame{
    \frametitle{Outline of Part I}
    \tableofcontents[part=1]}
  \subsection{Part II: Today's Lecture}
```

```

\frame{
  \frametitle{Outline of Part II}
  \tableofcontents[part=2]}

\part{Review of Previous Lecture}
\frame{\partpage}
\section[Previous Lecture]{Summary of the Previous Lecture}
\subsection{Topics}
\frame{...}
\subsection{Learning Objectives}
\frame{...}

\part{Today's Lecture}
\frame{\partpage}
\section{Topic A}
\frame{\tableofcontents[currentsection]}
\subsection{Foo}
\frame{...}
\section{Topic B}
\frame{\tableofcontents[currentsection]}
\subsection{bar}
\frame{...}
\end{document}

```

`\partpage`

所作所为和 `\titlepage` 命令相似，只是“宣告 (advertised)”当前部分 (current part) 而不是当前的演示稿 (current presentation)。

举例: `\frame{\partpage}`

Beamer-Template/-Color/-Font part page

即:

Beamer-Template/-Color/-Font 部分页

当使用 `\partpage` 命令时调用该模板。

下面的模板选项是预定好的:

- **[default]** [*<alignment>*] 在部分页 (part page) 显示当前部分序号 (part number)，并在下面显示当前部分的标题 (current part title)。会使用 `BEAMER-color` 和 `-font`，包括 `part page` 的背景色。至于 `title page` 模板，*<alignment>* 选项会传递给 `beamercolorbox`。

下面的命令对于该模板是有用的:

- `\insertpart` 插入当前部分名。
- `\insertpartnumber` 将当前部分的序号 (阿拉伯数字) 插入到模板。
- `\insertpartromannumber` 将当前部分的序号 (罗马数字) 插入到模板。

`\AtBeginPart{<text>}`

在每一部分的起始处插入给定的文本。

举例：

```
\AtBeginPart{\frame{\partpage}}
```

10.4 将教程分解成讲课

在论文 (article) 模式下使用 BEAMER 时，我们可能希望将整个教程 (whole course) 的讲稿 (lecture notes) 放在一个文件中。既然如此，只有一些帧才是特定讲课 (particular lecture) 的事实上的的一部分。

`\lecture` 命令使用得从一个文件中选取某些帧变得容易。该命令提取标签名 (label name) (和其它事物)。如果我们用该标签名声明 `\includeonlylecture`，那么只会显示跟随在相应的 `\lecture` 命令后面的帧。跟随在其它 `\lecture` 命令后面的帧则被抑制。

默认情况下，`\lecture` 命令没有其它作用。它不会创建任何帧或目录中的条目。然而，我们可使用 `\AtBeginLecture` 命令让 BEAMER 在 (每一) 讲课 (lecture) 的开始处插入一个标题页 (title page)。

`\lecture[<short lecture name>]{<lecture name>}{<lecture label>}`

即：

`\lecture[<讲课名简写形式>]{<讲课名>}{<讲课标签>}`

开始一讲课 (lecture)。通过 `\insertlecture <lecture name>` 变得可用。通过 `\insertshortlecture` 命令 `<short lecture name>` 变得可用。

举例：

```
\begin{document}
\lecture{Vector Spaces}{week 1}

\section{Introduction}
...
\section{Summary}

\lecture{Scalar Products}{week 2}

\section{Introduction}
...
\section{Summary}

\end{document}
```

ARTICLE 在论文 (article) 模式中，该命令无效。

`\includeonlylecture<lecture label>`

即：

`\includeonlylecture`<讲课标签>

抑制跟随在 `\lecture` 命令后面的所有 `\frame`、`frame`、`\section`、`\subsection` 和 `\part` 命令，除非讲课的标签 (lecture's label) 和 `<lecture label>` 相配。常常会包含 `\lecture` 之前的帧。该命令可以在导言区 (Preamble) 给出。

举例：`\includeonlylecture{week 1}`

ARTICLE 在论文 (article) 模式中，该命令无效。

`\AtBeginLecture`{*text*}

在每一讲课 (lecture) 开始处插入给定的文本。

举例：

```
\AtBeginLecture{\frame{\Large Today's Lecture: \insertlecture}}
```

ARTICLE 在论文 (article) 模式中，该命令无效。

10.5 添加目录

可以用 `\tableofcontents` 命令创建一个目录 (table of contents)³¹。不像标准的 L^AT_EX 目录命令，该命令带有一可选的参数，该参数放在方括号中，用它创建某些特别的效果。

`\tableofcontents`[*comma-separated option list*]

即：

`\tableofcontents`[*逗号分开的选项列表*]

在当前的帧插入一个目录

举例：

```
\section*{Outline}
\frame{\tableofcontents}

\section{Introduction}
\frame{\tableofcontents[currentsection]}
\subsection{Why?}
\frame{...}
\frame{...}
\subsection{Where?}
\frame{...}

\section{Results}
\frame{\tableofcontents[currentsection]}
\subsection{Because}
```

³¹要显中文的“目录”这两个字，可以在 `\begin{document}` 的后面（而不是在导言区中）使用下面的命令：
`\renewcommand\contentsname{目录}`

```
\frame{...}
\subsection{Here}
\frame{...}
```

可能会给出下面的选项：

- **currentsection** 使除当前节以外的所有节以半透明的方式显示。也使除当前节内的小节以外的所有小节以半透明的方式显示。该命令是指定了下面选项的简写形式：

```
sectionstyle=show/shaded,subsectionstyle=show/show/shaded
```

- **currentsubsection** 使除当前节中的当前小节以外的所有小节以半透明的方式显示。该命令是指定了下面选项的简写形式：`subsectionstyle=show/shaded`
- **firstsection=*<section number>*** 指定哪个节被编号成节 “1.”。如果有一个不应接受编号的首节（first section）（像概述节），这时这个选项很有用。默认情况下不显示节编号（Section numbers）。要显示节编号，我们必须安装一个不同的目录模板（Table of Contents templates）。
- **hideallsubsections** 隐藏所有的小节。该命令是指定了选项 `subsectionstyle=hide` 的简写形式
- **hideothersubsections** 隐藏除当前节以外的节的小节。该命令是指定了选项 `subsectionstyle=show/show/hide` 的简写形式。
- **part=*<part number>*** 显示部分 *<part number>* 的目录，取代当前部分（这是默认的）的目录。这个选项可以和其它的选项联用，虽然和 `current` 选项联用显然是没道理的。
- **pausesections** 在每一节前给出 `\pause` 命令。要递增地显示目录时，这个选项很有用。
- **pausesubsections** 在每一小节前给出 `\pause` 命令
- **sections=*<overlay specification>*** 显示在 *<overlay specification>* 中提到的节，例如，`sections={<2-4| handout:0>}` 只显示普通版本（normal version）的第 2、3、4 节，不显示讲义版本（Handout version）的任何东西，显示其它版本的所有东西。为图方便，如果我们忽略了尖括号，则规则（specification）被认定适应于所有版本。因此，`sections={2-4}` 会显示所有版本的第 2、3、4 节。
- **sectionstyle=*<style for current section>/<style for other sections>*** 指定如何显示节。允的 *<styles>* 是 `show`、`shaded`、`hide`。第一会正常地显示节的标题，第二以半透明的方式显示，第三完全将它抑制。我们可以忽略第二样式（style），这时所有的节会使用第一样式（这不是真正有益的）。
- **subsectionstyle=*<style for current subsection>/<style for other subsections in current section>/<style for subsections in other sections>*** 指定如何显示小节。会给定和 `sectionstyle` 选项相同的样式。我们可以忽略最后的样式，这时，第二种也适应于最后；当我们忽略后两种样式时，第一种样式适应于全部。

举例：`subsectionstyle=shaded` 使所有的小节带阴影。

举例：`subsectionstyle=hide` 隐藏所有的小节。

举例：`subsectionstyle=show/shaded` 使除当前节中的当前小节外的所有小节以半透明的方式显示。

举例：`subsectionstyle=show/show/hide` 抑制当前节外的所有小节。

举例：`subsectionstyle=show/shaded/hide` 抑制当前节外的所有小节，当前节的当前小节高亮显示。

- `subsubsectionstyle={style for current subsubsection}/{style for other subsubsections in current subsection}/{style for subsubsections in other subsections}` 指定如何显示小小节 (subsubsections)。可能会给出和 `sectionstyle` 选项相同的样式 (style)。可以忽略最后的样式 (the last style)，这时，倒数第二个样式成了最后一个样式，也可以忽略，因此，第一个样式适应于全部。该选项的运行方式与 `subsectionstyle` 相似。

在呈现演讲提纲 (talk outline) 而不想显示太多细节时，最后的例子很有用。

ARTICLE 在论文 (article) 模式中，可以忽略上述选项。

LYX 通过在目录后面直接插入一 \TeX -模式 (\TeX -mode) 的文本和方括号中的选项，我们就可以提供选项给 `\tableofcontents` 命令。

Parent Beamer-Template `section/subsection in toc`

即：

Parent Beamer-Template 目录中的节/小节

这是一个父模板，它的子模板是目录中的节 (`section in toc`) 和目录中的小节 (`subsection in toc`)。这意味着如果我们在这个模板中使用 `\setbeamertemplate` 命令，调用的是该命令而不是它的子模板，调用的参数与其子模板相同。

下面的模板选项是预定好的：

- **[default]** 在默认的设置中，使用 `fonts` 和 `colors section in toc` 和 `subsection in toc` 排版节和小节，虽然会忽略背景色。小节会缩进。
- **[sections numbered]** 和默认设置之不理相类似，但节序号不显示。小节不被编号。
- **[subsections numbered]** 这次，给小节编号，而不是节。然而，因为小节的编号是象“1.2”或“3.2”这样的“全编号”，如果每一节拥有至少一个小节，被编号的节则不会出错 (be missed)。
- **[circle]** 在节的前面绘制一小圆。在小圆中显示节编号。BEAMER-font 和 `color section number projected` 用于排版小圆，也就是说，圆获得了 (get) 背景色，圆中的文本获得了前景色。
- **[square]** 和 `circle` 选项相似，只不过是用小方形 (small squares) 取代小圆罢了。在小节前面显示小的未编号的小方形。
- **[ball]** 和 `square` 选项相似，不同的是用小球取代小方形。
- **[ball unnumbered]** 和 `ball` 选项相似，使用未编号的小球。该选项使目录看起来更象常规列表 (`itemize`)。

如果上述选项不能满足我们，可以直接更改模板目录中的节 (`section in toc`) 和目录中的小节 (`subsection in toc`)。

Parent Beamer-Template `section/subsection in toc shaded`

即：

Parent Beamer-Template 目录中的带阴影的节/小节

带有子模板目录中的带阴影的节 (`section in toc shaded`) 和目录中的带阴影的小节 (`subsection in toc shaded`) 的一个父模板。它们用于生成带阴影的节条目和带阴影的小节条目；像在 `\tableofcontents[currentsubsection]` 中的所有非当前的 (`non-current`) 小节一样。

下面的模板选项是预定好的：

- `[default][<opaqueness>]` 在默认的设置中，模板目录中的带阴影的节 (`section in toc shaded`) 和目录中的带阴影的小节 (`subsection in toc shaded`) 只显示这些模板的非阴影版本 (`nonshaded versions`) 显示的任何东西，可是只有 `<opaqueness>%` 的不透明 (`opaque`)。默认是 20%。

举例：`\setbeamertemplate{table of contents shaded}[default][50]` 使暗淡的条目 (`dimmed entries`) 50% 的透明。

10.6 添加参考书目

可以在 BEAMER 演示稿中使用 L^AT_EX 的 `bibliography` 环境和 `\cite` 命令。也许我们不得不“手工”排版参考书的部分条目。然而，我们可以使用 `bibtex` 创建“初步近似的”参考书目。复制 `main.bbl` 文件的内容到我们的演示稿中。如果我们不熟悉 `bibtex`，可以查阅 (`consult`) 它的文档。它是创建高质量引文 (`citations`) 的强有力的工具。

使用 `bibtex` 或我们的编辑器，在 `thebibliography` 环境中放置我们的参考文献 (`bibliographic references`)。该 (标准的 L^AT_EX) 环境带有一个参数，该参数是下面的参考书目条目中的长的 `\bibitem` 标签 (`label`)

```
\begin{thebibliography}{<longest label text>}
  <environment contents>
\end{thebibliography}
```

即：

```
\begin{thebibliography}{<长标签文本>}
  <environment contents>
\end{thebibliography}
```

在当前帧插入一个参考书目。用 `<longest label text>` 判定列表的缩进 (`indentation`)。然而，排版参考书目的几个预定义选项会忽略该参数，因为这几个预定义选项会用一个符号 (`symbol`) 代替参考文献 (`references`)。

在该环境中，使用 (标准的 L^AT_EX) `\bibitem` 命令创每一参考文献条目 (`reference item`)。在每一参考文献条目内，使用 (标准的 L^AT_EX) `\newblock` 命令分隔作者姓名、标题、引用的书籍/期刊 (`book/journal`)、其它笔录 (`notes`)。每一条这样的命令会生成新的一行或新的颜色或其它格式，这可通过参考书目的模板指定。

该环境必须放在一个帧内。如果一个帧容不下参考书目，我们可以将参考书目分开 (`split`) (新建一个帧并新建第二个 `thebibliography` 环境) 或使用 `allowframebreaks` 选项。更好的办法是，我们可以重新考虑是否需要呈现这么多参考文献。

举例：

```
\begin{frame}
```

```

\frametitle{For Further Reading}

\begin{thebibliography}{Dijkstra, 1982}
\bibitem[Salomaa, 1973]{Salomaa1973}
  A.-Salomaa.
  \newblock {\em Formal Languages}.
  \newblock Academic Press, 1973.

\bibitem[Dijkstra, 1982]{Dijkstra1982}
  E.-Dijkstra.
  \newblock Smoothsort, an alternative for sorting in situ.
  \newblock {\em Science of Computer Programming}, 1(3):223--233, 1982.
\end{thebibliography}
\end{frame}

```

四个模板控制着作者（author）、标题（title）、期刊（journal）、笔录（note）文本的外观。在条目的第一个块（block）的前面插入作者模板（author templates），第一个块就是第一次出现的 `\newblock` 命令的前面的所有文本。在第二个块的前面插入标题模板（title template），第二个块就是第一次出现的 `\newblock` 和第二次出现的 `\newblock` 命令之间的文本。对于期刊模板（journal）和笔录模板（note），情况相同。在块之前插入模板，通过插入命令我们无权使用块本身。相应的 `BEAMER-color` 和 `-font` 也插入到块前面。

Beamer-Template/-Color/-Font `bibliography entry author`

即：

Beamer-Template/-Color/-Font 参考书目条目作者

该模板插入到参考书目条目的作者的前面。同时安装颜色和字体。注意该模板的效果一直持续到参考书目条目的结束之处，或持续到下列模板之一撤消该效果之处。

默认情况下，该模板无作用。默认颜色是结构色（structure color）。

Beamer-Template/-Color/-Font `bibliography entry title`

即：

Beamer-Template/-Color/-Font 参考书目条目标题

该模板插入到参考书目条目标题的前面（更精确地讲，是插入到第一次出现的 `\newblock` 的后面）。默认情况下，该模板生成一换行符（line break）并新启一段落（paragraph）。默认颜色是普通文本的颜色（normal text）。

Beamer-Template/-Color/-Font `bibliography entry location`

即：

Beamer-Template/-Color/-Font 参考书目条日期刊

该模板插入到参考书目条目（第二个 `\newblock` 命令）的期刊的前面。默认情况下，该模板新启一个段落（paragraph）。默认颜色是结构色（structure color）的一个略透明的版本（slightly transparent version）。

Beamer-Template/-Color/-Font bibliography entry note

即:

Beamer-Template/-Color/-Font 参考书目条目笔记

该模板插入到参考书目条目的结束处的笔记文本 (note text) 的前面 (插入到第三个 `\newblock` 命令之前)。默认情况下, 该模板新启一个段落 (paragraph)。默认颜色是结构色 (structure color) 的一个略透明的版本 (slightly transparent version)。

```
\bibitem<⟨overlay specification⟩>[⟨citation text⟩]{⟨label name⟩}
```

即:

```
\bibitem<⟨叠层规则⟩>[⟨引文文本⟩]{⟨标签名⟩}
```

在主体呈现文本 (main presentation text) 中用 `\cite{⟨label name⟩}` 命令引用条目时, 将 `⟨citation text⟩` 插入。对于 BEAMER 演示稿, 经常出现这种情况。

使用 `\newblock` 命令分隔作者姓名、标题、引用的书籍/期刊 (book/journal)、其它笔记 (notes)。如果提供了 `⟨overlay specification⟩`, 只会在指定的幻灯片中显示条目。

举例:


```
\bibitem[Dijkstra, 1982]{Dijkstra1982}
  E.~Dijkstra.
  \newblock Smoothsort, an alternative for sorting in situ.
  \newblock {\em Science of Computer Programming}, 1(3):223--233, 1982.
```

Beamer-Template/-Color/-Font bibliography item



即:




Beamer-Template/-Color/-Font 参考书目条目

Color/font 父模板: item

该模板用于生成参考书目的条目 (bibliography item)。不像标准的 L^AT_EX, 参考书目默认的模板不会在参考书目的每一条目之前重复 (repeat) 引文文本 (citation text) (如 “[Dijkstra, 1982]”)。而是绘制一个可爱的小论文图标 (article symbol)  ³²。原因是观众不会想起任何简短的引文文本直到演讲结束。

下面的模板选项是预定好的:

- **[default]** 绘制一个可爱的小论文图标 (article symbol)  作为参考文献。可将此用于期刊论文 (journal articles)、书籍的一部分如会议论文集 (conference proceedings)、或科技报告等。
- **[article]** 默认使用 Alias。
- **[book]** 绘制一个可爱的小书籍图标 (book icon)  作为参考文献。

³²更改这些图标的语法如下: 在 thebibliography 环境中, `\setbeamertemplate{bibliography item}[book]` 生成小书籍图标 ; `\setbeamertemplate{bibliography item}[article]` 生成小论文图标 ; `\setbeamertemplate{bibliography item}[triangle]` 生成小三角形图标 。也可以用 `\beamertemplatebookbibitem` 生成小书籍图标; 用 `\beamertemplatearticlebibitem` 生成小论文图标; 用 `\beamertemplatetextbibitem` 用序号作图标; 用 `\beamertemplatearrowbibitem` 生成箭头 (小三角形) 图标。

- `[online]` 绘制一个漂亮的小球图标 (globe icon) 作为参考文献。网站 (websites) 等常用该图标。
- `[triangle]` 绘制一个小三角形 (triangle) ▶ 作为参考文献。这更多地用于标准的常规列表条目 (itemize items)。
- `[text]` 使用参考文献文本 (如 “[Dijkstra, 1982]”) 作为参考文献文本 (reference text)。这样做时必须明白我们在干什么。

下面的插入物 (insert) 可用于该模板:

- `\insertbiblabel` 插入当前的引文标签 (citation label)。

10.7 添加附录

使用 `\appendix` 命令可以在我们的演讲中添加一个附录 (appendix)。我们可以将在我们的演示稿中不想提到的帧和整个小节放入到附录中, 但这些帧和小节对回答问题却很有用。本质上, `\appendix` 命令开始一个名为 `\appendixname` 的部分 (part)。然而, 它也建立特定的超链接。和其它的部分一样, 附录独立于我们的实际演讲。

`\appendix<mode specification>`

即:

`\appendix<模式规则>`

在指定的模式中开始附录。用在该命令后的所有的帧、所有的 `\subsection` 命令、所有的 `\section` 命令都不会作为普通导航条 (navigation bars) 的一部分显示出来。

举例:

```
\begin{document}
\frame{\titlepage}
\section*{Outline}
\frame{\tableofcontents}
\section{Main Text}
\frame{Some text}
\section*{Summary}
\frame{Summary text}

\appendix
\section{\appendixname}
\frame{\tableofcontents}
\subsection{Additional material}
\frame{Details}
\frame{Text omitted in main talk.}
\subsection{Even more additional material}
\frame{More details}
\end{document}
```

11 组建演示稿: 交互的全局结构

11.1 添加超级链接和按钮

要在演讲结构中创建期望的非线性跳转 (anticipated nonlinear jumps), 我们可以在演示稿中添加超链接 (hyperlinks)。一个超链接是这样的文本 (通常呈现为一个按钮), 当我们点击它时, 演讲将跳转到其它幻灯片。要创建这样的按钮, 可通过下述三步实现

1. 使用 `\hypertarget` 命令或 (以前的) `\label` 命令指定一个跳转的目标。有时, 可以跳过这一步, 请继续下面的第二步。
2. 使用 `\beamerbutton` 命令或类似的命令生成按钮。点击这样生成的按钮没有任何效果。
3. 将按钮放在 `\hyperlink` 命令内。现在点击该按钮将跳转到目标处。

```
\hypertarget<<overlay specification>>{<target name>}{<text>}
```

即:

```
\hypertarget<<叠层规则>>{<目标名>}{<文本>}
```

如是提供了 `<overlay specification>`, 在指定的幻灯片中, 链接跳转到 `<target name>` 的目标就是 `<text>`。在其它幻灯片中, 文本以普通方式显示。注意无任何时候我们在拥有多张幻灯片的帧上使用该命令时, 我们都必须对 `\hypertarget` 命令添加叠层规则 (然而, `pdflatex` 正抱怨我们在不同的幻灯片中定义了相同的目标)。

举例:

```
\begin{frame}
  \begin{itemize}
    \item<1-> First item.
    \item<2-> Second item.
    \item<3-> Third item.
  \end{itemize}

  \hyperlink{jumptosecond}{\beamerbutton{Jump to second slide}}
  \hypertarget<2>{jumptosecond}{}
\end{frame}
```

ARTICLE 要在论文 (article) 模式中使用该命令, 我们必须在导言区声明 `\usepackage[hyperref]{beamerarticle}` 或 `\usepackage{hyperref}`。

`\label` 命令创建一个超链接目标 (hypertarget) 是其副作用 (side-effect)。`\frame` 命令的 `label=<name>` 选项为帧的每一张幻灯片创建一个名为 `<name><slide number>` 的标签是其副作用。因此, 上述例子可以写成:

```
\begin{frame}[label=threeitems]
  \begin{itemize}
    \item<1-> First item.
    \item<2-> Second item.
    \item<3-> Third item.
  \end{itemize}
\end{frame}
```

```
\end{itemize}
```

```
\hyperlink{threeitems<2>}{\beamerbutton{Jump to second slide}}
\end{frame}
```

下面的命令简述了按钮的作用。

```
\beamerbutton{<button text>}
```

即:

```
\beamerbutton{<按钮文本>}
```

绘制一帶有 *<button text>* 的按钮。

举例: `\hyperlink{somewhere}{\beamerbutton{Go somewhere}}`

ARTICLE 在论文 (article) 模式中, 该命令 (和下面的命令) 只插入它们的参数。

Beamer-Template/-Color/-Font `button`

即:

Beamer-Template/-Color/-Font `按钮`

当调用 `\beamerbutton` 命令时, 该模板用于生成按钮。在该模板内, 我们可以使用 `\insertbuttontext` 命令插入要传递给 `\beamerbutton` 的参数。

下面的模板选项是预定好的:

- **[default]** 排版一圆角按钮。使用 `BEAMER-color button` 的前景色和背景色, 也会使用 `BEAMER-font button`。按钮边界获得 `BEAMER-color button border` 的前景色。

下面的插入物 (inserts) 对该元素是有用的:

- `\insertbuttontext` 插入当前按钮的文本。在“Goto-Buttons”(见下)之内该文本的前面缀以插入物 `\insertgotosymbol`, 对于跳转 (skip) 和返回 (return) 按钮, 情况相同。
- `\insertgotosymbol` 该文本插入到跳转按钮 (goto buttons) 的开始处。重定义该命令可以更改符号 (symbol)。

举例: `\renewcommand{\insertgotosymbol}{\somearrowcommand}`

- `\insertskipsymbol` 该文本插入到跳转按钮 (skip buttons) 的开始处。
- `\insertreturnsymbol` 该文本插入到返回按钮 (return buttons) 的开始处。

Beamer-Color `button border`

即:

Beamer-Color `按钮边界`

该颜色的前景色用于生成按钮边界的颜色。

```
\beamerbutton{<button text>}
```

即:

`\beamertobutton{<按钮文本>}`

绘制带有 `<button text>` 的按钮。在该文本的前面，会插入一个小的标记 (symbol) (通常是一个指向右边的箭头)，指明点击该按钮后将跳转到演示稿的其它“区域”。

举例: `\hyperlink{detour}{\beamertobutton{Go to detour}}`

`\beamerskipbutton{<button text>}`

即:

`\beamerskipbutton{<按钮文本>}`

绘制带有 `<button text>` 的按钮。在该文本的前面，插入一个双向的箭头。点击该按钮后将跳过 (skip over) 演讲定义明确 (well-defined) 的部分。

举例:

```
\frame{
  \begin{theorem}
    ...
  \end{theorem}

  \begin{overprint}
  \onslide<1>
    \hfill\hyperlinkframestartnext{\beamerskipbutton{Skip proof}}
  \onslide<2>
    \begin{proof}
      ...
    \end{proof}
  \end{overprint}
}
```

`\beamerreturnbutton{<button text>}`

即:

`\beamerreturnbutton{<按钮文本>}`

绘制带有 `<button text>` 的按钮。插入一个小的标记 (symbol) (通常是一个指向左边的箭头)。点击该按钮后将绕道返回 (return)。

举例:

```
\frame<1>[label=mytheorem]
{
  \begin{theorem}
    ...
  \end{theorem}
}
```

```

\begin{overprint}
\onslide<1>
  \hfill\hyperlink{mytheorem<2>}{\beamergotobutton{Go to proof details}}
\onslide<2>
  \begin{proof}
    ...
  \end{proof}
  \hfill\hyperlink{mytheorem<1>}{\beamerreturnbutton{Return}}
\end{overprint}
}
\appendix
\againframe<2>{mytheorem}

```

要使用按钮“可点击”，我们必须将它放在像 `\hyperlink` 这样的命令内。`\hyperlink` 命令是 `hyperref` 宏包一个标准命令。我们可以使用 BEAMER 文档类定义的其它的超链接命令。

`\hyperlink<<overlay specification>>{(target name)}{(link text)}<<overlay specification>>`

即：

`\hyperlink<<叠层规则>>{(目标名)}{(链接文本)}<<叠层规则>>`

可能只会给出一个 `<overlay specification>`。以普通的方式排版 `<link text>`。如果我们点击该文本的任何地方，将跳转到使用了带 `<target name>` 参数的 `\hypertarget` 命令的幻灯片。如果提供了 `<overlay specification>`，在无叠层规则的幻灯片（non-specified slides）中超链接（包括 `<link text>`）被完全抑制。

下面的命令有一个预定义的目标，然而，它们之所作为极像 `\hyperlink`。特别是，它们也接受叠层规则，更多的是在末尾而不是在开始接受叠层规则。

`\hyperlinkslideprev<<overlay specification>>{(link text)}`

即：

`\hyperlinkslideprev<<叠层规则>>{(链接文本)}`

点击文本后往回跳一张幻灯片（jumps one slide back）。

`\hyperlinkslidenext<<overlay specification>>{(link text)}`

即：

`\hyperlinkslidenext<<叠层规则>>{(链接文本)}`

点击文本后跳转到下一张幻灯片（jumps one slide forward）。

`\hyperlinkframestart<<overlay specification>>{(link text)}`

即：

`\hyperlinkframestart<<叠层规则>>{(链接文本)}`

点击文本后跳转到当前帧的第一张幻灯片。

`\hyperlinkframeend<<overlay specification>>{\link text}`

即:

`\hyperlinkframeend<<叠层规则>>{\link text}`

点击文本后跳转到当前帧的最后一张幻灯片。

`\hyperlinkframestartnext<<overlay specification>>{\link text}`

点击文本后跳转到下一帧的第一张幻灯片。

`\hyperlinkframeendprev<<overlay specification>>{\link text}`

即:

`\hyperlinkframestartnext<<叠层规则>>{\link text}`

点击文本后跳转到前一帧的最后一张幻灯片。

前四条命令中的“frame”可以用“subsection”替换，也可以用“section”替换。

`\hyperlinkpresentationstart<<overlay specification>>{\link text}`

即:

`\hyperlinkpresentationstart<<叠层规则>>{\link text}`

点击文本后跳转到演示稿的第一张幻灯片。

`\hyperlinkpresentationend<<overlay specification>>{\link text}`

即:

`\hyperlinkpresentationend<<叠层规则>>{\link text}`

点击文本后跳转到演示稿的最后一张幻灯片。这不包含附录（appendix）。

`\hyperlinkappendixstart<<overlay specification>>{\link text}`

即:

`\hyperlinkappendixstart<<叠层规则>>{\link text}`

点击文本后跳转到附录的第一张幻灯片。如果没有附录，将跳转到文档的最后一张幻灯片。

`\hyperlinkappendixend<<overlay specification>>{\link text}`

即:

`\hyperlinkappendixend<<叠层规则>>{\link text}`

点击文本后跳转到附录的最后一张幻灯片。

`\hyperlinkdocumentstart<<overlay specification>>{\link text}`

即:

```
\hyperlinkdocumentstart<<叠层规则>>{<链接文本>}
```

点击文本后跳转到演示稿的第一张幻灯片。

```
\hyperlinkdocumentend<<overlay specification>>{<link text>}
```

即:

```
\hyperlinkdocumentend<<叠层规则>>{<链接文本>}
```

点击文本后跳转到演示稿的最后一张幻灯片, 或者, 如果有附录, 跳转到附录的最后一张幻灯片。

11.2 在后面的某点重复某帧

有时我们可能希望帧的某些幻灯片在主体演讲 (main talk) 中显示, 但该帧的某些“补充的”幻灯片只在附录中显示。这时, `\againframe` 命令很有用。

```
\againframe<<overlay specification>>[<<default overlay specification>>] [<options>]{<name>}
```

即:

```
\againframe<<叠层规则>>[<<默认的叠层规则>>] [<选项>]{<名称>}
```

PRESENTATION 重新开始一个帧, 该帧是以前用带 `label=<name>` 选项的 `\frame` 命令创建的。我们要使用该选项, 仅在帧内“手工”放置一个标签是不够的。可以使用该命令“继续”一个被其它帧中断的帧。该命令的作用是调用带有 `<overlay specification>`、`<default overlay specification>` (如果提供了)、`<options>` (如果提供了) 和原来帧内容的 `\frame` 命令。

举例:

```
\frame<1-2>[label=myframe]
{
  \begin{itemize}
    \item<alert@1> First subject.
    \item<alert@2> Second subject.
    \item<alert@3> Third subject.
  \end{itemize}
}

\frame
{
  Some stuff explaining more on the second matter.
}

\againframe<3>{myframe}
```

上述代码的作用是创建四张幻灯片。在最初的两张中, 条目 1 和 2 高亮显示。第三张幻灯片包含文本“Some stuff explaining more on the second matter.”。第四张幻灯片除第三点为高亮显示外和最初的两张完全相同。

举例：

```
\frame<1>[label=Cantor]
{
  \frametitle{Main Theorem}

  \begin{Theorem}
     $\alpha < 2^\alpha$  for all ordinals  $\alpha$ .
  \end{Theorem}

  \begin{overprint}
    \onslide<1>
      \hyperlink{Cantor<2>}{\beamergotobutton{Proof details}}

    \onslide<2->
      % this is only shown in the appendix, where this frame is resumed.
      \begin{proof}
        As shown by Cantor, ...
      \end{proof}

      \hfill\hyperlink{Cantor<1>}{\beamerreturnbutton{Return}}
    \end{overprint}
}

...
\appendix

\againframe<2>{Cantor}
```

在该例子中，证明的细节遵从（defer）附录中的一幻灯片。会创建超链接，因此可以跳转到证明（proof）并可以返回。

ARTICLE 在论文（article）模式中会忽略该命令。

LYX 使用样式“AgainFrame”插入一个 `\againframe`。 $\langle label name \rangle$ 是跟随样式名（style name）的文本， $\langle label name \rangle$ 不是放在 $\text{T}_\text{E}_\text{X}$ -模式中。然而，必须在 $\text{T}_\text{E}_\text{X}$ -模式中给定一个叠层规则，并放在标签名（label name）之前。

11.3 添加预期的缩放

当我们有一很复杂的图形，因为事实上，所有复杂细节值得（merit）解释，从而我们不希望简化该图形，这时，就很需要预期的缩放（Anticipated zooming）。这可以用 `\framezoom` 命令实现。当我们点击帧的某区域时将缩小（zoom out）该区域。我们也可以解释细节。点击缩小了的图形将复原到最初的图形。

```
\framezoom<\langle button overlay specification \rangle><\langle zoomed overlay specification \rangle>[\langle options \rangle]
(\langle upper left x \rangle, \langle upper left y \rangle) (\langle zoom area width \rangle, \langle zoom area depth \rangle)
```

即：

`\framezoom`<按钮叠层规则><缩放叠层规则>[<选项>](<上左 x >,<上左 y >)(<缩放区域宽>,<缩放区域高>)

必须在帧的开始处的某地方给出该命令。给出该命令后，据帧的当前幻灯片是否应用了 `<button overlay specification>` 或 `<zoomed overlay specification>`，将发生两件不同的事情。这些叠层规则不会重叠 (overlap)。

如果应用了 `<button overlay specification>`，在帧内将创建一可点击区域。该区域大小由 `<zoom area width>` 和 `<zoom area depth>` 设定，它们是两个标准的 $\text{T}_{\text{E}}\text{X}$ 尺寸（像 `1cm` 或 `20pt`）。该区域的左上角由 `<upper left x >` 和 `<upper left y >` 设定，它们也是两个标准的 $\text{T}_{\text{E}}\text{X}$ 尺寸。它们的测量与帧的首个普通文本的位置有关。因此，(0pt, 0pt) 位置就是普通文本（除顶部导航区、帧标是）的起始处。

默认情况下，按钮是可点击的，但这并不意味着任何特殊情况下它都是可点击的。我们可以用下面的 `<option>` 在按钮周围绘制一个边框：

- `border=<width in pixels>` 将在指定的按钮区域的周围绘制一个边框。默认的宽度是 1 pixel。该按钮的颜色是 `hyperref` 的 `linkbordercolor`。默认情况下 BEAMER 将该颜色设为 50% 灰。要改变这一点，我们可以用下面这个命令：

`\hypersetup{linkbordercolor={<red> <green> <blue>}}`，在这里，`<red>`、`<green>`、`<blue>` 的取值是 0 和 1 之间。

当我们点击用上述方法创建的按钮时，查看器应用程序 (viewer application) 将超跃 (hyperjump) 到通过 `<zoomed overlay specification>` 指定的第一帧处。对于应用了该叠层规则的幻灯片，将发生下面的事情：

和前面指定的区域大小一样的区域将被“缩小”以容纳帧的整个普通文本区域。和其它的东西一样，侧栏、顶部导航区、底部导航区、甚至帧标题保持它们的正常大小。缩放 (zooming) 以这样的方式完成，即整个指定的区域完全显示。如果该区域的长宽比和有用的 (available) 的文本区域不允许，会保持正确的长宽比，缩放区域会尽可能显示指定的区域。

在整个文本区域（它包含了缩放区域）的后面，放置了一个巨大的不可见的“背后 (Back)”按钮。因此，点击文本的任何区域将复原到初始（未缩放）的图形。

我们可以在一个帧内指定多个缩放区域。如此，我们应指定不同的 `<zoomed overlay specification>`，但可以指定相同的 `<button overlay specification>`。我们不可以嵌套 (nest) 缩放，从某种意义上说就是，在某 `<zoomed overlay specification>` 中的幻灯片中不可以有一个缩放按钮。然而，我们可重叠 (overlap) 甚至嵌套 `<button overlay specification>`。当点击一属于多个按钮的区域时，最后被点击的按钮将会“赢”（应该是最小的一个）。

如果我们不想在缩小的幻灯片中显示帧标题 (Frame Title)，可以在 `\frametitle` 命令中添加一叠层规则，该叠层规则只抑制幻灯片的标题。也可以用 `plain` 选项，使缩小的幻灯片填满整个页面

举例：一个简单的例子

```
\begin{frame}
  \frametitle{A Complicated Picture}

  \framezoom<1><2>(0cm,0cm)(2cm,1.5cm)
  \framezoom<1><3>(1cm,3cm)(2cm,1.5cm)
  \framezoom<1><4>(3cm,2cm)(3cm,2cm)

  \pgfimage[height=8cm]{complicatedimagefilename}
\end{frame}
```

举例：一个更复杂的例子，缩放部分填满整个帧。

```
\begin{frame}<1>[label=zooms]
  \frametitle<1>{A Complicated Picture}

  \framezoom<1><2>[border](0cm,0cm)(2cm,1.5cm)
  \framezoom<1><3>[border](1cm,3cm)(2cm,1.5cm)
  \framezoom<1><4>[border](3cm,2cm)(3cm,2cm)

  \pgfimage[height=8cm]{complicatedimagefilename}
\end{frame}
\againframe<2->[plain]{zooms}
```

12 组建演示稿：局部结构

L^AT_EX 提供了不同命令用于组建正文“局部”（text “locally”），如通过 `itemize` 环境。这些环境在 BEAMER 文档类中同样适应，虽然它们的外观会有点变化。而且，BEAMER 文档类还可以定义新命令，这有助于我们组建正文。

12.1 常规、排序、解说

用于创建列表（list）的预定义环境有三种，即排序或列举（`enumerate`）、常规或项目（`itemize`）、解说或描述（`description`）。前面两种可以嵌套（nest）至第三层，但嵌套至这样的深度将使幻灯片完全不可读。条目（item）之间的间距可以人为设定³³。

`\item` 命令是叠层规则感知的（overlay specification-aware）。如果提供了叠层规则，条目（item）只会在指定的幻灯片中显示，请参考下面的例子。如果 `\item` 命令带有可选的参数和叠层规则，那么叠层规则可以放在前面 `\item<1>[Cat]`，也可以放在后面如 `\item[Cat]<1>`。

```
\begin{frame}
  There are three important points:
  \begin{enumerate}
    \item<1-> A first one,
    \item<2-> a second one with a bunch of subpoints,
      \begin{itemize}
        \item first subpoint. (Only shown from second slide on!).
        \item<3-> second subpoint added on third slide.
        \item<4-> third subpoint added on fourth slide.
      \end{itemize}
    \item<5-> and a third one.
  \end{enumerate}
\end{frame}
```

```
\begin{itemize}[<<default overlay specification>>]
  <environment contents>
\end{itemize}
```

即：

```
\begin{itemize}[<<默认的叠层规则>>]
  <environment contents>
\end{itemize}
```

用于显示一个条目没有特定顺序的列表。在该环境中，用 `\item` 命令生成每一个条目。

如果给定了可选的参数 `<default overlay specification>`，在每一 `\item` 命令（没有附加叠层规则）出现处，将使用 `<default overlay specification>`。通过设定该规则为增加的（incremental）叠层规则（请参考第 9.6.4

³³BEAMER 设置 item 之间的间距的方法有二：

- ①使用 `\vspace` 命令：`\begin{itemize} \item item1 \vspace{.5cm} \item item2 \end{itemize}`
- ②使用 `\setlength` 命令：`\begin{itemize} \setlength{\itemsep}{.5cm} \item item1 \item item2 \end{itemize}`

节)，我们就可以完成象条目逐步显示这样的事情了。子环境（subenvironment）会继承 *(default overlay specification)*。自然地，我们可以在局部把它设置为 `<1->`。

举例：

```
\begin{itemize}
\item This is important.
\item This is also important.
\end{itemize}
```

举例：

```
\begin{itemize}[<+>]
\item This is shown from the first slide on.
\item This is shown from the second slide on.
\item This is shown from the third slide on.
\item<1-> This is shown from the first slide on.
\item This is shown from the fourth slide on.
\end{itemize}
```

举例：

```
\begin{itemize}[<+ | alert@+>]
\item This is shown from the first slide on and alerted on the first slide.
\item This is shown from the second slide on and alerted on the second slide.
\item This is shown from the third slide on and alerted on the third slide.
\end{itemize}
```

举例：

```
\newenvironment{mystepwiseitemize}{\begin{itemize}[<+ | alert@+>]}\end{itemize}
```

LYX 不幸的是，当前我们无法指定 `itemize` 环境的可选参数。但我们可以在该环境之前使用 `\beamerdefaultoverlayspecification` 命令以获得预期效果。

`itemize` 列表的外观由几个模板（template）支配着。第一个模板与小标记（little marker）的排版方式有关：

Parent Beamer-Template `itemize items`

即：

Parent Beamer-Template `itemize` 列表条目

这个模板是父模板（parent template），它的子模板是 `itemize item`、`itemize subitem`、`itemize subsubitem`。这意味着如果在该模板上我们使用 `\setbeamertemplate` 命令，该命令会被上述子模板（带有相同的参数）代替。

下面的模板选项是预定好的：

- **[default]** 默认的条目标记（item marker）是一个小三角形（triangle），它的颜色是 `BEAMER-color itemize item` 的前景色 [或者，对于子条目（subitems），是 `itemize subitem` 等]。注意这些颜色在特定的情况下（如在 `example` 块中或在 `alertenv` 环境中）会自动改变。
- **[triangle]** 默认为 Alias。
- **[circle]** 用小圆（little circles）或小点（little dots）作为条目标记。

- `[square]` 用小方形 (little squares) 作为条目标记。
- `[ball]` 用小球 (little balls) 作为条目标记。

Beamer-Template/-Color/-Font `itemize item`

即:

Beamer-Template/-Color/-Font `itemize` 列表条目

Color/font 父模板: `item`

该模板 (用 `item` 代替 `items`) 控制着第一层条目 (first-level item) 前面的标记 (marker) 是如何排版的。“第一层”是针对嵌套来说的。请参考 `itemize items` 模板可能会给出的 `<options>`。

当插入该模板时, 会安装 `BEAMER-font` 和 `-color itemize item`。通常, 当绘制某些特定的符号时, 模板会忽略 `font`, 如果 `\item` 命令给定了一个可选的参数 (像 `\item[First]` 这样), 该 `font` 可能很重要。

`font` 和 `color` 继承自 `item font` 和 `color`, 在这一节的末尾会解释 `item font` 和 `color`。

Beamer-Template/-Color/-Font `itemize subitem`

即:

Beamer-Template/-Color/-Font `itemize` 列表小条目

Color/font 父模板: `subitem`

和 `itemize item` 相似, 该模板只是针对第二层的条目。嵌套在一排序列表 (`enumerate`) 中的常规列表 (`itemize`) 的一个条目就是第二层的条目。

Beamer-Template/-Color/-Font `itemize subsubitem`

即:

Beamer-Template/-Color/-Font `itemize` 列表小小条目

Color/font 父模板: `subsubitem`

和 `itemize item` 相似, 该模板只是针对第三层的条目。

```
\begin{enumerate}[<<default overlay specification>][<mini template>]
  <environment contents>
\end{enumerate}
```

即:

```
\begin{enumerate}[<<默认的叠层规则>][<小模板>]
  <environment contents>
\end{enumerate}
```

用于显示一个具有顺序的一列条目。在该环境中, 每一条目必需使用一条 `\item` 命令。默认情况下, 在每一条目之前是一个渐增的阿拉伯数字, 在阿拉伯数字和条目之间是一个小圆点 (象 “1.” 和 “2.” 这样)。可以通过指定不同的模板来改变这种显示方式, 详参下面。

第一个可选参数 *(default overlay specification)* 的作用与 `itemize` 环境的相同。它会被 *(default overlay specification)* 中的 `<` 符号“检测”到。因此，如果只有一个可选参数且该参数不是始于 `<`，那么它被认为是 *(mini template)*。

(mini template) 的语法和 `enumerate` 宏包（会自动加载该宏包）中的小模板（mini templates）的语法相同。粗略地讲，会在每一条目之前显示 *(mini template)* 的文本，但小模板中的 `1` 会被当前的条目号（item number）代替，字母 `A` 会被大写的字母 *i*-th 代替，字母 `a`、`i`、和 `I` 会被相应的小写的字母代替。因此，小模板 `(i)` 将生成条目号 `(i)`、`(ii)`、`(iii)`、`(iv)` 等等。小模板 `A.)` 将生成条目号 `A.)`、`B.)`、`C.)`、`D.)` 等等。有关小模板的细节，请参阅 `enumerate` 包的文档。注意，同样有一模板控制着小模板（mini template）的外观。

举例：

```
\begin{enumerate}
\item This is important.
\item This is also important.
\end{enumerate}
```

```
\begin{enumerate}[(i)]
\item First Roman point.
\item Second Roman point.
\end{enumerate}
```

```
\begin{enumerate}[<+>][(i)]
\item First Roman point.
\item Second Roman point, uncovered on second slide.
\end{enumerate}
```

ARTICLE 要使用 *(mini template)*，必需先加载 `enumerate` 宏包。

LYX 系统规定的参数和 `itemize` 的相同。

Parent Beamer-Template `enumerate items`

即：

Parent Beamer-Template `enumerate` 列表条目

和 `itemize items` 相似，该模板是一父模板，它的子模板是 `enumerate item`、`enumerate subitem`、`enumerate subsubitem`。这些模板控制着排序列表（enumeration）的文本（序号³⁴）是如何排版的。

下面的模板选项是预定好的：

- **[default]** 默认的排序列表标记是，第一层使用 `1.`，`2.`，`3.`；第二层用 `1.1`，`1.2`，`1.3`；第三层用 `1.1.1`，`1.1.2`，`1.1.3`。

³⁴通过下面的方法改变该序号的计数器：

```
\begin{enumerate}
\addtocounter{enumi}{3}    %% 改变列表序号计数器
.....
\end{enumerate}
```

- `[circle]` 在小圆 (circle) 中放置序号, 颜色取自 `item projected` 或 `subitem projected` 或 `subsubitem projected`。
- `[square]` 在小方括号 (square) 中放置序号。
- `[ball]` 将序号“投影”在小球 (ball) 上。

Beamer-Template/-Color/-Font enumerate item

即:

Beamer-Template/-Color/-Font enumerate 列表条目

该模板控制着如何排版第一层条目 (first-level item) 前的数字 (number)。这里所说的层指的是排序嵌套 (enumeration nesting) 的层次。因此, 嵌在一个 `itemize` 内的一个 `enumerate` 是第一层的 `enumerate` (但它使用第二层的 `itemize/enumerate body`)。

当插入该模板时, 也就会插入 `BEAMER-font` 和 `-color enumerate item`。

下面的命令对该模板有用:

- `\insertenumlabel` 插入顶层 (top-level) enumeration 的当前数字 (number) (阿拉伯数字)。该插入物 (insert) 在下面的两个模板中也有效。

Beamer-Template/-Color/-Font enumerate subitem

即:

Beamer-Template/-Color/-Font enumerate 小条目

和 `enumerate item` 相似, 它只适应于第二层的条目 (second-level items)。

- `\insertsubenumlabel` 插入第二层 (second-level) enumeration 的当前数字 (阿拉伯数字)。

举例:

```
\setbeamertemplate{enumerate subitem}{\insertenumlabel-\insertsubenumlabel}
```

Beamer-Template/-Color/-Font enumerate subsubitem

即:

Beamer-Template/-Color/-Font enumerate 小小条目

和 `enumerate item` 相似, 它只适应于第三层的条目 (third-level items)。

- `\insertsubsubenumlabel` 插入第二层排序 (second-level enumeration) 的当前数字 (阿拉伯数字)。

Beamer-Template/-Color/-Font enumerate mini template

即:

Beamer-Template/-Color/-Font enumerate 小模板

该模板用于排版由 `mini template` (小模板) 生成的数字。

- `\insertenumlabel` 插入由该 `mini template` 模板生成的当前数字, 例如, 如果 `(mini template)` 是 (i), 该命令用在第四个条目内, `\insertenumlabel` 将生成 (iv)。

下面的模板控制如何排版 `itemize` 或 `enumerate` 的主体 (*body*)。

Beamer-Template `itemize/enumerate body begin`

即:

Beamer-Template `itemize/enumerate` 主体开始

该模板插入在第一层 `itemize` 或 `enumerate` 环境的开始处。而且, 在插入该模板之前, 会使用 `BEAMER-font` 和 `-color itemize/enumerate body`。

Beamer-Template `itemize/enumerate body end`

即:

Beamer-Template `itemize/enumerate` 主体结束

该模板插入在第一层 `itemize` 或 `enumerate` 环境的结束处。

也有用于第二或第三层 `itemize` 或 `enumerate` 的相应的模板, 如 `itemize/enumerate subbody begin`。

Parent Beamer-Template `items`

即:

Parent Beamer-Template 条目

该模板是 `itemize items` 和 `enumerate items` 的父模板。

举例: `\setbeamertemplate{items}[circle]` 将 `itemize` 或 `enumerate` 环境中的所有条目标记改成圆 (`circle`) (该圆具有相应的尺寸、颜色、字体)。

```
\begin{description}[<<default overlay specification>>][<long text>]
  <environment contents>
\end{description}
```

即:

```
\begin{description}[<<默认的叠层规则>>][<长文本>]
  <environment contents>
\end{description}
```

和 `itemize` 环境相似, 该环境用于显示一个列表, 该列表解释 (`explain`) 或定义 (`define`) 一些标签 (`labels`)。<long text> 的宽度用于设置缩进 (`indentation`)。通常, 可以选择 `description` 中最宽的标签将其复制并粘贴到这里。如果不给定该参数, 则使用默认的宽度, 可以使用带 `description width=<width>` 参数的 `\setbeamersize` 命令改变该默认的宽度。

至于 `enumerate`, <default overlay specification> 可以被一个开放的 < 侦测到。效果和 `enumerate`、`itemize` 相同。

举例:

```
\begin{description}
```

```

\item[Lion] King of the savanna.
\item[Tiger] King of the jungle.
\end{description}

```

```

\begin{description}[longest label]
\item<1->[short] Some text.
\item<2->[longest label] Some text.
\item<3->[long label] Some text.
\end{description}

```

举例：下面的例子和前一例子的效果相同：

```

\begin{description}[<+>] [longest label]
\item[short] Some text.
\item[longest label] Some text.
\item[long label] Some text.
\end{description}

```

LYX 因为不能在 LyX 中指定可选的参数，如果想指定宽度（width），则可以在环境之前马上使用下面的命令：

```

\setbeamersize{description width of={\langle text \rangle}}

```

举例：

```

\setbeamersize{description width of={longest label}}
\begin{description}
\item<1->[short] Some text.
\item<2->[longest label] Some text.
\item<3->[long label] Some text.
\end{description}

```

Beamer-Template/-Color/-Font description item

即：

Beamer-Template/-Color/-Font description 条目

模板用于排版 description 的条目。当调用该模板时，会安装 BEAMER-font 和 -color description item。

下面的模板选项是预定好的：

- **[default]** 默认情况下，会插入无任何修改的 description 的条目文本（item text）。

在该模板中有效的主要插入物是：

- **\insertdescriptionitem** 插入当前 description 条目的文本。

为简化更改条目的颜色或字体，不则类型的条目继承或使用下面的“一般（general）”BEAMER-color 和 fonts。

Beamer-Color/-Font item

即：

Beamer-Color/-Font 条目

Color 父模板: `local structure`

Font 父模板: `structure`

该 color/font 用作 (serves as) `itemize` 和 `enumerate` 环境条目的父本 (parent), 它也是目录条目 (items in table of contents) 的父本。因为它的颜色父本 (color parent) 是 `local structure`, `local structure` 的颜色改变则导致条目颜色的自动改变。

Beamer-Color/-Font `item projected`

即:

Beamer-Color/-Font 被投影的条目

Color/font 父模板: `item`

这是由模板使用的 `item` color 和 font 的一个特殊“版本”, 该模板生成 (render) 的条目带有文本 (如在一个 enumeration 中), 而该文本则“投影 (project)”到象圆球 (ball)、方块 (square) 或诸如此类的东西上。虽然普通的 (normal) `item` color 常有一个透明的背景 (transparent background), 但 `item projected` 通常有一个有颜色 (比如说白色) 的背景 (colored background)。

Beamer-Color/-Font `subitem`

即:

Beamer-Color/-Font 小条目

Color/font 父模板: `item`

和用于 `subitems` 的 `item` 相同, 也就是说, 用于缩进的第二层的条目。

Beamer-Color/-Font `subitem projected`

即:

Beamer-Color/-Font 被投影的小条目

Color/font 父模板: `item projected`

和用于 `subitems` 的 `item projected` 相同, 也就是说, 用于缩进的第二层的条目。

Beamer-Color/-Font `subsubitem`

即:

Beamer-Color/-Font 小小条目

Color/font 父模板: `subitem`

和用于 `subsubitems` 的 `subitem` 相同, 也就是说, 用于缩进的第三层的条目。

Beamer-Color/-Font `subsubitem projected`

即:

Beamer-Color/-Font 被投影的小小条目

Color/font 父模板: `subitem projected`

和用于 `subsubitems` 的 `subitem projected` 相同, 也就是说, 用于缩进的第三层的条目。

12.2 高亮显示

BEAMER 文档类预定义了用于高亮显示 (Highlighting) 文本的命令和环境。通过更改主题并使用这些命令可以很容易地改这文档的外观

```
\structure<<overlay specification>>{<text>}
```

即:

```
\structure<<叠层规则>>{<文本>}
```

给定的文本被标记为 (marked as) 结构 (structure) 的一部分, 也就是说, 它有助于观众了解演示稿的结构。如果给定了 *<overlay specification>*, 则该命令只作用于指定的幻灯片。

举例: `\structure{Paragraph Heading.}`

在内部 (Internally), 该命令将 文本 (*text*) 放入到一个 `structureenv` 环境内。

ARTICLE 以粗体文本 (bold text) 排版结构文本 (Structure text)。这可以通过修改模板来改变结构文本。

LYX 必须在 TeX-模式中插入该命令。

Beamer-Color/-Font structure

即:

Beamer-Color/-Font 结构

当排版结构化的文本 (structured text) 时使用该 color/font, 但它也作为其它颜色包括块顶部 (headings of blocks)、条目按钮 (item buttons)、标题 (titles) 的基础颜色 (basic color)。在大多数颜色主题中, 顶部导航区或底部导航区的导航元素的颜色衍生自 structure 的前景色。通过改变结构色 (structure color) 可以很容易改变演示稿的“基础颜色”, 这和普通文本 (normal text) 的颜色不同。请参考相关的颜色 `local structure` 及相关的字体 `tiny structure`。

在 `\structure` 命令内部, 会忽略颜色的背景 (background of the color), 但这对于其颜色继承自 `structure` 的元素来说不总是这样。因为不存在 `structure` 模板, 所以用 `structure begin` 和 `structure end` 代替。

Beamer-Color local structure

即:

Beamer-Color 局部结构

该颜色用于排版结构元素 (structural elements), 这样的结构元素会据“局部 (local) 环境”改变其颜色。例如, 一个 `itemize` 环境中的“按钮 (button)”据情况 (circumstances) 改变其颜色。如果它用在示例块 (example block) 内, 则它具有 `example text` 的颜色; 如果它当前用于“提醒 (alerted)”, 则它具有 `alerted text` 的颜色。由特定的环境设置该颜色, 该颜色用于排版像条目按钮 (item buttons) 这样的东西。因为该颜色用于条目 (items), 默认情况下 `item` 继承自该颜色, 条目会根据当前情况 (current situation) 自动改变它们的颜色。

如果我们自己的环境 (your own environment) 中的条目按钮和类似的结构元素具有不同的颜色, 则应改变这些环境内的颜色 `local structure`。

Beamer-Font `tiny structure`

即：

Beamer-Font 微结构

该特殊的字体用于“微小 (tiny)”结构文本 (structural text)。基本上，无论何时一个结构元素 (structural element) 使用微小字体 (tiny font) 时就会使用该字体。结构字体 (`structure`) 的微小版本是不合适的。例如，常常使用该字体的粗体版本。而且，有人希望用衬线的 (serif) 的小型大写 (smallcaps) 的结构文本，但仍用普通的 (normal) 无衬线 (sans-serif) 微小结构文本 (tiny structural text)。

```
\begin{structureenv}<<overlay specification>>
  <environment contents>
\end{structureenv}
```

即：

```
\begin{structureenv}<<叠层规则>>
  <environment contents>
\end{structureenv}

\structure 命令的环境版本 (environment version)
```

Beamer-Template `structure begin`

即：

Beamer-Template 结构开始

该文本插入到 `structureenv` 环境的开始处。

下面的模板选项是预定好的：

- `[default]`

`ARTICLE` 以粗体 (boldface) 排版文本。

Beamer-Template `structure end`

即：

Beamer-Template 结构结束

该文本插入到 `structureenv` 环境的结束处。

```
\alert<<overlay specification>>{<highlighted text>}
```

即：

```
\alert<<叠层规则>>{<高亮显示的文本>}
```

给定的文本高亮显示，典型地显示为红色。如果给定 `<overlay specification>`，则该命令只对指定的幻灯片有效。

举例: This is `\alert{important}`.

内部 (Internally), 该命令将高亮显示的文本 (*highlighted text*) 放入到一个 `alertenv` 环境中。

ARTICLE 以强调文本 (emphasized text) 的形式排版提醒文本 (Alerted text)。这可以通过修改模板来改变这一点, 请参考下面。

LYX 必须以 $\text{T}_\text{E}\text{X}$ -模式插入该命令 (这不是很方便)。

Beamer-Color/-Font alerted text

即:

Beamer-Color/-Font 提醒文本

当排版提醒文本时使用该 `color/font`。会在当前忽略背景。不存在 `alerted text` 模板, 但可以在提醒文本之前插入 `alerted text begin` 模板, 在提醒文本之后插入 `alerted text end` 模板。

```
\begin{alertenv}<<overlay specification>>
  <environment contents>
\end{alertenv}
```

即:

```
\begin{alertenv}<<叠层规则>>
  <environment contents>
\end{alertenv}
```

`\alert` 命令的环境版本。

Beamer-Template alerted text begin

即:

Beamer-Template 提醒文本开始

在 `alertenv` 环境的开始处插入文本。

下面的模板选项是预定好的:

- `[default]`

PRESENTATION 将颜色 `local structure` 更改成 `alerted text`。这可以使按钮或条目之类的东西具有和提醒文本颜色相同的颜色, 这看起来很舒服。请参考 `\structure` 命令。

ARTICLE 强调 (emphasize) 该文本。

Beamer-Template alerted text end

即:

Beamer-Template 提醒文本结束

该文本插入到 `alertenv` 环境的结束处。

12.3 块环境

BEAMER 文档类预定义了一个环境用于排版文本“块 (block)”，该文本块包含标题 (heading)。通过更改下面的模板就能很容易地改变块的外观：

Parent Beamer-Template blocks

即：

Parent Beamer-Template 块

更改父模板 (parent template) 就可以改变普通块 (normal blocks)、提醒块 (alerted blocks)、示例块 (example blocks)。

举例：`\setbeamertemplate{blocks}[default]`

举例：`\setbeamertemplate{blocks}[rounded][shadow=true]`

下面的模板选项是预定好的：

- `[default]` 为默认的设置，将块标题排版在其所在行。可以为块标题 (block title) 或块正文 (block body) 分别指定背景。对于提醒块 (alerted blocks) 和示例块 (example blocks)，可以使用相关的 BEAMER-colors 和 -fonts。
- `[rounded][<shadow=true>]` 生成“圆角 (rounded)”块。这意味着块的背景的角被“切掉”。如果给定了 `shadow=true` 选项，会在块的后面生成“阴影 (shadow)”。

```
\begin{block}<<action specification>>{(block title)}<<action specification>>
  <environment contents>
\end{block}
```

即：

```
\begin{block}<<行为规则>>{(块标题)}<<行为规则>>
  <environment contents>
\end{block}
```

可能只会给定一个 `<action specification>`。插入一个带有 `<block title>` 的块如定义 (definition) 或定理 (theorem)。如果 `<action specification>` 是父本，则给定的行为 (actions) 将会作用于指定的幻灯片，请参考第 9.6.3 节。下面的例子，定义 (definition) 将会从第 3 张幻灯片向前显示。

举例：

```
\begin{block}<3->{Definition}
  A \alert{set} consists of elements.
\end{block}
```

ARTICLE 块的名字有粗体显示。

LYX 必须在 T_EX-模式中给出块的参数。更精确地讲，在 T_EX-模式中，必需在参数的两侧分别放置左大括号 (opening brace) 和右大括号 (closing brace)。其中的文本可以用 L_AT_EX 排版。我们期望某天能改变这种状况。

Beamer-Template `block begin`

即：

Beamer-Template 块开始

在 $\langle environment contents \rangle$ 前面的块开始处插入该模板。在该模板中，可以通过下面的插入物 (insert) 访问块标题 (block title)：

- `\insertblocktitle` 将 $\langle block title \rangle$ 插入到模板。

当启动 (starts) 该模板时，不会安装指定的颜色或字体 (由于某些复杂的原因)。因此，该模板必须自己为标题安装正确的颜色和字体。

Beamer-Template `block end`

即：

Beamer-Template 块结束

在块的结束处插入该模板。

Beamer-Color/-Font `block title`

即：

Beamer-Color/-Font 块标题

该 `BEAMER-color/-font` 可用于排版块标题。因为不会自动安装 `color` 和 `font`，模板 `block begin` 必须自己安装 `color` 和 `font`。

默认的块模板 (block template) 和圆角版本 (rounded version) 接受 (honor) 该 `color` 的背景。

Beamer-Color/-Font `block body`

即：

Beamer-Color/-Font 块正文

该 `BEAMER-color/-font` 可用于排版块正文 (the body of the block)，即 $\langle environment contents \rangle$ 。至于 `block title`，必须通过模板 `block begin` 来安装 `color` 和 `font`。

```
\begin{alertblock}<<action specification>>{\langle block title \rangle}<<action specification>>
  \langle environment contents \rangle
\end{alertblock}
```

即：

```
\begin{alertblock}<<行为规则>>{\langle 块标题 \rangle}<<行为规则>>
  \langle environment contents \rangle
\end{alertblock}
```

插入一个提醒块 (alerted blocks)，其标题高亮显示 (Highlighting)。其行为和块 (block) 环境类似。

举例：


```

\begin{alertblock}{Wrong Theorem}
  $1=2$.
\end{alertblock}

```

ARTICLE 块名（block name）以粗体和强调显示。

LYX 其应用和 block 相似。

Beamer-Template **block alerted begin**

即：

Beamer-Template 提醒块开始

其应用和普通块（normal blocks）相似。

Beamer-Template **block alerted end**

即：

Beamer-Template 提醒块结束

其应用和普通块（normal blocks）相似。

Beamer-Color/-Font **block title alerted**

即：

Beamer-Color/-Font 提醒块标题

其应用和普通块（normal blocks）相似。

Beamer-Color/-Font **block body alerted**

即：

Beamer-Color/-Font 提醒块正文

其应用和普通块（normal blocks）相似。

```

\begin{exampleblock}<<action specification>>{\langle block title \rangle}<<overlay specification>>
  \langle environment contents \rangle
\end{exampleblock}

```

即：

```

\begin{exampleblock}<<行为规则>>{\langle 块标题 \rangle}<<叠层规则>>
  \langle environment contents \rangle
\end{exampleblock}

```

插入一个示例块（example blocks）。其所作所为和块（block）环境相类。

举例：在下面的例子中，块在第一张幻灯片中抑制了（它不会占用任何空间）。

```

\begin{exampleblock}{Example}<only@2->
  The set  $\{1,2,3,5\}$  has four elements.
\end{exampleblock}

```

ARTICLE 块名用斜体字显示。

LYX 其应用和 block 相同。

Beamer-Template `block example begin`

即：

Beamer-Template 示例块开始

其应用和普通块（normal blocks）相似。

Beamer-Template `block example end`

即：

Beamer-Template 示例块结束

其应用和普通块（normal blocks）相似

Beamer-Color/-Font `block title example`

即：

Beamer-Color/-Font 示例块标题

其应用和普通块（normal blocks）相似。

Beamer-Color/-Font `block body example`

即：

Beamer-Color/-Font 示例块正文

其应用和普通块（normal blocks）相似。

LYX 叠层规则（Overlay Specifications）必须在环境开始的右边给出，并且是在 $\text{T}_{\text{E}}\text{X}$ -模式中。

12.4 定理环境

BEAMER 文档类预定义了几个环境，如 `theorem`、`definition`、`proof`，这些预定义的环境可分别用于排版像定理（theorems）、定义（definition）、证明（proofs）这样的内容。预定的全部环境有：`theorem`、`corollary`、`definition`、`definitions`、`fact`、`example`、`examples` 等。预定的德语的（German）块环境（block environments）有：`Problem`、`Loesung`、`Definition`、`Satz`、`Beweis`、`Folgerung`、`Lemma`、`Fakt`、`Beispiel`、`Beispiele` 等。

如何使用这些预定义的环境的典型例子如下：

```
\begin{frame}
  \frametitle{A Theorem on Infinite Sets}

  \begin{theorem}<1->
    There exists an infinite set.
  \end{theorem}
```

```
\begin{proof}<2->
  This follows from the axiom of infinity.
\end{proof}
```

```
\begin{example}<3->[Natural Numbers]
  The set of natural numbers is infinite.
\end{example}
\end{frame}
```

下面只讲述英文版的预定义环境的用法。德文版的用法类似。

```
\begin{theorem}<<action specification>>[<additional text>]<<action specification>>
  <environment contents>
\end{theorem}
```

即：

```
\begin{theorem}<<行为规则>>[<附加的文本>]<<行为规则>>
  <environment contents>
\end{theorem}
```

插入一个定理 (theorem)。可能只会给出一个 *<action specification>*。如果给出，则在单词“定理 (Theorem)” (置于圆括号中，虽然可以通过更改模板来改变它) 后面显示 *<additional text>*。

定理的外观 (appearance) 由 `theorem begin` 和 `theorem end` 控制，请参阅后面的内容以获得它们是如何控制定理外观的。每一个定理置于块 (Block) 环境中，因此，也会应用块的模板 (templates for blocks)。

用于该环境的定理的样式 (theorem style) (此概念来自于 `amsthm`) 是 `plain`。在这个样式中，定理的主体 (body) 用斜体 (italics) 排版，定理的标题 (head) 用粗体 (bold) 排版，但这种状况常被模板覆盖 (overruled)。

如果 `envcountsect` 选项是以 `presentation` 模式中的类选项或以 `article` 模式中 `beamerarticle` 宏包的选项这样的形式给出，则给定理编号 (numbering) 局限于每一节 (section)，这些节带有节号 (section number)，这些节号的前面缀有定理编号 (theorem number)。然而，定理的编号在演示稿 (presentation) 或论文 (article) 中是连续的³⁵。我们推荐在 `article` 模式中使用该选项。

默认情况下，在 `presentation` 模式中，不会显示定理编号³⁶。

举例：

```
\begin{theorem}[Kummer, 1992]
  If  $\mathbb{A}^n$  is  $n$ -enumerable, then  $\mathbb{A}$  is recursive.
\end{theorem}
```

³⁵用如下命令定制定理 (显示“定理”，编号按照节标题独立编号)：

```
\newtheorem{mytht}{定理}[section]
\begin{mytht}
.....
\end{mytht}
```

³⁶在 `presentation` 模式中要开始使用定理编号，可以在导言区添加命令：`\setbeamertemplate{theorems}[numbered]`

```
\begin{theorem}<2->[Tantau, 2002]
  If  $\mathbb{A}^2$  is  $\mathbb{A}$ -fa-enumerable, then  $\mathbb{A}$  is regular.
\end{theorem}
```

LYX 如果出现，则必须在 T_EX-模式中在环境的开始处给出可选的参数和行为规则。

`corollary`、`fact`、`lemma` 等环境的所作所为和上述相同。

```
\documentclass[envcountsect]{beamer}
```

在每一节局部给定理、定义等编号。这样，第二节的第一个定理将显示为 Theorem 2.1（假定在前面的节中没有定义、引理、推论）。

```
\begin{definition}<<action specification>>[<additional text>]<<action specification>>
  <environment contents>
\end{definition}
```

即：

```
\begin{definition}<<行为规则>>[<附加的文本>]<<行为规则>>
  <environment contents>
\end{definition}
```

除使用定理样式 `definition` 之外，其它的所作所为与 `theorem` 环境相似。在 `definition` 样式中，定理主体用直立字体（upright font）排版。

`definitions` 环境的所作所为与 `definition` 环境相似。

```
\begin{example}<<action specification>>[<additional text>]<<action specification>>
  <environment contents>
\end{example}
```

即：

```
\begin{example}<<行为规则>>[<附加的文本>]<<行为规则>>
  <environment contents>
\end{example}
```

除使用定理样式 `example` 之外，其它所作所为与 `theorem` 环境相似。使用 `example` 定理样式的副作用是在一个 `exampleblock` 而不是 `block` 内放置 `<environment contents>`。

`examples` 环境的所作所为与 `example` 环境相似。

PRESENTATION 用于排版定理的默认模板会抑制定理编号（theorem number），即使该编号“可用于”排版（在所有预定义的环境中，它是默认的，但是，如果我们使用了 `\newtheorem*` 预定义了我们自己的环境，则编号不可用）。

ARTICLE 在 `article` 模式中，会自动给定理编号。通过指定 `envcountsect` 类选项，定理可以在每一节的局部进行编号，这是个好主意，除非常短的论文（article）外。

```
\begin{proof}<<action specification>>[<proof name>]<<action specification>>
  <environment contents>
```

```
\end{proof}
```

即:

```
\begin{proof}<<行为规则>>[<证明的名称>]<<行为规则>>
  <environment contents>
\end{proof}
```

排版一个证明 (proof)。如果给定了可选参数 $\langle proof name \rangle$ ，则它会替换单词“证明 (Proof)”。这和普通的定理 (normal theorems) 不同，在普通的定理中，可选参数是放在括号 (bracket) 中的。

在定理的末尾会显示一个 $\backslash qed$ 符号，除非我们在证明中 (这和在 `amsthm` 中极其相似) 已经声明了 $\backslash qedhere$ 。默认的 $\backslash qed$ 符号是一个不封闭的矩形 (open rectangle)。要完全抑制该符号，可以在导言区写入 $\backslash def\qedsymbol\{\}$ 。为获得封闭的矩形，请声明:

```
\setbeamertemplate{qed symbol}{\vrule width1.5ex height1.5ex depth0pt}
```

如果使用了 `babel` 和不同的语言，则文本“证明 (Proof)”会被选定语言的任何合适的内容替换。

举例:

```
\begin{proof}<2->[Sketch of proof]
  Suppose ...
\end{proof}
```

Beamer-Template/-Color/-Font qed symbol

即:

Beamer-Template/-Color/-Font qed 符号

在证明的末尾显示符号。

我们可以用下面的命令定义新的环境:

```
\newtheorem*{<environment name>}[<numbered same as>]{<head text>}[<number within>]
```

即:

```
\newtheorem*{<环境名>}[<与 ... 编号相同>]{<标题文本>}[<在 ... 内编号>]
```

该命令的使用方法与在 `amsthm` 宏包中的使用方法几乎相同 (事实是，该命令来自于 `amsthm` 宏包)，请参阅其文档。不同之处是，在 BEAMER 中用该命令声明的环境是叠层规则敏感的 (specification-aware)，而且，当排版时，会根据 BEAMER 的模板进行排版。

ARTICLE 在 `article` 模式中，用该命令声明的环境也是叠层规则敏感的 (specification-aware)。

举例: $\backslash newtheorem\{observation\}[theorem]\{Observation\}$

我们也可以使用 `amsthm` 的 $\backslash newtheoremstyle$ 命令定义新的定理样式 (theorem styles)。注意，定理的默认模板会忽略标题字体 (head font) 的任何设置，但会接受 (honor) 正文字体 (body font) 的设置。

如果我们希望像 `theorem` 这样以不同的方式 (例如，在每一节内进行编号) 定义环境，则可以使用下面的文档类选项使预定的环境失效:

```
\documentclass[notheorems]{beamer}
```

关闭像 `theorem` 这样的默认块的定义，但仍然加载 `amsthm` 并使定理对叠层敏感。

像 `beamerarticle` 宏包的选项一样，该选项也是可用的并具有相同效果。

ARTICLE 在 `article` 版本中，`amsthm` 宏包有时和文档类相冲突 (`clash`)。这时，我们可以用下面的选项以关闭加载 `amsthm`，该选项作为用于 `BEAMER` 的文档类选项和作为用于 `beamerarticle` 的宏包选项是可用的。

```
\documentclass[noamsthm]{beamer}
```

不加载 `amsthm` 也不加载 `amsmath`。像 `theorem` 或 `proof` 这样的环境将不可用。

```
\documentclass[noamssymb]{beamer}
```

不加载 `amssymb`。该选项主要便于加载专门的字体宏包。注意，如果使用了 `itemize` 环境，由必须定义 `\blacktriangleright`。

Parent Beamer-Template `theorems`

即：

Parent Beamer-Template 定理 s

该模板是 `theorem begin` 和 `theorem end` 的父模板，请参阅开始以获得和定理模板的设置相关的细节。

举例：`\setbeamertemplate{theorems}[numbered]`

下面的模板选项是预定好的：

- **[default]** 默认情况下，定理按以下排版：接受正文 (`body`) 字体的设置，忽略标题 (`head`) 字体的设置。不印出定理编号。
- **[normal font]** 除忽略正文 (`body`) 字体的设置外，其它和上述默认情况相似。这样，使用的字体也常常用于块 (`blocks`)。
- **[numbered]** 除在已编号的环境中印出定理编号外，其它和上述默认情况相似。
- **[ams style]** 该选项将定理放入 `block` 或 `exampleblock` 中，但排版定理和在 `amsthm` 中排版定理相似。这样，标题字体和正文字体依赖于排版 `theorem` 和编号 `theorems` 的设置。

Beamer-Template `theorem begin`

即：

Beamer-Template 定理开始

无论何时用 `\newtheorem` 命令声明要排版一个环境时，都会在环境开始处插入该模板，并在环境结束处插入 `theorem end` 模板。当使用一个像 `theorem` 这样的环境并存在一个叠层规则时，则该叠层规则直接跟于 *(block beginning template)* 之后并被调用 (`invocation`)。如果 `theorem` 环境带有可选的参数，情况也是这样。通过插入 `\inserttheoremaddition`，该可选的参数即可用。

该模板有多个可用的插入物 (`inserts`)，请参考下面。

在开始该模板前，字体将被设置成正文字体 (`body font`)，该正文字体由要排版的环境指定 (`prescribe`)。

举例：下面的例子用于排版像 `amsthm` 这样的定理：

```
\setbeamertemplate{theorem begin}
{%
  \begin{\inserttheoremblockenv}
  {%
    \inserttheoremheadfont
    \inserttheoremname
    \inserttheoremnumber
    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
    \inserttheorempunctuation
  }%
}
\setbeamertemplate{theorem end}{\end{\inserttheoremblockenv}}
```

举例：在下面的例子中，所有“建议（suggestions）”用于环境的字体将被抑制或被忽略，定理编号也被抑制。

```
\setbeamertemplate{theorem begin}
{%
  \normalfont% ignore body font
  \begin{\inserttheoremblockenv}
  {%
    \inserttheoremname
    \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
  }%
}
\setbeamertemplate{theorem end}{\end{\inserttheoremblockenv}}
```

下面的插入物（inserts）可用于该模板：

- `\inserttheoremblockenv` 该插入物会扩展为（expand to）block，但如果排版了一个具有 `example` 定理样式（theorem style）的定理时，该插入物会扩展为 `exampleblock`。因此，当排版定理时，我们可以使用该插入物决定使用哪一个环境
- `\inserttheoremheadfont` 该插入物会扩展为字体更改命令（font changing command），该字体更改命令将字体切换成定理标题所用的字体。通过不插入该插入物，我们就可以忽略标题字体（head font）。
- `\inserttheoremname` 该插入物会扩展为要排版的环境的名称，如“定理（Theorem）”或“推论（Corollary）”。
- `\inserttheoremnumber` 如果当前定理没有编号，则该插入物会扩展为当前定理的编号，当前定理前面有空白（space）或没有任何东西（nothing）。
- `\inserttheoremaddition` 如果没有可选的参数，则该插入物会扩展为配给环境的可选参数，或该插入物为空（empty）。
- `\inserttheorempunctuation` 该插入物会扩展为用于当前环境的标点符号（punctuation character），通常是句号（period）。

Beamer-Template `theorem end`

即:

Beamer-Template 定理结束

在定理 (theorem) 的结束处插入该模板。

12.5 加框的文本和盒子文本

要在文本的周围画一个框 (矩形), 可以用 L^AT_EX 的标准命令 `\fbox`, 也可以用 `\frame` (在 BEAMER 的帧内, `\frame` 命令将其含义更改为标准的 L^AT_EX `\frame` 命令)。fancybox 宏包提供了多种类型的方框 (frame), 该宏包定义了以下命令: `\shadowbox`、`\doublebox`、`\ovalbox`、`\Ovalbox`。请参阅 L^AT_EX 手册 (Companion) 以获得这些命令的具体使用方法。

BEAMER 文档类定义了两个环境用于创建盒子 (box)。

```
\begin{beamercolorbox}[(options)]{<beamer color>}
  <environment contents>
\end{beamercolorbox}
```

即:

```
\begin{beamercolorbox}[(选项)]{<beamer 颜色>}
  <environment contents>
\end{beamercolorbox}
```

该环境可方便地用于排版使用了 BEAMER-color 的文本。基本上, 下面的两命令块 (command blocks) 具有相同功能:

```
\begin{beamercolorbox}{beamer color}
  Text
\end{beamercolorbox}
```

```
{
  \usebeamercolor{beamer color}
  \colorbox{bg}{
    \color{fg}
    Text
  }
}
```

换言之, 该环境安装了 `<beamer color>` 并使用 background 作为盒子的背景, 使用 foreground 作为盒子的前景。然而, 事实上, 可以给定为数众多的 `<options>` 以指定如何渲染 (render) 盒子。

如果 `<beamer color>` 的背景色是空的, 则在文本的后面不会绘制背景 (Background), 也就是说, 背景是“透明的 (transparent)”。

在默认的内部主题和外部主题中, 该命令被广泛地应用于排版顶部导航区 (headline) 和底部导航区 (footline)。不希望在普通的框中 (in normal frames) 使用该命令 (例如, 它在 `article` 模式中无效)。我们可能更愿意使用诸如块 (blocks) 或定理 (theorems) 这样的结构元素 (structuring elements), 它们可以按需插入带颜色的盒子 (colored boxes)。

举例：下面的例子用于排版占有两行的顶部导航区（headline），第一个盒子显示文档标题，第二个盒子显示作者姓名：

```
\begin{beamercolorbox}[ht=2.5ex,dp=1ex,center]{title in head/footer}
  \usebeamerfont{title in head/footer}
  \insertshorttitle
\end{beamercolorbox}%
\begin{beamercolorbox}[ht=2.5ex,dp=1ex,center]{author in head/footer}
  \usebeamerfont{author in head/footer}
  \insertshortauthor
\end{beamercolorbox}
```

举例：排版一个快易贴（postit）：

```
\setbeamercolor{postit}{fg=black,bg=yellow}
\begin{beamercolorbox}[sep=1em,wd=5cm]{postit}
  Place me somewhere!
\end{beamercolorbox}
```

可以给定下列 *options*：

- **wd**=*{width}* 设置盒子的宽度。该命令有两个作用：第一个是，将 T_EX 的 `\hspace` 设为 *width*；第二个是，在盒子排版完成后，它的宽度设定为 *width*（不管它变成什么样子）。因为设定 `\hspace` 不会自动更改 L^AT_EX 的行距量纲（linewidth dimensions），我们可能被这个宽度愚弄，这时我们可以考虑在该环境中使用微页（minipage）。

如果该宽度超过了普通文本的宽度，通过指定 `\textwidth` 的值，最终盒子的宽度将重置为 `\textwidth` 的宽度，但在盒子的左右末端插入智能的负平移（intelligent negative skips）。这样，我们就可以使用比文本更宽的盒子，并在普通文本中直接插入盒子而无需理会那些烦人的提示，而且在放置盒子时凭感觉即可。

- **dp**=*{depth}* 设置盒子的深度（depth），废除（override）盒子的真正深度。首先正常地（normally）排版盒子，然后改变其深度。该选项对保证盒子的尺寸很有用。
如果没有给定该选项，则盒子具有它的“自然（natural）”深度，该深度由排版决定。例如，一个只包含字母“a”的盒子其深度与一个只包含字母“g”的盒子的深度不同。
- **ht**=*height* 设置盒子的高度（height），废除（override）盒子的真正高度。注意，“高度（height）”不包含深度（depth）（细节请参考 T_EX-book）。如果想让一个单行的盒子（one-line box）一直具有相同的尺寸，将高度设置为 2.25ex，深度设为 1ex 是个好的选择。
- **left** 使盒子中的文本左对齐（left-aligned），文本的右边界则（完全）参差不齐（ragged）。这是默认的选项。要使右边界更平整，可以使用 `rightskip` 选项。
- **right** 使盒子中的文本右对齐（right-aligned），文本的左边界则（完全）参差不齐（ragged）。
- **center** 使盒子中的文本居中对齐。
- **leftskip**=*left skip* 在盒子内部安装 *left skip* 作为左侧平移（left skip）。T_EX 的左侧平移是一个插入到每一行左侧末端的粘连（粘性之物，glue）。细节请参考 T_EX-book。
- **rightskip**=*right skip* 安装 *right skip* 作为右侧平移（right skip），要获得不太参差不齐（ragged）的右边界，请试着声明：`\rightskip=0pt plus 4em`

- `sep=<dimension>` 设置文本周围的额外间距 (extra space)。该间距被添加到“盒子里面”，这意味着，如果指定 `sep` 为 1cm 并将盒子插入到垂直列表 (vertical list)，则盒子的左边界将与幻灯片文本的左边界对齐，虽然盒子内文本的左边界与盒子左边界的右侧之间有 1cm 的间距。同样的，盒子内文本与普通文本 (normal text) 的右边界有 1cm 的间距。
- `colsep=<dimension>` 设置文本周围的额外“分色间距 (color separation space)”。该间距的行为与由 `sep` 添加的间距的行为相同，不同的是只有当盒子具有彩色背景 (colored background) (即 `<beamer color>` 的背景为非空) 时才插入该分色间距。该命令可以和 `sep` 一起使用，这时，它们的效果会叠加。
- `colsep*=<dimension>` 设置盒子水平方向外面 (horizontally outside the box) 的文本的周围的额外“分色间距 (color separation space)”。这意味着，如果盒子具有背景，该背景将根据 `<dimension>` 伸出 (protrude) 到文本的两侧，但在排版时 T_EX 不会考虑该伸出的背景。

该选项用法的一个典型例子是，在普通文本 (normal text) 中间插入一个带有彩色背景的盒子。既然这样，如果设置了背景色，则会在文本的后面绘出背景，并在该文本的周围出现特定的额外间距 (背景会超出该文本的边界，这有点难看)，我们当然希望普通文本和背景具有相同的水平位置 (horizontal position) 而不依赖是否有背景。这种情况下，`colsep*=4pt` 是个好的选项。

- `shadow=<true or false>` 在盒子后面绘制阴影 (shadow)。目前，该选项只有和 `rounded` 选项连用时才有效，但这是可以改变的。
- `rounded=<true or false>` 如果安装了背景，则会切掉盒子的边界 (borders)。该命令在内部调用 `beamerboxesrounded`。
- `ignorebg` 忽略 `<beamer color>` 的背景色，也就是说，如果将背景色设置成“透明 (transparent)”或“空 (empty)”则忽略该背景色。
- `vmode` 当开始盒子 (box starts) 时，让 T_EX 处于垂直模式 (vertical mode)。通常，开始盒子时 T_EX 处于水平模式 (horizontal mode) (除非给定该选项，否则会在盒子开始处自动插入一个 `\leavevmode`)。只有 T_EXperts 需要该选项，因此，如果我们使用该选项，我们必须知道我们正在做什么。

```
\begin{beamerboxesrounded} [<options>] {<head>}
  <environment contents>
\end{beamerboxesrounded}
```

即：

```
\begin{beamerboxesrounded} [<选项>] {<头部>}
  <environment contents>
\end{beamerboxesrounded}
```

该环境内的文本被一个圆角的矩形框住。BEAMER-color 及指定的 `lower` 选项用于大的矩形区域。用 BEAMER-color 背景作背景，用 BEAMER-color 的前景作前景。如果 `<head>` 不为空 (empty)，将使用 BEAMER-color 及指定的 `upper` 选项作为前景和背景在盒子的上面绘制 `<head>`。可能会给出下面的选项：

- `lower=<beamer color>` 将 BEAMER-color 用于盒子的下面 (lower) (主体) 部分，用 BEAMER-color 背景作背景，用 BEAMER-color 的前景作盒子主体部分的前景。如果其中之一为空，则使用当前的背景或前景。盒子永不透明 (transparent)。

- `upper=<beamer color>` 将 BEAMER-color 用于盒子的上面 (upper) (顶) 部分, 只有 `<head>` 为非空时才使用该选项。
- `width=<dimension>` 将盒子内的文本的宽度设为指定的 `<dimension>`。默认情况下, 使用 `\textwidth`。注意, 盒子向左右两侧各伸出 4pt。
- `shadow=<true or false>` 设为 `true` 时, 绘制阴影 (shadow)。

如果没有给定 `<head>`, 将会完全抑制 head 部分。

举例:

```
\begin{beamerboxesrounded}[upper=block head,lower=block body,shadow=true]{Theorem}
  $A = B$.
\end{beamerboxesrounded}
```

ARTICLE 该环境在论文 (article) 模式中无效。

12.6 图和表

我们可以像平常一样使用标准的 L^AT_EX figure 和 table 环境。然而, 会忽略任何放置规则 (placement specification)。在 figure 和 table 环境开始处会立即插入图 (Figures) 和表 (Tables)。如果图表太多在帧中容不下时, 必须手工分开它们并将它们放置于不同的帧, 或使用 `allowframebreaks` 选项。

举例:

```
\begin{frame}
  \begin{figure}
    \pgfuseimage{myfigure}
    \caption{This caption is placed below the figure.}
  \end{figure}

  \begin{figure}
    \caption{This caption is placed above the figure.}
    \pgfuseimage{myotherfigure}
  \end{figure}
\end{frame}
```

Beamer-Template/-Color/-Font caption

即:

Beamer-Template/-Color/-Font 标题

该模板用于渲染 (render) 标题 (caption)。

下面的模板选项是预定好的:

- `[default]` 在标题文本 (caption text) 的前面排版标题名 (caption name) [像 “Figure (图)”、“Abbildung (德语中的插图)”、“Table (表)” 这样的单词, 这些单词的外形依排版的图、表、插图和安装的语言不同而不同³⁷]。不会排版编号 (number), 因为这些编号在普通的演示稿中是毫无意义的。

³⁷要显示中文的“图”、“表”, 可以在 `\begin{document}` 的后面 (而不是在导言区中) 使用下面的命令:
`\renewcommand\figurename{图}, \renewcommand\tablename{表}。`

- `[numbered]` 在标题的前面添加图或表编号³⁸。只有当观众手上的已打印的讲义 (Handout) 或演讲记录 (lecture notes) 具有相同的编号时, 才使用该选项。
- `[caption name own line]` 正如其名称所建议的那样, 该选项在它自己的行放置标题名 (如 “Figure”)。

在该模板内, 可以使用下面的插入物 (inserts):

- `\insertcaption` 插入当前标题的文本。
- `\insertcaptionname` 插入当前的标题名。标题名的单词是 “Table” 或 “Figure”, 如果使用了 `babel` 宏包, 则使用其译文 (translation)。
- `\insertcaptionnumber` 插入当前图或表的编号。

Beamer-Color/-Font `caption name`

即:

Beamer-Color/-Font 标题名

这些 `BEAMER-color` 和 `-font` 用于排版标题名 (caption name) (像 “Figure” 这样的单词)。`caption` 模板会直接 “使用 (use)” 它们, `\insertcaptionname` 命令不会自动安装这些 `BEAMER-color` 和 `-font`。

12.7 将帧分成多栏

`BEAMER` 文档类提供了多个命令和环境用于将一帧 (也许是其一部分) 分成多栏 (multiple columns)。这些命令与 `LATEX` 用于创建栏的命令无关。栏对于在描述 (description) / 解释 (explanation) 后面放置的图形 (graphic) 特别有用。

用于创建栏 (columns) 的主环境 (main environment) 称为 `columns` 环境, 在这个环境中里, 我们可以嵌入几个 `columns` 环境, 嵌入的每一个 `columns` 环境创建一个新栏; 我们也可以使用 `\column` 创建新栏。

```
\begin{columns}[(options)]
  <environment contents>
\end{columns}
```

即:

```
\begin{columns}[(选项)]
  <environment contents>
\end{columns}
```

一个多栏 (multi-column) 区域。在该环境里, 我们只能放置 `column` 环境或 `\column` 命令 (见下)。可能会给出下面的 `<options>`:

- `b` 使栏的底边垂直对齐。
- `c` 使栏居中对齐。这是默认选项, 除非使用了全局的 `t` 选项。
- `onlytextwidth` 与 `totalwidth=\textwidth` 等价。

³⁸实现语句是: `\setbeamertemplate{caption}[numbered]`

- `t` 以栏的首行对齐。如果使用了全局的 `t` 选项则该选项是默认选项。
- `T` 和 `t` 选项相似，但 `T` 选项以首行的顶部对齐，而 `t` 选项以所谓的首行的基线对齐。如果使用 `t` 选项时发生了奇怪的事情（例如，如果使用了 `t` 选项后图形突然“落下”而不是“上升”），请用 `T` 选项。
- `totalwidth=<width>` 设置栏的总宽度为 `<width>`，而不是整个页面的宽度。

举例：

```
\begin{columns}[t]
  \begin{column}{5cm}
    Two\\lines.
  \end{column}
  \begin{column}{5cm}
    One line (but aligned).
  \end{column}
\end{columns}
```

举例：

```
\begin{columns}[t]
  \column{5cm}
    Two\\lines.

  \column[T]{5cm}
    \includegraphics[height=3cm]{mygraphic.jpg}
\end{columns}
```

ARTICLE 在论文（`article`）模式中，`columns` 环境将被忽略。

LYX 用“Columns”或“ColumnsTopAligned”创建一个 `columns` 环境。要传递选项，在 `TEX` 模式（`TEX-mode`）下在该环境的右侧放置方括号（square brackets），并在方括号中放置选项。

要创建栏，我们可以使用 `column` 环境或 `\column` 命令。

```
\begin{column}[<placement>]{<column width>}
  <environment contents>
\end{column}
```

即：

```
\begin{column}[<对齐方式>]{<栏宽>}
  <environment contents>
\end{column}
```

以 `<column width>` 的宽度创建一单栏（single column）。封闭的 `columns` 环境的垂直放置（vertical placement）可以被特别指定的 `<placement>`（`t` 和 `T` 用于顶部对齐，`c` 用于居中对齐，`b` 用于底部对齐）否定。

举例：下面的代码与上述例子具有相同效果：

```
\begin{columns}
  \begin{column}[t]{5cm}
```

```

    Two\\lines.
\end{column}
\begin{column}[t]{5cm}
    One line (but aligned).
\end{column}
\end{columns}

```

ARTICLE 在论文 (article) 模式中, 该命令将被忽略。

LYX “Column” 样式插入命令版本 (command version), 见下。

```
\column[{placement}]{{column width}}
```

即:

```
\column[{对齐方式}]{{栏宽}}
```

开始一单栏 (single column)。该命令的参数 *{placement}* 和选项 *{column width}* 与上述 `column` 环境的相同。栏会在下一 `\column` 命令或下一 `column` 环境出现时或在当前 `column` 环境结束时自动结束。

举例:

```

\begin{columns}
  \column[t]{5cm}
    Two\\lines.
  \column[t]{5cm}
    One line (but aligned).
\end{columns}

```

ARTICLE 在论文 (article) 模式中, 该命令将被忽略。

LYX 在 “Column” 样式中, *{column width}* 必需是以普通的文本给出, 而不能在 `TEX` 模式中给出。

12.8 文本和图形的绝对定位

通常, BEAMER 使用 `TEX` 普通的排版机制 (normal typesetting mechanism) 在页面上放置文本 (text) 和图形 (graphics)。某些特定的情况下, 可能希望在页面按指定的绝对 (*absolutely*) 位置放置特定的文本或图形。这意味着, 指定的绝对位置是以幻灯片的左上角 (upper left corner) 为基准的。

`textpos` 宏包提供了多个命令用于按绝对位置放置文本, 该宏包可以和 BEAMER 一起使用。当命名用该宏包时, 必须指定 `overlay` 选项, 可能需要指定 `absolute` 选项。使用该宏包的细节, 请参考 `textpos` 宏包的文档。

12.9 抄录文本和脆弱文本

如果想在帧内使用 `{verbatim}` 环境, 可以添加 `[fragile]` 选项给 `{frame}` 环境。既然这样, 必须使用 `{frame}` 环境 (而不是 `\frame` 命令), 而且必须独占一行。使用该选项会将帧内容 (frame contents) 写入一个外部文件 (external file) 并从该文件读回 (read back)。详细内容请参考 `{frame}` 环境的描述。

当帧包含任何 “脆弱 (fragile)” 文本时必须使用 `[fragile]` 选项, 脆弱文本 (fragile text) 是指 “不能用 `TEX` 解释 (interpret) 文本的通常方式解释的” 任何文本, 例如, 如果使用了一个宏包, 该宏包 (局部) 重定义了字符如 `&` 的含义, 则必须使用 `[fragile]` 选项。

在 `{verbatim}` 环境内，显然不能使用 `\alert<2>` 这样的命令来高亮显示部分代码，因为这样的代码文本是抄录文本主（verbatim text）。有多个宏包如 `alltt` 或 `listings` 可以绕过（circumvent）该问题。作为一个简单的例子，下面的环境可以使用，它是 BEAMER 定义的：

```
\begin{semiverbatim}
  <environment contents>
\end{semiverbatim}
```

该环境中的文本会像逐字显示的文本（Verbatim Text）那样排版出来。然而，`\`、`{` 和 `}` 等字符仍保留它们的含义。因此，可以像下面这样声明

```
\alert<1->{std::cout << "AT&T likes 100% performance";}
```

要排版这三个字符 `\`、`{` 和 `}`，可以使用 `\\`（在该环境中重定义了该命令 - 你不需要它）、`\{` 和 `\}` 命令。因此，要排版 “`\alert<1>{X}`”，可以写下 `\\alert<1>\{X\}`。`\\alert<1>\{X\}`。

12.10 摘要

在 BEAMER 中，`{abstract}` 环境是对叠层规则敏感的（overlay specification-aware）：

```
\begin{abstract}< <action specification> >
  <environment contents>
\end{abstract}
```

即：

```
\begin{abstract}< <行为规则> >
  <environment contents>
\end{abstract}
```

可以使用该环境排版一个摘要（abstract）。

Beamer-Color/-Font abstract

即：

Beamer-Color/-Font 摘要

这些 BEAMER-color 和 -font 用于排版摘要。如果设置了背景色，默认情况下，该背景色将用作整个摘要的背景。

Beamer-Template/-Color/-Font abstract title

即：

Beamer-Template/-Color/-Font 摘要标题

Color 父模板：titlelike

该模板用于标题（title）中，默认情况下，插入 `\abstractname`³⁹ 单词，并且居中。会忽略背景色

³⁹要显中文的“摘要”，可以在 `\begin{document}` 的后面（而不是在导言区中）使用下面的命令：
`\renewcommand\abstractname{摘要}`

Beamer-Template abstract begin

即：

Beamer-Template 摘要开始

在每一摘要的开始处摘要标题 (abstract title) 之前插入该模板及 $\langle environment contents \rangle$ 。

Beamer-Template abstract end

即：

Beamer-Template 摘要结束

在摘要的结束处 $\langle environment contents \rangle$ 之后插入该模板。

12.11 诗歌、引用（首行再缩进）、引用（首行不缩进）

L^AT_EX 定义了三个环境用于排版引用 (quotations) 和诗歌 (verses): `verse`、`quotation`、`quote`。这三个环境在 BEAMER 文档类中同样可用，beamer 文档类是叠层规则敏感的 (overlay specification-aware)。如果给定了叠层规则，诗歌或引用只会在指定的幻灯片是显示，在其余幻灯片中隐藏。`quotation` 和 `quote` 的不同之处是，`quotation`（首行再缩进的引用）是段落缩进的，而 `quote`（首行不缩进的引用）是不缩进的。

不能通过更改下述的 BEAMER-colors 和 -fonts 来改变这三个环境中的字体和颜色。不像标准的 L^AT_EX 环境，默认的字体主题 (font theme) 以斜体衬线 (italic serif) 字体排版 verse，以斜体字体（使用衬线还是无衬线字体由标准的文档字体决定）排版 quotations 和 quotes。

```
\begin{verse}<<action specification>>
  <environment contents>
\end{verse}
```

即：

```
\begin{verse}<<行为规则>>
  <environment contents>
\end{verse}
```

使用该环境排版诗歌 (verse)。

Beamer-Color/-Font verse

即：

Beamer-Color/-Font 诗歌

这些 BEAMER-color 和 -font 用于排版 verse。如果设定了背景色，则该背景色将用作整个 verse 的背景。默认的字体是斜体衬线 (italic serif) 字体。

Beamer-Template verse begin

即：

Beamer-Template 诗歌开始

在 verse 的开始处插入该模板。

Beamer-Template verse end

即：

Beamer-Template 诗歌结束

在 verse 的结束处插入该模板。

```
\begin{quotation}<<action specification>>
  <environment contents>
\end{quotation}
```

即：

```
\begin{quotation}<<行为规则>>
  <environment contents>
\end{quotation}
```

使用该环境排版多段落的（multi-paragraph）quotations。在呈现多段落引文前须考虑清楚。

Beamer-Color/-Font quotation

即：

Beamer-Color/-Font 首行缩进的引用

这些 BEAMER-color 和 -font 用于排版首行缩进的引用（quotation）。

Beamer-Template quotation begin

即：

Beamer-Template 首行缩进的引用开始

在 quotation 的开始处插入该模板。

Beamer-Template quotation end

即：

Beamer-Template 首行缩进的引用结束

在 quotation 的结束处插入该模板。

```
\begin{quote}<<action specification>>
  <environment contents>
\end{quote}
```

即：

```
\begin{quote}<<行为规则>>
```

`<environment contents>`

`\end{quote}`

使用该环境排版单段落的（single-paragraph）quotation。

Beamer-Color/**-Font** `quote`

即：

Beamer-Color/**-Font** 首行不缩进的引用

这些 `BEAMER-color` 和 `-font` 用于排版首行不缩进的引用（`quote`）。

Beamer-Template `quote begin`

即：

Beamer-Template 首行不缩进的引用开始

在 `quote` 的开始处插入该模板。

Beamer-Template `quote end`

即：

Beamer-Template 首行不缩进的引用结束

在 `quote` 的结束处插入该模板。

12.12 脚注

首先提个醒：使用脚注（footnote）常常不是个好主意。脚注会打断阅读流程（the flow of reading）。

可以使用平常的 `\footnote` 命令。已经讨论过了该命令带有一个额外的选项，以便在帧底部而不是在当前微页（minipage）底部放置脚注。

`\footnote<overlay specification>[options]{text}`

即：

`\footnote<叠层规则>[选项]{文本}`

在当前帧插入一条脚注。脚注常显示于当前帧的底部，而不会“转移”到其它帧。照例，可以给定一个数字（number）作为 `<options>`，以使脚注使用该数字。BEAMER 文档类会添加一个额外选项：

- **frame** 使脚注显示于帧的底部。这是默认的行为，但在微页（minipages）和特定的块（blocks）内，它的行为不同。在微页中，脚注常常是作为微页的一部分显示的，而不是作为帧的一部分显示的。

如果给定了一个 `<overlay specification>`，那么脚注 `<text>` 只会显示于指定的幻灯片上。文本中的脚注符号（footnote symbol）会显示于所有幻灯片。

举例：`\footnote{On a fast machine.}`

举例：`\footnote[frame,2]{Not proved.}`

举例：`\footnote<.->{Der Spiegel, 4/04, S.~90.}`

Beamer-Template/-Color/-Font `footnote`

即：

Beamer-Template/-Color/-Font 脚注

该模板用于渲染 (render) 脚注。在该模板内可以使用下面两个插入物 (inserts)：

- `\insertfootnotetext` 插入当前的脚注文本 (footnote text)。
- `\insertfootnotemark` 插入当前的脚注符号 (footnote mark) (像一个上标数字)，该符号是自动计算出来的。

Beamer-Color/-Font `footnote mark`

即：

Beamer-Color/-Font 脚注符号

当渲染正文 (text) 中和在脚注开始处的脚注符号时使用该 `BEAMER-color/-font`。

13 图形

下面将讨论为 BEAMER 创建图形 (graphics) 的不同方式的利与弊。下述大部分信息不只适应于 BEAMER, 也适应于其它文档类。

13.1 包含外部图形文件与直接嵌入图形

创建包含图形的 $\text{T}_{\text{E}}\text{X}$ 文档的方式有两种: 第一种, 图形存放于一个外部文件 (external file) 中, $\text{T}_{\text{E}}\text{X}$ 文档包含 (*include*) 该外部文件; 第二种, 在 $\text{T}_{\text{E}}\text{X}$ 文档中直接嵌入 (*inline*) 图形, 这意味着 $\text{T}_{\text{E}}\text{X}$ 文件包含 (contain) 一群像“画一条从这里到那里的红线”这样的命令。下面讨论这两种方式的利与弊。

我们可以使用外部程序如 `xfig`、GIMP、Inkscape 创建图形。这些程序有一个选项用于导出 (*export*) 图形, 该格式的图形可以插入到演示稿中。

有利之处主要是:

- 可以使用一个强有力的程序创建高质量的图形。

弊端主要是:

- 必须考虑多个文件, 至少有两个, 即程序的图形数据文件 (graphic data file) 和导出的图形文件 (graphic file), 导出的图形文件必须能被 $\text{T}_{\text{E}}\text{X}$ 读取。
- 用外部程序更改图形后, 不会在演示稿中自动更新相应的图形。这样, 我们就必须重新导出 (reexport) 图形并重新运行 (rerun) $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 。
- 很难获得准确的线宽、字体、字体的尺寸。
- 创建作为图形一部分的公式 (formulas) 很难或不可能。

我们可以用所有用于插入图形的标准的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 命令, 如 `\includegraphics` (确保载入了 `graphics` 或 `graphicx` 宏包)。而且, `pgf` 宏包提供了包含图形的命令。这些命令中的任何一个均工作良好, 无论选择哪个均可。在一个 `.pdf` 文件中像 `\pgfdeclareimage`、`\includegraphics` 这样的命令也能一次性 (only once) 包含一个图形, 即使这样的命令使用了多次 (实际上, `graphics` 宏包在这一点上比 `pgf` 更智能)。然而, 在当前只有 `pgf` 宏包才能包含部分透明的 (transparent) 图形。

在这节的末尾将阐述如何包含特殊格式如 `.eps` 或 `.jpg` 的图形。

LYX 我们可以使用通常的“Insert Graphic (插入图形)”命令插入图形。

在 BEAMER 中, `\includegraphics`、`\pgfuseimage`、`\pgfimage` 等命令是叠层规则敏感的 (overlay-specification-aware)。如果没有应用叠层规则, 则这些命令没有其效果。这对于创建一个简单的动画 (animation) 是很有用的, 我们可以将动画的每一幅图像放在一个不同的文件中:

```
\begin{frame}
  \includegraphics<1>[height=2cm]{step1.pdf}%
  \includegraphics<2>[height=2cm]{step2.pdf}%
  \includegraphics<3>[height=2cm]{step3.pdf}%
\end{frame}
```

创建图形的另一种方式是在 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 文件中直接插入画图命令 (graphic drawing commands)。有很多宏包可以帮助我们完成这件事, 它们的复杂 (sophistication) 程度各不相同。这种方式不存在上述的包含外部图形文件

的不利之处，但这种方式的主要弊端是难以使用（hard to use）这些宏包。从某种意义上说，我们“用程序画（program）”图需要的只是一点实践（practice）而已。

mind: 当选择一个图形宏包（graphic package）时，请记住下面这些事情：

- 多个图形宏包生成的图形质量低劣。L^AT_EX 的标准 `picture` 环境尤其是这样。
- 生成高质量图形的强有力的宏包常和 `pdflatex`、`lualatex`、`xelatex` 不兼容。
- 功能强大又便于使用的宏包大概要算 `pstricks`，但它和 `pdflatex`、`lualatex`、`xelatex` 不兼容，这是一个底层的问题（fundamental problem）。这是因为 PDF 和 PostScript 两者的基础（fundamental）不同，不可能写一个“用于 `pstricks` 的 `pdflatex` 后端（back-end）”。（`lualatex` 和 `xelatex` 的情况类似）。无论如何，`PST-PDF`、`XETEX-PSTRICKS`、`PDFTRICKS` 等宏包这时能帮上忙，它们帮助简化从用户的角度来看事物。

有三种可能的办法解决上述问题，每一种均有各自的利弊。

- 使用 `PGF` 宏包。该宏包能生成高质量的图形并与 `pdflatex`、`lualatex`、`xelatex` 兼容，也能与普通的 `latex` 兼容。它没有 `pstricks`（如上所述，是因为基础不同）功能强大，也不易使用，但大多数情况下它能胜任。
- 使用 `LUAMPLIB` 宏包和 `lualatex`。它提供了一个环境，该环境可以让我们在文档中直接键入（type）MetaPost 代码。
- 使用 `pstricks` 宏包并紧随 `latex` 和 `dvips`，或使用上述的变通方法（workarounds）。

[LYX](#) 目前内嵌的图形必须放在一个大的 T_EX-模式的盒子中才能插入到文档中。这不很方便。

13.2 包含 .eps 或 .ps 格式的图形文件

如果使用 `latex` 和 `dvips`，则可以包含的外部图形文件的扩展名为 `.eps`（封装了 PostScript）或 `.ps`（PostScript），如果使用 `pdflatex` 则没有这种要求。这才是真正的既为普通的 `graphics` 宏包又为 `pgf`。当使用 `pgf` 时，不要添加扩展名 `.eps`。当使用 `graphics` 时，则要添加扩展名。

如果要使用 `.eps` 图形及 `pdflatex`，则可以使用 `ps2pdf` 程序将该图形文件转换成 `.pdf` 文件。一些现代的分发版（distributions）启用了 `writeln18`，它允许 `pdflatex` 自动完成上述转换过程。注意，如果程序能生成 `.eps`，那么直接生成一个 `.pdf` 文件是个好主意。

13.3 包含 .pdf、.jpg、.jpeg 或 .png 格式的图形文件

`.pdf`、`.jpg`、`.jpeg`、`.png` 这四种格式的图形只能由 `pdflatex` 包含。如前所述，当使用 `pgf` 时无需添加这些扩展名，但使用 `graphics` 时则要添加。如果图形文件是这四种格式中的一种，希望/必须使用 `latex` 和 `dvips`，则首先必须将它转换成 `.eps`。

13.4 包含 .mps 格式的图形文件

扩展名为 `.mps`（MetaPost PostScript）的图形文件是一类特殊的封装（Encapsulated）PostScript 文件。它由 MetaPost 程序生成。正如我们所知道的，T_EX 是这样一个程序，它能将简单的纯文本（simple plain text）转换成漂亮的排版文件（typeset documents）。MetaPost 程序和 T_EX 相似，不同的是 MetaPost 将简单的纯文本转换成漂亮的图形。

MetaPost 将扩展名为 `.mp` 的简单的纯文本文件转换成 `.mps` 文件（虽然由于某些莫测高深的原因而无需添加扩展名）。`.mp` 文件必须包含用 MetaFont 程序语言写成的文本。`.mps` 文件其实也就是 `.eps` 文件，我们就可以使用普通的 `\includegraphics` 命令包含这些图形文件。

然而，当使用 `pdflatex` 时，我们也可以包含这样一个文件。通常，`pdflatex` 不能处理 `.eps` 文件，但由 MetaPost 生成的 `.mps` 文件具有这样一个简单又特殊的结构使得 `pdflatex` 能处理 `.mps` 文件。`graphics` 宏包具有过滤（filters）功能，它能将 PostScript 的输出实时地（on-the-fly）转换成 PDF。对于该工作，文件的扩展名必须用 `.mps` 替代 `.eps`。下面的命令可使 `graphics` 宏包让一无所知的文件（如扩展名为 `.1` 的文件，它是 MetaPost 最喜欢生成的文件）假装（assume）成 `.mps` 文件：

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

目前只有 `graphics` 宏包才具有这个特殊的功能，`pgf` 宏包没有。

13.5 包含 `.mmp` 格式的图形文件

`TEX`-文件不能直接包含 `.mmp` 格式（Multi-MetaPost）的图形。而且，像 `.mp` 这样的文件也必须先用 MetaPost 程序转换。`.mp` 和 `.mmp` 之间最重要的不同之处是，在一个 `.mmp` 中可以存放多个图形（multiple graphics）（事实上，在一个 `.mp` 中也可以存放多个图形，但按惯例这样的文件称为 `.mmp` 文件）。当用 MetaPost 运行一个 `.mmp` 文件时，产生的文件不是一个封装的 PostScript 文件，而是多个文件，它们的扩展名是 `.0`、`.1`、`.2` 等等。大意是 `.0` 包含主要的图形（main graphic），其余的图形包含叠层元素（overlay material），它递增地（incrementally）添加给这个主要的图形。

要包含这一系列结果文件（resulting files），我们可以使用来自于 `mpmulti` 宏包或 `xmpmulti` 宏包的 `\multiinclude` 命令。该程序是如何工作的请参阅第 14.1.3 节。

14 动画、声音、幻灯片过渡

14.1 动画

14.1.1 包含外部的动画文件

如果我们用某个外部程序（如一个渲染程序）创建了动画（animation），则可以用演示稿程序（如 Acrobat Reader）显示这些动画。不幸的是，目前还没有一个方便的途径（portable way）完成上述过程，即便是 Acrobat Reader 也不具有在所有平台（platforms）上显示动画的功能。

要在演示稿中包含（include）一个动画，我们可以使用 `multimedia.sty` 宏包，它是 BEAMER 宏包的一部分，但我们必须明确地加载 `multimedia.sty` 宏包。尽管该宏包是作为 BEAMER 宏包的一部分分发的，但它是完全自给自足的（perfectly self-sufficient），它能独立于 BEAMER 而被使用。

```
\usepackage{multimedia}
```

这是一个独立的（stand-alone）宏包，它能执行多个命令以在一个 PDF 文档中包含外部的动画和声音文件。该宏包能与 `dvips + ps2pdf`、`pdflatex` 一起使用，虽然只有在 `pdflatex` 中才支持特定的声音。

当载入（including）该宏包时，我们还必须载入 `hyperref` 宏包，因为只希望在导言区的末尾载入 `hyperref` 宏包，载入 `multimedia` 宏包不会自动载入 `hyperref` 宏包。然而，可以在载入 `hyperref` 宏包之前或之后载入 `multimedia` 宏包。因为 BEAMER 会自动加载 `hyperref` 宏包，所以用 BEAMER 创建演示稿时我们无需当心 `hyperref` 宏包的加载这样的事情。

要在一个 PDF 文件中包含动画，我们可以使用 `\movie` 命令，下面会阐述该命令。据所用的选项不同，该命令可以设置（setup）PDF 文件使得浏览程序（viewer application）（如 Acrobat Reader）能演示动画或调用一个外部程序。调用一个外部程序虽然更不灵活（flexible），却是浏览程序无法显示动画时必须采取的措施。

```
\movie[options]{poster text}{movie filename}
```

即：

```
\movie[选项]{广告文本}{动画文件名}
```

该命令将文件名为 *movie filename* 的动画插入到 PDF 文件中。动画文件必须放在一个浏览程序（viewer application）能找到的地方，该地方最好是最终的 PDF 文件所在的目录。动画文件不会嵌入（embedded）到 PDF 文件中，某种意义上实际的动画数据是 `main.pdf` 文件的一部分。动画文件从此必须随 PDF 文件一同拷贝和传递。（然而，常说的动画“嵌入”到文档中，其实是指当浏览该文档时点击动画后动画开始播放。）

动画会占用一个矩形区域（rectangular area），其尺寸由 `width=` 和 `height=` 选项决定，或由 *poster text* 的尺寸决定。*poster text* 可以是任何的 $\text{T}_\text{E}\text{X}$ 文本，例如，它可能是一个 `\pgfuseimage` 命令，或是一个 `\includegraphics` 命令，或是一个 `pgfpicture` 环境，或仅仅是单纯的文本（plain text）。*poster text* 被排入一个盒子中，该盒子可以插入到普通的文本中，动画占据的矩形（movie rectangle）会溢出（over）该盒子。因此，如果 *poster text* 是一幅来自动画的图像，则该图像直到动画播放时才会显示，这时该图像才会被动画替换。然而，有个不同的可以说是更好的方法创建一个广告画图像（poster image），该广告画图像由 `poster` 选项命名，稍后会讲述该选项。

如果 *poster text* 盒子的长宽比（aspect ratio）和动画的长宽比不一样，则动画的长宽比不会自动矫正。大部分动画的长宽比是 4:3 或 16:9。

虽然名为动画，但一个动画可能仅有声音而没有图像。这样， $\langle poster\ text\rangle$ 就只能是一个代表声音的符号 (symbol)。还有一个不同的专门用于在 PDF 文件中包含声音的 `\sound` 命令，请对阅第 14.2 节。

除非给定更多的选项，否则只有在点击动画时它才会开始播放。浏览程序能否真正播放动画取决于该程序及其版本，例如，在 Linux 平台上，Acrobat Reader 5 及其以前的版本无法播放任何动画及声音。另一方面，在 MacOS 平台上，Acrobat Reader 6 可以播放任何 QuickTime 能播放的动画。PDF 标准 (standard) 允许在一个 PDF 文档中嵌入动画，但这不是 Acrobat Reader 的特性。我们当然希望在将来还有其它的浏览程序如 `xpdf` 和基于 `poppler` 的浏览程序 (Okular、Evince) 能支持嵌入的动画。

举例：`\movie{\pgfuseimage{myposterimage}}{mymovie.avi}`

举例：`\movie[width=3cm,height=2cm,poster]{}{mymovie.mpg}`

如果浏览程序无法渲染 (render) 动画，但外部程序却可以，这时我们可以使用 `externalviewer` 选项。该选项会询问浏览程序以启动 (launch) 一个外部程序来显示动画而不是自己来显示。因为该外部程序会新开一个窗口，这没有由浏览程序直接播放动画漂亮 (除非使用了诡计抑制了浏览程序的框架)。选择哪个外部程序由浏览程序决定，决定的依据是： $\langle movie\ filename\rangle$ 的扩展名，以及匹配浏览程序的扩展名映射表 (mapping table)。该扩展名映射表可以据浏览程序进行修改，请参阅浏览程序的阅版本注释 (release notes)。

可能会给出下面的 $\langle options\rangle$ ：

- **autostart**. 使显示页面 (page) 时立即播放动画。这种方式最多只能开始播放一个动画。当页面不再显示时，动画也会立即停止。当我们使用 `\movie` 命令包含 (include) 一个页面关闭后仍要播放的声音时，变会带有问题。既然这样，就必须使用 `\sound` 命令。
- **borderwidth**= $\langle TeX\ dimension\rangle$. 在动画的周围绘制一个厚度为 $\langle TeX\ dimension\rangle$ 的边界。某些版本的 Acrobat Reader 有 bug 以致无法显示厚度小于 0.5bp (大约 0.51pt) 的边界。
- **depth**= $\langle TeX\ dimension\rangle$. 覆盖 $\langle poster\ text\rangle$ 盒子的深度 (depth)，并将其设为给定的尺寸。
- **duration**= $\langle time\rangle$ s. 指定动画要显示的秒数。 $\langle time\rangle$ 必须是小数 (fractional)，且后跟有一个字母 `s`，例如，`duration=1.5s` 使动画显示 1.5 秒。和 `start` 选项联用，我们就可以“删去”一部分动画。
- **externalviewer**. 如前所述，该选项启动 (launch) 一个外部程序新开一个独立的窗口播放动画。大多数选项如 `duration` 或 `loop` 没有作用，因为没有将它们传递给浏览程序。
- **height**= $\langle TeX\ dimension\rangle$. 覆盖 $\langle poster\ text\rangle$ 盒子的高度 (height)，并将其设为给定的尺寸。
- **label**= $\langle movie\ label\rangle$. 给动画指定一个标签 (label)，以便以后的 `\hyperlinkmovie` 命令参考，它用于停止播放动画或显示动画的不同部分。 $\langle movie\ label\rangle$ 不是普通的标签。它不能太花俏 (fancy)，因为它是逐字逐句地 (literally) 插入到 PDF 代码中。尤其是它不能包含关的圆括号 (closing parentheses)。
- **loop**. 使动画循环播放。通常，动画在结束时停止播放。
- **once**. 使动画在结束时停止播放。这是默认的选项。
- **open**. 当动画播放完毕后播放器仍处于打开状态。
- **palindrome**. 当到达动画结束处时开始倒放 (playing backwards)，到达动画开始处重新播放一遍，等等。
- **poster**. 询问浏览程序当动画不播放时显示动画的第一幅图像。通常，当动画不播放时不显示任何东西 (因此，会显包含 $\langle poster\ text\rangle$ 的盒子)。对于一个没有任何图像 (除声音外) 的动画或首幅图像没有任何信息 (uninformative) 的动画来说该选项没有太多用处。

- **repeat** 和 **loop** 选项相同。
- **showcontrols**=*(true or false)*. 当播放动画时在动画下面显示一个控制条 (control bar)。我们也可以使用 **showcontrols** 代替 **showcontrols=true**。默认情况下, 不显示控制条。
- **start**=*(time)*s. 跳过 *(time)* 秒以前的动画, 例如, **start=10s,duration=5s** 将显示第 10 秒至第 15 秒之间的动画。
- **width**=*(TeXdimension)* 和 **height** 选项相似, 该选项用于设置广告盒子 (poster box) 的宽度。

举例: 下面的例子创建一个幻灯片的“背景声音 (background sound)”。

```
\movie[autostart]{}{test.wav}
```

举例: 动画的两个额外按钮用于显示动画的不同部分。

```
\movie[label=cells,width=4cm,height=3cm,poster,showcontrols,duration=5s]{}{cells.avi}
```

```
\hyperlinkmovie[start=5s,duration=7s]{cells}{\beamerbutton{Show the middle stage}}
```

```
\hyperlinkmovie[start=12s,duration=5s]{cells}{\beamerbutton{Show the late stage}}
```

一个动画可作为特殊链接的链接目标, 特殊链接也就是用下面的命令生成的链接:

```
\hyperlinkmovie[(options)]{(movie label)}{(text)}
```

即:

```
\hyperlinkmovie[(选项)]{(动画标签)}{(文本)}
```

将 *(text)* 设为一个动画链接 (movie hyperlink)。当我们点击时 *(text)*, 带有 *(movie label)* 标签的动画将开始播放 (据 *(options)* 或停止, 或暂停, 或重新开始播放)。动画必须和链接处于同一页面中。

可能会给定下面的 *(options)*, 其中的多个选项和 **\movie** 命令的选项相同; 如果配给链接 (link) 的选项与配给动画 (movie) 的选项不同, 则配给链接 (link) 的选项具有优先权 (precedence):

- **duration**=*(time)*s. 对于 **\movie** 命令, 该选项使动画播放持续给定的秒数。
- **loop** 和 **repeat**. 对于 **\movie**, 该选项使动画循环播放。
- **once**. 对于 **\movie**, 该选项使动画只播放一次。
- **palindrome**. 对于 **\movie**, 该选项使动画来回 (forth and back) 播放。
- **pause**. 如果当前正在播放动画, 则使动画的重播暂停。如果当前不在播放动画, 则不做任何事情。
- **play**. 使动画在指定的任何位置开始播放。如果已经在播放动画, 将会停止并在指定的位置重新开始播放。这是默认的选项。
- **resume**. 如果以前暂停播放动画, 则重新开始播放。如果以前没有暂停播放动画, 但没有开始播放或已经正在播放, 则不做任何事情。
- **showcontrols**=*(true or false)*. 对于 **\movie**, 该选项显示一个控制条, 但在重播时不显示。
- **start**=*(time)*s. 对于 **\movie**, 如果使用了 **play** 选项开始播放动画, 则该选项将给定秒数以前的动画忽略。
- **stop**. 将动画的重播 (playback) 停止。

14.1.2 快速演替幻灯片时形成动画

我们也可以通过 BEAMER 宏包的叠层命令 (overlay commands) 这样简便的方法创建一系列幻灯片, 当快速有序地显示这些幻灯片时, 就形成了动画 (Animation)。这是一种灵活的 (flexible) 方法, 便这种动画有点停滞 (static), 因为一张张地推进 (advance) 幻灯片需要一些时间。要解释动画的每一“图像 (picture)”时这种方法很有用。当我们”手动”也就是说按下前进按钮推进幻灯片时, 至少得等一秒钟才会显示下一张幻灯片。

更多“逼真的 (lively)”动画是通过浏览程序创建的。一些程序只支持显示特定秒数的幻灯片 (对于 Acrobat Reader 来说, 它只工作在全屏模式中)。将秒数设为 0, 就可以创建快速演替的 (rapid succession) 幻灯片。

为帮助创建动画, 可以使用下面的命令: `\animate` 和 `\animatevalue`。

`\animate`<*overlay specification*>

即:

`\animate`<*叠层规则*>

尽可能快地显示由 *overlay specification* 指定的幻灯片。

举例:

```
\begin{frame}
  \frametitle{A Five Slide Animation}
  \animate<2-4>

  The first slide is shown normally. When the second slide is shown
  (presumably after pressing a forward key), the second, third, and
  fourth slides ``flash by.'' At the end, the content of the fifth
  slide is shown.

  ... code for creating an animation with five slides ...
\end{frame}
```

ARTICLE 在 `article` 模式中会忽略该命令。

`\animatevalue`<*start slide*-<*end slide*> {<*name*>}{<*start value*>}{<*end value*>}

即:

`\animatevalue`<*开始的幻灯片*>-<*结束的的幻灯片*> {<*名称*>}{<*开始的值*>}{<*结束的值*>}

name 必须是一个计数器 (counter) 或一个尺寸 (dimension) 的名称。它在两个值之间变化。对于指定的范围内的幻灯片, 会将计数器或尺寸设为一个插入值 (interpolated value), 该插入值的依据是当前的幻灯片序号 (number)。在 *start slide* 之前的幻灯片, 计数器或尺寸设为 *start value*; 在 *end slide* 之后的幻灯片, 计数器或尺寸设为 *end value*。

举例:

```
\newcount\opaqueness
\begin{frame}
```

```

\animate<2-10>
\animatevalue<1-10>{\opaqueness}{100}{0}
\begin{colormixin}{\the\opaqueness!averagebackgroundcolor}
  \frametitle{Fadeout Frame}

  This text (and all other frame content) will fade out when the
  second slide is shown. This even works with
  {\color{green!90!black}colored} \alert{text}.
\end{colormixin}
\end{frame}

\newcount\opaqueness
\newdimen\offset
\begin{frame}
  \frametitle{Flying Theorems (You Really Shouldn't!)}

  \animate<2-14>

  \animatevalue<1-15>{\opaqueness}{100}{0}
  \animatevalue<1-15>{\offset}{0cm}{-5cm}
  \begin{colormixin}{\the\opaqueness!averagebackgroundcolor}
  \hskip\offset
  \begin{minipage}{\textwidth}
    \begin{theorem}
      This theorem flies out.
    \end{theorem}
  \end{minipage}
\end{colormixin}

  \animatevalue<1-15>{\opaqueness}{0}{100}
  \animatevalue<1-15>{\offset}{-5cm}{0cm}
  \begin{colormixin}{\the\opaqueness!averagebackgroundcolor}
  \hskip\offset
  \begin{minipage}{\textwidth}
    \begin{theorem}
      This theorem flies in.
    \end{theorem}
  \end{minipage}
\end{colormixin}
\end{frame}

```

[ARTICLE](#) 在 `article` 模式中会忽略该命令。

如果动画“图像 (graphics)”存放于一个单独的 (individual) 外部图形文件中，我们必须确保使用了 `\multiinclude` 命令（在第 14.1.3 节阐述了该命令）以及 `\animate` 命令。例如，我们可以像下面那样创建一个动画，假设已创建了名为 `animation.1` 到 `animation.10` 的图形文件：

```

\begin{frame}
  \animate<2-9>
  \multiinclude[start=1]{animation}
\end{frame}

```

14.1.3 包含存放在多个图形文件中的外部动画

一些动画以下述方式存放于外部文件：对于动画的每一阶段（stage），均有一个包含一幅图像（image）的图像文件与之对应。我们可以用来自于 ppower4 宏包的 `mpmulti.sty` 样式文件方便地包含（include）一系列的图像。由 Klaus Guntermann 创作的 `mpmulti.sty` 样式引进（introduces）了一个名为 `\multiinclude` 的命令，该命令能获得像 `mygraphic` 这样的图像文件的基名（base name），然后搜索名为 `mygraphic.0`、`mygraphic.1` 等等这样的文件，直到没有这样的文件为止。然后使用 `\includegraphics` 命令包含这些文件，这些文件会“一个接一个地叠放在一起”。而且，最重要的是在每一图像后面插入一个 `\pause` 命令。该命令已在 ppower4 宏包中定义好了，它与 BEAMER 宏包中的 `\pause` 命令具有相同的效果。正因如此，ppower4 和 BEAMER 两者均会首先显示基本图像（basic graphic），然后在每一幻灯片中显示下一图像。

如果我们试图直接使用 `mpmulti.sty`，将会遭遇这样一个问题即包含名为 `pause.sty` 的文件，该文件是 ppower4 宏包的一部分。

我们必须确保使用了来自 BEAMER 的 `xmpmulti.sty`。该文件与 `mpmulti` 几乎相同，只有两点不同：第一点，`xmpmulti.sty` 不包含 `pause.sty`，`pause.sty` 理论上和 BEAMER 相冲突，虽然 BEAMER 包含一个变通方法（workaround）以回避（sidesteps）该问题；第二点，通过给定一个特殊的默认叠层规 `xmpmulti.sty` 扩展了 `\multiinclude` 命令。该效应的解释如下。

```
\usepackage{xmpmulti}
```

定义 `\multiinclude` 命令。该宏包的代码归功于 Klaus Guntermann，而我的代码只起了补遗（additions）的作用。该宏包可以和 BEAMER 及 ppower4 一起使用，例如，如果在一个 ppower4 的演示稿中包含了 `pause` 宏包，则 `xmpmulti` 宏包可作为 `mpmulti` 宏包的替代品。

```
\multiinclude[<{default overlay specification}>][<{options}>]{<{base file name}>}
```

即：

```
\multiinclude[<{默认的叠层规则}>][<{选项}>]{<{基文件名}>}
```

除可能指定一个 `<{default overlay specification}>` 外，该命令与来自 ppower4 宏包的 `\multiinclude` 命令相同。

如果没有指定叠层规则，该命令会搜索名为 `<{base file name}>.<{number}>` 的文件，`<{number}>` 是从 0 开始增大的数字。只要找到了这些文件，就会配给（issue）一个 `\includegraphics` 命令给它们。跟在第一个文件后面文件会放在第一个文件“之上”。在任何两次调用 `\includegraphics` 命令之间，均会插入一个 `\pause` 命令。我们可以通过给定合适的 `<{options}>` 来修改这种行为，请参阅下面。

举例：假定 MetaPost 创建了名为 `gra.0`、`gra.1`、`gra.2` 的文件。然后我们就可以创建由三张幻灯片组成的帧，在这三张幻灯片中如下所述的那样递增地（incrementally）显示图像：

```

\begin{frame}
  \multiinclude{gra}
\end{frame}

```

提供了 $\langle default overlay specification \rangle$ 后效果如下：首先，在两个图像之间不会插入 `\pause` 命令，相反，每个图像由一个 `actionenv` 环境包围，该环境的叠层规则设为 $\langle default overlay specification \rangle$ 。

举例：我们可以用 `\multiinclude[<+>]{gra}` 命令获得和前面的例子相同的效果。

举例：要获得 $\langle default overlay specification \rangle$ 更有趣的用法，可以考虑下面的用法：

```
\multiinclude[<alert@+| +>]{gra}
```

这通常使图像新近添加的部分变成红色（假定我们没有使用图像自身的特殊的颜色）。

举例：要使每一个图像完全替换前一个图像，我们可以使用：`\multiinclude[<+>]{gra}`。

可能会给定下述 $\langle options \rangle$ （这些选项与来自 `ppower4` 宏包的初始命令的选项相同）：

- `pause= $\langle command \rangle$` 用 $\langle command \rangle$ 替换默认的暂停命令 `\pause`。如果给定了一个 $\langle default overlay specification \rangle$ ，则默认的暂停命令为空（empty）；否则默认的暂停命令为 `\pause`。注意像 `\pauselevel` 这样的命令在 `\beamer` 中不可用。
- `graphics= $\langle options \rangle$` 传递 $\langle options \rangle$ 给 `\includegraphics` 命令。
举例：`\multiinclude[graphics={height=5cm}]{gra}`
- `format= $\langle extension \rangle$` 使我们搜索的文件名由 $\langle base file name \rangle.\langle number \rangle$ 更改为 $\langle base file name \rangle-\langle number \rangle.\langle extension \rangle$ 。注意圆点（dot）更改成了连字符（hyphen）。该选项允许我们包含（include）.jpg 文件。
- `start= $\langle number \rangle$` 指定开始的 $\langle number \rangle$ 。默认为 0。
- `end= $\langle number \rangle$` specifies the end $\langle number \rangle$. The default is infinity.

注意：如果没有使用 `format=` 选项，`\includegraphics` 命令会变得有点不知所措，图像文件究竟是什么格式的呢。毕竟，图像文件以隐藏的（cryptic）“格式后缀”.0 或.1 结尾。使用下面的命令，我们就可以告诉 `\includegraphics` 命令它一无所知的文件后缀其实就是 .mps：

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

14.2 声音

我们可以在演示稿中包含声音（sounds）。这样，当我们打开一张幻灯片或点击特定的按钮时就会播放该声音。在 `multimedia` 宏包中已定义了用于包含声音的命令，关于 `multimedia` 宏包的介绍请参考第 14.1.1 节。

正如第 14.1.1 节指出的那样，把声音当作动画（movie）并用 `\movie` 命令处理，就可以将该声音包含在一个 PDF 演示稿中。虽然在大多数情况下这很完美，但在下面这两种情况下上述的处理方法无法让人满意：

1. 当关闭页面时，会关闭任何正在播放的动画（movie）。因此，我们就不能用 `\movie` 命令创建要持续播放很长时间的聲音。
2. 我们无法同时播放两个动画。

PDF 规范采用特殊的聲音对象（sound objects），该聲音对象的处理方式与动画对象（movie objects）的处理方式大不相同。我们可以用 `\sound` 命令创建一个聲音对象，`\sound` 命令和 `\movie` 命令相似。还有另一个命令 `\hyperlinksound`，它和 `\hyperlinkmovie` 命令相似。虽然理论上使用 `\sound` 命令更好，但使用该命令前必须确保以下这些事情：

- 能同时播放多个声音。尤其是，能同步 (in parallel) 播放一般声音 (general sound) 和 (希望无声的) 动画。
- 在关闭当前页面时 (虽然这不是必须的)，仍能重复播放一个声音。
- 声音文件的数据能完整地嵌入 (embedded in) 到一个 PDF 文件中，这样就无需“随身携带”其它文件。
- 声音对象和 dvips、ps2pdf 不兼容，只和 pdflatex 兼容。
- 对播放声音的哪部分的能力较弱。尤其是，不会显示控制条，我们也就无法指定开始播放的时间及播放持续的时间。
- 一些版本的 Acrobat Reader 存在隐错 (bug)，使得提供声音文件编码 (encoding) 的精确细节变得很有必要，我们必须提供采样率 (the sampling rate)、声道 (channels) 数 (单声道或立体声)、每样本位 (number of bits per sample)、样本编码方法 (sample encoding method) (raw、signed、Alaw 或 μ law)。
- 如果我们不知道这些数据或提供的数据错误，将不能正确地播放声音。
- 好像我们只能包含未经压缩的声音数据，它可能是巨大的。PDF 规范没有这样的要求，但某些版本的 Acrobat Reader 无法播放任何压缩的数据。支持的数据格式是 .aif 和 .au。

`\sound[(options)]{(sound poster text)}{(sound filename)}`

即：

`\sound[(选项)]{(声音广告文本)}{(声音文件名)}`

该命令在 PDF 文件中插入名为 *(sound filename)* 的声音。对于 `\movie`，当播放声音时要能访问该声音文件。不像 `\movie`，我们无论如何都可以使用 `inlinesound` 选项在 PDF 文件中嵌入声音。

对于动画 (movie)，会将 *(sound poster text)* 置入一个盒子中，当点击该盒子时开始播放动画。然而，我们也可以设置该盒子为空 (empty) 并只使用 `autostart` 选项。一旦重新播放一个声音，就可以通过播放另一声音或使用 `\hyperlinkmute` 命令来停止前一声音。

支持的声音格式依赖浏览程序。某此版本的 Acrobat Reader 支持 .aif 和 .au。有时我们必须指定像采样率这样的信息，虽然这样的信息必须从声音文件中提取，虽然 PDF 标准要求浏览程序这样做。就这一点而言，某些版本的 Acrobat Reader 好像不符合标准 (non-standard-conforming)。

该命令只能和 `pdflatex` 一起联用。如果我们使用了 `dvips`，仍会显示广告画 (poster)，但是点击它不会起作用，也不会以任何方式嵌入声音。

举例：`\sound[autostart,samplingrate=22050]{}{applause.au}`

可能会给出下面的 *(options)*：

- **autostart**. 当打开页面时立即播放声音。
- **automute**. 当离开当前页面时使所有声音变小 (muted)。
- **bitspersample**=*(8 or 16)*. 指定声音文件每个样本的位数 (the number of bits per sample)。如果为 16，则无需指定该选项。
- **channels**=*(1 or 2)*. 指定声音是单声道还是立体声。如果声音是单声道，则无需指定该选项。
- **depth**=*(TeX dimension)*. 覆盖 *(sound poster text)* 盒子的深度并将其设为给定的尺寸。

- **encoding**=*<method>*. 指定编码方法 (encoding method), 它可以是 Raw、Signed、muLaw、ALaw。如果是 muLaw, 则无需指定该选项。
- **height**=*<TeX dimension>*. 覆盖 *<sound poster text>* 盒子的高度并将其设定为尺寸。
- **inlinesound** 将声音数据直接存储在文件 PDF 中。
- **label**=*<sound label>*. 指定一个标签 (label) 给声音以便于后来给 `\hyperlinksound` 命令作参考, 该命令可用于播放一个声音。*<sound label>* 不是一个普通的声音。
- **loop** 或 **repeat**. 当到达声音结束处时重新播放声音。
- **mixsound**=*<true or false>*. 如果设为 **true**, 除其它正在播放的声音外, 播放该声音。如果设为 **false**, 在播放该声音之前所有其它声音 (虽然不是来自动画) 都停止。默认为 **false**。
- **samplingrate**=*<number>*. 指定声音文件每秒的样本数 (the number of samples per second)。如果为 44100, 则无需指定该选项。
- **width**=*<TeX dimension>*. 和 **height** 选项相似, 只不过该选项用于设置广告盒子 (poster box) 的宽度。

举例: 下面的例子为幻灯片创建一个“背景声音”, 假定 `applause.au` 编码正确 (每秒样本数为 44100、单声道、 μ law 编码、每样本位数为 16)。

```
\sound[autostart]{}{applause.au}
```

和动画相像, 声音可作为特定声音链接的目标。

```
\hyperlinksound[<options>]{<sound label>}{<text>}
```

即:

```
\hyperlinksound[<选项>]{<声音标签>}{<文本>}
```

使 *<text>* 成为一个声音链接 (sound hyperlink)。当点击 *<text>* 时, 带有 *<sound label>* 标签的声音将开始播放。

可能会给出下面的 *<options>*:

- **loop** 或 **repeat**. 当到达声音结束时重新播放声音。
- **mixsound**=*<true or false>*. 如果设为 **true**, 除其它正在播放的声音外, 播放该声音。如果设为 **false**, 在播放该声音之前所有其它声音 (虽然不是来自动画) 都停止。默认为 **false**。

因为没有直接的方式停止声音的重播。下面的命令很有用:

```
\hyperlinkmute{<text>}
```

使 *<text>* 成为一个链接, 当点击该链接时, 停止重播所有声音。

14.3 幻灯片过渡

PDF 和 Acrobat Reader 提供了一个标准化的方法用于定义幻灯片过渡 (*slide transition*)。这样的过渡是一种视觉效果 (visual effect)，它用于显示幻灯片，例如，不管什么在显示之前都会慢慢“溶解 (dissolve)”并代之幻灯片的内容，而不是立即显示幻灯片。

有多个命令用于指定当显示幻灯片时使用什么效果。考虑下面的例子：

```
\frame{\pgfuseimage{youngboy}}
\frame{
  \transdissolve
  \pgfuseimage{man}
}
```

`\transdissolve` 命令使第二帧的幻灯片以“溶解的方式”显示。注意，溶解是第二帧的属性 (property)，而不是第一帧的属性。我们可以在帧的任何地方放置该命令。

过渡命令是叠层规则敏感的 (overlay-specification-aware)。我们可像下面这样，将两个帧折叠 (collapse) 成一个帧：

```
\begin{frame}
  \only<1>{\pgfuseimage{youngboy}}
  \only<2>{\pgfuseimage{man}}
  \transdissolve<2>
\end{frame}
```

这种状态是，在第一张幻灯片会显示小男孩 (young boy)，在第二张幻灯片会显示老 (old man)，当显示第二张幻灯片时，它会以“溶解的方式”显示。

下面罗列了用于创建过渡效果的不同命令。所有这些命令均带一个可选的参数，该参数可能包含一个 $\langle key \rangle = \langle value \rangle$ 的配对 (pairs) 列表。可能会使用下面的选项：

- `duration = \langle seconds \rangle`. 指定过渡效果所需的 $\langle seconds \rangle$ 数。默认值为 1 秒，但一个更短的时间 (如 0.2 秒) 更合适。浏览程序特别是 Acrobat，能以略显古怪的方式解释该选项。
- `direction = \langle degree \rangle`. 用于“直射 (directed)”效果。该选项指定效果的直射角度，允许的取值是 0、90、180、270，如果用于闪光效果 (glitter effect)，也可以取值 315。

ARTICLE 在 `article` 模式中会忽略所有这些命令。

LYX 必须在 `TEX`-模式中插入这些命令。

```
\transblindshorizontal<\langle overlay specification \rangle>[\langle options \rangle]
```

即：

```
\transblindshorizontal<\langle 叠层规则 \rangle>[\langle 选项 \rangle]
```

拉开水平百叶窗，以这种方式显示幻灯片。

举例：`\transblindshorizontal`

```
\transblindsvvertical<\langle overlay specification \rangle>[\langle options \rangle]
```


即:

`\transblindsvertical<⟨叠层规则⟩>[⟨选项⟩]`

拉开垂直百叶窗，以这种方式显示幻灯片。

举例: `\transblindsvertical<2,3>`

`\transboxin<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transboxin<⟨叠层规则⟩>[⟨选项⟩]`

一个矩形从四周向中心逐渐缩小，以这种方式显示幻灯片。

举例: `\transboxin<1>`

`\transboxout<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transboxout<⟨叠层规则⟩>[⟨选项⟩]`

一个矩形从中心向四周逐渐扩大，以这种方式显示幻灯片。

举例: `\transboxout`

`\transdissolve<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transdissolve<⟨叠层规则⟩>[⟨选项⟩]`

慢慢地溶解以前显示的内容，以这种方式显示幻灯片。

举例: `\transdissolve[duration=0.2]`

`\transglitter<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transglitter<⟨叠层规则⟩>[⟨选项⟩]`

闪光 (glitter) 以指定的方向扫过 (sweeps)，以这种方式显示幻灯片。

举例: `\transglitter<2-3>[direction=90]`

`\transreplace<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transreplace<⟨叠层规则⟩>[⟨选项⟩]`

直接替换前一张幻灯片 (默认的行为)。

`\transsplitverticalin<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transsplitverticalin<⟨叠层规则⟩>[⟨选项⟩]`

由两条垂线分别从左右两外侧向中间扫过，以这种方式显示幻灯片。

举例: `\transsplitverticalin`

`\transsplitverticalout<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transsplitverticalout<⟨叠层规则⟩>[⟨选项⟩]`

从中间裂开，两条垂线分别向左右两外侧扫过，以这种方式显示幻灯片。

举例: `\transsplitverticalout`

`\transsplithorizontalin<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transsplithorizontalin<⟨叠层规则⟩>[⟨选项⟩]`

两条水平线分别从上下两侧向中间扫过，以这种方式显示幻灯片。

举例: `\transsplithorizontalin`

`\transsplithorizontalout<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transsplithorizontalout<⟨叠层规则⟩>[⟨选项⟩]`

两条水平线从中间分别向上下两侧扫过，以这种方式显示幻灯片。

举例: `\transsplithorizontalout`

`\transwipe<⟨overlay specification⟩>[⟨options⟩]`

即:

`\transwipe<⟨叠层规则⟩>[⟨选项⟩]`

一条线向指定的方向扫过，然后“清除”以前的内容，以这种方式显示幻灯片。

举例: `\transwipe[direction=90]`

我们也可以用下面的叠层规则敏感的命令指定幻灯片显示多长时间:

`\transduration<⟨overlay specification⟩>{⟨number of seconds⟩}`

即:

`\transduration<⟨叠层规则⟩>{⟨秒数⟩}`

在全屏模式中，指定的幻灯片共显示 `⟨number of seconds⟩` 秒。如果设为 0 秒，则显示指定的幻灯片时将一闪而过。这种方法可用于创建假动画 (pseudo-animation)。

举例: `\transduration<2>{1}`

部分 III

更改外观的方法

BEAMER 提供了更改演示稿外观的方法，这些方法可以对演示稿的各个水平的细节进行更改。在最高层，主题 (*themes*) 可以方便地全面改变演示稿的外观。在最底层，模板 (*templates*) 允许我们对每一细节进行个性化的更改。

演示稿“外观”的两个重要方面是：`colors` 和 `fonts`。当然，`color` 和 `font` 主题可用于全面地控制演示稿的颜色和字体。

15 主题

15.1 五类主题

使用不同的主题 (*Themes*) 可以很容易地改变演示稿的外观。BEAMER 文档类使用五类不同的主题⁴⁰:

演示主题 (Presentation Themes) 理论上, 一个演示主题 (presentation theme) 控制着演示稿的每一细节的外观。也就是说, 某个主题控制着列表中数字的颜色、背景是什么、使用什么字体、列表标记是球形还是矩形等等。因此, 当我们选择了一个主题时, 我们的演示稿将以某种方式显示。演示主题非常融洽地整合了颜色主题 (color theme)、字体主题 (font theme)、内部主题 (inner theme)、外部主题 (outer theme)

颜色主题 (Color Themes) 一个颜色主题 (color theme) 只控制着演示稿使用的颜色。如果我们选择了某个特定的演示主题及颜色主题, 那么只会更改演示稿的颜色。一个颜色主题可以非常详细地指定颜色: 例如, 指定按钮的边界的颜色、按钮的背景色、按钮文本的颜色。

字体主题 (Font Themes) 一个字体主题 (font theme) 控制着演示稿使用的字体的属性。可以分别指定演示稿中所有文本元素使用的字体

内部主题 (Inner Themes) 一个内部主题 (inner theme) 控制着排版演示稿的什么元素。这些元素就是“内建”于帧的所有元素, 它们不是顶部导航区 (headline)、底部导航区 (footline) 或侧栏 (sidebar) 的一部分。“内建”于帧的元素包括所有的 enumerations 排序环境、itemize 环境、块 (Block) 环境、定理 (Theorem) 环境、目录 (table of contents)。例如, 一个内部主题会指定一个排序列表 (enumeration) 的数字不带圆点, 其后跟随一个小圆, 但不会指定数字或小圆使用什么颜色 (这是颜色主题要做的事情), 也不会指定使用什么字体 (这是字体主题要做的事情)。

外部主题 (Outer Themes) 一个外部主题 (outer theme) 控制着幻灯片的“外部”或“边界”的显示方式。它指定是否显示顶部导航区 (headline)、底部导航区 (footline)、侧栏 (sidebar), 指定在其中显示什么内容; 在哪里显示徽标 (Logo)、导航符 (navigation symbol)、导航条 (navigation bars) 等等。外部主题也指定在哪里放置帧标题 (frametitle)、如何显示帧标题。

不同的主题放在 `beamer/themes` 目录⁴¹的五个子目录中, 它们是: `theme`、`color`、`font`、`inner`、`outer`。主题存储在普通的样式文件 (style file) 中。然而, 要使用一个主题, 必需使用下面特定的命令:

```
\usetheme[options]{name list}
```

即:

```
\usetheme[选项]{名称列表}
```

安装一名为 `<name>` 的演示主题。目前, 该命令等效于为 `beamertheme<name>.sty` 样式文件⁴²的每一 `<name>` 使用 `\usepackage` 命令, 这里的 `<name>` 是 `<name list>` 中的一个。

⁴⁰有众多的网页介绍这些 BEAMER 自带的主题, 并配有插图, 做幻灯片时可据此来选择合适的搭配方案, 极为直观, 如: <http://www.hartwork.org/beamer-theme-matrix/>; 在 <http://latex.simon04.net/> 这里展示了 Rogier Koppejan 在阿姆斯特丹大学 (University of Amsterdam) 学习时自创的主题。

⁴¹在装有 CTEX 套装的 Windows 7 平台中, 该目录的位置是如 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes`。

⁴²在装有 CTEX 套装的 Windows 7 平台中, 这些文件位于如 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\theme` 中, 如 `beamerthemeAnnArbor.sty`、`beamerthemeWarsaw.sty` 等文件。

`\usecolortheme[options]{name list}`

即:

`\usecolortheme[选项]{名称列表}`

所作所为与 `\usetheme` 相同, 只是适应于颜色主题。颜色样式文件命名为 `beamercolortheme<name>.sty`⁴³。

`\usefonttheme[options]{name}`

即:

`\usefonttheme[选项]{名称}`

所作所为与 `\usetheme` 相同, 只是适应于字体主题。字体样式文件命名为 `beamerfonttheme<name>.sty`。

`\useinnertheme[options]{name}`

即:

`\useinnertheme[选项]{名称}`

所作所为与 `\usetheme` 相同, 只是适应于内部主题。内部样式文件命名为 `beamerinnertheme<name>.sty`。

`\useoutertheme[options]{name}`

即:

`\useoutertheme[选项]{名称}`

所作所为与 `\usetheme` 相同, 只是适应于外部主题。外部样式文件命名为 `beameroutertheme<name>.sty`。

如果我们不使用上述任何一个命令, 则默认地会使用五类主题中的 *default* 主题, 它是一个稳重的 (sober) 的主题。下面将阐述 BEAMER 文档类的演示主题。将分几节分别阐述元素 (element)、布局 (layout)、颜色 (color)、字体 (font)。

15.2 无导航条的演示主题

一个演示主题 (presentation theme) 控制着演示稿的每一细节的显示。通常, 选择了某个演示主题后, 我们无需指定与演示稿外观有关的任何东西 — 主题的创作者已经为我们考虑了一切。然而, 我们可以通过使用不同的颜色、字体、元素、布局主题 (layout theme), 或者通过直接改变特定的颜色、字体、模板来改变演示稿的外观。

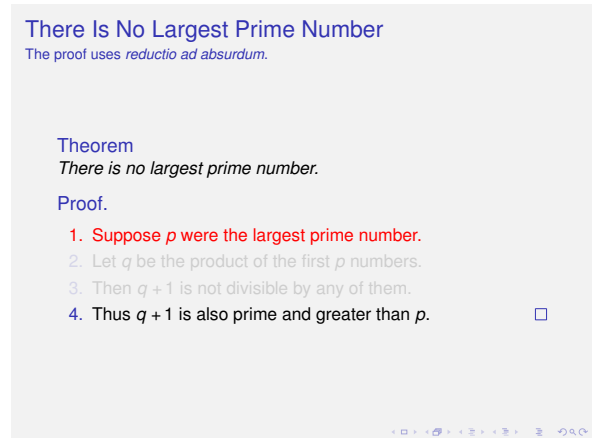
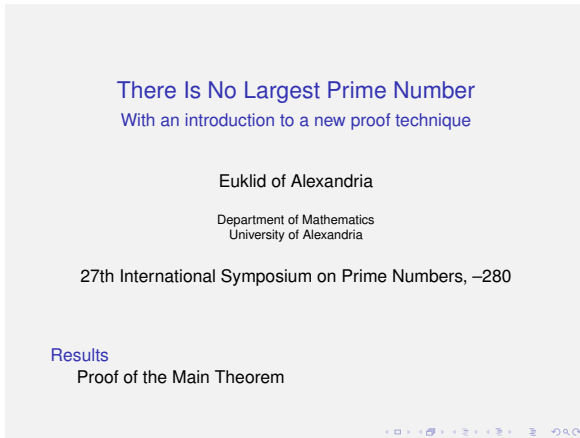
提尔·坦图 (Till Tantau) 教授想尽了一切办法如何命名一个主题。他决定用不同的约定俗成的名字来代替累赘的名字: 除两种特殊的情况外, 所有演示主题均按城市命名。在这些城市中或其附近举办过他或他的合作者参加过的讨论会或研讨会。

下列没有提及作者的 theme 均由提尔·坦图教授创作。如果一个主题不是由我们 (而是由其它某个人) 创作, 这时会指明其创作者。我们常常会略微改动或“修正”某个主题, 这时也会列出该主题的最初创作者。

⁴³如在装有 CTEX 套装的 Windows 7 平台中, 位于如 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\color` 中的 `beamercolorthemestructure.sty`、`beamercolorthememealbatross.sty` 等文件。

`\usetheme{default}`

举例：

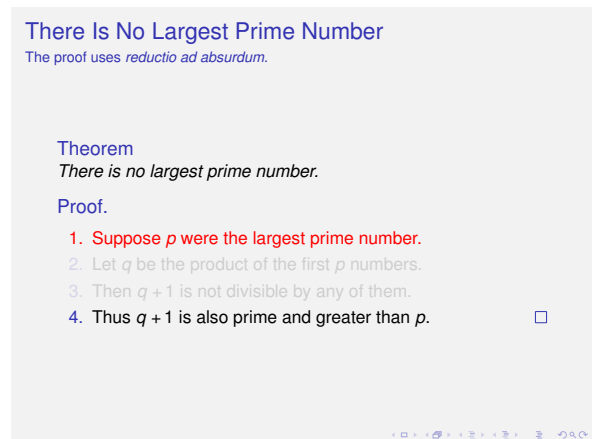
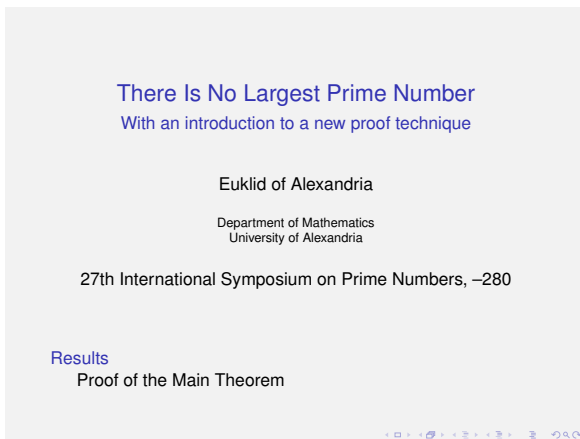


主题的名称：default（默认）。

正如名字所提示的那样，默认情况下安装该主题。该主题比较稳重严肃（sober no-nonsense），它使用的颜色和字体变化较少。除很长的演讲外，该主题适应于所有类型的演讲。

`\usetheme[headlineheight=<headline height>,footlineheight=<footline height>]{boxes}`

举例：



对于该主题，我们可以指定任意数量的用于顶部导航区（headline）和底部导航区（footline）的盒子（boxes）的模板。可以用下面的命令为另一盒子添加一个模板：

`\addheadlinebox{<beamer color>}{<box template>}`

即：

`\addheadlinebox{<beamer 颜色>}{<盒子模板>}`

每次调用该命令时，就会在顶部导航区添加一个盒子，先添加的盒子显示在左边。全部盒子大小相同。

`<beamer color>` 用于设置盒子的前景色和背景色

举例：

```
\addheadbox{section in head/footer}{\tiny\quad 1. Box}
```

```
\addheadbox{structure}{\tiny\quad 2. Box}
```

要实现上述命令的功能，还有一个办法就是安装一个包含两个 `beamercolorbox` 的 head 模板：

```
\setbeamertemplate{headline}
{\leavevmode
\begin{beamercolorbox}[width=.5\paperwidth]{section in head/footer}
  \tiny\quad 1. Box
\end{beamercolorbox}%
\begin{beamercolorbox}[width=.5\paperwidth]{structure}
  \tiny\quad 2. Box
\end{beamercolorbox}
}
```

情况越复杂，上述命令就越有弹性（flexibility）。

```
\addfootbox{<beamer color>}{<box template>}
```

即：

```
\addfootbox{<beamer 颜色>}{<盒子模板>}
```

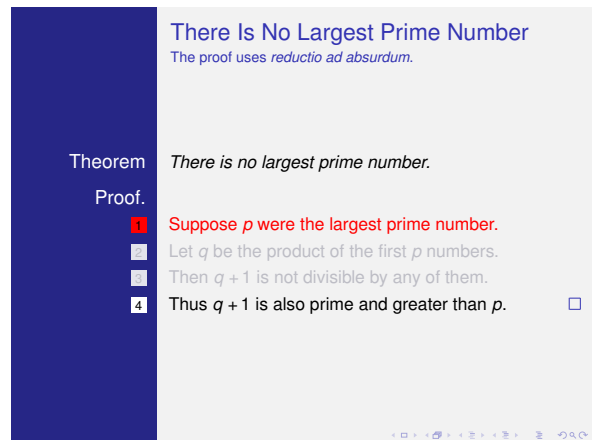
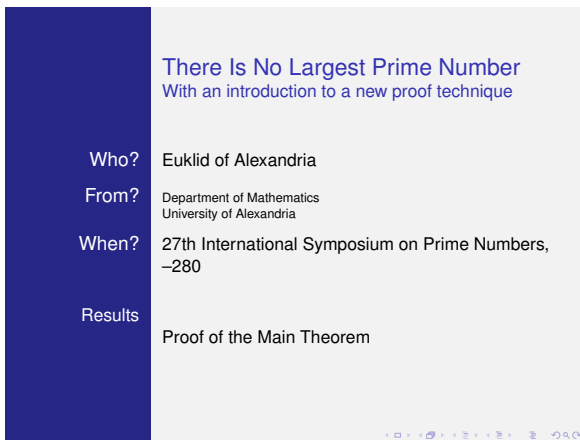
举例：

```
\addfootbox{section in head/footer}{\tiny\quad 1. Box}
```

```
\addfootbox{structure}{\tiny\quad 2. Box}
```

```
\usetheme[<options>]{Bergen}
```

举例：



主题的名称：Bergen（卑尔根）。

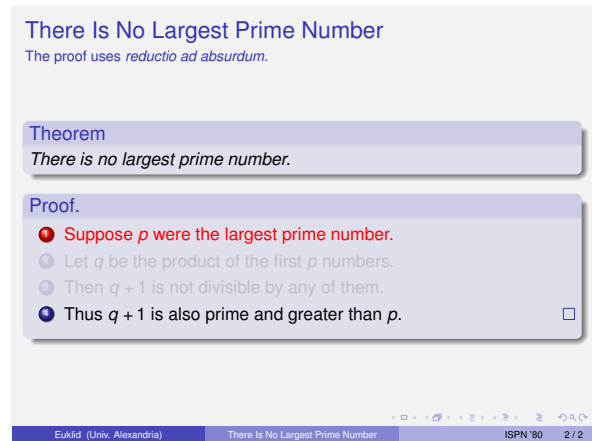
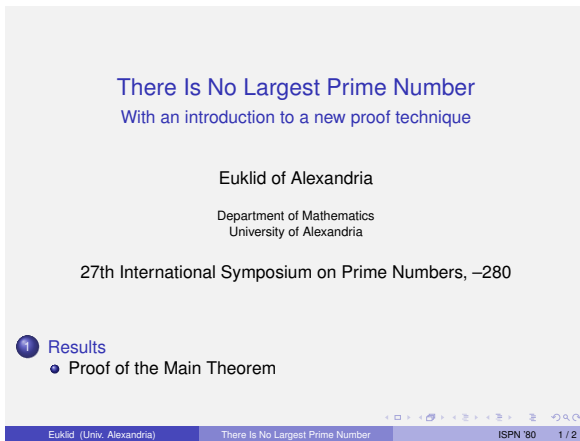
主题的作者：IWPEC。

卑尔根（Bergen）主题是基于 `inmargin` 和 `rectangles` 内部主题的。该主题不常用，因为它有比其它主题有较多的空白，而且该主题不适合分栏（columns）。我们可以查阅（consult）关于 `inmargin` 内部主题的评价（remarks）。

卑尔根（Bergen）是挪威的第二大城市。该主题是 IWPEC 于 2004 年创作的。

`\usetheme[options]{Boadilla}`

举例：



主题的名称：Boadilla（博阿迪利亚）。

主题的作者：Manuel Carro（曼努埃尔·卡洛）。

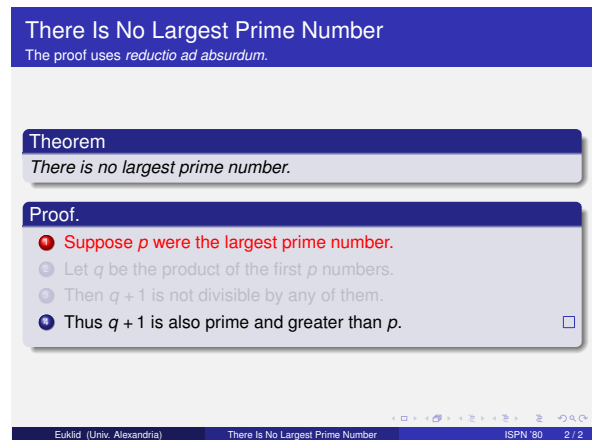
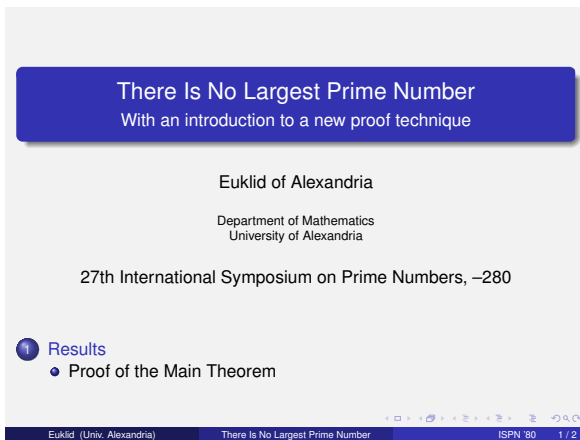
Boadilla 主题可以在有限的空间内容纳更多的信息。可能会给出下面的 *options*⁴⁴：

- `secheader` 插入顶部导航区（headline），并在其中显示当前节和当前小节。默认情况下，不显示顶部导航区（headline）。

Boadilla（博阿迪利亚）是西班牙首都马德里（Madrid）附近的一个村庄，大学的计算机科学系座落于此。

`\usetheme[options]{Madrid}`

举例：



主题的名称：Madrid（马德里）。

主题的作者：Manuel Carro（曼努埃尔·卡洛）。

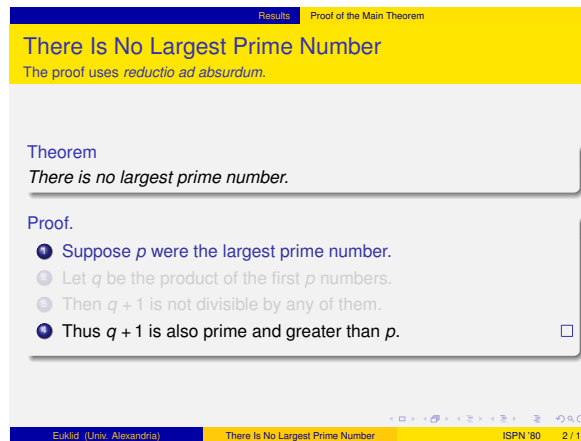
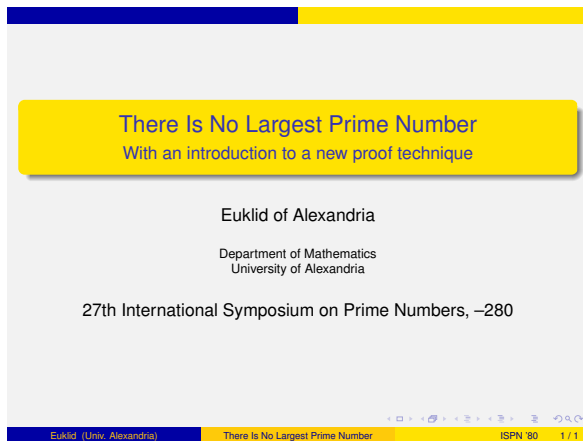
除使用更深的颜色及列表图标（itemize icons）没有变外，其它和 Boadilla（博阿迪利亚）主题相似。可能会给出的 *options* 和 Boadilla（博阿迪利亚）主题相同。

马德里（Madrid）是西班牙的首都。

⁴⁴相应的语句是：`\usetheme[secheader]{Boadilla}`

`\usetheme{AnnArbor}`

举例:



主题的名称: Ann Arbor (安阿伯)。

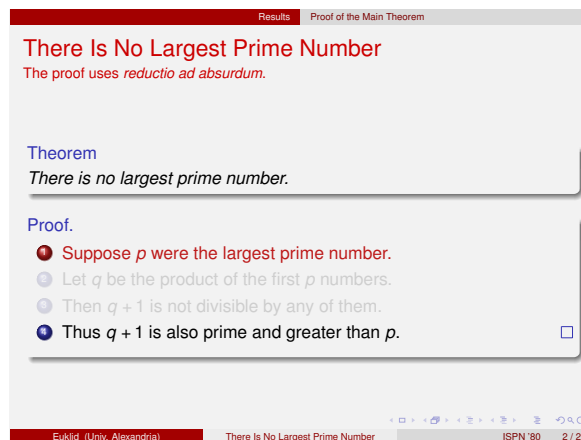
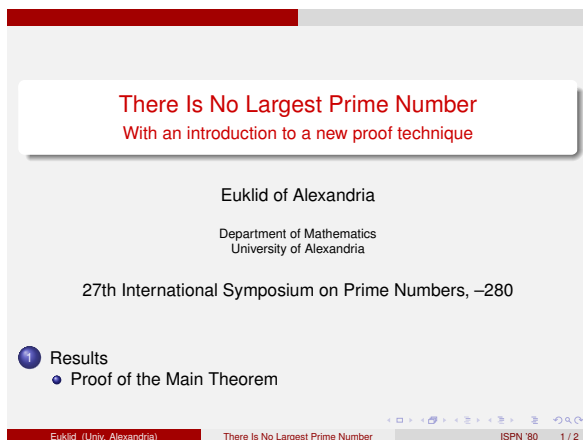
主题的作者: Madhusudan Singh (马杜苏登·辛格)。

和 Boadilla (博阿迪利亚) 主题相似, 但使用密执安大学 (University of Michigan) 的颜色。

密执安大学位于美国的安阿伯。

`\usetheme{CambridgeUS}`

举例:



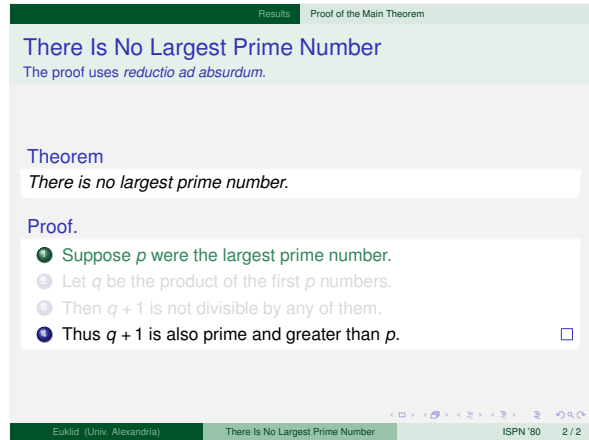
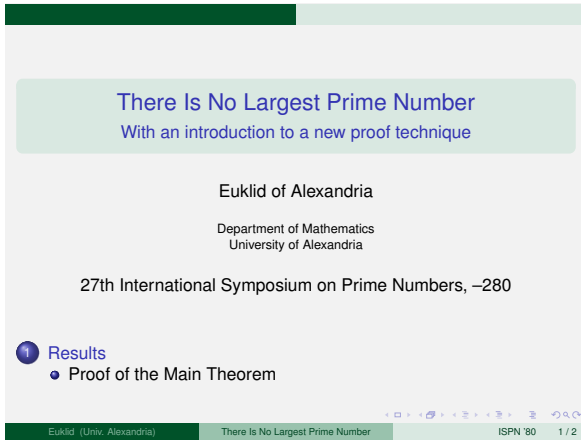
主题的名称: CambridgeUS (坎布里奇 US)。

主题的作者: Madhusudan Singh (马杜苏登·辛格)。

和 Boadilla (博阿迪利亚) 主题相似, 但使用麻省理工学院 (Massachusetts Institute of Technology, MIT) 的颜色。

`\usetheme{EastLansing}`

举例:



主题的名称: EastLansing (东兰辛)。

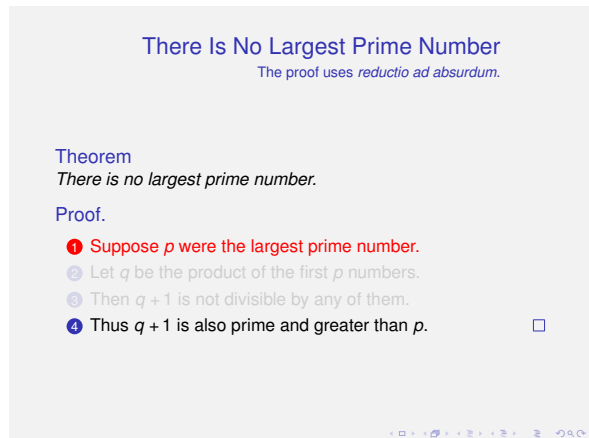
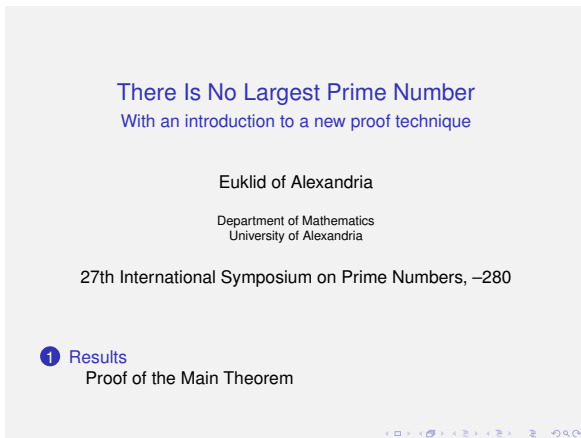
主题的作者: Alan Munn (艾伦·曼)。

和 Boadilla (博阿迪利亚) 主题相似, 但使用密执安州立大学 (Michigan State University) 的颜色。

密执安州立大学位于东兰辛 (East Lansing)。

`\usetheme{Pittsburgh}`

举例:



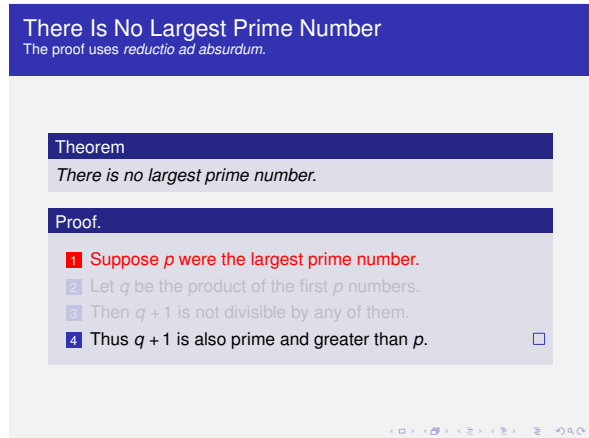
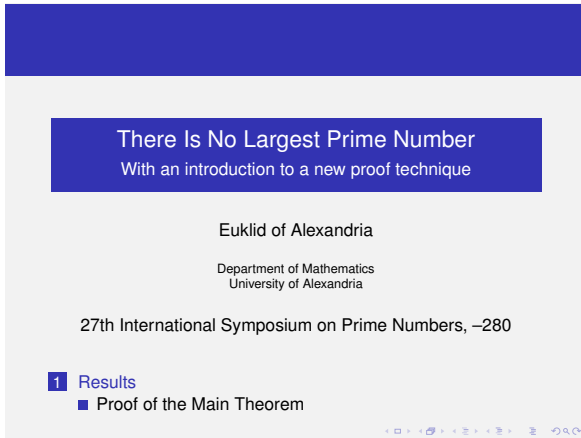
主题的名称: Pittsburgh (匹兹堡)。

是一个稳重的主题。右对齐的帧标题 (Frame Title) 在每一帧的内部产生一种有趣的“紧凑”感。

Pittsburgh (匹兹堡) 是美国东部一个城市。2004 年在这里举办了第二次单核苷酸多态性 (SNPs) 和单倍体型 (haplotypes) 的重组 (RECOMB) 会议。

`\usetheme[options]{Rochester}`

举例:



主题的名称: Rochester (罗切斯特)。

是一个主要 (dominant) 的不包含导航元素的主题。使用不同的颜色主题 (color theme) 后会降低其灵活性。

可能会给出下面的 $\langle options \rangle$ ⁴⁵:

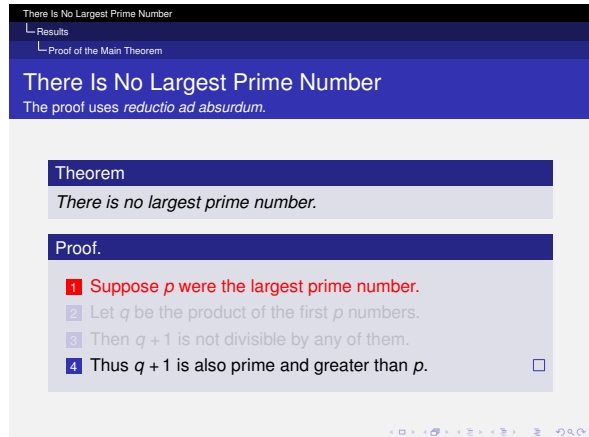
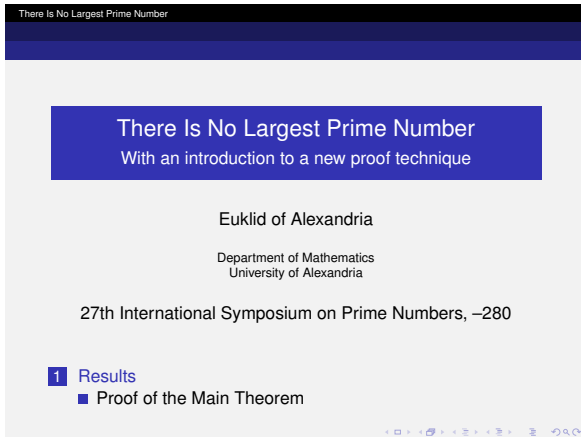
- `height= $\langle dimension \rangle$` 设置帧标题条 (frame title bar) 的高度。

Rochester (罗切斯特) 是离美国纽约较远的一座城市, 提尔 (Till) 教授在于 2001 年访问过这里。

15.3 顶部带有树形导航条的演示主题

`\usetheme{Antibes}`

举例:



主题的名称: Antibes (安提贝)。

是主要的顶部带有树状 (tree-like) 导航的主题。矩形元素 (rectangular elements) 映射 (mirror) 了顶部的导航。使用不同的颜色主题会降低该主题的灵活性。

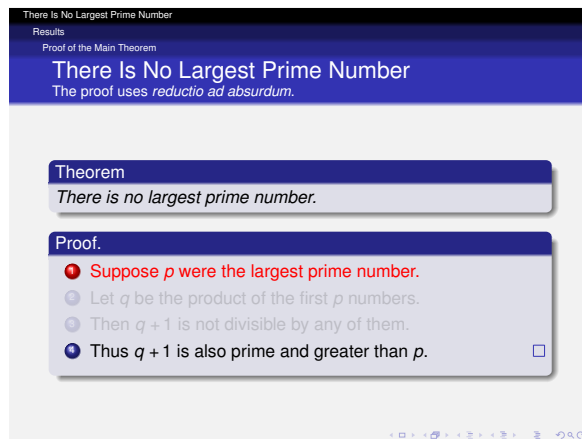
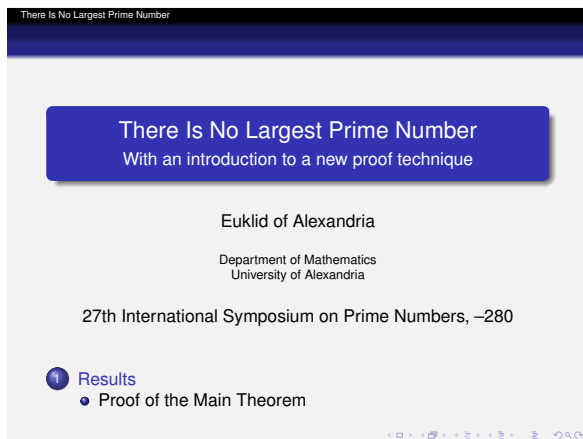
Antibes (安提贝) 是法国南部的一座城市。2002 年在这里举办了 STACS⁴⁶。

⁴⁵一个示例语句: `\usetheme[height=15pt]{Rochester}`

⁴⁶STACS: Symposium on Theoretical Aspects of Computer Science, 计算机科学理论方面的研讨会。

`\usetheme{JuanLesPins}`

举例：



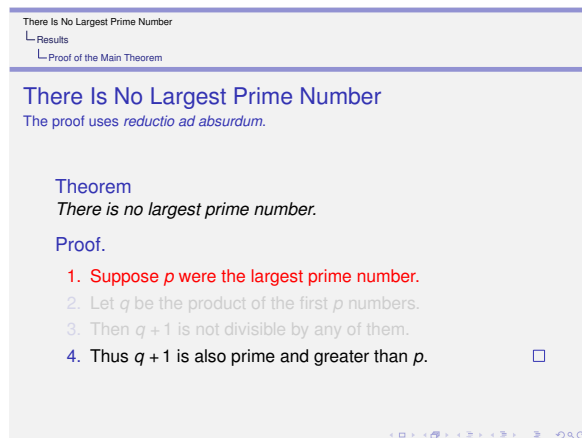
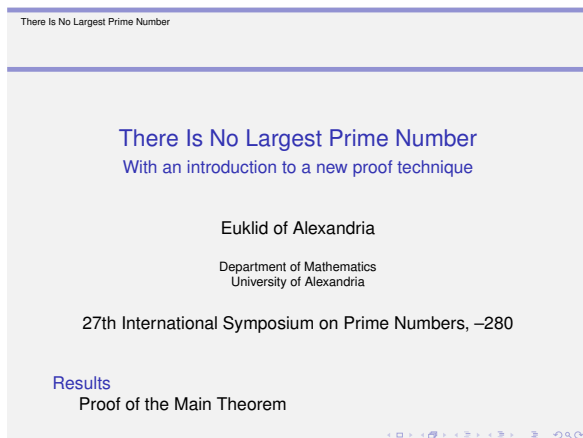
主题的名称：JuanLesPins（瑞昂莱潘）。

为 Antibes（安提贝）主题的变体，该主题的外观比较“光滑”。选用不同的颜色主题会降低其灵活性。

Juan-Les-Pins（瑞昂莱潘）是 Antibes（安提贝）附近一个舒适的村庄。2002 年在这里举办了 STACS。

`\usetheme{Montpellier}`

举例：



主题的名称：Montpellier（蒙彼利埃）

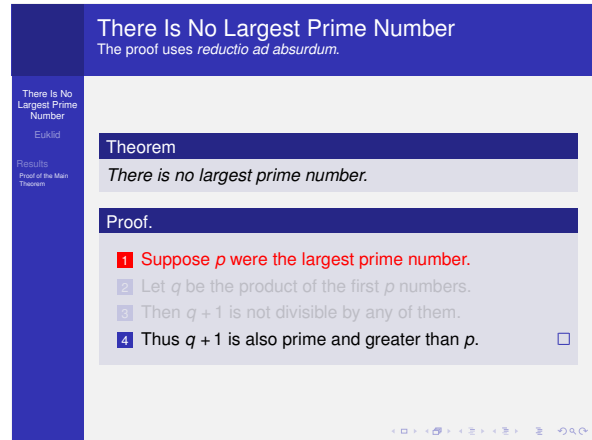
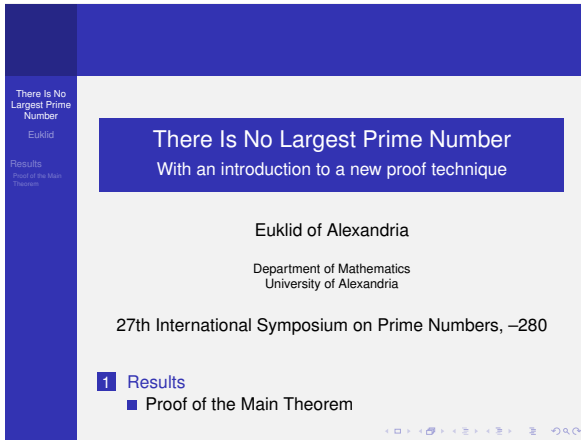
是一个带基本导航提示的稳重的主题。使用不同的颜色主题可以使顶部导航区（headline）更灵活。

Montpellier（蒙彼利埃）位于法国南部。2004 年在这里举办了 STACS。

15.4 侧边带有目录的演示主题

`\usetheme[options]{Berkeley}`

举例：



主题的名称：Berkeley（伯克利）。

是主流主题之一。多数情况下导航条（Navigation Bar）位于左侧。帧标题（Frame Title）的高度占据两行半，因此，在使用长标题（long titles）时须小心。会在拐角处放置徽标（logo）。以矩形区域为布局。使用不同的颜色主题会降低该主题的灵活性。

默认情况下，侧栏（sidebar）的目录中的当前条目会以不同的颜色高亮显示。一个好的方法是，通过为当前要点（current point）选用不同的背景色来高亮显示当前条目。颜色主题 `sidebartab` 会安装适当的（appropriate）颜色，只需声明 `\usecolortheme{sidebartab}`。

`\usecolortheme{sidebartab}`

该颜色主题对所有会在侧栏中显示目录的主题均有效。

该主题对于像演讲（lectures）这样长的讲话（talks）很有用，因为侧栏中的目录会一直显示。

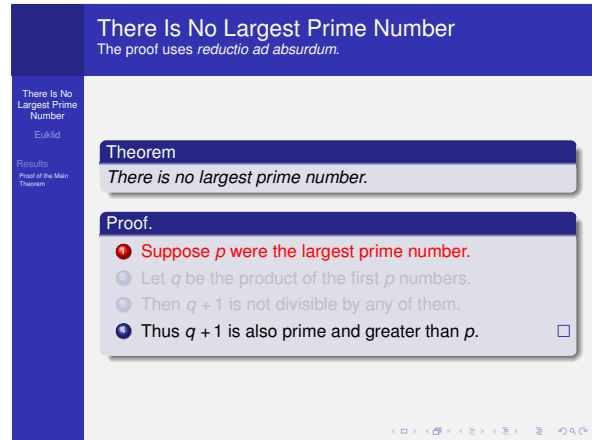
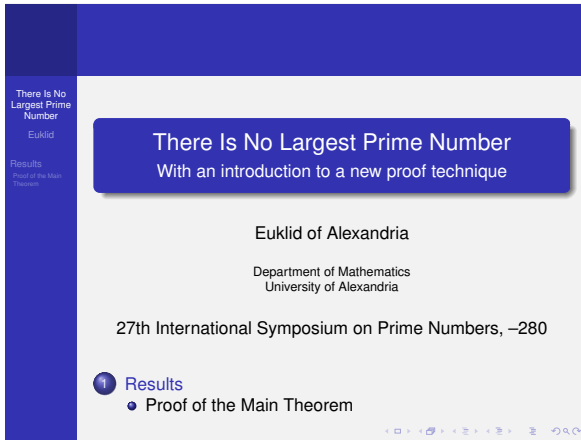
可能会给出下面的 `\options`：

- `hideallsubsections` 只在侧栏中显示节（sections）。如果我们想节省空间这将很有用。
- `hideothersubsections` 只在侧栏中显示当前节的小节（subsections）。如果想节省空间这将很有用。
- `left` 将侧栏置于左侧（默认）。
- `right` 将侧栏置于右侧。
- `width=<dimension>` 设置侧栏的宽度。如果将它设为 0，将不会显示侧栏。

Berkeley（伯克利）位于美国西海岸，靠近 San Francisco（圣弗兰西斯科，即旧金山）。提尔教授在 2004 年访问了 Berkeley。

`\usetheme[<options>]{PaloAlto}`

举例：



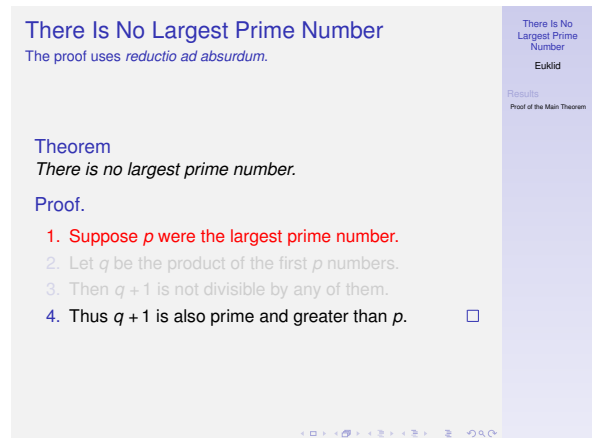
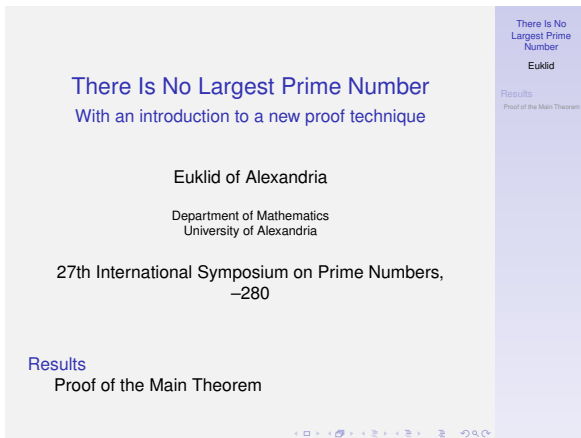
主题的名称: PaloAlto (帕罗奥多)。

Berkeley (伯克利) 主题的变体。可能会给出和 Berkeley (伯克利) 主题相同的 $\langle options \rangle$ 。

PaloAlto (帕罗奥多) 也靠近 San Francisco (圣弗兰西斯科, 即旧金山)。在这里举办了 Bay Area Theory Workshop 2004 (海湾地区理论研讨会 2004)。

`\usetheme[$\langle options \rangle$]{Goettingen}`

举例:



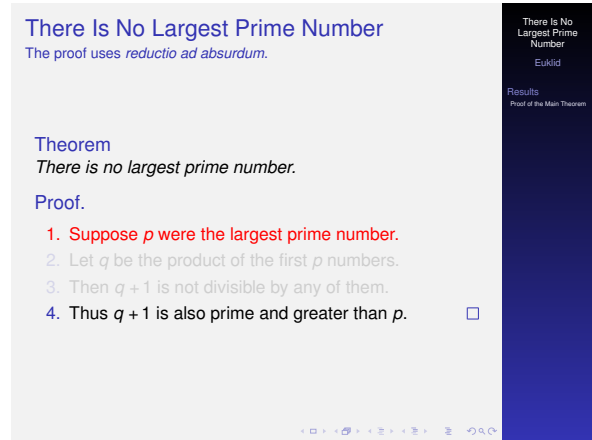
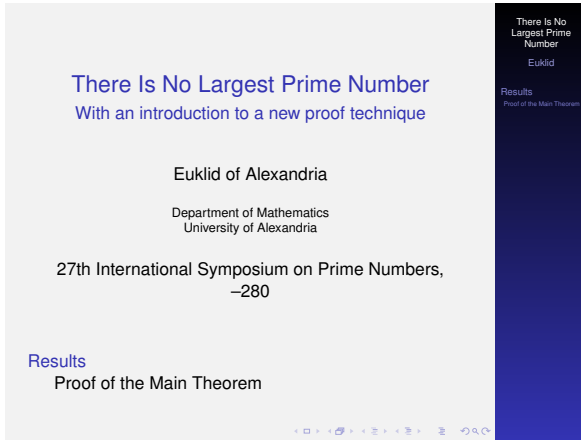
主题的名称: Goettingen (哥廷根)。

一个相对稳重的主题, 该主题适应于一个要求具有完整目录侧栏的长演讲。可能会给出和 Berkeley (伯克利) 主题相同的 $\langle options \rangle$:

Göttingen (哥廷根) 是德国的一个城市。在这里举办了第 42 届 Theorietag。

`\usetheme[$\langle options \rangle$]{Marburg}`

举例:



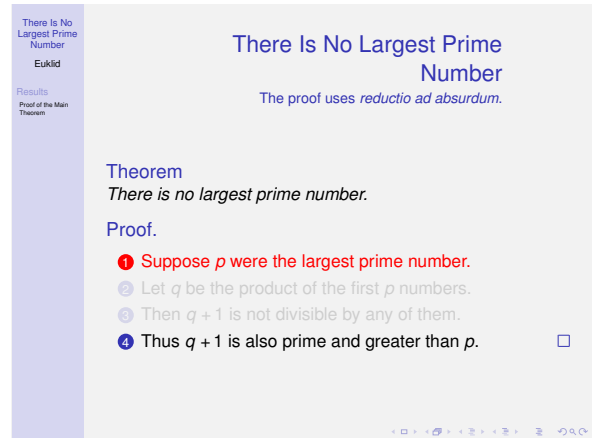
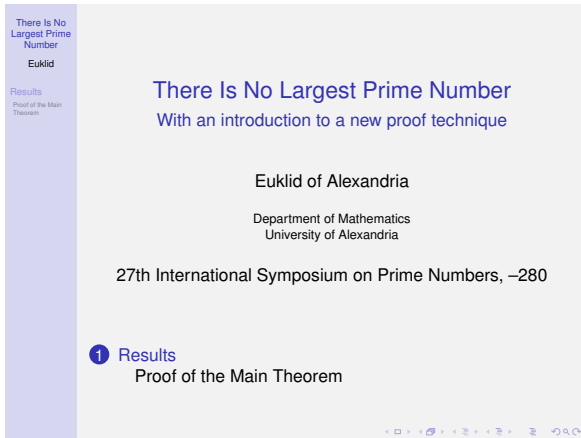
主题的名称: Marburg (马尔堡)。

Goettingen (哥廷根) 的变体。可能会给出相同的 $\langle options \rangle$ 。

Marburg (马尔堡) 是德国的一个城市。在这里举办了第 46 届 Theorietag。

`\usetheme[$\langle options \rangle$]{Hannover}`

举例:



主题的名称: Hannover (汉诺威)。

该主题左侧的侧栏和右对齐的 (right-flushed) 帧标题保持平衡。

可能会给出下面的 $\langle options \rangle$:

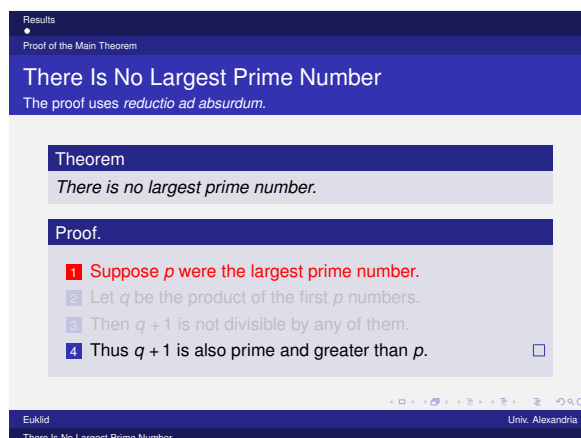
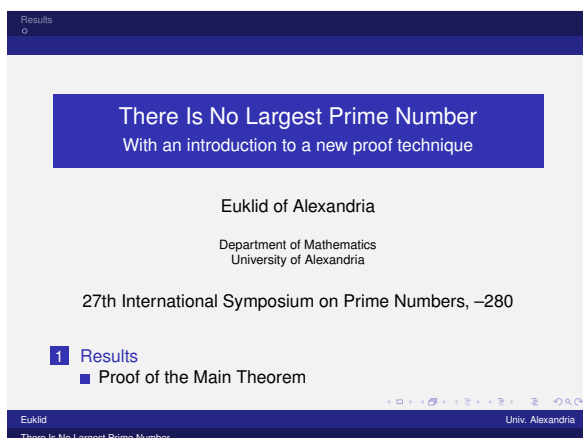
- `hideallsubsections` 在侧栏中只显示节 (sections)。如果要节省空间, 这将很有用。
- `hideothersubsections` 在侧栏中只显示当前节的小节 (subsections)。如果要节省空间, 这将很有用。
- `width= $\langle dimension \rangle$` 设置侧栏的宽度。

Hannover (汉诺威) 是德国的一个城市。在这里举办了第 48 届 Theorietag。

15.5 带有微帧导航的演示主题

`\usetheme[$\langle options \rangle$]{Berlin}`

举例：



主题的名称：Berlin（柏林）。

一个主流（dominant）主题，该主题带有强烈的色彩并大量使用矩形区域。顶部和底部导航区能给出大量信息但幻灯片的实际内容占有较少的空间。该主题适合于观众不太了解演讲的标题这样的大会（conferences）。通过使用不同的颜色主题可以降低该主题的地位。

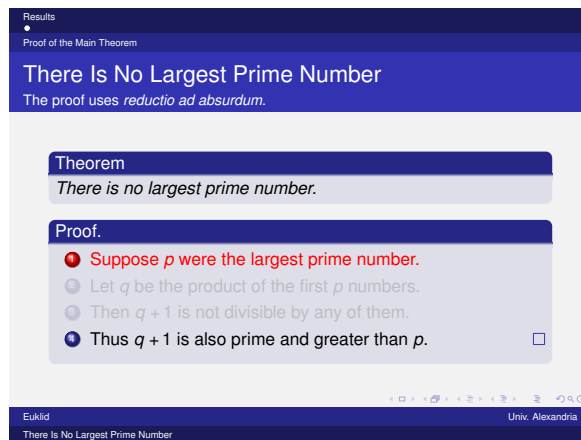
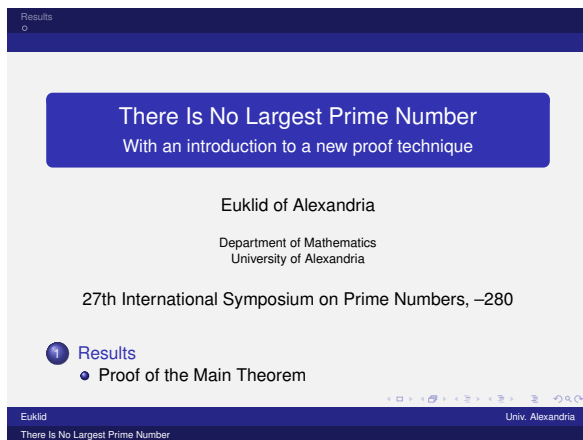
可能会给出下面的 *<options>*：

- **compress** 使顶部导航区的微帧只占一行。这有利于节省空间。

Berlin（柏林）是 Germany（德国）的首都。

`\usetheme[<options>]{Ilmenau}`

举例：



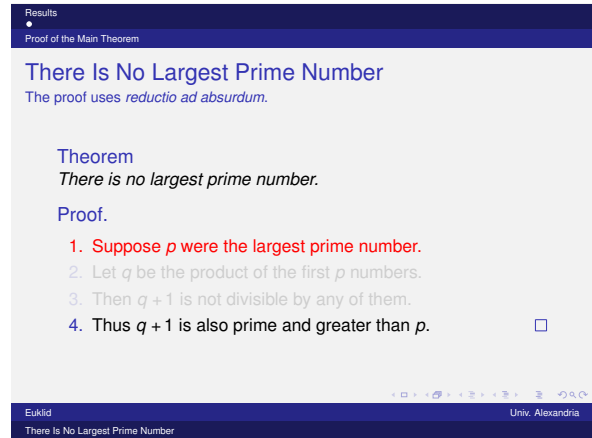
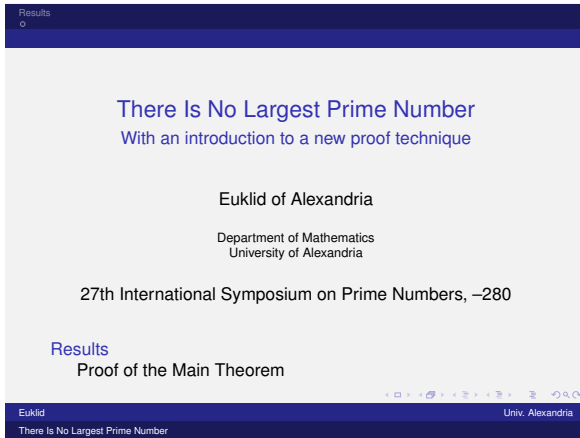
主题的名称：Ilmenau（伊尔姆瑙理）。

Berlin（柏林）主题的一个变体。可能会给出和 Berlin（柏林）主题相同的 *<options>*。

Ilmenau（伊尔姆瑙理）是德国的一座城市，在此举办了第 40 届 Theorietag。

`\usetheme{Dresden}`

举例：



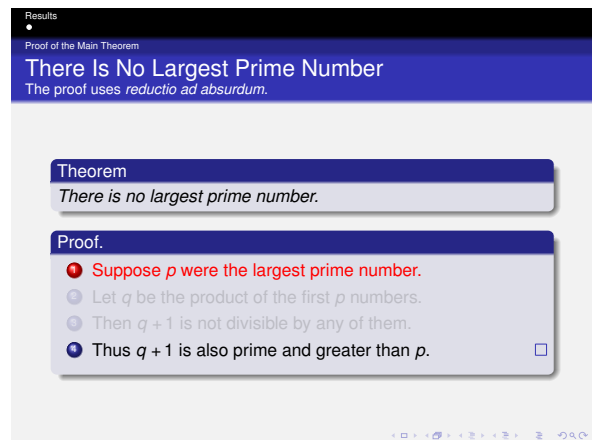
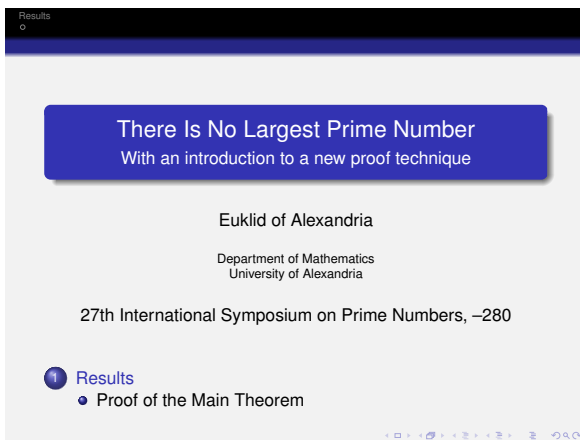
主题的名称: Dresden (德累斯顿)。

Berlin (柏林) 主题的一个变体。该主题导航区的顶/底部之间有一明显的分隔线, 该主题的主体文本是稳重的。可能会给出和 Berlin (柏林) 主题相同的 $\langle options \rangle$ 。

Dresden (德累斯顿) 是德国的一座城市, 在此举办了 STACS 2001。

`\usetheme{Darmstadt}`

举例:



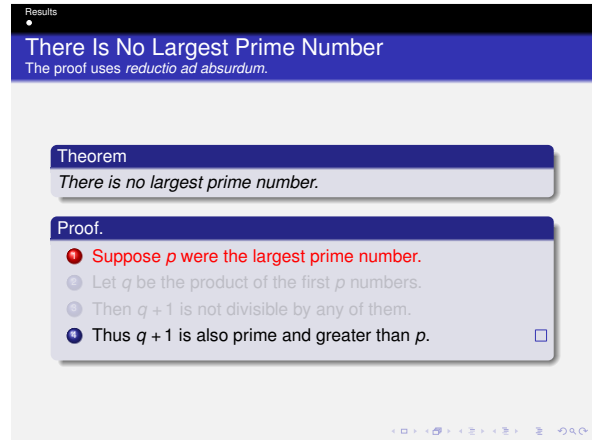
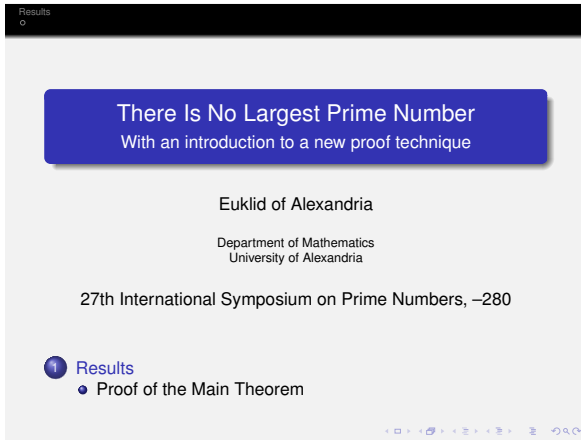
主题的名称: Darmstadt (达姆施塔特)。

该主题的导航区的上部 (navigational upper part) 和信息主体部分 (informational main part) 之间有一个明显的分隔线。通过使用不同的颜色主体, 可以使该分隔线不明显。

Darmstadt (达姆施塔特) 是德国的一座城市。

`\usetheme{Frankfurt}`

举例:



主题的名称: Frankfurt (法兰克福)。

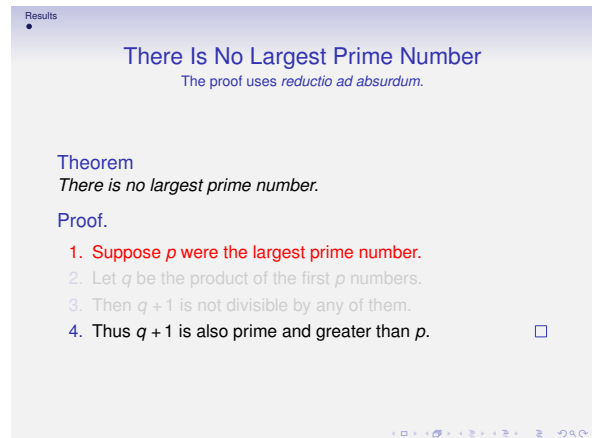
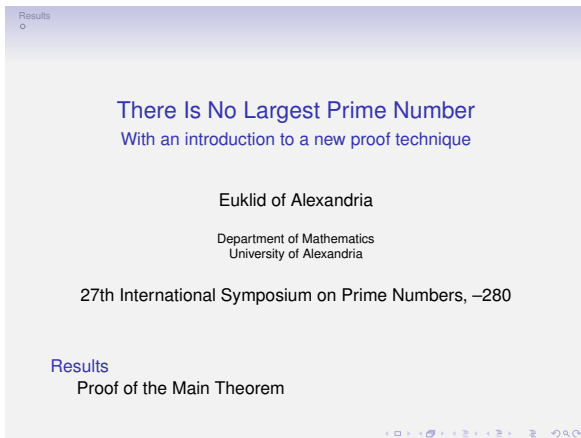
该主题是 Darmstadt (达姆施塔特) 主题的一个变体, 通过删除一些小节信息 (subsection information)

该主题显得更不那么杂乱 (cluttered)。

Frankfurt (法兰克福) 是德国的一座城市。

`\usetheme{Singapore}`

举例:



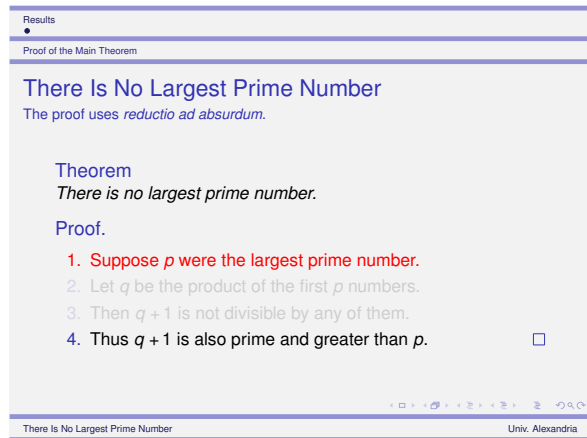
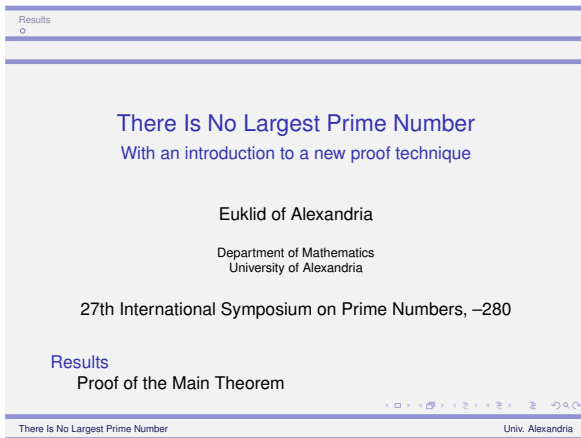
主题的名称: Singapore (新加坡)。

一个不太稳重的主题, 该主题的导航区不明显 (dominate)。

Singapore (新加坡) 位于东南亚。在此举办了 COCOON 2002。

`\usetheme{Szeged}`

举例:



主题的名称：Szeged（塞格德）。

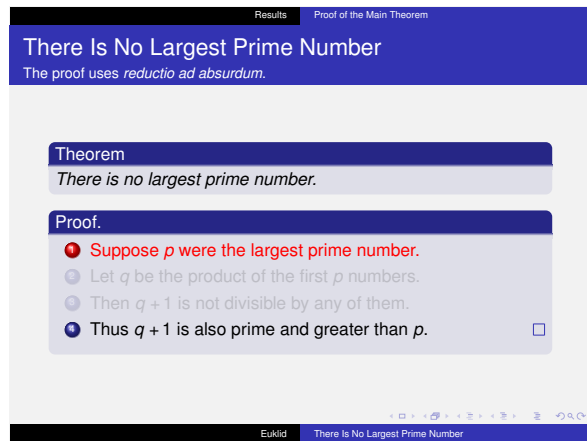
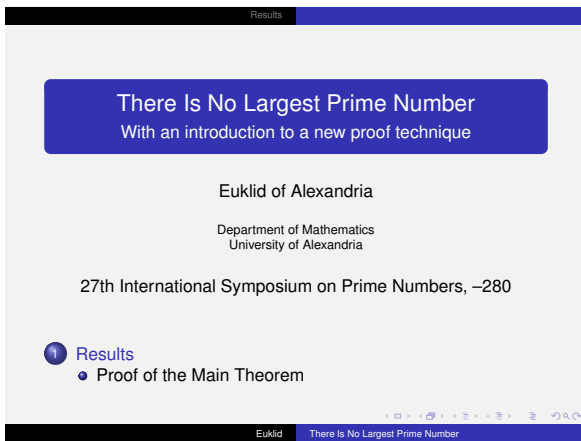
一个稳重的主题，该主题拥有几条明显的水平线。

Szeged（塞格德）是匈牙利南部的一座城市。在此举办了 DLT⁴⁷ 2003。

15.6 带有节和小节列表的演示主题

`\usetheme{Copenhagen}`

举例：



主题的名称：Copenhagen（哥本哈根）。

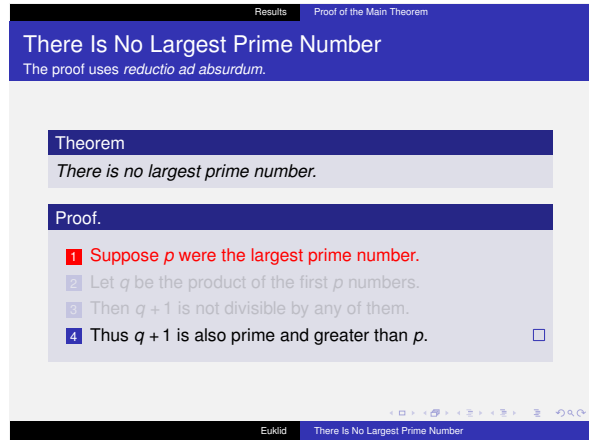
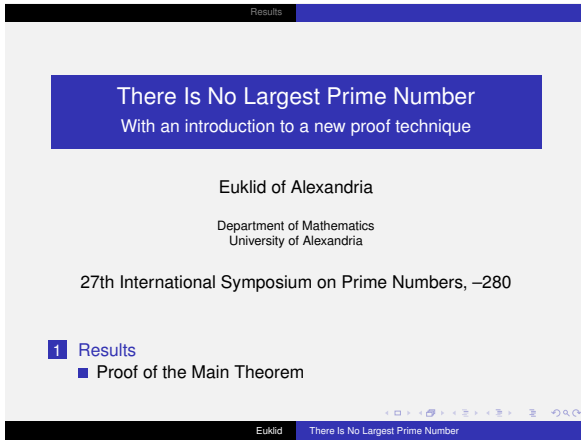
一个不太主流的（not-quite-too-dominant）主题。该主题在顶部给出了当前节和当前小节的简短信息，在底部给出了标题和作者的简短信息。因没有使用阴影，使演示稿看起来很“平淡（flat）”。在该主题中使用不同的颜色主题会使该主题变得更不主流。

Copenhagen（哥本哈根）是 Denmark（丹麦）的首都。它通过 Øresund bridge（厄勒海峡大桥）连接 Malmö（马尔默）。

`\usetheme{Luebeck}`

举例：

⁴⁷DLT: Developments in Language Theory, 语言理论进展。



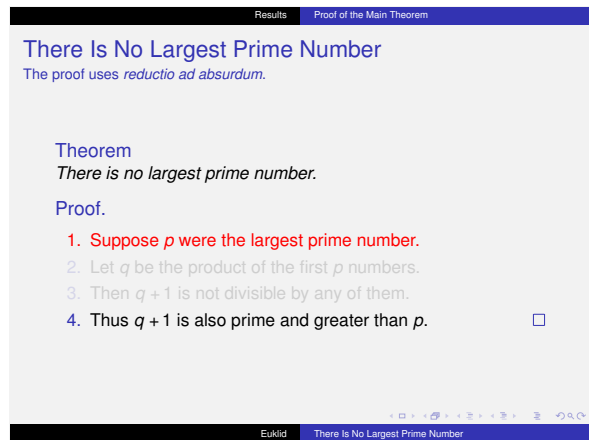
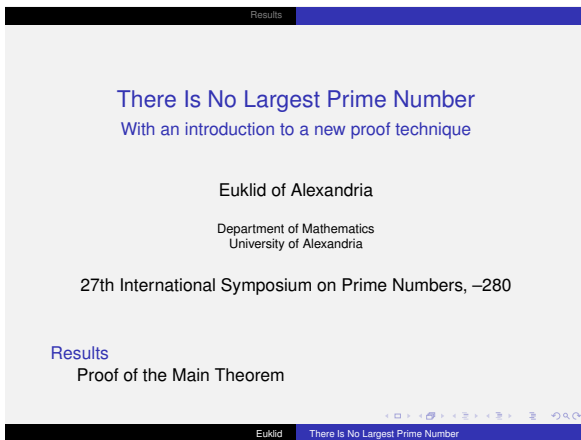
主题的名称: Luebeck (吕贝克)。

Copenhagen (哥本哈根) 主题的一个变体。

Lübeck (吕贝克) 是德国北部的一座城市。在此举办了第 41 届 Theorietag。

`\usetheme{Malmoe}`

举例:



主题的名称: Malmoe (马尔默)。

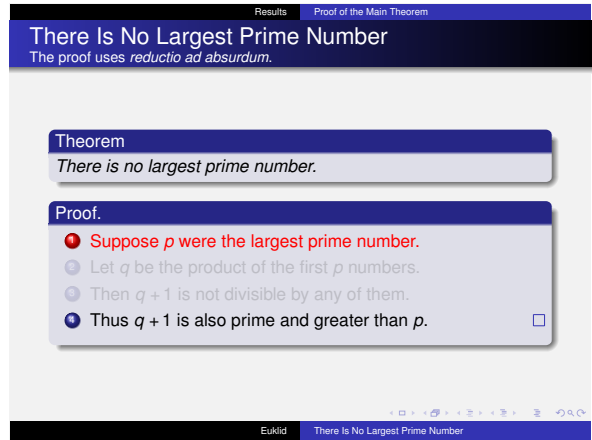
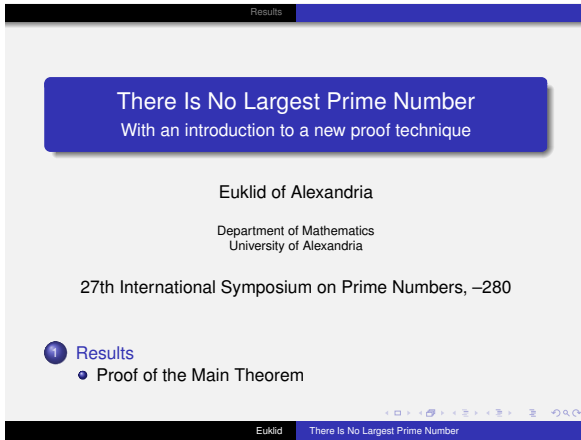
Copenhagen (哥本哈根) 主题的一个变体, 该主题很稳重。

Malmö (马尔默) 是 Sweden (瑞典) 南部的一座城市。在此举办过 FCT⁴⁸ 2001。

`\usetheme{Warsaw}`

举例:

⁴⁸FCT: Fundamentals of Computation Theory, 计算理论基础。



主题的名称：Warsaw（华沙）。

Copenhagen（哥本哈根）主题的一个主流变体。

Warsaw（华沙）是 Poland（波兰）的首都。在此举办过 MFCS⁴⁹ 2002。

15.7 能兼容的演示主题

早期版本的 BEAMER 包含了更多的主题。这些主题仍然有用（即向后兼容），但它们的应用不同（它们也会安装相应的颜色、字体、内部和外部主题）。它们可能不拥有在将来也可能不支持的颜色主题。下面罗列了哪些新主题用以替代旧主题。（在切换时，我们必须使用带有选项 `onlysmall` 的字体主题 `structurebold`）

旧主题	替换选项
none	使用 <code>compatibility</code> 。
bars	试用 <code>Dresden</code> 替换。
classic	试用 <code>Singapore</code> 替换。
lined	试用 <code>Szeged</code> 替换。
plain	试用 <code>none</code> 或 <code>Pittsburgh</code> 替换。
sidebar	对亮版本（light version）试用 <code>Goettingen</code> 替换，对暗版本（dark version）试用 <code>Marburg</code> 替换。
shadow	试用 <code>Warsaw</code> 替换。
split	试用 <code>Malmoe</code> 替换。
tree	试用 <code>Montpellier</code> 替换，对于导航条版本（bars version），试用 <code>Antibes</code> 或 <code>JuansLesPins</code> 替换。

⁴⁹MFCS: Mathematical Foundations of Computer Science, 计算机科学的数学基础。

16 内部主题、外部主题、模板

本节讨论 BEAMER 文档类中可用的内部主题 (inner themes) 和外部主题 (outer themes)。这些主题会为演示稿 (presentation) 的不同元素 (elements) 安装特定的模板 (*templates*)。模板的机制 (templates mechanism) 将在该节末尾讨论。

在我们讨论细节之前，先约定些这节会用到的术语 (terminology)。在 BEAMER 中，一个元素 (*element*) 是以特定方式排版的演示稿的一部分。元素的例子如帧标题 (Frame Title)、作者姓名 (author's name)、脚注符号 (footnote sign) 等。每一元素的外观由其相应的模板 (*template*) 控制。模板由内部主题、外部主题安装，内部主题 (*inner themes*) 只安装“主文本中的 (inside the main text)”元素的模板；外部主题 (*outer themes*) 只安装“主文本周围的 (inside the main text)”元素的模板。因此，内部主题和外部主题本质上是相同的。

16.1 内部主题

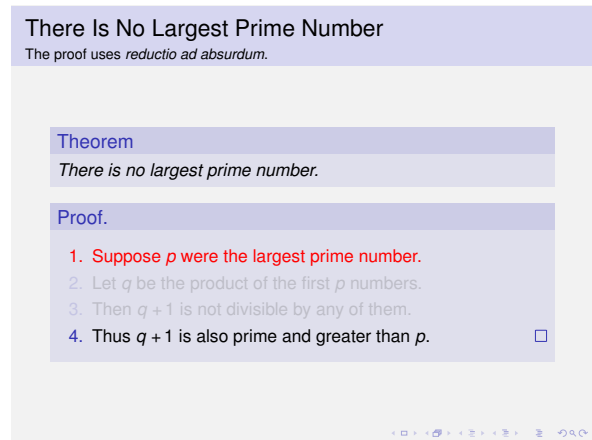
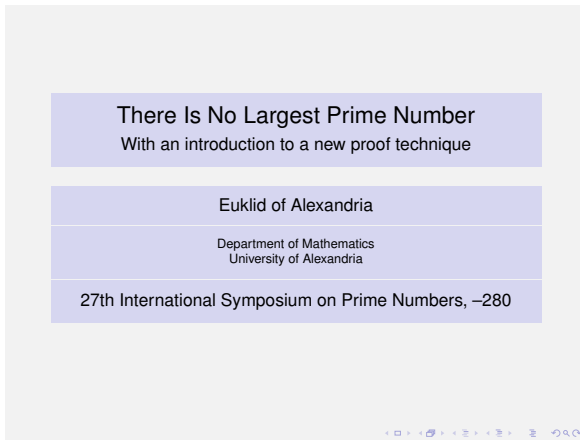
一个内部主题安装的模板控制 (dictate) 着如何排版下面这些元素：

- 标题 (Title) 和部分页面 (part pages)。
- Itemize 环境。
- Enumerate 环境。
- Description 环境。
- 块 (Block) 环境。
- 定理 (Theorem) 和证明 (proof) 环境。
- 图 (Figures) 和表 (Tables)。
- 脚注 (Footnotes)。
- 参考书目条目 (Bibliography entries)。

在下面的例子中，使用的颜色主题 `seahorse` 和 `rose` 显示了在哪里和如何获得 (honor) 背景色。而且，在默认的主题中，背景色已指定给所有元素。在默认的颜色主题中，所有大量的矩形区域都是透明的。

```
\useinnertheme{default}
```

举例：



主题的名称：default（默认）。

默认的元素主题（element theme）相当稳重。略显奢华（extravagance）的就是在 `itemize` 环境中的条目标记使用小三角形（triangle）而不是通常的圆点（dot）。

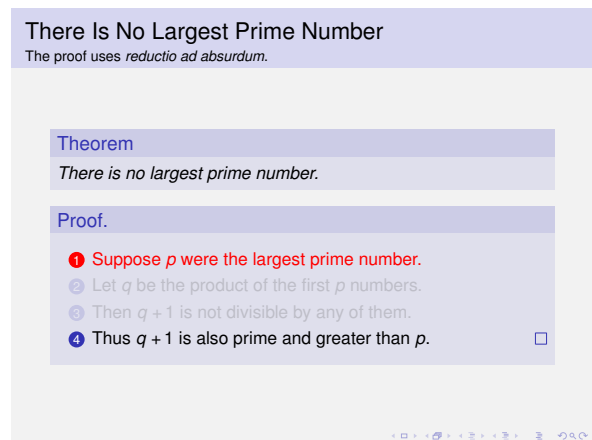
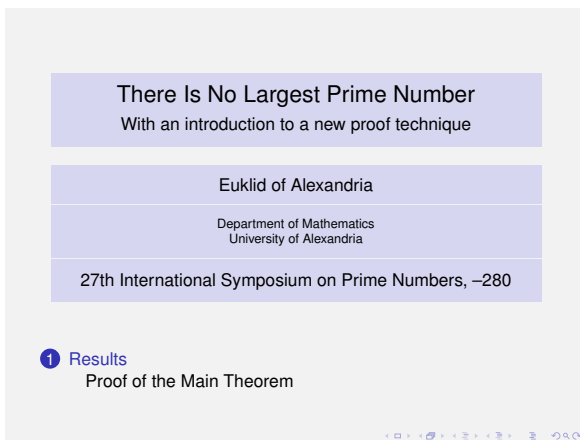
在某些情况下，主题将拥有元素的背景色规则（background color specifications），例如，如果我们指定块标题（block titles）的背景色为绿色（green），则块标题将拥有绿色的背景。下列元素可以拥有当前背景色规则：

- 封面（title page）中的标题（Title）、作者（author）、大学（institute）、日期（date）等字段。
- 块（Block）环境中的标题和正文。

可以拥有当前背景色规则的元素在将来会越来越多。

`\useinnertheme{circles}`

举例：

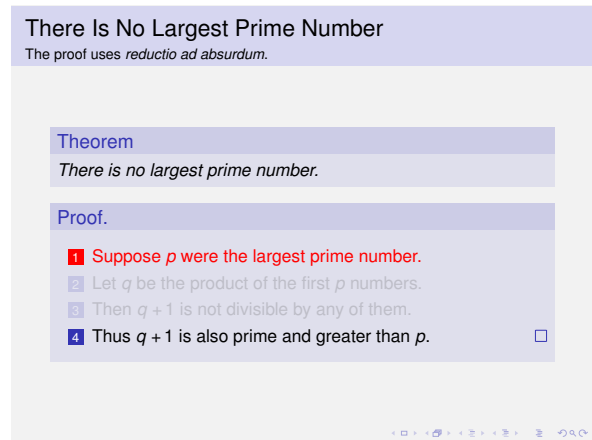
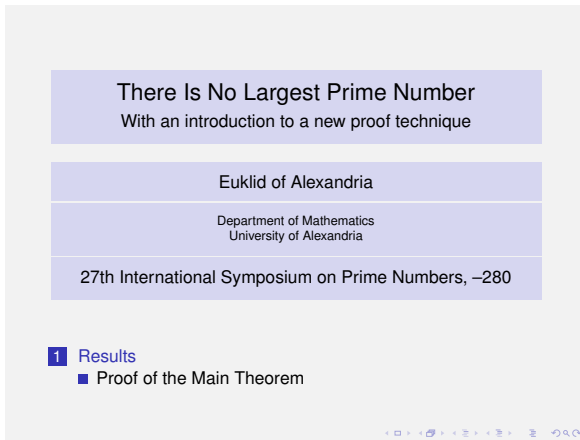


主题的名称：circles（圆）。

在该主题中，`itemize` 和 `enumerate` 的条目标记是小圆（circle）。同样，目录条目也是以圆开始的。

`\useinnertheme{rectangles}`

举例：

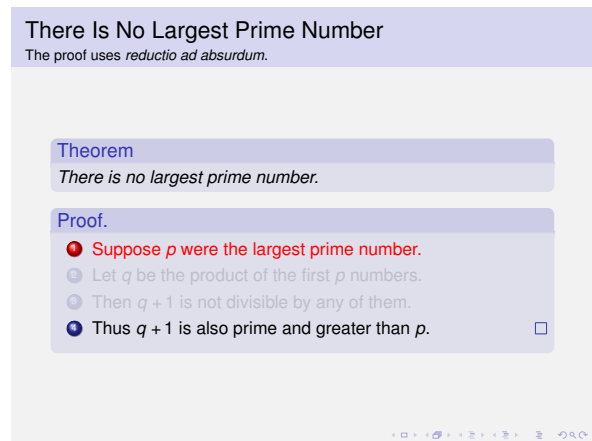
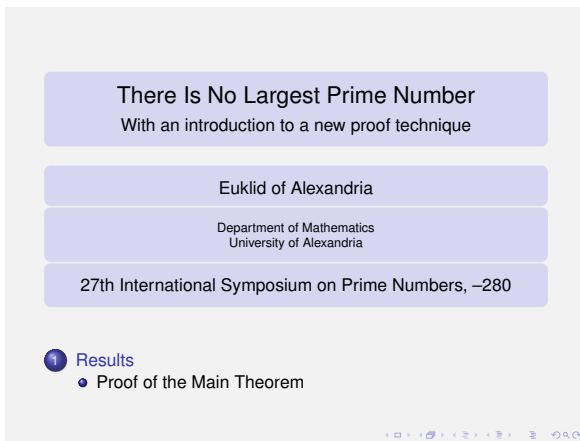


主题的名称: rectangles (矩形)。

在该主题中, `itemize` 和 `enumerate` 的条目标记是小矩形 (rectangle), 目录的条目也是以小矩形开始的。

`\useinnertheme[options]{rounded}`

举例:



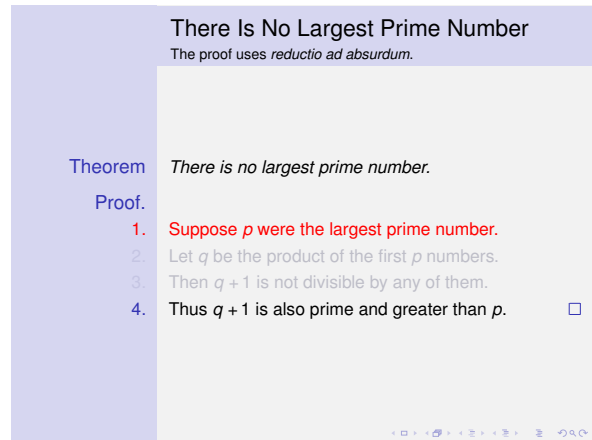
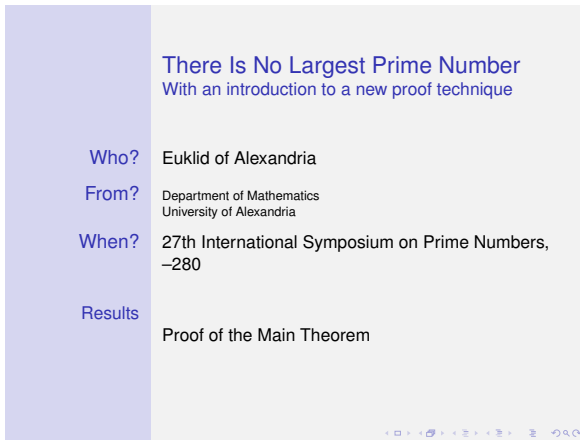
主题的名称: rounded (圆角)。

在该主题中, `itemize` 和 `enumerate` 的条目标记是小球 (balls), 目录的条目也是以小球开始的。如果为块 (blocks) 指定了背景 (background), 那么背景矩形 (background rectangles) 的拐角会被切掉。可能会给出下面的 `options`:

- `shadow` 为所有的块添加阴影。

`\useinnertheme{inmargin}`

举例:



主题的名称：inmargin（悬挂）。

该主题的设计思想是在左侧显示“结构（structuring）”信息，在右侧显示“常规（normal）”信息。为此目的，块（blocks）被重新定义了，块标题显示于左侧，块正文显示于右侧。

页边放置文本的代码是脆弱的（fragile）。我们必须“手工”调整页边空白，而且风险自担。

Itemize 的条目被重定义而显示于左侧。然而，通过改变一些间距参数（spacing parameters）只会改变位置，而用于绘制条目的代码不会改变。正因为这样，我们可以先加载其它的内部主题然后加载该主题。

该主题是一个“脏的（dirty）”内部主题，因为它会搞糟内部主题的一些东西。特别是它会改大左侧栏的宽度。然而，该主题仍可与大多数外部主题合用。

该主题使用栏（columns）时可能会出现问題，结果可能和我们预期的不一样。

16.2 外部主题

一个外部主题控制着（大概是这样）帧的全部布局。它指定了导航元素（navigational elements）指向哪里（如微目录或微帧）、导航元素的外观。通常，一外外部主题指定了如何生成下列元素：

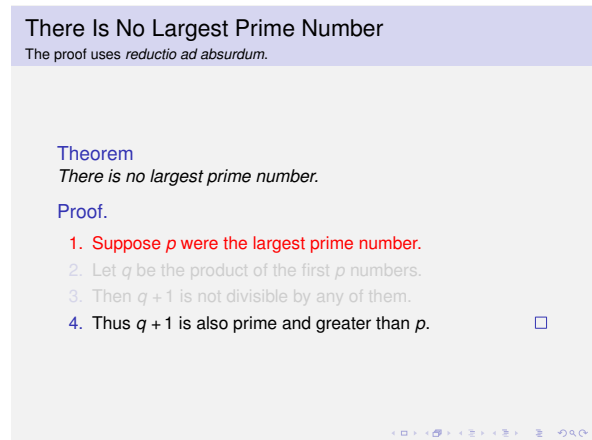
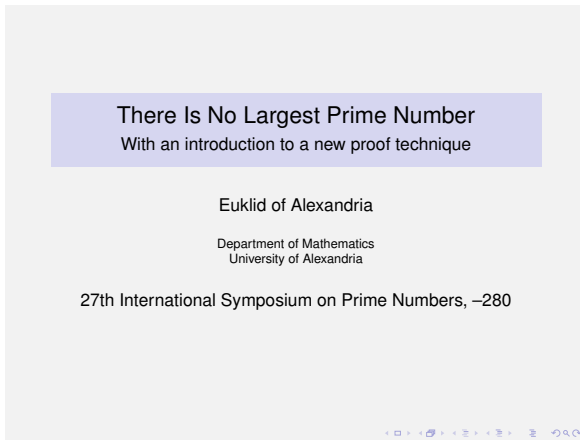
- 顶部导航区（headline）和底部导航区（footline）。
- 侧栏（Sidebars）。
- 徽标（Logo）。
- 帧标题（Frame Title）。

一个外部主题不会指定像 `itemize` 环境这样的东西是如何生成的，因为它是内部主题的事情。

下面的例子使用了颜色主题 `seahorse`。因为默认的颜色主题会使背景为空，大部分外部主题和默认的颜色主题使用时会显得结构不紧凑（unstructured）。

```
\useoutertheme{default}
```

举例：

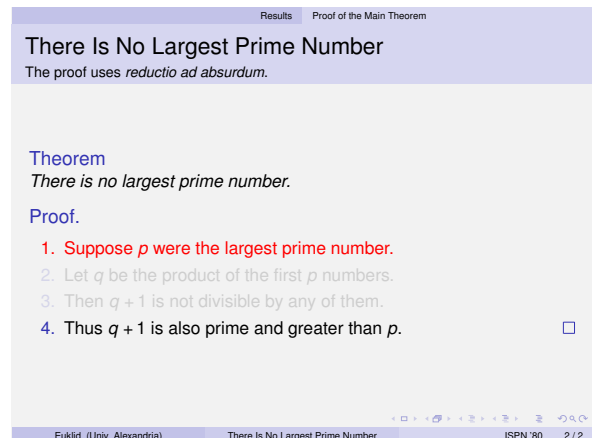
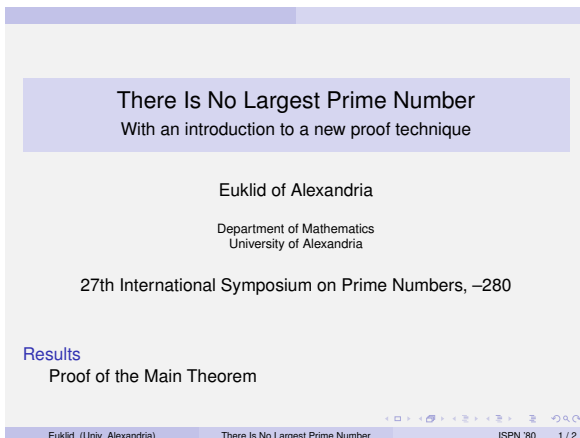


主题的名称: default (默认)。

默认的布局主题 (layout theme) 是非常稳重 (sober) 和简约的 (minimalistic), 帧标题左对齐, 不会安装顶部导航区和底部导航区, 即使是主题拥有帧标题的背景色。如果指定了颜色, 在帧标题的后面会放置一个条 (bar), 该条占据了整个页面。帧小标题 (frame subtitle) 的背景色会被忽略。

`\useoutertheme{infolines}`

举例:



主题的名称: infolines (信息行)。

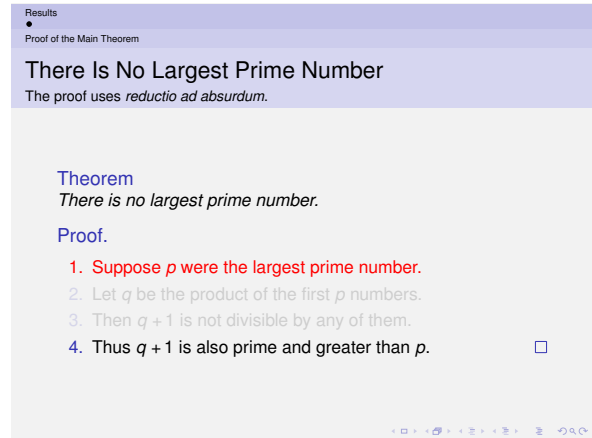
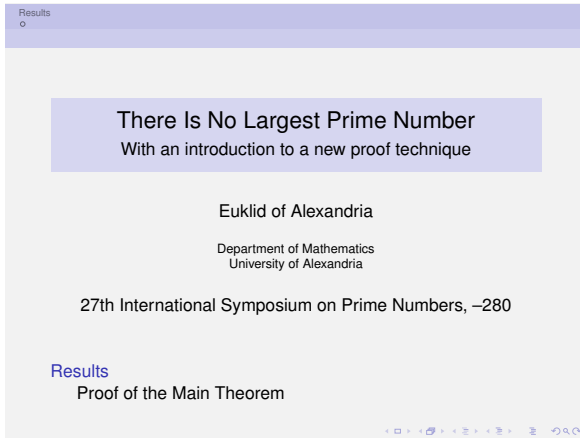
该主题会安装顶部导航区 (headline), 在此区显示当前节和当前小节。该主题也会安装底部导航区 (footline), 在此显示下面这些信息⁵⁰: 作者姓名 (author's name)、大学 (institution)、演示稿的标题 (presentation's title)、当前的日期 (date)、帧计数 (frame count), 在此也可以放置计时器 (即时钟)。该主题只使用很少的空间。

顶部导航区 (headline) 和底部导航区 (footline) 使用的颜色来自于 `palette primary`、`palette secondary`、`primary tertiary` (如何改变这种状况请参阅第 17 节)。

`\useoutertheme[options]{miniframes}`

举例:

⁵⁰可以通过修改 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\outer` 中的 `beamerouterthemeinfolines.sty` 文件来更改显示于底部导航区的信息 (包括添加一个时钟)。附件是译者修改好的一个示例文件。



主题的名称: miniframes (微帧)。

该主题会安装顶部导航区 (headline), 在此会显示一个水平导航条 (navigational bar)。水平导航条包含演示稿的每一节的条目。在每一节的条目下面, 以小圆代表节的不同的帧。帧是按小节智能排列的 (arranged subsection-wise), 也就是说, 每一小节的帧排成一行。如果给定文档类选项 `compress`⁵¹, 每一节的帧将排成单行。导航条的颜色来自 `section in head/foot`。

在导航条的下面, 以一行显示当前小节的标题。其颜色来自于 `subsection in head/foot`。

在底部, 以两行显示诸如作者姓名、大学、论文标题这样的信息。具体显示什么由给定的 $\langle options \rangle$ 决定。而颜色由诸如 `author in head/foot` 这样的 BEAMER-COLORS 决定。

如果设置了 `separation line` 的背景色, 则会在顶部导航区和底部导航区的顶部和底部, 以及导航条和小节名之间, 放置分隔线 (separation lines)。该分隔线的高度是 3pt。我们可以通过设置恰当的颜色如 `lower separation line head` 来更好地控制分隔线的颜色。

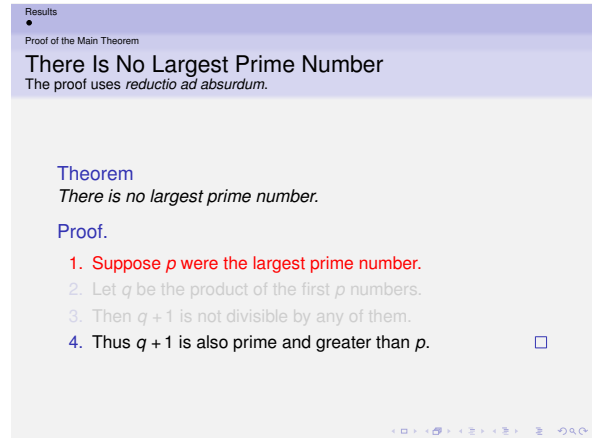
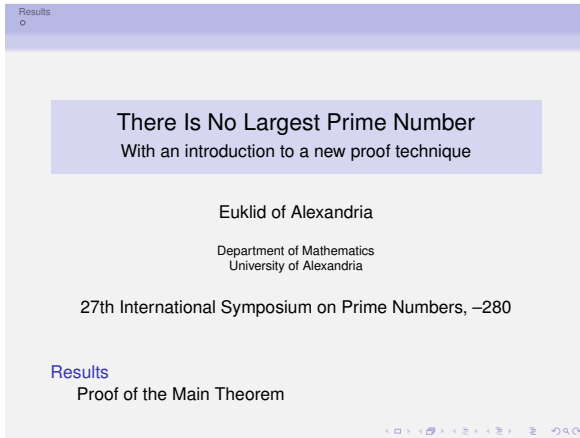
可能会给出下面的 $\langle options \rangle$:

- `footline=empty` 抑制底部导航区 (默认)。
- `footline=authorinstitute` 在底部导航区显示作者姓名和大学。
- `footline=authortitle` 在底部导航区显示作者姓名和标题。
- `footline=institutetitle` 在底部导航区显示大学和标题。
- `footline=authorinstitutetitle` 在底部导航区显示作者姓名、大学、标题。
- `subsection= $\langle true or false \rangle$` 显示或抑制在顶部导航区显示小节的行。默认情况下是显示。

`\useoutertheme [$\langle options \rangle$] {smoothbars}`

举例:

⁵¹ 一个示例语句是: `\documentclass[17pt,compress]{beamer}`



主题的名称：smoothbars（光滑条）。

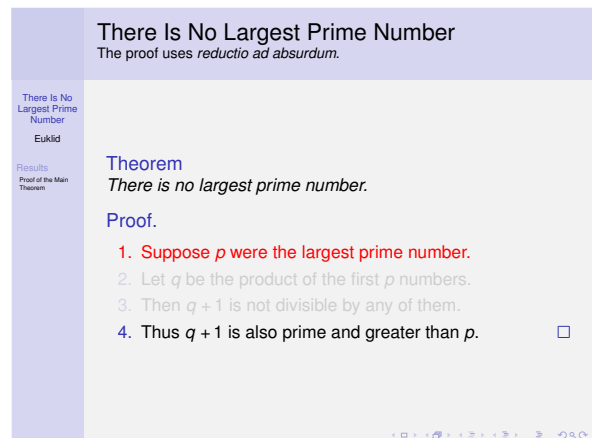
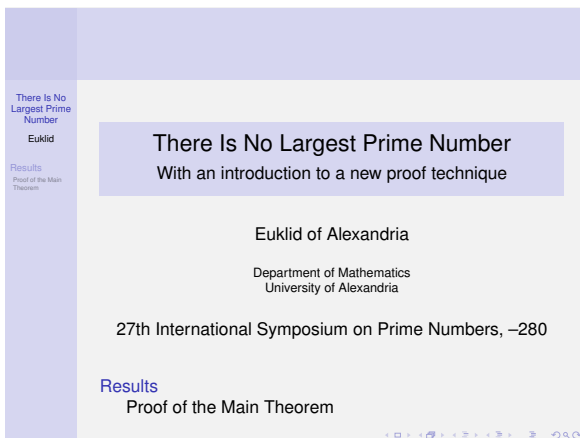
该主题的行为很像 miniframes 主题，至少相对于顶部导航区（headline）来说是这样的。所不同的是：在导航条的背景色、（可选的）小节名的横条、帧标题的背景之间安装平滑的过渡（smooth transition）。不会创建底部导航区（footline）。我们可以先加载 miniframes 主题从而获得其底部导航区（footline），然后加载 smoothbars 主题。

可能会给出下面的 $\langle options \rangle$ ：

- `subsection= $\langle true \text{ or } false \rangle$` 显示或抑制在顶部导航区显示小节的行。默认情况下是显示。

`\useoutertheme [$\langle options \rangle$] {sidebar}`

举例：



主题的名称：sidebar（侧栏）。

在该主题中，会显一个侧栏，它包含一个小的目录，该目录包含当前节、小节（高亮显示）。帧标题垂直居于中于顶部的矩形区域中，该矩形区域所占据的空间在所有的帧中是相同的。徽标（Logo）则放置于侧栏和帧标题所形成的“拐角”处。

使用 $\langle options \rangle$ 可以改变上述布局。如果将侧栏的宽度设为 $0pt$ ，则不会显示侧栏，帧标题也就不会“摇摆不定（wobble）”，因为帧标题在所有幻灯片中占据相同大小的空间。相反，如果我们将帧标题所在的矩形的高度设为 $0pt$ ，则不会使用该矩形，帧标题也就以普通的方式插入（即在每一张幻灯片中，帧标题将按其实际需要占用一定的空间）。

侧栏的背景色来自于 `sidebar`，帧标题的背景色则来自于 `frametitle`，徽标拐角 (Logo corner) 的背景色则来自于 `logo`。

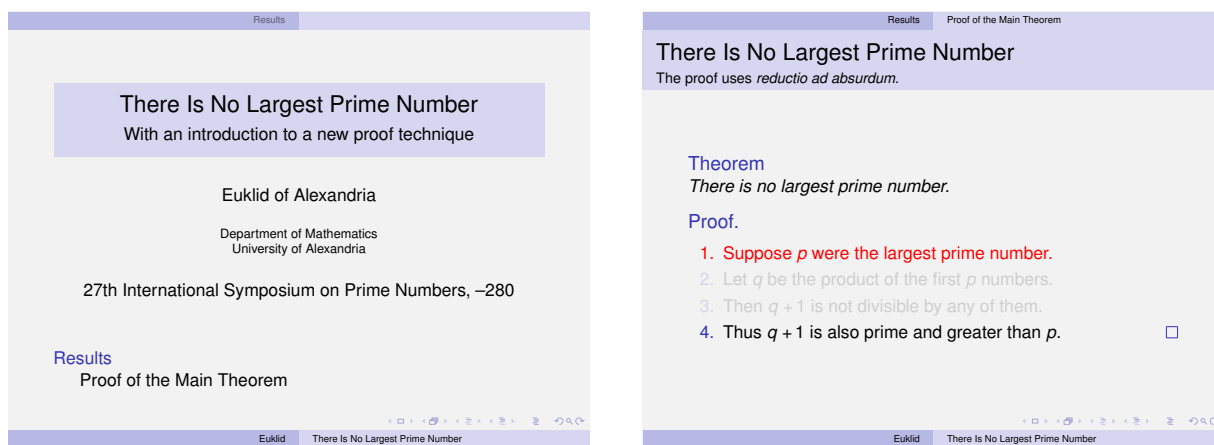
目录条目的颜色来自于 `BEAMER-color section in sidebar` 和 `section in sidebar current`，这和用于小节 (subsections) 的 `BEAMER-colors` 是一样的。如果一个目录条目在一行容不下时，它会自动“断行 (linebroken)”。

可能会给出下面的 `<options>`:

- `height=<dimension>` 指定帧标题所在矩形的高度。如果将该高度设为 `0pt`，则不会创建帧标题所在的矩形，帧标题也就以普通的方式插入（即在每一张幻灯片中，帧标题将按实际需要占用一定的空间）。该高度的默认值是帧标题字体基线高度 (base line heights) 的 2.5 倍。因此，有足够的空间用于容纳两行的帧标题和一行的副标题 (subtitle)。
- `hideothersubsections` 抑制在目录中显示除当前节的小节以外的所有小节（译者注：即在目录中只显示当前节的小节）。如果有很多小节，该选项就大有用处。
- `hideallsubsections` 抑制在目录中显示所有小节。
- 在左侧放置侧栏。在由左向右阅读的文化中，首先阅读的是左边。注意：因为左侧的目录常常不是帧的重要组成部分，所以我们无法要求别人首先阅读左侧的目录。虽然如此，该选项是默认的选项。
- `right` 在右侧放置侧栏。
- `width=<dimension>` 指定侧栏的宽度。如果将该宽度设为 `0pt`，则不会显示侧栏。该宽度的默认值是帧标题字体基线高度 (base line heights) 的 2.5 倍。

`\useoutertheme{split}`

举例：



主题的名称：split (裂开)。

该主题会安装顶部导航区，在其左侧显示节 (sections)，在其右侧显示当前节的小节 (subsections)，如果给定了文档类选项 `compress`⁵²，节和小节会放置于一行中；通常每一节或小节占据一行。

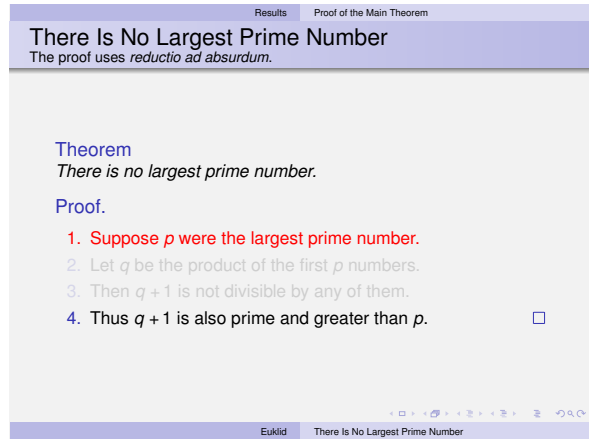
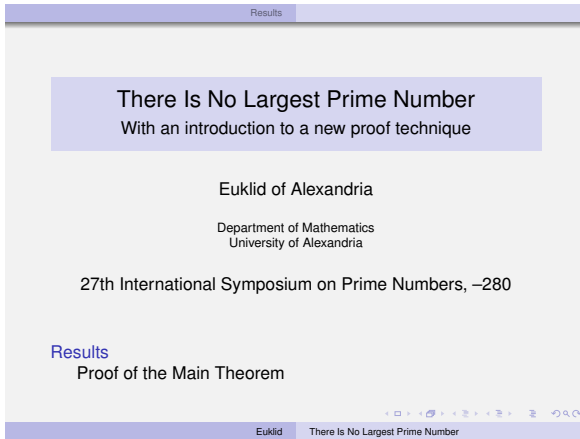
在底部导航区 (footline) 的左侧显示作者 (author)，在右侧显示演讲的标题 (talk's title)。

颜色来自于 `palette primary` 和 `palette quarternary`。

⁵²一个示例语句是：`\documentclass[17pt,compress]{beamer}`

`\useoutertheme{shadow}`

举例：

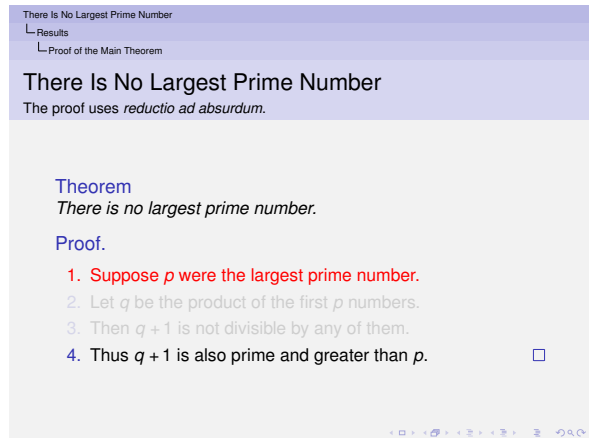
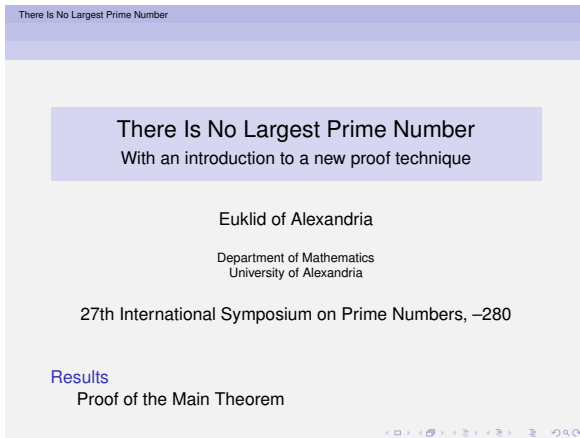


主题的名称：shadow（阴影）。

该主题扩展了 `split` 主题，具体的情况是这样：在帧标题的后面放置水平阴影，在顶部导航区的底部添加小的“阴影”。

`\useoutertheme[<options>]{tree}`

举例：



主题的名称：tree（树形）。

该主题的顶部导航区占据三行，分别显示演讲的标题、当前节、当前小节。颜色来自 `title in head/footer`、`section in head/footer`、`subsection in head/footer`。

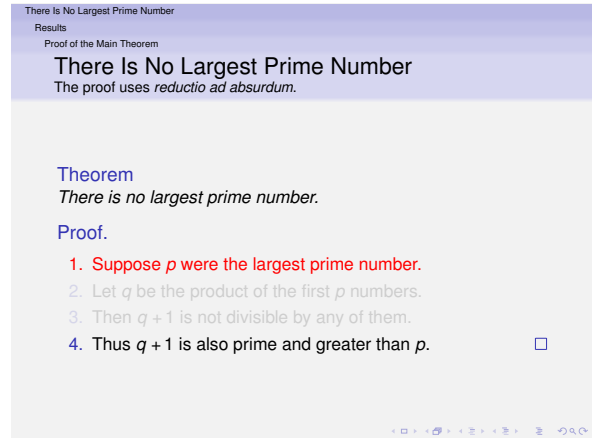
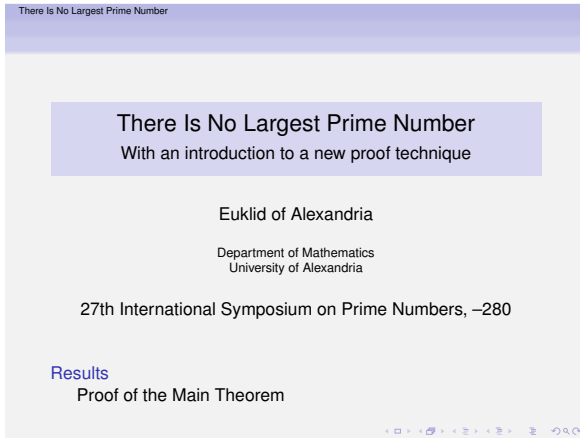
而且，如果设置了 `separation line` 的背景，则在上述三行的上下两侧放置高为 3pt 的分隔线。通过设置 `upper separation line head` 和 `lower separation line head`，可能获得多种多样的分隔线。

可能会给出下面的 `<options>`：

- `hooks` 在节和小节的前面生成小“牵引钩（hooks）”。这些小牵引钩呈递增的树状（tree-like）。

`\useoutertheme{smoothtree}`

举例：



主题的名称: smoothtree (光滑树形)。

该主题和 tree 主题相似。主要的不同是背景色改变是平滑的。

16.3 改变演示稿不同元素的模板

本节阐述 BEAMER 的模板是如何工作的。

16.3.1 Beamer 的模板管理概述

如果想改变某个或某些元素的外观，我们无需创建新的内部或外部主题，只需修改相应的模板即可。

一个模板指定了演示稿的元素是如何排版的，例如，`frametitle` 模板指定了在哪里放置帧标题，使用什么字体等等。

顾名思义，当排版一个帧标题时我们可以通过编写正确的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 代码手工指定一个模板，所要做的只是使用命令 `\insertframetitle`。

举例：假定想让帧标题显示为红色、居中、粗体，可以手工输入下面的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 代码：

```
\begin{frame}
  \begin{centering}
    \color{red}
    \textbf{The Title of This Frame.}
  \par
  \end{centering}

  Blah, blah.
\end{frame}
```

要以上述方式在全部幻灯片中显示帧标题，可以用下面的简便方法改变帧标题：

```
\setbeamertemplate{frametitle}
{
  \begin{centering}
    \color{red}
    \textbf{\insertframetitle}
  \par
}
```

```

\end{centering}
}

```

我们可以用下面的代码来获得期望的效果:

```

\begin{frame}
\frametitle{The Title of This Frame.}

Blah, blah.
\end{frame}

```

当生成帧时, BEAMER 用帧标题模板 (frame title template) 的代码排版帧标题, 并用当前帧标题替换每一个 `\insertframetitle`。

可以将该例子深化一步。如果无需“装备 (hardwire)”帧标题的颜色, 但可以独立指定该颜色, 模板的代码会变得更有趣。这样, 颜色主题就可以改变该颜色。因为这对于大多数模板来说是个普遍的问题, 当使用 `frametitle` 模板时, BEAMER 会自动建立 `BEAMER-color frametitle`。因此, 如果在某个地方将 `BEAMER-color frametitle` 设为红色 (red), 我们就可以移除 `\color{red}` 命令。

```

\setbeamercolor{frametitle}{fg=red}
\setbeamertemplate{frametitle}
{
\begin{centering}
\textbf{\insertframetitle}
\par
\end{centering}
}

```

接下来, 我们也可以将字体“主题化 (themable)”。像颜色一样, 在排版 `frametitle` 模板之前, 会安装 `BEAMER-font frametitle`。这样, 我们可以重写代码如下:

```

\setbeamercolor{frametitle}{fg=red}
\setbeamerfont{frametitle}{series=\bfseries}
\setbeamertemplate{frametitle}
{
\begin{centering}
\insertframetitle\par
\end{centering}
}

```

用户 (users)、主题 (themes)、任何人 (whoever) 均可以不惹 (mess with) 代码的情况下轻而易举地改变帧标题的颜色或字体。

[ARTICLE](#) 在论文 (article) 模式中, 模板机理的大部分被关闭而失效。然而, 有些模板还是有用的。如果情况是这样, 就要特别指明。

在设置一个模板时, 下面的这些提示或许有帮助:

- 通常，我们可能会从一个已有的模板复制代码。代码常处理一些我们没想到的事情。默认的内部和外部主题可能是有益的起点 (start point)。而且，`beamerbaseauxtemplates.sty` 文件⁵³已包含了有趣的“辅助 (auxiliary)”模板。
- 当从其它模板复制代码并在我们的文档（不是其它 style 文件）的导言区插入该代码时，我们也许不得不“开启 (switch on)” at-字符 (@)，请在 `\setbeamertheme` 命令之前及命令 `\makeatother` 之后添加 `\makeatletter` 命令。
- 和帧组成部分（顶部导航区、侧栏等）相关的大部分模板只能在导言区改变。其它模板可以在整个文档中改变。
- 会自动算出顶部导航区 (headline) 和底部导航区 (footline) 模板的高度。是这样实现的：先排版模板并“看一看”它的高度。重新计算是在开始文档时，加载全部宏包之后甚至这些宏包执行 `\AtBeginDocument` 初始化之后。
- 在模板中使用盒子 (boxes) 常会引发争论 (hassle)。我们可以查阅 T_EX 书籍以获取与“Making Boxes”相关的细节。如果顶部导航区 (headline) 比较单一 (simple)，我们可以试将任何东西放入 `pgfpicture` 环境中，该环境使放置工作变得更容易。

16.3.2 使用 Beamer 的模板

作为一名 BEAMER 文档类的用户，我们无需自己直接“使用 (use)”或“调用 (invoke)”模板。例如，在排版帧过程中的任何地方由 BEAMER 自动调用帧标题模板 (frame title template)。其它大部分模板也是这样。然而，无论如何，如果我们想自己调用模板，则可以使用下面的命令：

```
\usebeamertemplate***{<element name>}
```

即：

```
\usebeamertemplate***{<元素名>}
```

如果没有星号，会在当前位置直接插入 `<element name>`。该文本必需用 `\setbeamertheme` 命令事先指定。在没有调用该命令之前不会插入任何文本。

举例：

```
\setbeamertheme{my template}{correct}
```

...

```
Your answer is \usebeamertemplate{my template}.
```

如果添加一个星号，将发生三件事：第一，模板会放入到一个 T_EX 组 (T_EX-group) 内，因此，会限制在模板内使用的命令的大部分副作用 (side effects)；第二，在该组内会使用名为 `<element name>` 的 BEAMER-color，还会选定前景色；第三，会使用 BEAMER-font `<element name>`。这个带一个星号的版本通常最为好用。

如果添加两个星号，发生的事情和添加了一个星号的差不多一样。不过除此以外，颜色会由命令 `\setbeamercolor*` 使用。(按此颜色指前述名为 `<element name>` 的 BEAMER-color)。如果没有根据 BEAMER-color `<element name>` 指定前景色或背景色，则会将颜色重置为普通文本的颜色。这样，在这个带两个星号的版本中，使用该模板时，模板使用的颜色就可以独立于当前使用的颜色。

⁵³在装有 C_TE_X 套装的 Windows 7 平台中，该文件位于如 `D:\CTEX\MiKTeX\tex\latex\beamer\base` 中

添加三个星号会将一个星号添加到 `\setbeamerfont*` 命令。这将重置模板使用的字体为普通文本使用的字体，除非 BEAMER-font $\langle element name \rangle$ 指定的字体不同。这个带三个星号的版本是可用的“最受保护 (most protected)”的版本。

```
\ifbeamertemplateempty{\(beamer template name)\}{\langleexecuted if empty\rangle}{\langleexecuted otherwise\rangle}
```

即：

```
\ifbeamertemplateempty{\(beamer 模板名)\}{\langleexecuted if empty\rangle}{\langleexecuted otherwise\rangle}
```

该命令检测是否定义了一个模板和设置成了非空文本 (non-empty text)。如果文本为空或根本就没有定义模板，则执行 $\langle executed if empty \rangle$ 。否则执行 $\langle executed otherwise \rangle$ 。

```
\expandbeamertemplate{\(beamer template name)}
```

即：

```
\expandbeamertemplate{\(beamer 模板名)}
```

该命令的功能和 `\usebeamertemplate{\(beamer template name)}` 相同。不同的地方是该命令直接执行扩展功能 (performs a direct expansion) 而不扫描星号 (star)。在内部这是很重要的，例如，一个 `\edef`。如果不知道 `\def` 和 `\edef` 之间的不同，我们可能不需要该命令。

16.3.3 设置 Beamer 的模板

要设置一个 BEAMER-模板，可以使用下面的命令：

```
\setbeamertemplate{\(element name)}[\(predefined option)]\langleargs\rangle
```

即：

```
\setbeamertemplate{\(元素名)}[\(预定义选项)](参数)
```

最简单的情形，如果没有给定 $\langle predefined option \rangle$ ， $\langle args \rangle$ 必须是一个简单的参数，模板 $\langle element name \rangle$ 的文本被设置成该文本。通过 `\usebeamertemplate` 命令请求该模板之后就会使用该文本。

举例：

```
\setbeamertemplate{answer}{correct}
```

...

```
Your answer is \usebeamertemplate*{answer}.
```

如果指定了 $\langle predefined option \rangle$ ，该命令的行为就有点不同了，这是因为别人已经用命令 `\defbeamertemplate` 预定义了一个模板。通过指定预定义模板的名字为可选的参数 $\langle predefined option \rangle$ ，我们就可以将模板 $\langle element name \rangle$ 设置成该模板。

举例：将参考书目的条目 (bibliography items) 变成小书籍图标 (book icons)。

该命令使在以后调用 `\usebeamertemplate{bibliography item}` 时，如要插入一书籍 (book) 则可插入预定的代码 (predefined code)。

一些预定义的模板选项自带有参数，这种情况下，参数是以 $\langle args \rangle$ 的形式给出。

举例：模板 `background` 的 $\langle predefined option \rangle$ `grid` 带有可选参数：

```
\setbeamertemplate{background}[grid][step=1cm]
```

该例子中的第二个参数位于方括号中，它是可选参数。

在元素的说明 (descriptions of elements) 中，如果存在可能的 *predefined option*，该说明就会显示如何使用 *predefined option* 及其参数，但 `\setbeamertemplate{xxxx}` 则被忽略。因此，上述例子在背景元素 (background element) 的说明中被记录成以下模样：

- `[grid]` [*step options*] 使亮的网格线 (light grid) 为 ...。

```
\addtoamertemplate{<element name>}{<pre-text>}{<post-text>}
```

即：

```
\addtoamertemplate{<元素>}{<前置文本>}{<后置文本>}
```

该命令在作为模板 (*element name*) 的文本之前添加 *pre-text*，在其之后添加 *post-text*。这样我们就可以对现有的模板作有限的修改。

举例：下面的命令有相同的效果：

```
\setbeamertemplate{my template}{Hello world!}
```

```
\setbeamertemplate{my template}{world}
```

```
\addtoamertemplate{my template}{Hello }{!}
```

如果安装了新模板，则会删除其附加的东西 (additions)。另一方面，我们可以重复使用该命令以添加复杂的东西 (multiple things)。

```
\defbeamertemplate<<mode specification>>*{<element name>}{<predefined option>}  
  [<argument number>] [<default optional argument>]{<predefined text>}  
  [<action>]{<action command>}
```

即：

```
\defbeamertemplate<<模式规则>>*{<元素名>}{<预定义选项>}  
  [<参数号>] [<默认可选参数>]{<预定义文本>}  
  [<功能>]{<功能命令>}
```

该命令为模板 (*element name*) 安装一个 *predefined option*。一旦使用了该命令，用户就可以通过 `\setbeamertemplate` 命令访问预定义的模板。

举例：`\defbeamertemplate{itemize item}{double arrow}{ \rightarrow }`

调用上述命令后，下面的两条命令具有相同的效果：

```
\setbeamertemplate{itemize item}{ $\rightarrow$ }
```

```
\setbeamertemplate{itemize item}[double arrow]
```

有时，当安装一个预定义的模板时需要一个参数。例如，我们要定义一个预定义模板让它生成方块 (square) 作为 `itemize` 的条目标记，并可设置该方块的尺寸。我们可以指定 *predefined option* 的 *argument number*，方法和 `\newcommand` 命令相同：

```
\defbeamertemplate{itemize item}{square}[1]{\hrule width #1 height #1}
```

```
% The following have the same effect:
\setbeamertemplate{itemize item}[square][3pt]
\setbeamertemplate{itemize item}{\hrule width 3pt height 3pt}
和 \newcommand 命令相同，我们也可以指定 default optional argument:
\defbeamertemplate{itemize item}[square][1][1ex]{\hrule width #1 height #1}
```

```
% The following have the same effect:
\setbeamertemplate{itemize item}[square][3pt]
\setbeamertemplate{itemize item}{\hrule width 3pt height 3pt}
```

```
% So do the following:
\setbeamertemplate{itemize item}[square]
\setbeamertemplate{itemize item}{\hrule width 1ex height 1ex}
```

该命令的一颗星点缀的版本会安装预定义模板的选项，然后会为该选项立即调用 `\setbeamertemplate`。这对于默认模板来说是很有用的。如果存在多个必需的参数，它们会被设置成 `\relax`。

在某些情况下，如果选择了一个预定义模板选项，我们不仅希望安装模板文本，而且希望具有额外的“功能 (actions)”。例如，必须定义一个阴影 (shading) 便于以后每一次使用阴影时无需重新定义它。要实现该项“功能”，我们可以在关键词 `[action]` 后面使用可选参数 *action*。这样，在使用 `\defbeamertemplate` 后，我们就可以添加文本 `[action]` 和任何命令（如命令 `a`），当通过 `\setbeamertemplate` 命令选择 *predefined option* 时命令 `a` 会被执行。

举例：

```
\defbeamertemplate{background canvas}{my shading}[2]
{
  \pgfuseshading{myshading}% simple enough
}
[action]
{
  \pgfdeclareverticalshading{myshading}{\the\paperwidth}
  {color(0cm)=(#1); color(\the\paperheight)=(#2)}
}
...

\setbeamertemplate{background canvas}{myshading}{red!10}{blue!10}
% Defines the shading myshading right here. Subsequent calls to
% \usebeamertemplate{background canvas} will yield
% ``\pgfuseshading{myshading}``.
```

ARTICLE 通常，该命令在论文 (article) 模式中无效。然而，如果给定了 *mode specification*，该命令就可以用在指定的模式中。这样，该命令的行为和命令 `\` 类似。如果没有指定其它的模式规则，命令 `\` 就会获取内含的模式规则 `<presentation>`。

举例：`\defbeamertemplate{my template}{default}{something}` 在论文 (article) 模式中无效。

举例：`\defbeamertemplate<article>{my template}{default}{something}` 在演示稿 (presentation) 模式中无效，但在论文 (article) 模式中有效。

举例：`\defbeamertemplate<all>{my template}{default}{something}` 适应于所有的模式。

通过不同的名称可以使用相同的模板选项（template option）是很有用的。正因为这样，我们才可以用下面的命令创建别名（aliases）：

```
\defbeamertemplatealias{<element name>}{<new predefined option name>}{<existing predefined option name>}
```

即：

```
\defbeamertemplatealias{<元素名>}{<新的预定义选项名>}{<已有的预定义选项名>}
```

使两个预定义选项具有相同的效果。

不象颜色和字体（colors and fonts），模板之间是没有继承关系的。这归因于一个元素的模板对另一元素的模板作用很少。然而，某些特定的元素“行为类似（behave similarly）”，一个元素如 `\setbeamertemplate` 通过继承适应于所有的模板集（set of templates）。例如，和 `itemize subsubitem`、`enumerate item` 相比，虽然 `itemize item` 的效果可能略有不同，仍可用 `\setbeamertemplate{items}[circle]` 让所有的条目（items）使用 `circle` 选项。

BEAMER-模板机理（BEAMER-template mechanism）通过父模板（parent templates）实现简单的继承。在元素说明（element descriptions）中，通过检测括弧内的标记（mark in parentheses）来标明（indicate）父模板（parent templates）。

```
\defbeamertemplateparent{<parent template name>}[<predefined option name>]{<child template list>}[<argument number>][<default optional argument>]{<arguments for children>}
```

即：

```
\defbeamertemplateparent{<父模板名>}[<预定义选项名>]{<子模板列表>}[<参数数量>][<默认可选参数>]{<children 的参数>}
```

该命令的作用是每当调用 `\setbeamertemplate{<parent template name>}{<args>}` 时，就会为 `<child template list>` 中的每一个 `<child template name>` 调用 `\setbeamertemplate{<child template name>}{<args>}`。如果调用带有预定义选项名（predefined option name）（未必和 `<predefined option name>` 相同）选项的 `\setbeamertemplate` 命令，`<arguments for children>` 就会开始起作用。如果调用带有某些预定义选项名（some predefined option name）选项的 `\setbeamertemplate` 命令，则会调用带有 `<arguments for children>` 的 children。请看下面的两个例子：

举例：下面是典型的简单的惯用法：

```
\defbeamertemplateparent{itemize items}{itemize item,itemize subitem,itemize subsubitem}{}
\setbeamertemplate{itemize items}[circle]
```

% 第一条命令和其它三条命令具有相同的效果：

```
\setbeamertemplate{itemize items}[circle]
```

```
\setbeamertemplate{itemize item}[circle] % 事实上会添加“empty”参数。
```

```
\setbeamertemplate{itemize subitem}[circle]
```

```
\setbeamertemplate{itemize subsubitem}[circle]
```

举例：下面的例子，会传递一个参数给其 children：

```
\defbeamertemplateparent{sections/subsections in toc shaded}
{section in toc shaded,subsection in toc shaded}[1][20]
{[#1]}

% 第一条命令和其它两条命令具有相同的效果：
\setbeamertemplate{sections/subsection in toc shaded}[default][35]

\setbeamertemplate{section in toc shaded}[default][35]
\setbeamertemplate{subsection in toc shaded}[default][35]

% 再者：
\setbeamertemplate{sections/subsection in toc shaded}[default]

\setbeamertemplate{section in toc shaded}[default][20]
\setbeamertemplate{subsection in toc shaded}[default][20]
```

具体点，会发下面的事情：当 `\setbeamertemplate` 遇到父模板时，BEAMER 会首先检测是否跟有预定义选项（predefined option）。如果没有，会读入单参数（single argument），并为该模板的所有 children 调用 `\setbeamertemplate`。如果后面跟有预定义模板选项集（predefined template option set），BEAMER 会求值 *⟨argument for children⟩*。包含的参数像 #1 或 #2 这样。这些参数（parameters）用参数（arguments）（这些参数后跟为父模板对 `\setbeamertemplate` 的调用）填充。参数的数量必须和 *⟨argument number⟩* 相同。可以用普通的方式（in the usual way）指定一个可选的参数（optional argument）。一旦 *⟨arguments for the children⟩* 计算出来，就会为预定义模板的所有 children 调用 `\setbeamertemplate` 及计算出来的参数。我们可能想知道当特定的预定义选项带有特定数量的参数时会发生什么事情，而其它的预定义选项却带有不同数量的参数。既然是这样，上述的机理不能区分预定义选项和 *⟨arguments for children⟩* 中包含哪一个或多个参数。正因如此，当调用 `\defbeamertemplateparent` 时我们可给定预定义参数 *⟨predefined option name⟩*。如果指定了该可选参数，则模板的父身份（parenthood）只适应于该特定的 *⟨predefined option name⟩*。这样，如果为该 *⟨predefined option name⟩* 调用 `\setbeamertemplate`，则会使用 *⟨argument for children⟩*。对于其它预定义选项名（predefined option names）则会使用不同的定义（definition）。我们可以想象成：删除可选的 *⟨predefined option name⟩* 意味着“该 *⟨argument for children⟩* 适应于所有预定义选项名，这些预定义选项名没有特别不同的定义”。

17 颜色

BEAMER 的颜色管理 (color management) 允许我们指定每一元素 (如目录中的节、侧栏中微目录的小节) 的颜色。因颜色系统的功能太强大, 日常用得不多。为简化颜色系统的使用, 我们最好使用预定义的颜色主题 (predefined color theme), 因为它们为我们考虑了所有的事情。

下面将首先讲述颜色主题 (color themes)。剩下的章节讲述颜色管理的内部机制。如果我们准备创作自己的颜色主题, 或者对预定义主题很感兴趣而且非要将数学文本 (mathematical text) 排版成可爱的粉色, 那么我们就必须阅读这些章节。

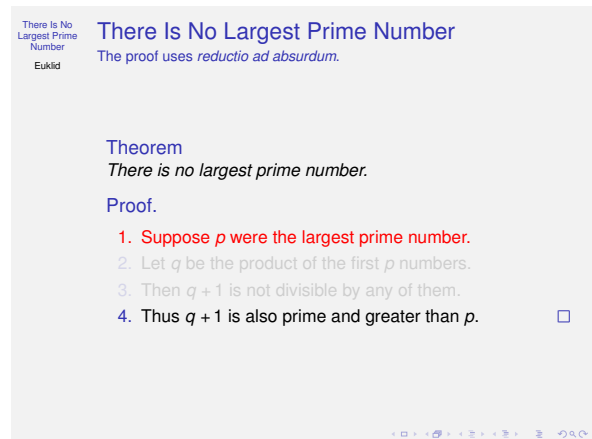
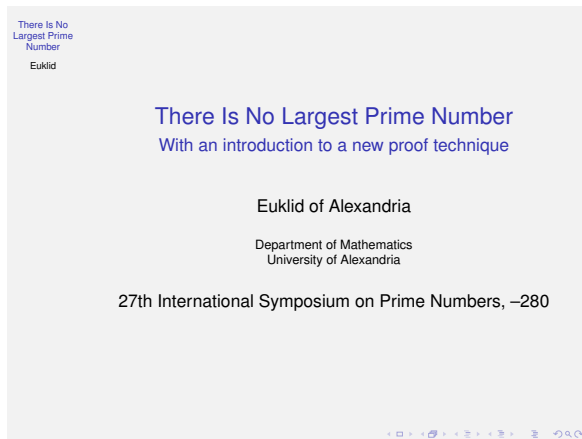
17.1 颜色主题

为了显示不同颜色主题在侧栏中的效果, 下面的例子同时使用了颜色主题和外部主题 sidebar。

17.1.1 默认和专用的颜色主题

`\usecolortheme{default}`

举例:



主题的名称: default (默认)。

默认 (default) 的颜色主题非常沉稳。它只安装了少量特定的颜色和更少的背景。默认的颜色主题在不同的 BEAMER-colors 之间建立了默认的父本关系 (parent relations)。

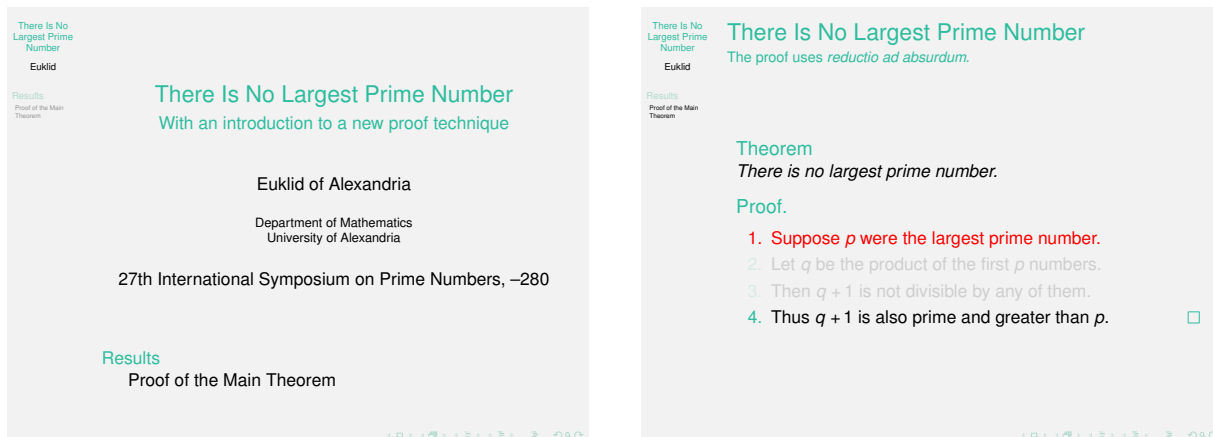
默认 (default) 的颜色主题的主要颜色设置如下:

- normal text (普通文本) 是白底黑字 (black on white)。
- alerted text (提醒文本) 是红色的。
- example text (举例文本) 是青绿色 (green 带 50% black)。
- structure (结构) 设置成中灰蓝色 (MidnightBlue) 的亮版本 (light version) (更精确地讲就是: 20% red, 20%green 和 70% blue)。

在严肃的 (no-nonsense) 演示稿中使用该主题。默认情况下自动加载该主题, 在加载其它颜色主题后无需“重加载 (reload)”该颜色主题。

`\usecolortheme[options]{structure}`

举例:



主题的名称: `structure` (结构)。

上述例子是用 `\usecolortheme[named=SeaGreen]{structure}`⁵⁴创建的。

该主题提供了便利的方法改变结构性元素 (structural elements) 的颜色。更精确地讲, 它仅改变 `BEAMERCOLOR structure` 的前景色。也可以直接调用 `\setbeamercolor` 以达到同样的效果, 但该主题使同样的事情变得容易。

该主题材提供了多个 *options*, 用于指定结构性元素的颜色:

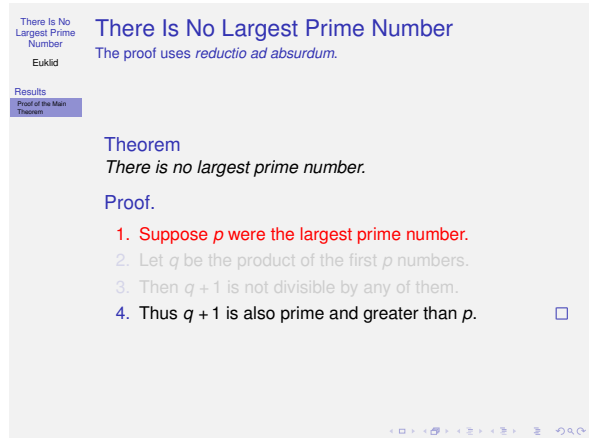
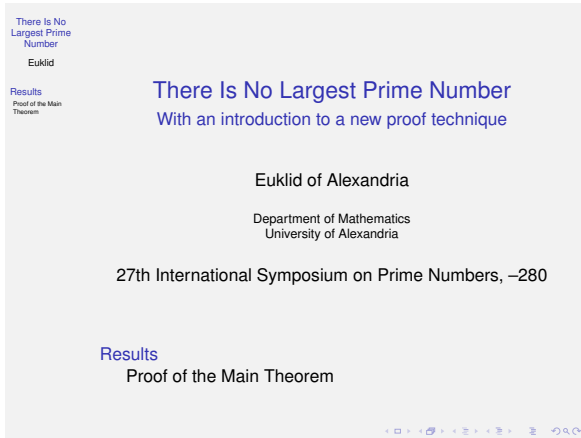
- `rgb={rgb tuple}` 设定 `structure` 前景 (foreground) 为 `rgb` 元色。这里的数字为 0 至 1 之间的小数, 例如, `rgb={0.5, 0, 0}` 输出暗红。
- `RGB={rgb tuple}` 所作所与 `rgb` 相同, 所不同的是数字为 0 至 255 之间的整数, 例如, `RGB={128,0,0}` 输出暗红。
- `cmyk={cmyk tuple}` 设定 `structure` 前景 (foreground) 为 `CMYK` 元色。这里的数字为 0 至 1 之间的小数, 例如, `cmyk={0,1,1,0.5}` 输出暗红。
- `cmY={cmY tuple}` 和 `cmyk` 类似, 所不同的是不指定黑色组份 (black component)。
- `hsb={hsb tuple}` 设定 `structure` 前景 (foreground) 为 `hsb` 元色。这里的数字为 0 至 1 之间的小数, 例如, `hsb={0,1,.5}` 输出暗红。
- `named={color name}` 设定 `structure` 前景 (foreground) 为已命名的颜色 (named color)。该颜色必须是以前用 `\DefineNamedColor` 命令定义好地。安装文档类选项 `xcolor=dvipsnames` 或 `xcolor=svgnames` 后, 将会 (分别) 安装一个长长的标准的 `dvips` 或 `SVG` 颜色名列表。请参阅 `dvipsnam.def`⁵⁵。

`\usecolortheme{sidebartab}`

举例:

⁵⁴SeaGreen: 海洋绿、海绿色。

⁵⁵在装有 CTEX 套装的 Windows 7 平台中, 该文件位于如 `D:\CTEX\MiKTeX\tex\latex\graphics` 中。 [点击打开这个文件](#)



主题的名称：sidebartab（侧栏目录）。

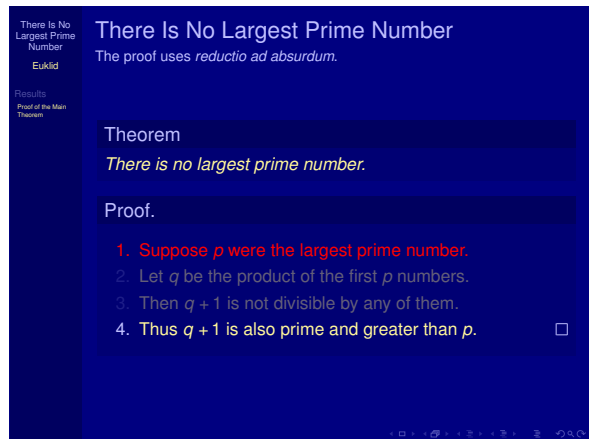
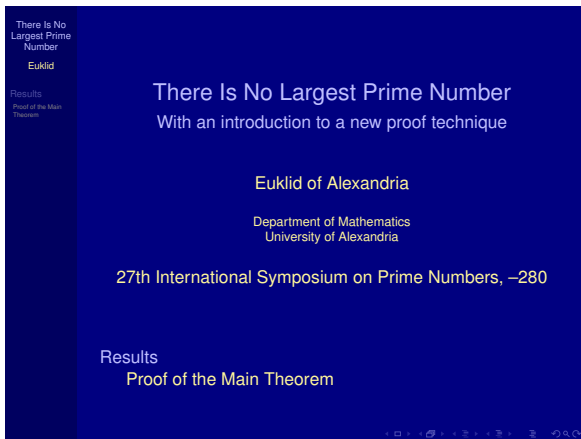
该主题会改变侧栏中目录的当前条目的颜色，通过显示该条目后的不同的背景以使该条目高亮显示。

17.1.2 完整的颜色主题

一个“完整”的颜色主题完整地指定了帧的全部组成部分的所有颜色。它安装指定的颜色（specific colors）而不是从 `structure` BEAMER-color 获取（derive）颜色。完整的颜色主题取名于会飞的动物（flying animals）。

`\usecolortheme{albatross}`

举例：



主题的名称：albatross（信天翁）。

该颜色主题是一个“暗的（dark）”或“反转的（inverted）”的主题，它在蓝色的背景上使用黄色作为主色（main colors）。该主题也为块（blocks）安装略暗的背景色，这样略暗的背景对于使用了阴影的演示主题（presentation themes）来说是必需的，但（Till 教授认为）对于其它演示主题来说可能不受欢迎（undesirable）。该主题和 lily 颜色主题共同使用时，块的背景可能被移除。

我们使用这样一个暗背景亮文字（light-on-dark）的主题时，必需知道其缺点：

- 如果演讲的场所是“昏暗的（darkened）”，使用这样的主题将使观众记录或阅读笔记（take or read notes）变得困难。
- 因为场所昏暗，瞳孔就会放大，这样眼睛就难于聚焦。因此文本就难于阅读。

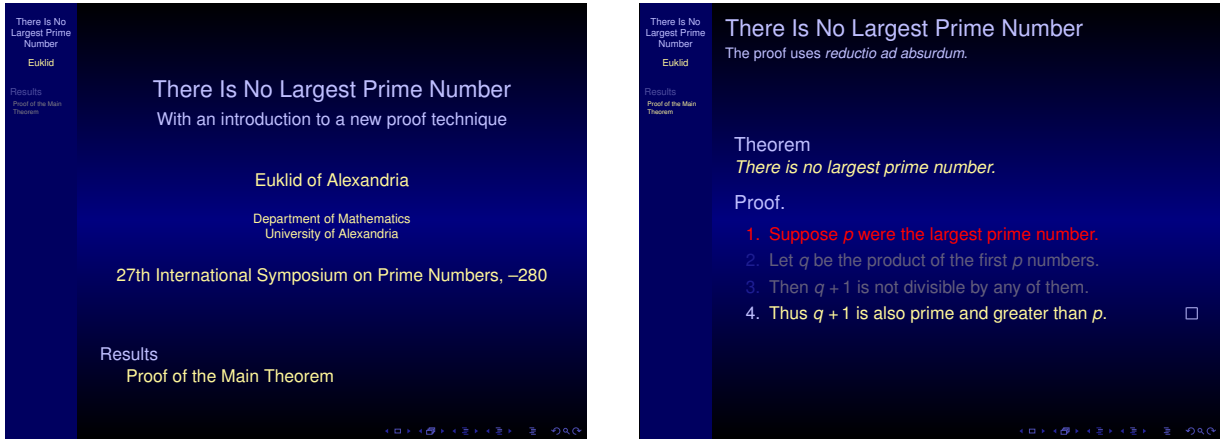
- 难于打印这样的幻灯片。

另一方面，暗背景亮文字的演示稿比普通的亮背景黑文字即白底黑字（black-on-white）的演示稿常常显得更“漂亮（stylish）”。

可能会给出下面的 *(options)*:

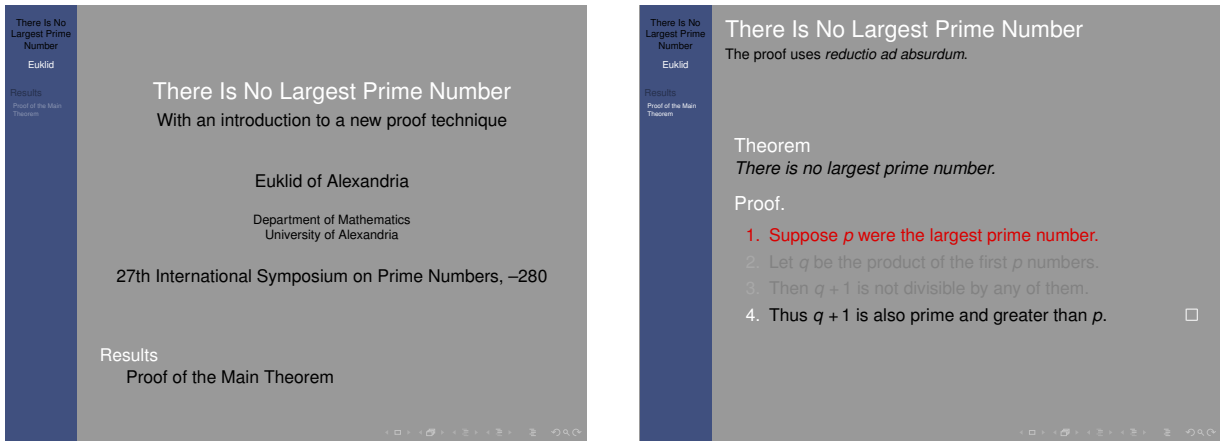
- **overlystylish** 安装背景画布（Background canvas），Till 教授认为这是很漂亮的。但是并非将其意愿强加于人。使用该选项时，如同时使用 `lily` 颜色主题则效果更佳。

举例：同时使用 `overlystylish` 选项和 `lily` 颜色主题：



`\usecolortheme{beetle}`

举例：



主题的名称：beetle（甲虫）。

“该主题的主题思想（theme behind this theme）”的要点是在灰色的背景上使用黑白文字。白色文字用于指明重点（emphasis），黑色文字用于普通文本（normal text）。像顶部导航区（headline）和底部导航区（footline）这样的“外部材料（outer stuff）”使用淡蓝色⁵⁶。要改变这些颜色，可以通过更改

⁵⁶淡蓝色：RGB=64,80,127。

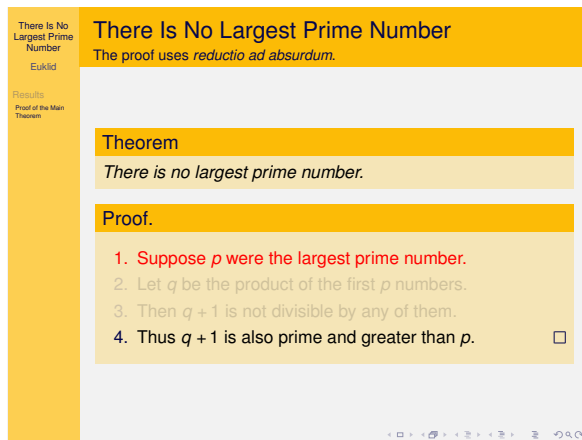
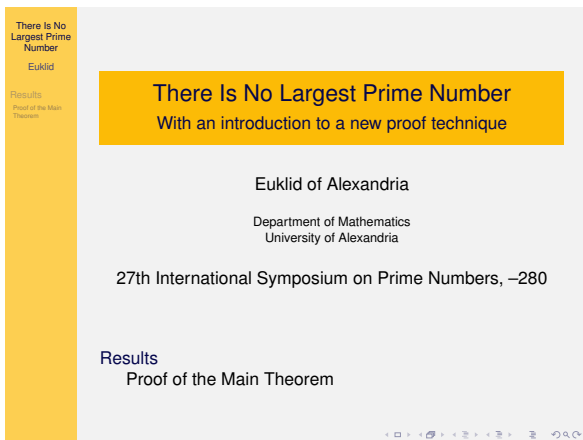
`palette primary` 的背景来实现⁵⁷。

小心使用该主题，因为白/灰和黑/灰的对比（`contrasts`）不如其它主题强烈。确保实际的演示稿具有足够的对比。


可以通过更改 `normal text` 的背景来改变“浅灰（grayish）”的背景⁵⁸。

```
\usecolortheme{crane}
```

举例：

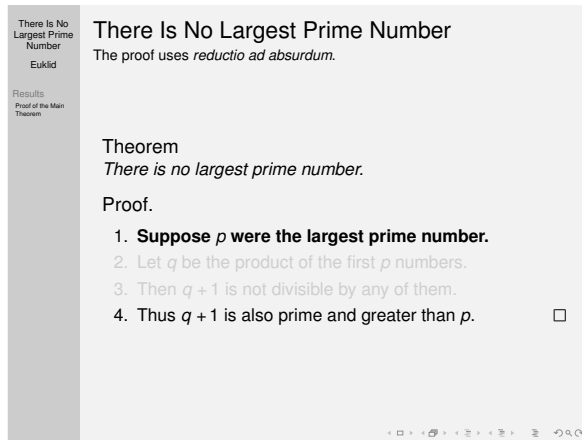
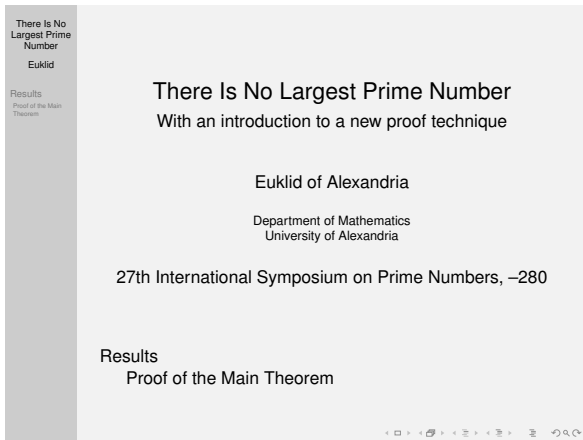


主题的名称：crane（鹤）。

该主题使用德国汉莎（Lufthansa）航空公司的颜色，汉莎航空公司的徽标是鹤 。然而，该主题不是汉莎航空公司的官方主题。

```
\usecolortheme{dove}
```

举例：



⁵⁷在装有 CTEX 套装的 Windows 7 平台中，`D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\color` 中的 `beamercolorthemebeetle.sty` 文件中有下面两条命令，只需更改第一条命令中的 RGB 值即可。

```
\definecolor{beetle@other}{RGB}{64,80,127}, \setbeamercolor*{palette primary}{fg=white,bg=beetle@other}.
```

⁵⁸更改 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\color` 中 `beamercolorthemebeetle.sty` 文件中的命令 `\setbeamercolor{normal text}{fg=black,bg=black!40}` 的 bg 值即可。

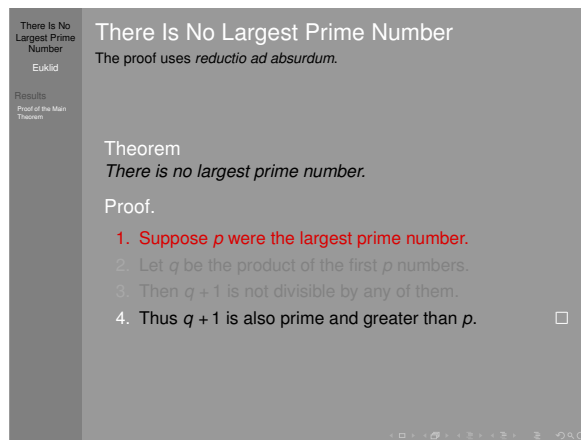
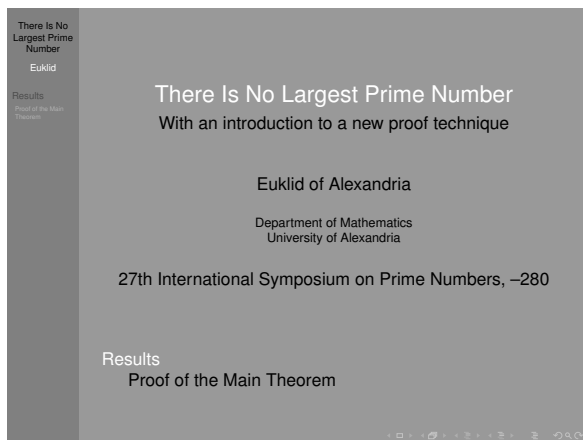
主题的名称: dove (鸽子)。

访主题几乎是一个黑白主题, 使用该主题的演示稿易于用黑白打印机打印。该主题不可避免地使用灰度 (grayscale), 但永不使用彩色 (color)。该主题将提示文本 (alerted text) 的字体更改为粗体。

使用该主题时, 必须使用文档类选项 `gray`, 该选项可确保所有颜色转变为灰度 (grayscale)。也必须使用字体主题 `structurebold`。

`\usecolortheme{fly}`

举例:

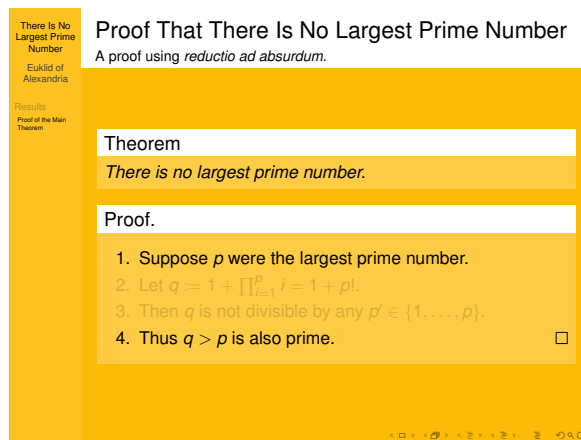
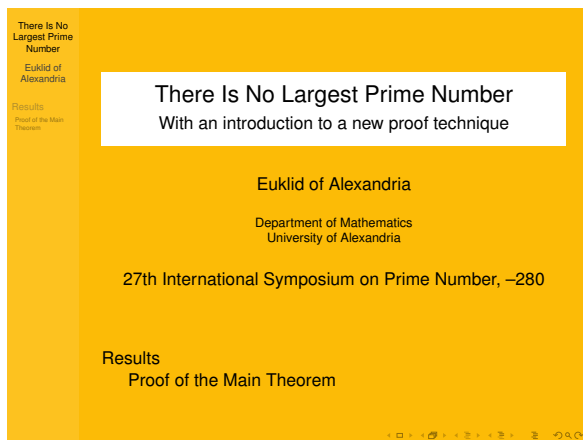


主题的名称: fly (苍蝇)。

该主题是自始至终使用了白/黑/灰的 beetle 主题的“最终”版本。该主题和使用阴影的主题不相配。

`\usecolortheme{monarca}`

举例:

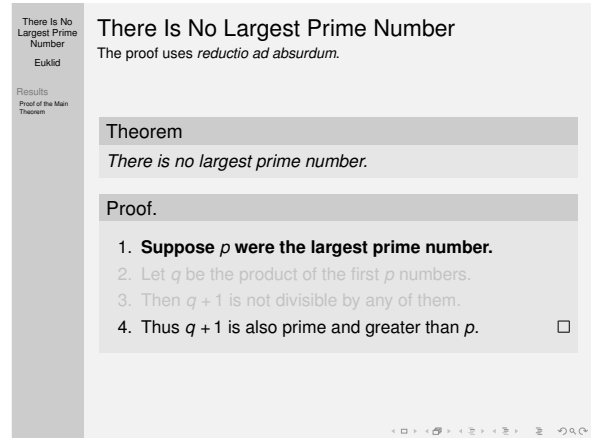
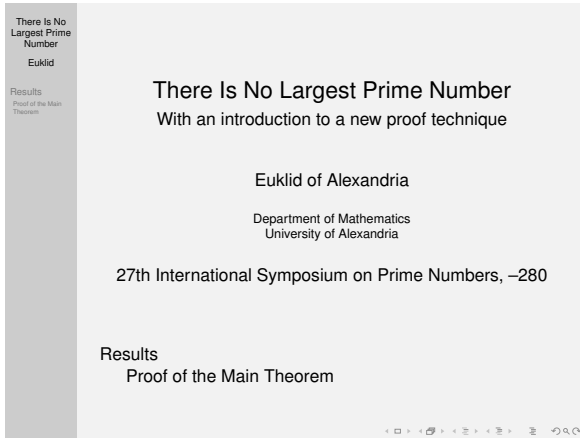


主题的名称: monarca (黑脉金斑蝶)。

主题的作者: Max Dohse (麦克斯·多泽) 该颜色主题使用了黑脉金斑蝶 (Monarch butterfly) 的颜色。

`\usecolortheme{seagull}`

举例:

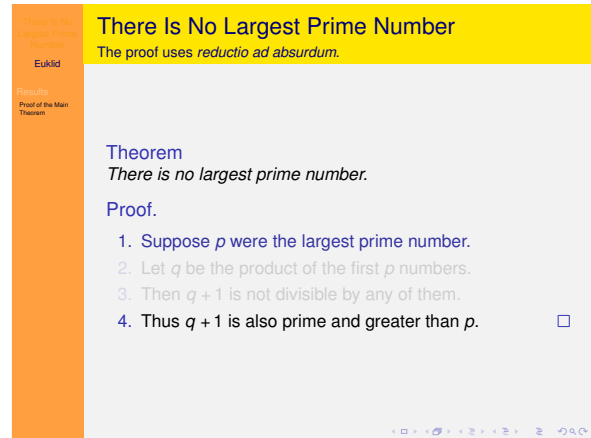
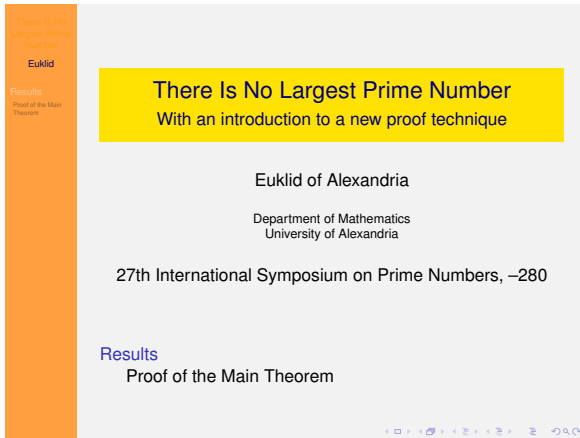


主题的名称: seagull (海鸥)。

和 dove (鸽子) 颜色主题相似, 该主题的演示稿易于用黑白打印机打印。然而, 它使用不同的灰度阴影, 这样的幻灯片 (transparency) 看起来可能漂亮也可能不漂亮。

`\usecolortheme{wolverine}`

举例:



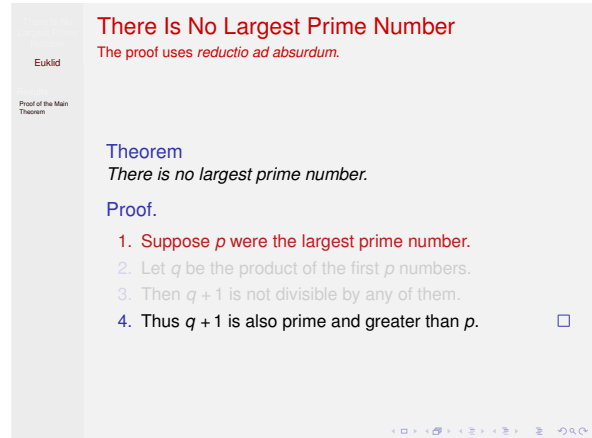
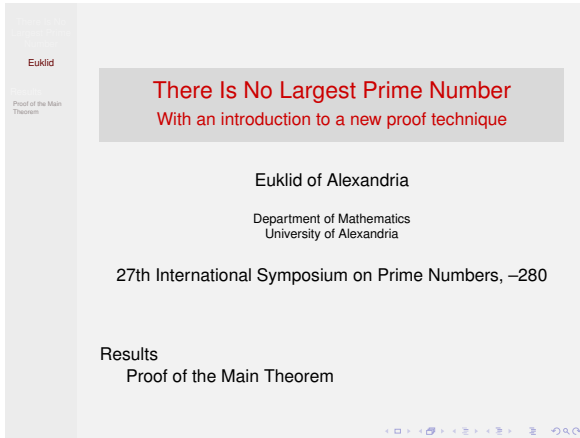
主题的名称: wolverine (貂熊)。

主题的作者: Madhusudan Singh (马杜苏登·辛格)。

该主题使用密执安大学 (University of Michigan) 的吉祥物貂熊 (wolverine) 的颜色。

`\usecolortheme{beaver}`

举例:



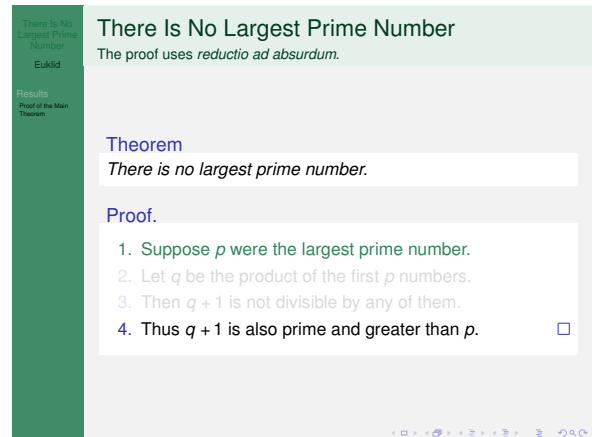
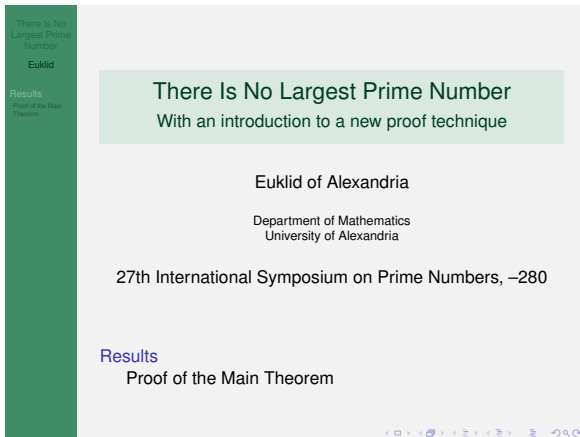
主题的名称: beaver (河狸)。

主题的作者: Madhusudan Singh (马杜苏登·辛格)。

该主题使用麻省理工学院 (Massachusetts Institute of Technology, MIT) 的吉祥物河狸 (beaver) 的颜色。

`\usecolortheme{spruce}`

举例:



主题的名称: spruce (针枞)。

主题的作者: Alan Munn (艾伦·曼)

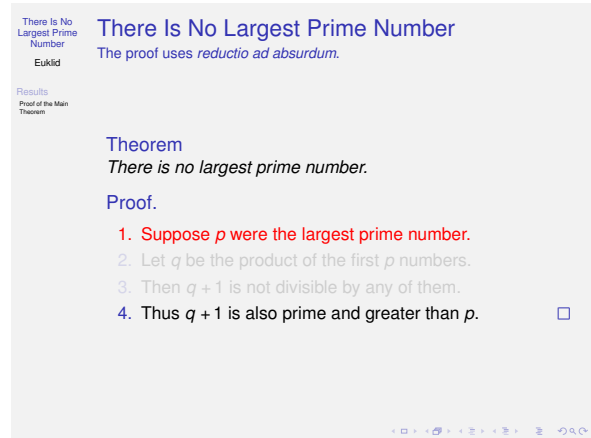
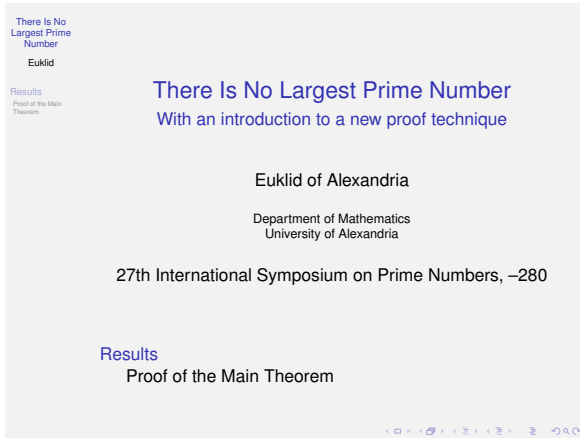
该颜色主题使用了密执安州立大学 (Michigan State University) 的颜色。

17.1.3 内部颜色主题

内部颜色主题 (Inner color themes) 只指定内部主题 (inner themes) 使用的颜色。最引人注目的是, 它们指定块 (blocks) 所使用的颜色。内部颜色主题也可与其它 (颜色) 主题一起使用。如果它们用于更改由演示主题 (presentation theme) 或其它颜色主题安装的内部颜色, 则会在加载其它主题之后指定这些内部颜色主题。内部颜色主题取名于花名。

`\usecolortheme{lily}`

举例:

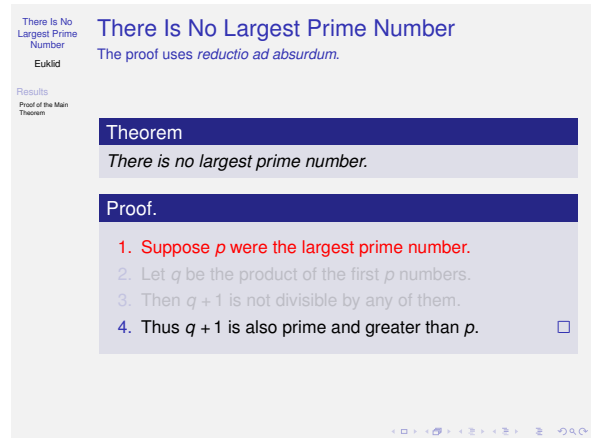
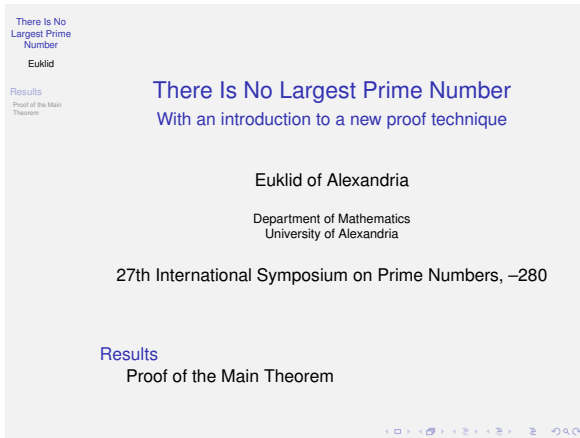


主题的名称：lily（百合）。

该主题主要用于卸载由其它主题安装的块颜色（block colors），恢复 default（默认）主题使用的颜色。使用该主题将移除所有块的背景色（background colors）。

`\usecolortheme{orchid}`

举例：

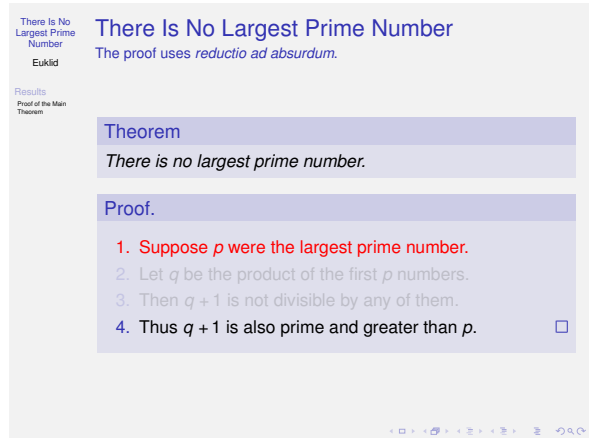
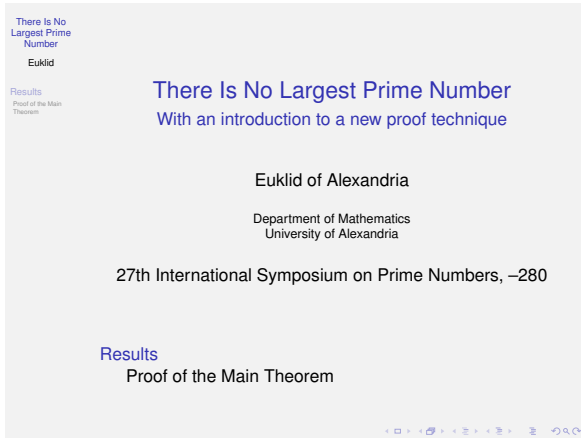


主题的名称：orchid（兰花）。

该主题安装一个暗底白字（white-on-dark）的块标题。普通块（normal blocks）标题的背景设置成了结构色的前景色（foreground of the structure color），而前景色设置成了白色。提醒块（alerted blocks）的背景设置成了红色。示例块（example blocks）的背景设置成了绿色。块主体（body of blocks）获得了几乎透明的（transparent）背景。

`\usecolortheme{rose}`

举例：



主题的名称：rose（玫瑰）。

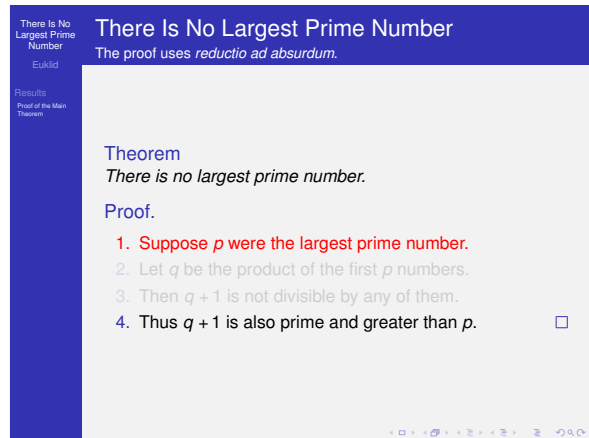
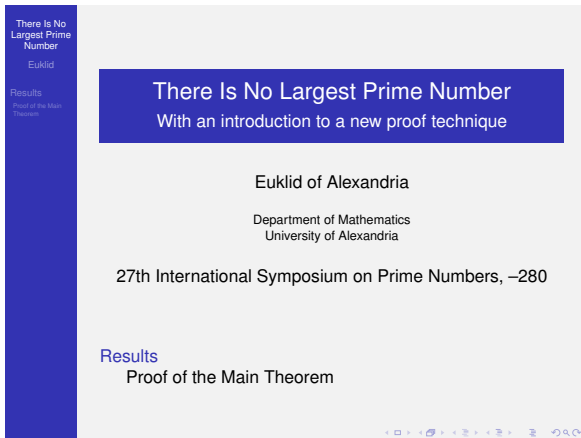
该主题为块标题（block titles）和块主体（block bodies）安装几乎透明的背景。该主题较 orchid（兰花）主题更不会“咄咄逼人（aggressive）”。背景色（background colors）衍生自结构 BEAMER-color（structure BEAMER-color）的前景色。

17.1.4 外部颜色主题

一个外部颜色主题（outer color theme）更改了模板的颜色（palette colors），该模板的颜色是顶部导航区、底部导航区、侧栏默认情况下所使用颜色的基础。外部颜色主题通常不会改变内部元素（inner elements）的颜色，除非对于 titlelike。外部颜色主题取名于海洋动物名。

`\usecolortheme{whale}`

举例：



主题的名称：whale（鲸）。

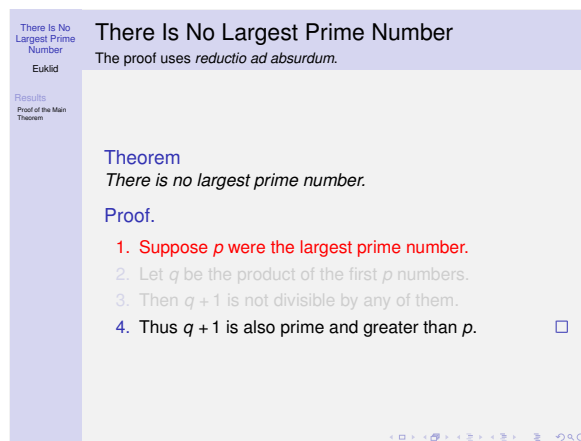
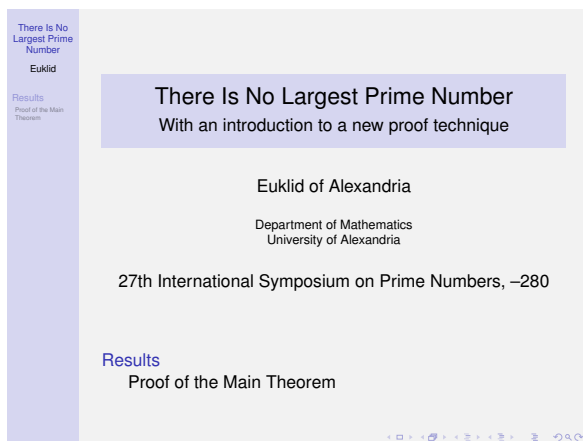
为顶部导航区、底部导航区、侧栏安装暗底白字（white-on-dark）的模板。这里使用的背景设置成了结构 BEAMER-color（structure BEAMER-color）和黑色之间的颜色。前景设置成了白色。

虽然该颜色主题可能显得咄咄逼人（aggressive），但我们会注意到，帧周围的暗条（dark bar）在演示稿（presentation）及纸（paper）中会有不同的外观：投影到墙上的演示稿被黑色包围，这样，与投影在纸上相比，暗条与周围的黑色不会产生对比。实际上，使用该主题将使帧的主体部分（main part of the frame）更容易吸引眼球。

和该主题及相关的块相配的是 `orchid`（兰花）主题。当然，和 `rose`（玫瑰）主题相配也是很有趣的。

```
\usecolortheme{seahorse}
```

举例：



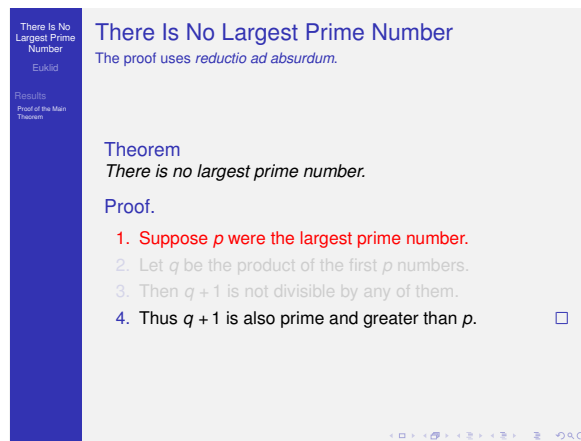
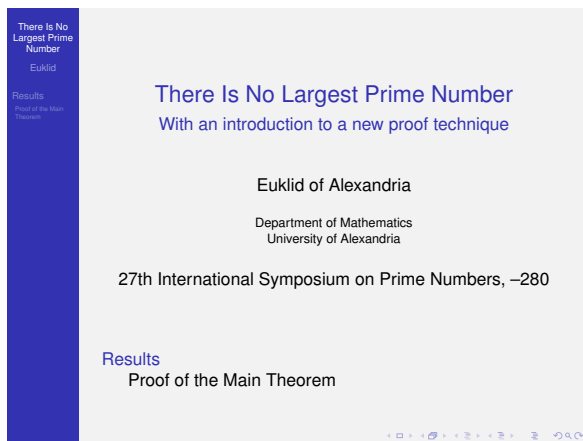
主题的名称：seahorse（海马）。

为顶部导航区、底部导航区、侧栏安装几乎透明的背景。和 `whale`（鲸）主题相比，使用该主题将使导航元素（navigational elements）更不“显眼（dominant）”。

该主题和 `rose`（玫瑰）或 `lily`（百合）颜色主题很相配。提尔（Till）教授认为该主题如和 `orchid`（兰花）主题相配，则会过分强调（overemphasizes）块。

```
\usecolortheme{dolphin}
```

举例：



主题的名称：dolphin（海豚）。

主题的作者：Manuel Carro（曼努埃尔·卡洛）。

该主题在某些地方处于 `whale`（鲸）和 `seahorse`（海马）两主题之间。和 `seahorse`（海马）主题一样，该主题与 `rose`（玫瑰）或 `lily`（百合）颜色主题很相配。

17.2 改变演示稿不同元素的颜色

本节阐述 BEAMER 颜色管理器是如何工作的。

17.2.1 Beamer 的颜色管理概述

在 BEAMER 的原理 (philosophy) 中, 演示稿的每一元素可以有不同的颜色。不幸的是, 事实证明只是简单地为演示稿的每一元素指定一种颜色不是个好主意。首先, 我们希望对演示稿元素的颜色进行全程改变, 如当条目标记 (item indicators) 变为提醒 (alerted) 或在示例块 (example block) 中时, 对条目标记 (item indicators) 的颜色进行改变。第二, 一些元素天生但不总是就有两种颜色, 即前景和背景。第三, 一些元素没有任何专门的颜色, 只是简单地“沿用 (run along)”其周围的颜色。最后, 为每一元素指定专门的颜色 (special color) 将使全局改变颜色 (如将全部不同的蓝色类东西改变成红色类东西) 变得困难, 也会使后来的功能扩展 (later extensions) 变得更困难。

由于上述原因, BEAMER 中的一个元素的颜色就是一个结构对象 (structured object), 可以称之为 BEAMER-color。每一个 BEAMER-color 由两部分组成: foreground (前景) 和 background (背景)。前景和背景都可以为“空 (empty)”, 这意味着当使用该颜色时, 无论什么前景或背景如以前是有效的 (active), 则它们仍然有效。

BEAMER-colors 可继承自其它 BEAMER-colors, 默认的主题扩展了该功能 (feature), 例如, 有一名为 **structure** 的 BEAMER-color, 并且所有类型的元素继承了该颜色。这样, 如果更改了 **structure**, 则继承了该颜色的所有元素的颜色将作相应的改变。如果颜色 A 继承自另一颜色 B, 则颜色 A 仍会废除 (override) 前景或背景。

也可能以一种更复杂 (sophisticated) 的方式“继承”自另一 BEAMER-color, 这更像间接地使用另一 BEAMER-color。我们可以这样指定: 标题 (beamer-color) 的背景是普通文本背景的 90% 和 **structure** 前景的 10%。

继承 (Inheritance) 和使用 (using) 其它 BEAMER-colors 是在动态中完成的, 这意味着如果父 BEAMER-colors 在演示稿的全程中不停地改变, 则其派生的颜色将自动改变。

常常会自动加载默认的颜色主题, 默认的颜色主题会安装大量的 BEAMER-colors 及与其有继承关系的 BEAMER-colors。这些颜色在本手册中会得到阐述。帧标题 (frametitles) 使用的颜色会在帧标题这一节阐述, 如此等等。

17.2.2 使用 Beamer 的颜色

一个 BEAMER-color 不是一个由 `color` 和 `xcolor` 宏包定义的普通的颜色, 因此, 它不能在 `\color` 或 `\colorlet` 这样的命令中直接使用。相反, 如果要使用一个 BEAMER-color, 必须调用 `\usebeamercolor`⁵⁹ 命令, 在下面阐述该命令。`\usebeamercolor` 命令会建立两个 (普通的) 颜色, 分别命名为 `fg` (对于前景)、`bg` (对于背景)。我们可以声明 `\color{fg}` 安装前景色, 声明 `\color{bg}` 安装背景色。我们也可以在上下文中使用 `fg` 和 `bg`, 在这样的上下文中我们常常使用象 `red` 这样的颜色。如果一个 BEAMER-color 没有前景或背景, `fg` 或/和 `bg` 将不改变。

在模板中, 该命令将和 `[fg]` 选项一同被调用。

```
\usebeamercolor*[{fg or bg}]{\beamer-color name}
```

即:

```
\usebeamercolor*[{fg 或 bg}]{\beamer-color 名}
```

⁵⁹ 在 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\inner` 中的 `beamerinnerthemedefault.sty`、`beamerinnerthemeinmargin.sty` 等文件中就有这样的命令。 [点击打开 beamerinnerthemedefault.sty 文件。](#)

该命令（可能）将两个颜色 `fg` 和 `bg` 改变成 $\langle beamer-color name \rangle$ 的前景色和背景色。如果 `BEAMER-color` 没有指定前景，则不会改变 `fg`；如果 `BEAMER-color` 没有指定背景，则不会改变 `bg`。

我们希望在使用该命令后直接使用颜色 `fg` 或 `bg`。对于这种情况，可选参数 $\langle fg or bg \rangle$ 很有用，它是 `fg` 或 `bg`。给定该选项将立即安装前景 `fg` 或背景 `bg`。因此，命令

```
\usebeamercolor[fg]{normal text}
```

是下面命令的简写形式

```
\usebeamercolor{normal text}
\color{fg}
```

如果使用该命令的带星号的版本（starred version），在调用该命令前会使用 `BEAMER-color normal text`。这样可以确保除不幸的欺骗外（barring evil trickery），在调用该命令时无论使用什么颜色，都会独立安装颜色 `fg` 和 `bg`。

该命令有一些特别的副作用（side-effects）。第一，调用该命令前，会将（常规）颜色 `parent.bg` 的值设置成 `bg` 的值。这样，我们就可以通过颜色 `parent.bg` 访问调用该命令之前使用的颜色。

第二，会将特别的颜色（special colors） $\langle beamer-color name \rangle.fg$ 的值全局（globally）地设置成 `fg` 的值，将 $\langle beamer-color name \rangle.bg$ 的值全局地设置成 `bg` 的值。这就允许我们在使用了另一 `BEAMER-color` 后访问某一 $\langle beamer-color name \rangle$ 的前景或背景。然而，应尽可能在小范围内并最低限度地参照这些特别的全局颜色（special global colors），因为在下一次调用 `\usebeamercolor` 之前，`BEAMER-color` 的更改不会在 $\langle beamer-color name \rangle.fg$ 和 $\langle beamer-color name \rangle.bg$ 颜色的更改中反映出来。而且，如果 $\langle beamer-color name \rangle$ 不指定一个前景色或背景色，在最后一次调用 `\usebeamercolor` 时该特别颜色（special colors）则会取前景或背景色的值。

因此，尽量不要养成总是写 `\color{structure.fg}` 的习惯，至少不要在附近没有 `\usebeamercolor{structure}` 时写 `\color{structure.fg}`。

举例：

```
This text is {\usebeamercolor[fg]{alerted text} alerted}. The
following box uses the fore- and background of frametitles:
{
  \usebeamercolor[fg]{frametitle}
  \colorbox{bg}{Frame Title}
}
```

ARTICLE 在论文（article）模式中该命令无效。

```
\ifbeamercolorempy[ $\langle fg or bg \rangle$ ]{ $\langle beamer-color name \rangle$ }{ $\langle if undefined \rangle$ }{ $\langle if defined \rangle$ }
```

即：

```
\ifbeamercolorempy[ $\langle fg 或 bg \rangle$ ]{ $\langle beamer-color 名 \rangle$ }{ $\langle 如果没有定义 \rangle$ }{ $\langle 如果定义了 \rangle$ }
```

该命令⁶⁰可用于检测某些 $\langle beamer-color name \rangle$ 的前景或背景是否是非空的（non-empty）。如果定义了 $\langle beamer-color name \rangle$ 的前景或背景，则会执行 $\langle if defined \rangle$ 的代码，否则，执行 $\langle if undefined \rangle$ 的代码。

⁶⁰在 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\inner` 中的 `beamerinnerthemedefault.sty` 文件中就有这样的命令。点击打开 `beamerinnerthemedefault.sty` 文件。

举例：

```
\ifbeamercolorempy[bg]{frametitle}
{ % ``Transparent background''
  \usebeamercolor[fg]{frametitle}
  \insertframetitle
}
{ % Opaque background
  \usebeamercolor[fg]{frametitle}
  \colorbox{bg}{\insertframetitle}
}
```

17.2.3 设置 Beamer 的颜色

要设置或更改一个 BEAMER-color，可以使用 `\setbeamercolor` 命令⁶¹。

```
\setbeamercolor*{<beamer-color name>}{<options>}
```

即：

```
\setbeamercolor*{<beamer-color 名>}{<选项>}
```

设置或更改一个 BEAMER-color。<beamer-color name> 必须是合法的常规文本（不能是意义含糊的文本，也不能用标点符号，但可以包含空格），因此 `normal text` 是有效的 <beamer-color name>，`My Color Number 2` 也是有效的。

最简单的情形是，通过 `fg=` 选项只指定了前景，通过 `bg=` 选项只指定了背景。

举例：`\setbeamercolor{normal text}{fg=black,bg=mylightgrey}`

举例：`\setbeamercolor{alerted text}{fg=red!80!black}`⁶²

该命令具有累积效应，因此，如下的两条命令

```
\setbeamercolor{section in toc}{fg=blue}
\setbeamercolor{section in toc}{bg=white}
```

和下面的这条命令具有相同的效果

```
\setbeamercolor{section in toc}{fg=blue,bg=white}
```

很自然，第二次调用带有相同类型 <option> 的该命令会设置成不同的值，并废除（overrides）第一次调用该命令时设置成的值。

带星号的版本首先会重置全部的设置，因此，它会“断开（switching off）”积累的效应。可以使用该带星号的版本完全地重置一些 BEAMER-color 的定义。

可能会给出下面的 <options>：

⁶¹在 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\color` 中的 `*.sty` 文件中就有很多 `\setbeamercolor` 命令。

⁶²在 `D:\CTEX\MiKTeX\tex\latex\beamer\base\themes\color` 中 `beamercolorthemedefault.sty` 文件中就有这样的命令。[点击打开该文件](#)。

- **fg**=*⟨color⟩* 将 *⟨beamer-color name⟩* 的前景色设置成给定的（常规）*⟨color⟩*。*⟨color⟩* 可以是像 `red!50!black` 这样的颜色表达式，请参阅（[点击打开](#)）XCOLOR 宏包的手册。如果 *⟨color⟩* 为空（empty），则 *⟨beamer-color name⟩* “没有特别的前景”，当使用该颜色时，不会改变当前有效的前景。通过这样指定的前景色会废除（override）任何继承来的前景色。
- **bg**=*⟨color⟩* 所作所为与 fg 选项相同，只不过是相对于背景而言。
- **parent**=*⟨parent beamer-color(s)⟩* 指定 *⟨beamer-color name⟩* 必须继承自指定的 *⟨parent beamer-color(s)⟩*。当使用 *⟨beamer-color name⟩* 时，由父本（parents）设定的任何前景色和/或背景色将被使用。如果多个父本指定了一个前景，则最后一个父本指定的前景是“有效的（wins）”。对于背景，原理相同。

举例：

```
\setbeamercolor{father}{fg=red}
\setbeamercolor{mother}{bg=green}
\setbeamercolor{child}{parent={father,mother}}
\begin{beamercolorbox}{child}
  Terrible red on green text.
\end{beamercolorbox}

\setbeamercolor{father}{fg=blue}
\begin{beamercolorbox}{child}
  Now terrible blue on green text, since parent was changed.
\end{beamercolorbox}
```

注意，父本的前景或背景的更改将会改变与其相关的后代（child）的前景或背景（除非被其推翻）。一个 BEAMER-color 不仅可以有父本（parents），而且可以有祖本（grandparents）等等。

- **use**=*⟨another beamer-color⟩* 用于确保在评价（evaluate）前景色或背景色规则（specification）之前正确地安装另一 BEAMER-color。

假若我们希望条目（items）的前景色由结构性元素（structural elements）前景色的 50% 和普通前景色（normal foreground color）的 50% 混合而成，我们可试试：

```
\setbeamercolor{item}{fg=structure.fg!50!normal text.fg}
```

然而，这不一定会得到期望的结果：如果 BEAMER-color `structure` 改变了，则（常规）颜色 `structure.fg` 不会立即更新。要确保常规颜色（normal color）`structure.fg` 是正确的，请使用下面的命令：

```
\setbeamercolor{item}{use={structure,normal text},fg=structure.fg!50!normal text.fg}
```

这能确保当计算 `item` 的前景时，可以正确地安装 `structure.fg` 和 `normal text.fg` 颜色。

下面的例子显示其中的不同之处：

```
\setbeamercolor{grandfather}{fg=red}
\setbeamercolor{grandmother}{bg=white}
\setbeamercolor{father}{parent={grandfather,grandmother}}
\setbeamercolor{mother}{fg=black}
{
  \usebeamercolor{father}\usebeamercolor{mother}
```

```

%% Defines father.fg and mother.fg globally
}
\setbeamercolor{my color A}{fg=father.fg!50!mother.fg}
\setbeamercolor{my color B}{use={father,mother},fg=father.fg!50!mother.fg}

{\usebeamercolor[fg]{my color A} dark red text}
{\usebeamercolor[fg]{my color b} also dark red text}

\setbeamercolor{grandfather}{fg=green}

{\usebeamercolor[fg]{my color A} still dark red text}
{\usebeamercolor[fg]{my color B} now dark green text}

```

17.3 数学文本的颜色

默认情况下，数学文本（mathematical text）没有任何特别的颜色（special color）——它只继承了“周围的（surrounding）”颜色。有人更喜欢数学文本具有特别的颜色。但我们不建议这样做，因为在普通文本（normal text）中数学文本不应突出显示（standout），BEAMER 要改变数学文本的颜色是非常容易的。它只要改变下面的颜色即可：

Beamer-Color `math text`

即：

Beamer-Color 数学文本

该颜色是 `math text inlined` 和 `math text displayed` 的父本。默认情况下该颜色为空（empty）。详情请参阅下面。

Beamer-Color `math text inlined`

即：

Beamer-Color 行内数学文本

Color 父模板：`math text`

如果设置了该颜色的前景，将使用该颜色排版行内数学文本（inlined mathematical text）。这是通过一些 `\everymath` hackery 实现的，但并不总是这样。如果不是这样，就必须找到解决此问题的方法。背景在当前会被忽略。

Beamer-Color `math text displayed`

即：

Beamer-Color 已显示的数学文本

Color 父模板：`math text`

和 `math text inlined` 相似，该模板只是针对所谓的“已显示（displayed）”的数学文本。已显示的数学文本是指 `\[` 和 `\]` 之间、或 `$$` 和 `$$` 之间、或 `equation`、`align` 这样的环境内的文本。该颜色的设置有几分脆弱（fragile），风险由自己承担。背景在当前会被忽略。

Beamer-Color `normal text in math text`

即：

Beamer-Color 数学文本内的普通文本

如果设置了该颜色的前景，将用该颜色排版数学文本（由 `\text` 命令声明）内的普通文本（normal text）。背景在当前会被忽略。

17.4 调色板

在设计一个颜色主题（color theme）时，将面临如下的问题：希望顶部导航区（headline）的颜色由黑色（black）渐变成蓝色（blue）；无论如何顶部导航区的最上面为黑色，其下面为深蓝（dark blue），最下面为蓝色。不幸的是，不同的外部主题（outer themes）会在顶部放置不同的东西。一个主题在顶部放置作者（author），另一主题可能在此放置文档标题（document title）。不可能将“黑色”、“深蓝”、“蓝色”分别指派给顶部导航区的不同元素。不管我们如何指派它们，对于外部主题来讲，事情看起来会出错。

要解决该问题，BEAMER 使用了 *palette colors* 的层（layers）。颜色主题（Color themes）只改变这些调色板的颜色（palette colors），例如，一个颜色主题可能将 `BEAMER-color palette primary` 设为蓝色（blue），将 `palette secondary` 设为深蓝色（dark blue），将 `palette tertiary` 设为黑色（black）。外部主题（Outer themes）现在就可以让无论如何都要显示在顶部导航区最上面的东西的颜色继承自 `palette primary`，其下面的东西的颜色继承自 `palette secondary`，最下面的东西的颜色继承自 `palette tertiary`。这样，颜色主题就可以改变相当复杂的外部主题的外观，并一如既往地这样做。

注意，我们仍可以独自改变每一元素的颜色，只需重新定义（overriding）顶部导航区元素的颜色。从某种意义上说，调色板的颜色仅仅是“建议”的颜色，即一个外部主题建议元素使用什么颜色。

关于外部主题使用的调色板的颜色的详细情况如下：

Beamer-Color `palette primary`

即：

Beamer-Color 调色板第一

外部主题（outer themes）把导航元素（navigational elements）及其它元素建立在四个调色板的基础上。“第一（primary）”调色板用于大多数重要的导航元素，导航元素改变频繁并吸引观众的大部分注意力。“第二（secondary）”调色板和“第三（tertiary）”调色板更不那么重要，“第四（quaternary）”调色板则最不重要。

默认情况下，调色板的颜色不拥有从 `structure.fg` 到 `black` 的背景和前景范围（ranges）。

对于侧栏（sidebar），有一套特别的调色板颜色，请参阅 `palette sidebar primary`。

Beamer-Color `palette secondary`

即：

Beamer-Color 调色板第二

请参阅 `palette primary`。

Beamer-Color `palette tertiary`

即：

Beamer-Color 调色板第三

请参阅 palette primary。

Beamer-Color palette quaternary

即：

Beamer-Color 调色板第四

请参阅 palette primary。

Beamer-Color palette sidebar primary

即：

Beamer-Color 调色板侧栏第一

和 palette primary 相似，只有外部主题把侧栏中的元素的颜色建立在四个侧栏调色板颜色的基础之上。

Beamer-Color palette sidebar secondary

即：

Beamer-Color 调色板侧栏第二

请参阅 palette sidebar primary。

Beamer-Color palette sidebar tertiary

即：

Beamer-Color 调色板侧栏第三

请参阅 palette sidebar primary。

Beamer-Color palette sidebar quaternary

即：

Beamer-Color 调色板侧栏第四

请参阅 palette sidebar primary。

17.5 杂色

这节罗列了一些“基本 (basic)”的颜色，它们不“属于”某个特别的命令。

Beamer-Color /-Font normal text

即：

Beamer-Color/-Font 普通文本

该颜色用于普通文本 (normal text)。在文档的开始以 `\normalcolor` 形式安装前景色。通过默认背景画布 (background canvas) 将该颜色的背景用作演示稿的背景, 请参阅第 8.2.7 节。背景具有常规颜色 (normal color) `bg` 的默认值。

因为该颜色是所有其它 BEAMER-colors 的“根基 (root)”, 必须安装前景和背景。尤其是, 要获得透明的 (transparent) 背景画布, 请将 `BEAMER-color background canvas` 的背景设置为空 (empty), 而不是将该颜色的背景设置为空。

在当前不会使用 BEAMER-font。尤其是, 重新定义 (redefining) 该字体不会有任何结果。在将来或许有可能改变这种状况。

Beamer-Color/-Font example text

即:

Beamer-Color/-Font 示例文本

排版示例块 (example block) 中的文本时使用该 color/font。

Beamer-Color/-Font titlelike

即:

Beamer-Color/-Font 标题类

该 color/font 是 `structure color/font` 的专化型 (specialized form)。它是所有“标题类 (like titles)”元素的基础。它包括帧标题 (frame title) 和副标题 (subtitle), 这与文档标题 (document title) 和小标题 (subtitle) 一样。

Beamer-Color separation line

即:

Beamer-Color 分隔线

该颜色的前景用于分隔线 (separating lines)。如果前景为空, 则不会生成分隔线。

Beamer-Color upper separation line head

即:

Beamer-Color 顶部导航区分隔线的上面

Color 父模板: `separation line`

用于顶部导航区分隔线的最上面的特殊盒子 (special case)。

Beamer-Color middle separation line head

即:

Beamer-Color 顶部导航区分隔线的中间

Color 父模板: `separation line`

用于顶部导航区分隔线的中间的特殊盒子 (special case)。

Beamer-Color `lower separation line head`

即:

Beamer-Color 顶部导航区分隔线的最下面

Color 父模板: `separation line`

用于顶部导航区分隔线的最下面的特殊盒子 (special case)。

Beamer-Color `upper separation line foot`

即:

Beamer-Color 底部导航区分隔线的最上面

Color 父模板: `separation line`

用于底部导航区分隔线的最上面的特殊盒子 (special case)。

Beamer-Color `middle separation line foot`

即:

Beamer-Color 底部导航区分隔线的中间

Color 父模板: `separation line`

用于底部导航区分隔线的中间的特殊盒子 (special case)。

Beamer-Color `lower separation line foot`

即:

Beamer-Color 底部导航区分隔线的最下面

Color 父模板: `separation line`

用于底部导航区分隔线的最下面的特殊盒子 (special case)。

17.6 透明效果

默认情况下, 隐藏的条目 (covered items) 在放映演示稿的过程中不会显示。因此, 如果我们写了 `\uncover<2>{Text.}` 这样一条命令, 则文本 (text) 只会在第二张幻灯片中显示。在其它幻灯片中, 文本会显示为背景色 – 即根本就不会显示出来。如果背景颜色不均匀, 该功能则很有用。

然而, 有时我们希望隐藏的条目不要完全地隐藏。而且, 希望这些条目以暗淡 (dim) 的方式早已 (already) 显示。这样, 观众就可以知道马上要讲的内容是什么, 却不会分心。而且, 我们可能希望“再一次”隐藏的文本保持一定程度的可见。

理想的情况是，有一个选项可以让隐藏的文本变得“透明（transparent）”。这意味着当显示隐藏的文本时，也会显示它后面的背景。不幸的是，`pgf` 不支持真正的透明。实际上，透明是由要显示的对象的颜色和当前的背景色（已安装颜色 `bg` 是背景色的平均）混合后形成的，要安装该功能，可以使用下面的命令：

```
\setbeamercovered{transparent}
```

该命令允许我们指定如何生成（render）一个隐藏的条目。我们甚至可能指定如何生成条目，在显示该条目之前要花多长时间，以及再一次隐藏要花多长时间。除图像中的颜色之外，透明效果会自动应用于所有颜色，因为图像有一个工作区（workaround），请参阅 PGF 宏包的文档。

```
\setbeamercovered{<options>}
```

即：

```
\setbeamercovered{<选项>}
```

该命令提供了多个不同的选项，其中最重要的选项是 `transparent`。所有选项在内部映射至两个选项 `still covered` 和 `again covered`。

更详细地，可能会给出下面的 `<options>`：

- `invisible` 为默认的选项，它使隐藏的文本（covered text）“完全地消失（completely disappear）”。
- `transparent=<opaqueness>` 以“透明（transparent）”的方式排版隐藏的文本。默认情况下，将 85% 的背景色混入全部颜色（all colors）中，或将文本的 `<opaqueness>` 设置成 15%。我们可以指不同的 `<percentage>`，0 代表“完全透明（totally transparent）”，100 代表“完全不透明（totally opaque）”。不幸的是，该值对每一投影仪（projector）有几分“特别（specific）”。在我们的电脑上看起来很棒但在演讲时就不一定了。
- `dynamic` 使所有隐藏的文本完全透明（quite transparent），但这个过程是“动态的（dynamic）”。文本显现（uncovered）以前，时间越长，透明度（transparency）就越大（stronger）。
- `highly dynamic` 作用和 `dynamic` 相同，但该选项的作用更强。
- `still covered=<not yet list>` 指定如何生成（render）还没有显示的隐藏的条目。`<not yet list>` 必须是 `\opaqueness` 命令的列表，请参阅该命令的描述，如下。

举例：

```
\setbeamercovered{%
  still covered={\opaqueness<1>{15}\opaqueness<2>{10}\opaqueness<3>{5}\opaqueness<4->{2}},
  again covered={\opaqueness<1->{15}}}
```

- `again covered=<once more list>` 指定如何生成（render）已经显示不止一次的隐藏的条目，也就是说，这些条目在以前显示后来又隐藏了。

```
\opaqueness<<overlay specification>>{<percentage of opaqueness>}
```

即：

```
\opaqueness<<叠层规则>>{<不透明百分比>}
```

`<overlay specification>` 指定哪张幻灯片中的隐藏文本具有 `<percentage of opaqueness>` [译者注：这句的原文是：The `<overlay specification>` specifies on which slides covered text should have which `<percentage of`

opaqueness]). 不象其它的叠层规则, 这里的叠层规则是“相对的 (relative)”。例如, 这里的规则“3”是指“前面三张幻灯片中的内容 (things) 将会显示 (uncovered)”, 各自地, “三张幻灯片中的内容再一次隐藏”。更精确地讲, 如果一个条目在一张以上幻灯片中显示, 然后再一次隐藏, 只有“首次的显示 (first moment of uncovering)”用于计算多长时间后再一次隐藏该条目。

不透明度 (opaqueness) 取值为 100 时代表完全不透明, 取值为 0 时为完全透明 (transparent)。目前, 因为真正的透明尚未实现, 该命令会使用所有颜色混入当前 `bg` 的 $\langle \textit{percentage of opaqueness} \rangle$ 。在将来, 该命令可能实现真正的透明。

能用在不透明区域 (opaque area) 内的可代替前述命令的 PGF 扩展是 $\langle \textit{percentage of opaqueness} \rangle \textit{opaque}$ 。如果出现嵌套调用 (nested calls), 只有最内层的不透明度规则 (innermost opaqueness specification) 才可用。

举例:

```
\setbeamercovered{still covered={\opaqueness<1->{15}},again covered={\opaqueness<1->{15}}}  
\pgfdeclareimage{book}{book}  
\pgfdeclareimage{book.!15opaque}{filenameforbooknearlytransparent}
```

以 15% 不透明 (opaque) 显示两张幻灯片中的所有内容。

18 字体

第一小节介绍了 BEAMER 附带的预定好的字体主题 (font themes)，通过字体主题很容易改变演示稿使用的字体。第二小节阐述了更专业的更改演示稿使用的字体基本属性的命令。第三小节解释了如何更详细地控制演示稿中每一个元素使用的字体。

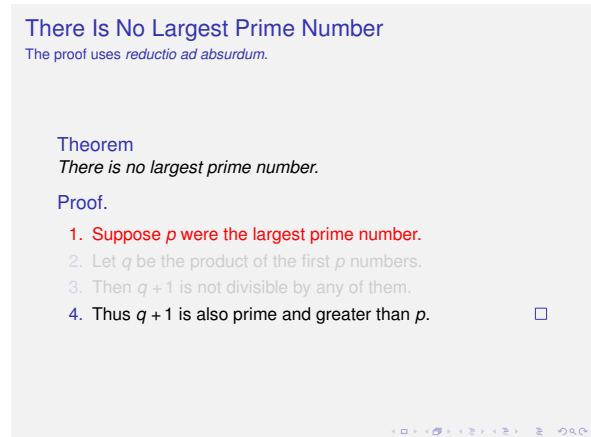
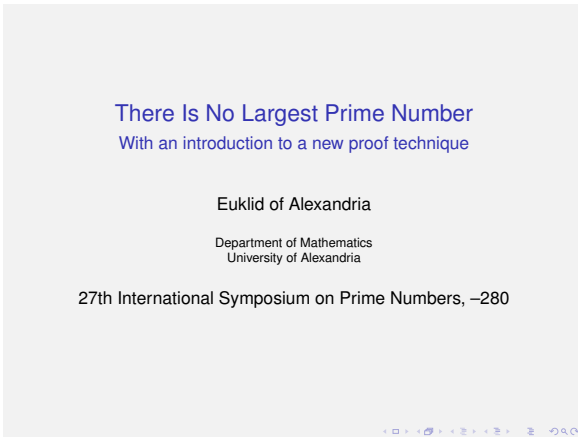
18.1 字体主题

BEAMER 附带有许多字体主题 (即字体主题集)。当我们使用这样一个字体主题时，将按下面的描述改变特定的字体 (certain fonts)。我们可以同时使用多个字体主题。由于历史原因，我们无法更改字体主题使用的字体的所有外观 (aspects) — 在某些情况下，可能需要特定的命令及选项，在下一小节会阐述这些特定的命令及选项。

下面的字体主题只改变特定的字体属性，它们不会选择特别的字族 (font family) (虽然这也是有可能实现的，而且能实现该功能的字体主题在将来可能会添加到 BEAMER 中)。当前，要改变字族，我们必须加载特定的宏包，这会在下一节得到阐述。

```
\usefonttheme{default}
```

举例：



主题的名称：default (默认)。

默认的字体主题会为演示稿的所有文本安装无衬线字体 (sans serif font)，并在像标题 (titles)、顶部和底部导航区这样的元素中使用不同的字体的尺寸 (font sizes)，但不会将粗体 (boldface) 或斜体 (italics) 用于“高亮显示 (Highlighting)”。可以使用 serif 主题将某些或所有文本改成衬线字体 (serif font)。

注意：`\mathrm` 命令总是生成直立 (非倾斜)、衬线文本，而 `\mathsf` 命令总时生成直立、无衬线文本。`\mathbf` 命令将生成直立、粗体、无衬线或衬线文本，这视使用 `mathsans` 还是使用 `mathserif` 而不同。

要生成直立、粗体、无衬线或衬线文本，视使用 `mathsans` 还是使用 `mathserif` 而不同，可以使用来自 `amsmath` 宏包的 `\operatorname` 命令。如果我们将无衬线 (sans-serif) 切换成衬线 (serif) 或相反，用 `\operatorname` 命令直接代替 `\mathrm` 或 `\mathsf` 命令则可以调整直立的 (upright) 数学文本 (mathematical)。

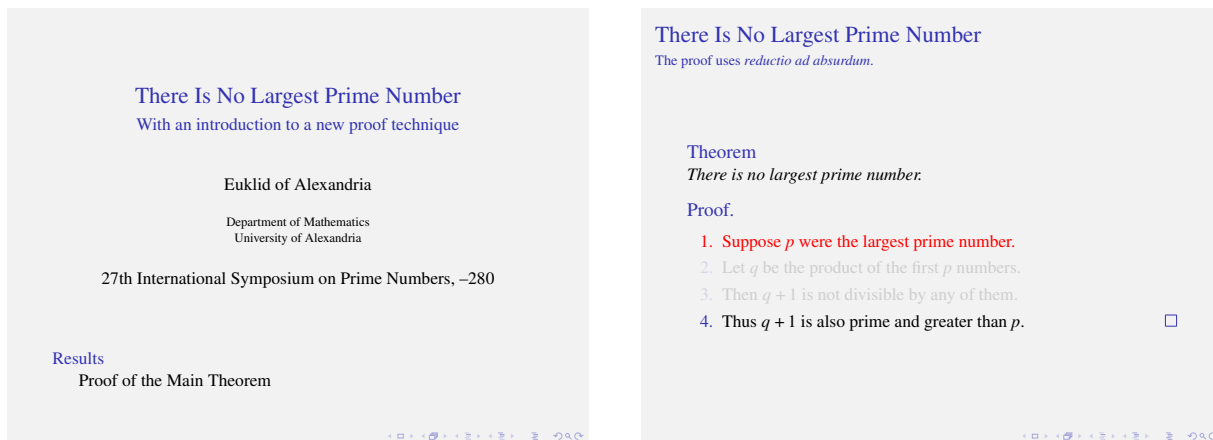
```
\usefonttheme{professionalfonts}
```

主题的名称: professionalfonts (专业字体)。

该字体主题实际上并不改变任何字体。其实,它是抑制由 BEAMER 执行的特定的内部替换 (internal replacements)。如果我们使用了“专业字体 (professional fonts)” (即: fonts that you buy 和 that come with a complete set of every symbol in all modes), 则无法让 BEAMER 干涉 (meddle) 我们所使用的专业字体。BEAMER 通常用更合适的字符竖沟 (character glyphs) 替换数学文本中特定的字符竖沟。例如, BEAMER 通常替换数学文本中变量的字符的竖沟, 这些字符来自主字体 (main font) 的斜体字符。如果我们的专业字体包已经考虑了这一点, 则会关闭 BEAMER 对该专业字体的干涉。注意, 如果加载了下列宏包之一, 则会自动关闭 BEAMER 的上述替换功能: arevmath、hvmath、lucidabr、lucimatx、mathpmt、mathpple、mathtime、mtpro、mtpro2。如果我们个人的专业字体包不在上述之列, 请使用 professionalfont 选项 (并写电邮给我们, 以将我们的字体包加入上述之列)。

```
\usefonttheme[<options>]{serif}
```

举例:



主题的名称: serif (衬线)。

该主题使所有文本用默认的衬线字体 (serif font) 排版 (除非我们指定了特定的 $\langle options \rangle$)。我们可以参阅第 5.6.2 节后决定是否使用衬线字体。

可能会给出下面的 $\langle options \rangle$

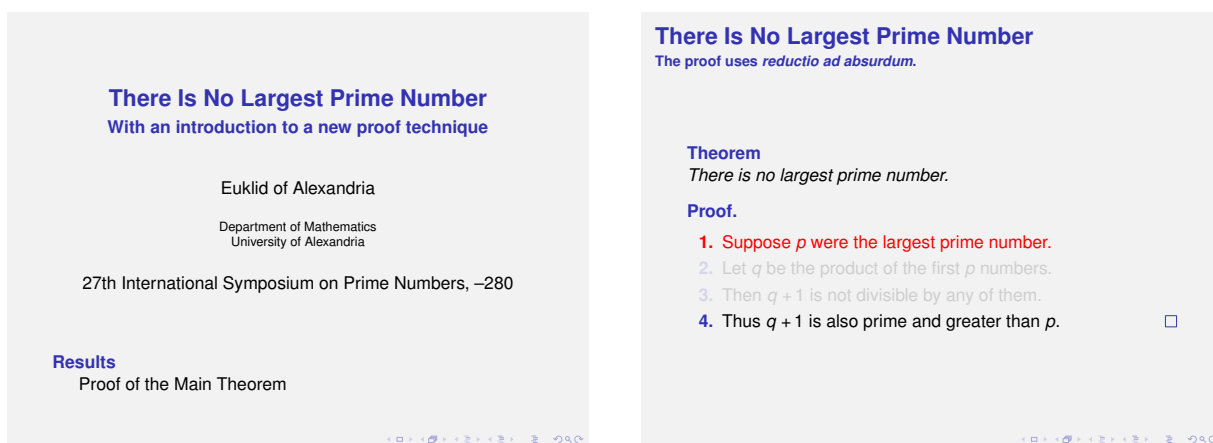
- **stillssansserifmath** 仍用无衬线字体 (sans serif) 排版数学文本。如果我们同时使用 stillssansseriftext 选项, stillssansserifmath 选项才有意义, 因为在衬线文本中使用无衬线数学文本会显得很难看。
- **stillssansserifsmall** 仍用无衬线排版“小 (small)”文本, 这主要涉及顶部导航区、底部导航区、侧栏 (sidebar) 这些地方的文本。使用该选项是明智的, 因为无衬线字体的小文本易于阅读。
- **stillssansseriflarge** 仍用无衬线排版演示稿标题 (presentation title) 或帧标题 (Frame Title) 这样的“大 (large)”文本。标题 (titles) 用无衬线而正文 (text) 用衬线这种字体联用 (fonts combinations) 在印刷中很流行。
- **stillssansseriftext** 仍用无衬线排版普通文本 (normal text) (即除上述三种文本以外的文本)。如果使用了该选项, 我们也应使用上述前面两个选项 (即 stillssansserifmath、stillssansserifsmall)。然而, 如不使用 stillssansseriflarge 选项, 则会将无衬线的文本恢复为一个衬线 (可能是斜体的) 标题 [译者注: 该句原文是: you get a serif (possibly italic) title over a sans serif text]。这是一个有趣的保护色效应 (visual effect)。自然而然, “有趣的印刷效果

(typographic effect)”意味着如果我们选择了糟糕的字体联用 (fonts combinations) 或糟糕的字体尺寸, 则变成了“糟糕的印刷效果”。当我们拥有一定印刷经验后则可以避免发生上述糟糕的事情。我们也可以向别人请教。

- **onlymath** 这是一个选择好上述选项 (除外第一个选项) 的捷径。因此, 使用该选项时只会用衬线字体排版数学文本。回顾以前, 默认情况下, 用无衬线字体排版数学方程式。大多数情况下, 如果用衬线字体排版数学文本, 而其周围的文本则用无衬线字体排版, 这样的排版令人喜爱, 也很容易阅读。然而, 在数学文本中, 用于渲染 (render) 变量的字体在该变量不同的含义时可能有所差别, 这时, 有必要用衬线字体排版数学文本。而且, 如果有大量的数学文本, 并且是用大家熟悉的衬线字体排版, 那么观众也许很快就能“分析 (parse)”它们。

`\usefonttheme[{options}]{structurebold}`

举例:



主题的名称: `structurebold` (结构粗体)。

该字体主题用粗体 (bold) 排版标题 (titles)、顶部导航区、底部导航区、侧栏中的文本。

可能会给出下面的 *{options}*:

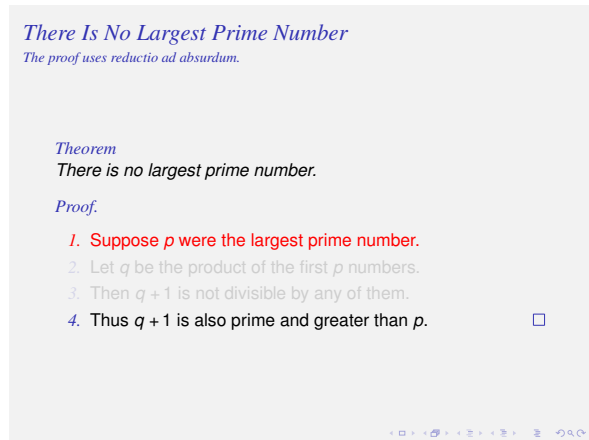
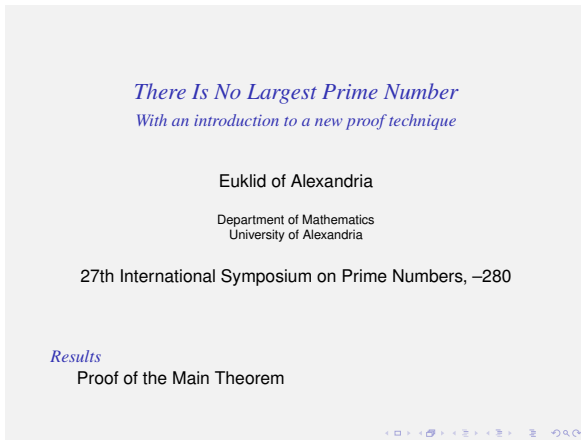
- **onlysmall** 用粗体排版“小 (small)”文本。更精确地讲, 只有顶部导航区、底部导航区、侧栏中的文本用粗体排版。该选项对大标题 (Large titles) 无效。
- **onlylarge** 用粗体排版“大 (large)”文本。适用于主标题 (main title)、帧标题、目录中的节条目。

正如第 5.6.1 指出的那样, 如果字体没有适当地按比例缩小或不适合暗背景亮文本 (light-on-dark text), 我们应使用该主题 (尽可能带有 `onlysmall` 选项)。

默认情况下一般的主题 (normal themes) 不会安装该字体主题, 便旧版本的兼容的主题会安装该字体主题。因为我们可以已经在已经加载了该字体主题后重新加载该字体主题, 我们不能同时使用该主题和旧版本的兼容的主题将标题设为粗体。

`\usefonttheme[{options}]{structureitalicserif}`

举例:

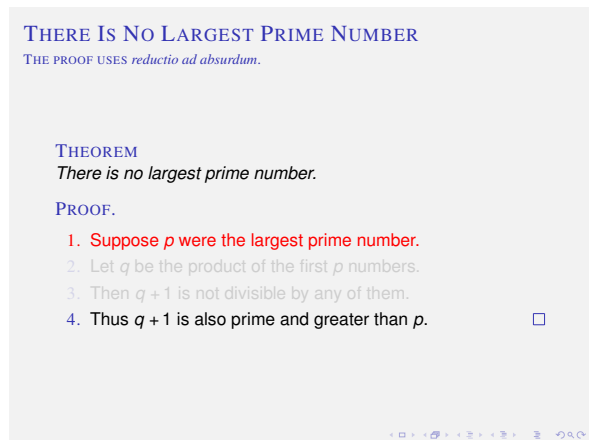
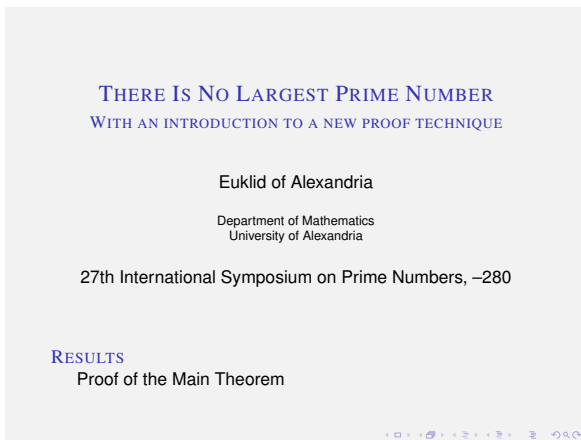


主题的名称：structureitalicserif（结构斜体衬线）。

该主题和 `structurebold` 字体主题相似，但 `structurebold` 字体主题将文本排版成粗体（bold），而该字体主题用标准的衬线字体将文本排版成斜体。该主题支持的 `<options>` 和 `structurebold` 字体主题相同。支持（pros）和反对（cons）使用斜体请参阅第 5.6.3 节。

```
\usefonttheme[<options>]{structuresmallcapsserif}
```

举例：



主题的名称：structure**smallcaps**serif（结构小型大写衬线）。

该主题的所作所为和 `structurebold` 字体主题相同，只不过是该主题使用小型大写（small caps）和衬线字体。该主题支持的 `<options>` 和 `structurebold` 字体主题相同。支持（pros）和反对（cons）使用小型大写请参阅第 5.6.3 节。

18.2 不通过字体主题而改变字体

虽然大部分字体制定（font decisions）可以通过字体主题实现，但由于历史原因，某些字体制定只能通过文档类（class）选项或加载特定的宏包（packages）实现。下面将阐述这些选项。或许在将来这些选项会被主题取代。

18.2.1 选择字体的尺寸为常规

正如第 5.6.1 节所述，用点（points）衡量演示稿默认的字体尺寸不是一个好主意。然而，BEAMER 却是这样做的，它将默认的字体尺寸设为 11pt。这看起来有点小，但帧的尺寸默认情况下是 128mm×96mm，而浏览器会将字体放大（enlarges）。指定默认的字体尺寸小于 11pt 就可以在每一幻灯片中放置更多的东西，而指更大的字体尺寸则更不适合于每一帧。

要指定字体尺寸，我们可以使用下面的选项：

```
\documentclass[8pt]{beamer}
```

这样指定的字体尺寸很小。必需先安装 `extsize` 宏包。

```
\documentclass[9pt]{beamer}
```

这样指定的字体尺寸也很小。必需先安装 `extsize` 宏包。

```
\documentclass[10pt]{beamer}
```

如果我们想更适合每一帧，请使用该选项。无需安装 `extsize` 宏包。

```
\documentclass[smaller]{beamer}
```

和 10pt 选项相同。

```
\documentclass[11pt]{beamer}
```

默认的字体尺寸。我们无需指定该选项。

```
\documentclass[12pt]{beamer}
```

使所有的字体更大一点，这有助于阅读。下面的选项不太适合每一帧。

```
\documentclass[bigger]{beamer}
```

和 12pt 选项相同。

```
\documentclass[14pt]{beamer}
```

使所有的字体有几分（somewhat）大。需安装 `extsize` 宏包。

```
\documentclass[17pt]{beamer}
```

这是 PowerPoint 和 OpenOffice.org Impress 的默认字体尺寸。需安装 `extsize` 宏包。

```
\documentclass[20pt]{beamer}
```

这样设置的字体是巨大的（huge）。需安装 `extsize` 宏包。

18.2.2 选择字族

默认情况下，BEAMER 使用计算机现代字体（Computer Modern fonts）。要改变这一点，我们可以使用已准备好的 L^AT_EX 的字体机制（font mechanism）宏包之一。例如，要将计算机现代字体更改为 Times/Helvetica，只需在导言区（Preamble）添加如下两条命令：

```
\usepackage{mathptmx}
```

```
\usepackage{helvet}
```

注意，如果我们不使用 `serif` 字体主题，将选用 Helvetica（而不是 Times）作为文本字体（text font）。

也许我们安装了其它可用的字体。通常，至少下列宏包可用：`arev`、`avant`、`bookman`、`chancery`、`charter`、`euler`、`helvet`、`lmodern`、`mathtime`、`mathptm`、`mathptmx`、`newcent`、`palatino`、`pifont`、`utopia`。

18.2.3 选择字体的编码

相同的字体⁶³可以进行不同的编码（encodings）⁶⁴，编码（粗略地讲）是文本的字符（characters）⁶⁵映射为字形（are mapped to glyphs）（即特定尺寸的特定字体的特定字符的实际形状）的一个途径。在 \TeX 中，有两种编码常用于拉丁字符（Latin characters）：T1 编码（T1 encoding）和 OT1 编码（旧 T1 编码，OT1 encoding）。

理论上，新 T1 编码（newer T1 encoding）比旧 T1 编码（old OT1 encoding）更好。例如，包含元音变化（umlauts）（如著名的德语单词 *Fräulein*）的联用词（hyphenation of words）只有用 T1 编码时才正常。不幸的是，属于 T1 编码的计算机现代字体 EC 字体（EC fonts），在小安装（small installs）中是作为 MetaFont 源（MetaFont sources）分发的，EC 字体也只有每一字形（glyph）的位图校正（bitmap rendition）。正是这个原因，在这样的小安装中使用 T1 编码的 EC 字体生成的 PDF 文件在渲染时是很糟糕的（poorly）。

可以以不同的完备程度安装 \TeX Live（跨平台，取代旧的 `teTeX` 用于 UNIX/Linux）和 `MiKTeX`（用于 Windows 平台）。可以安装与计算机现代字体相关的下列宏包：`cm-super` 字体、`lmodern`（Latin Modern）字体、`lgc` 字体，后者包含拉丁、希腊、西里尔（Cyrillic）字母。其它有关的字体宏包，如 `txfonts` 和 `pxfonts`，它们是 Times 和 Palatino PostScript 字体的两个扩展集（extended sets），这两个字体宏包包含数学字形（mathematical glyphs）的扩展集。大部分其它标准的 PostScript 字体也可以用 T1 编码。

在能使 T1 编码的计算机现代字体可用的宏包中，建议使用 `lmodern` 宏包。如果使用该宏包，则可以使用多个额外的字体（如无衬线的粗体数学字体）和额外的符号（如特有的海鸠符号）。

要选用 T1 编码，只需使用 `\usepackage[T1]{fontenc}` 命令。因此，如果我们安装了 LM 字体，就可使用下面的命令获得漂亮的轮廓字体（outline fonts）并能正确的用连字符连接：

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

然而，应注意，旧版本的 LM 包（LM bundle）不包含用于连字（ligatures）如“fi”的正确字形（correct glyphs），这会带来麻烦。仔细检查所有的连字是否显示正确，如不正确，可以更新我们的安装。

上述内容也适应于 `pdflatex` 和 `latex+dvips`。和前面的引擎不同，`xelatex` 和 `lualatex` 支持 OpenType 字体，这意味着在我们的文档中使用系统字体（system fonts）变得相对容易。在该手册的最后会阐述相关的细节。眼下，我们可以浏览一下 `fontspec` 宏包的文档，`fontspec` 宏包也支持 `xelatex` 和 `lualatex` 引擎。也应注意，当 `lualatex` 或 `xelatex` 和 EU2 或 EU1 编码一同使用时，默认情况下我们会获得各自的 OpenType 拉丁现代字体（Latin Modern fonts）。

18.3 改变演示稿不同元素的字体

这一节将阐述 BEAMER 的字体管理器（font management）是如何工作的。

⁶³字体（font），又称书体，是指文字的风格式样、体式，如汉字有篆书、隶书、楷书、草书、行书等。

⁶⁴字符是可使用多种不同字符方案或代码页来表示的抽象实体，例如，Unicode UTF-16 编码将字符表示为 16 位整数序列，而 Unicode UTF-8 编码则将相同的字符表示为 8 位字节序列。UTF-8、UTF-16 就是编码之一。

⁶⁵字符（characters）是指计算机中使用的字母、数字、字和符号，包括：1、2、3、A、B、C、!、·、#、¥、%、……、-、*、（）、-、+ 等等。

18.3.1 Beamer 的字体管理概述

BEAMER 的字体机制 (font mechanism) 有点象 BEAMER 的颜色机制 (color mechanism), 便不完全相同。对于颜色 (colors), 每一 BEAMER 元素如帧标题、文档标题、脚注等都有一个特定的 BEAMER-font。而且可以分别指定各个元素的字体。另一方面, 字体也可以使用继承 (inheritance), 因此, 很容易使用“象标题的东西 (titlelike things)”或“象条目的东西 (itemizelike things)”全局地改变字体。

虽然 BEAMER-color 拥有特定的前景 (foreground) 和背景 (background), 但其中之一可以为空 (empty)。BEAMER-font 拥有尺寸 (size)、形状 (shape)、衬线 (series)、字簇 (family), 其中的每一个都可以为空。beamer-fonts 中的继承关系不是必须的, 这和 BEAMER-colors 不同, 虽然我们设法使它们相配。

多重继承 (Multiple inheritance) 在字体 (fonts) 中扮演重要角色, 在颜色 (colors) 中却不是这样。一个字体可以继承两个不同字体的属性。如果一个字体指定粗体 (boldface) 而其它字体指定了大 (large) 的属性, 则子字体 (child font) 会显示为大和粗体。

对于字体, 一个元素 (element) 使用的字体的描述 (description) 会在该元素的描述之后给出。

18.3.2 使用 Beamer 的字体

要使用 BEAMER-font, 只需使用 `\usebeamerfont` 命令。在每一元素的模板内部, 已经调用了 `\usebeamerfont` 命令, 因此我们不会经常使用该命令。

```
\usebeamerfont*{<beamer-font name>}
```

即:

```
\usebeamerfont*{<beamer-font 名>}
```

该命令将当前字体 (current font) 更改为由 `<beamer-font name>` 指定的字体。`<beamer-font name>` 可以是不太古怪 (not-too-fancyfu) 的文本, 它也可以包含空隔 (spaces), 例如, 可以是 `frametitle` 或 `section in toc` 或 `My Font 1`。BEAMER-fonts 可以 (也必需) 具有与 BEAMER-templates 和 BEAMER-colors 一样的名称。

举例: `\usebeamerfont{frametitle}` 在该命令的不带星号的 (unstarred) 版本中, 会根据 `<beamer-font name>` 指定的属性更改字体, 但未指明的 (unspecified) 属性保持不变。例如, 如果只指定了必需用“粗体 (bold)”而没有指定其它属性, 这时如果当前的字体是大的 (large), 那么, 命令 `\usebeamerfont` 将使当前的字体变成大和粗体 (large and bold)。

在该命令的带星号的 (starred) 版本中, 在应用字体属性前会重置 (reset) 字体。因此, 上述 BEAMER-font 的例子只会将字体设为“粗体 (boldface)”, 声明 `\usebeamerfont*` 常常将当前字体设置成普通尺寸 (normal-size)、普通形状 (normal-shape)、粗体 (bold)、默认字族 (default-family) 的字体。

18.3.3 设置 Beamer 字体

changing BEAMER-fonts. 对于 BEAMER-colors, 有一个核心命令 (central command) 用于设置和更改 BEAMER-fonts, 即:

```
\setbeamerfont*{<beamer-font name>}{<attributes>}
```

即:

`\setbeamerfont*{<beamer-font 名>}{<属性>}`

该命令用于设置或重置 BEAMER-font *<beamer-font name>* 的特定属性。在不带星号的 (unstarred) 版本中, 该命令只添加那些在前一调用 (previous call) 中未提及的属性。因此, 下面的两命令块 (command blocks) 具有相同效果:

举例:

```
\setbeamerfont{frametitle}{size=\large}
\setbeamerfont{frametitle}{series=\bfseries}
```

```
\setbeamerfont{frametitle}{size=\large,series=\bfseries}
```

在带星号的 (starred) 版本中, 会首先完全地重置字体属性, 也就是说, 将字体属性重置为空 (empty)。

可能会给出下面的 *<attributes>*:

- **size=<size command>** 设置 BEAMER 字体的尺寸属性 (size attribute)。*<size command>* 必需是用于设置字体尺寸或为空的标准的 L^AT_EX 命令。可用的命令有: `\tiny`、`\scriptsize`、`\footnotesize`、`\small`、`\normalsize`、`\large`、`\Large`、`\huge`、`\Huge`。BEAMER 也采用 (introduce) 两个字体尺寸命令 `\Tiny` 和 `\TINY`, 这两个命令用于设置实在小的 (really small) 文本, 提醒一下我们, 当使用这两个命令时, 我们必须确切地知道我们正在干什么。
注意, 指定一个空命令 (empty command) 及指定 `\normalsize` 二者的区别: 指定尺寸属性为“空 (empty)”意味着当使用该字体时不会改变字体的尺寸; 而指定 `\normalsize` 则意味着不管何时使用该字体, 都会将字体的尺寸设置为普通尺寸 (normal size)。
- **size*={<size in pt>}{<baselineskip>}** 将字体的尺寸属性设置为给定的 *<size in pt>*, 将行间距 (baseline skip) 设置为给定的值。注意, 据我们所使用的字体类型 (kind of font), 不是所有的字体尺寸均可用。而且, 特定的 (certain) 字体的尺寸并不比其它的字体尺寸更称心如意; 标准的命令已经为我们选好了合适的字体尺寸。除非我们有很好的理由, 否则不要使用该选项。该命令与 `size={\fontsize{<size in pt>}{<baselineskip>}}` 等效。
- **shape=<shape command>** 设置字体的形状属性 (shape attribute)。这里的命令必须是象 `\itshape`、`\slshape`、`\scshape`、`\upshape` 这样的命令。
- **shape*={<shape attribute abbreviation>}** 用 L^AT_EX 的属性缩写 (abbreviations for attributes) 设置字体的形状属性。该命令与 `shape={\fontshape{<shape attributes abbreviation>}}` 等效。
- **series=<series command>** 设置字体的“衬线 (series)”属性。这里的命令必须是象 `\bfseries` 这样的命令。
- **series*={<series attribute abbreviation>}** 和 `series={\fontseries{<series attributes abbreviation>}}` 等效。
- **family=<family command>** 设置字族 (family) 属性。这里的命令必须是象 `\rmfamily` 或 `\sffamily` 这样的命令。
- **family*={<family name>}** 将字族属性设置为给定的 *<family name>*。该命令和 `family={\fontfamily{<family name>}}` 等效。通常, *<family name>* 是安装在我们系统中的字族名 (font family name) 的缩写 (abbreviation) 形式, 它有几分神秘 (cryptic)。例如, Times 的 *<family name>* 是 `ptm`。没有人能记住这些名字, 所以辛苦地查阅它们是很正常 (perfectly normal) 的事。

- `parent={\{parent list}}` 指定父本字体 (parent fonts) 列表。当使用 BEAMER-font 时, 会首先使用其父本 (parents)。因此, 由一个父本设置的任何字体属性, 都会被 BEAMER-font 继承, 除非该属性被字体覆盖 (overwrite)。

举例:

```
\setbeamerfont{parent A}{size=\large}
\setbeamerfont{parent B}{series=\bfseries}
\setbeamerfont{child}{parent={parent A, parent B},size=\small}
```

```
\normalfont
This text is in a normal font.
\usebeamerfont{parent A}
This text is large.
\usebeamerfont{parent B}
This text is large and bold.
\usebeamerfont{parent B}
This text is still large and bold.
\usebeamerfont*{parent B}
This text is only bold, but not large.
\usebeamerfont{child}
This text is small and bold.
```

部分 IV

创建配套的材料

BEAMER 文档类的目的是简化创建用投影机放映的演示稿 (presentation) 的过程。然而，演示稿常常不是独立的，它的配套材料包括：

- 演示稿 (presentations) 常常附带有讲义 (*handout*)，讲义是文本形式的，在演讲时和/或演讲后，观众可以阅读它。
- 我们可能希望做些笔记 (note)，当观众看着演示稿时，我们则看着我们电脑中相应的笔记。
- 我们可能希望将我们的演讲内容打印出来，或是为自己看或是为了检查错误。
- 我们可能希望创建演讲的胶片版本 (transparencies version) 以方便回放 (fall-back)。

这部分将讨论 BEAMER 是如何帮助我们实现以上目的。

19 为自己添加笔记

笔记 (*note*) 是这样的文本，它用于在放映幻灯片时提醒我们接下来要说什么或要记住什么。笔记常打印在纸上，当然，如果支持双屏显示，在放映机 (projector) 中放映主演示稿 (main presentation) 时，在便携机 (laptop) 上则可以显示笔记。

19.1 指定笔记的内容

要在幻灯片或帧上添加笔记，可以使用 `\note` 命令。该命令可以用在帧内，也可以用在帧外，但二者的行为却不同：用在帧内时，会在当前幻灯片的后面累加 (accumulate) 和追加 (append) 一个单独的笔记页 (note page)；用在帧外时，每一 `\note` 命令则会直接插入一个单独的带有给定参数的笔记页，并将其作为正文内容。在帧内使用 `\note` 命令比在帧外使用该命令更好，因为只有帧内的 `\note` 命令才能从文档类选项 `onlyslideswithnotes` 中获益，请看下面。

LYX 在 LyX 中，在 NoteItem 风格 (NoteItem style) 中，只有帧内的带有 [item] 选项的 `\note` 命令才可用。

在帧内，`\note<text>` 命令的效果如下：在特定幻灯片在帧内使用该命令时，会在该幻灯片的后面创建一个笔记页 (note page)，该笔记页包含了 `<text>`。因为可以在 `\note` 命令中添加叠层规则 (overlay specification)，我们可以指定在哪一幻灯片后面显示笔记。如果在一张幻灯片中使用了多个 `\note` 命令，它们会“累加 (accumulate)”并显示于同一笔记。

为使笔记累加 (accumulation of notes) 更方便，我们可以使用带有 [item] 选项的 `\note` 命令。带有 [item] 选项的笔记会累加到一个 `enumerate` 列表中，该列表跟有用 `\note` 命令插入的文本。

下面的例子会生成一个笔记页，该笔记页有两个条目 (entries)，出现在第二张幻灯片后面。

```
\begin{frame}
  \begin{itemize}
    \item<1-> Eggs
    \item<2-> Plants
      \note[item]<2>{Tell joke about plants.}
      \note[item]<2>{Make it short.}
    \item<3-> Animals
  \end{itemize}
\end{frame}
```

在帧外，`\note` 命令创建单独的笔记页 (note page)。这“不依赖”在前一帧是否使用了 `\note` 命令。如果我们在一个帧内声明了 `\note` 命令，在其后又跟有一个 `\note` 命令，则会创建两个 (*two*) 笔记页。

下面讲述帧内 `\note` 命令的语法及效果：

```
\note<<overlay specification>>[<options>]{<note text>}
```

即：

```
\note<<叠层规则>>[<选项>]{<笔记文本>}
```

帧内的效果：该命令追加 `<note text>` 到笔记，该笔记紧跟在当前幻灯片之后。在一张幻灯片中多次使用该命令会产生累积 (accumulate) 效应。如果不指定 `<overlay specification>`，则会在当前帧的所有幻灯片中添加笔记。这可不是我们想要的，因此，添加像 `<1>` 这样的规则常常是个好主意。

可能会给出下面的 `<options>`：

- `item` 将笔记作为一个列表的条目放置于笔记页的末尾。

举例: `\note<2>{Do not talk longer than 2 minutes about this.}`

ARTICLE 在论文 (article) 模式中会忽略笔记。

LYX 使用 `NoteItem` style 插入笔记条目。

接下来, 讲述帧外 `\note` 命令的语法和效果:

`\note[<options>]{<note text>}`

即:

`\note[<选项>]{<笔记文本>}`

在帧外, 该命令创建一个笔记页。该命令不受 `notes=onlyframeswithnotes` 选项的影响, 请看下面。

可能会给出下面的 `<options>`:

- `itemize` 将整个笔记页围入到一个 `itemize` 环境中。这只是图方便 (This is just a convenience)。
- `enumerate` 将整个笔记页围入到一个 `enumerate` 环境中。

举例:

```
\frame{some text}
\note{Talk no more than 1 minute.}
```

```
\note[enumerate]
{
  \item Stress this first.
  \item Then this.
}
```

ARTICLE 在论文 (article) 模式中会忽略笔记。

下面的元素讲述如何生成 (render) 笔记页:

Beamer-Template/-Color/-Font note page

即:

Beamer-Template/-Color/-Font 笔记页

该模板用于排版笔记页。该模板应包含关于插入物 (insert) `\insertnote` 的陈述 (mentioning), 该陈述即包含笔记文本 (note text)。为了在笔记页中挤进更多内容, 我们应考虑将 `BEAMER-font note page` 的尺寸更改为更小 (something small)。默认为 `\small`。

下面的模板选项是预定好的:

- `[default]` 默认的模板显示右上角的最后一张幻灯片和一些“提示 (hints)”, 这些提示可以帮助我们使笔记页和当前显示的幻灯片相匹配。

- `[compress]` 该选项的输出和默认 (default) 选项的输出相似, 该选项为了更适合每一笔记页而牺牲了易读性 (legibility)。
- `[plain]` 只插入笔记文本 (note text), 不插入提示 (hints)。

下面的两个插入物 (inserts) 对笔记页很有用:

- `\insertnote` 将当前笔记的文本插入到模板。
- `\insertslideintonotes{<magnification>}` 将最后一张幻灯片的“缩略图 (mini picture)”插入到当前的笔记中。最后一张幻灯片将按给定的缩放比例 (magnification) 缩放。

举例: `\insertslideintonotes{0.25}`

这会生成一缩小的幻灯片, 它的宽和高都是正常尺寸的四分之一。

19.2 指定多个笔记的内容

有时, 我们可能希望在每一笔记或一系列笔记中的至少每一笔记中显示一些文本。要实现这一点, 我们可以使用下面的两条命令:

`\AtBeginNote{<text>}`

即:

`\AtBeginNote{<文本>}`

在该命令所属的每一笔记的起始处插入 `<text>`。要中止其效应, 可以使用 `\AtBeginNote{}` 命令, 也可以在 `TEX` group 内围入该区域。

建议在 `<text>` 末尾处添加 `\par` 命令或一个空行 (empty line), 要不然, 其它笔记文本 (note text) 会直接紧随 `<text>` 之后, 而两者之间却没有换行符 (line break)。

举例:

```
\section{My Section}
```

```
\AtBeginNote{Finish this section by 14:35.\par}
```

```
\begin{frame}
```

```
...
```

```
\note{some note}
```

```
\end{frame}
```

```
\begin{frame}
```

```
...
```

```
\note{some other note}
```

```
\end{frame}
```

```
\AtBeginNote{}
```

`\AtEndNote{<text>}`

即:

`\AtEndNote{<文本>}`

除在末尾 (底部) 插入文本外, 该命令的所作所为和 `\AtBeginNote` 相同, 我们可以在 `<text>` 的起始处添加一个 `\par`。

19.3 指定显示哪个笔记和帧

通常我们不希望笔记（note）成为演示稿（presentation）的一部分，如果我们要让演示稿包含笔记，我们就必须在导言区明确地声明这一点。这使用下面的 BEAMER 选项来实现：

```
\setbeameroption{hide notes}
```

不显示笔记。在演示稿中，这是默认的选项。

```
\setbeameroption{show notes}
```

在输出文件中包含笔记。输出文件也包含幻灯片，间插（interleaved）有笔记页。

```
\setbeameroption{show notes on second screen=<location>}
```

给定该选项后，会创建演示稿的双屏版本（two screen version），详细内容请参阅第 22 节。默认情况下，在放置于右侧（right）的第二屏幕上显示笔记，指定不同的 *location*，可以在左侧（left）、底部（bottom）、顶部（top）放置第二屏幕。

举例：

```
\documentclass{beamer}
\usepackage{pgfpages}
\setbeameroption{show notes on second screen}
\begin{document}
\begin{frame}
  A frame.
  \note{This is shown on the right.}
\end{frame}
\end{document}
```

具体点，会发生下面的事情：演示稿以正常（normally）的方式排版，并显示于主屏幕（main screen），更精确地讲，演示稿显示于 `pgfpages` 的逻辑页码 0（`pgfpages`'s logical page number zero）上。第二屏幕（逻辑页码 1）初始化为空（empty）。

不管什么时候排版笔记页，或者因为一个帧包含 `\note` 命令，或者因为一个帧后跟一个 `\note` 命令，笔记页都会以普通方式（normally）排版。然后笔记页将显示于第二屏幕。（基本过程是这样但实际的细节更复杂。）

该行为（behavior）的一个重要效应是后跟有一个帧的笔记页会显示于该帧之后。通常，这是我们希望。然而，如果一张幻灯片有多个笔记页，则当前只会显示最后一个笔记页。在将来这或许能有改进。

举例：

```
\begin{frame}
  First frame.
\end{frame}
\note{This note is not shown at all (currently).}
\note{This note is shown together with the first frame.}

\begin{frame}
```

```
Second frame.  
\note{This note is shown together with the second frame.}  
\end{frame}
```

```
\begin{frame}  
No note text is shown for this frame.  
\end{frame}
```

如果我们真正需要一张幻灯片具有多个笔记页，则可以使用像这样复杂的代码：

```
\begin{frame}<1-3>  
First frame.  
\note<1>{First page of notes for this frame.}  
\note<2>{Second page of notes for this frame.}  
\note<3>{Third page of notes for this frame.}  
\end{frame}
```

```
\setbeameroption{show only notes}
```

在输出文件中只包含笔记并抑制所有的帧。在打印笔记时这是很有用的。如果我们指定了该命令，则不会更新 `.aux` 和 `.toc` 文件。因此，如果我们添加一个节（section）并对演示稿执行 `reTeX`，添加的该节不会在导航条（Navigation Bar）中反映出来（无论如何我们都不会看见添加的节，因为只输出了笔记）。

20 创建胶片

BEAMER 文档类的主要目的是创建用投影机（有时称作 beamers，于是有了这个名字）放映的演示稿（presentations）。然而，以防硬件损坏，将演示稿打印成胶片（transparencies）作为备份（backup）很有好处。演讲的胶片版本（transparencies version）通常比主版本（main version）具有更少的幻灯片（slides），因为切换幻灯片耗时更多。但演讲的胶片版本又比讲义版本（handout version）具有更多的幻灯片。例如，在一个讲义中，动画将浓缩进一张幻灯片中，而在胶片版本中会打印成多张幻灯片。

为了创建胶片版本，可以指定文档类的 `trans` 选项。如果我们没有指定别的其它选项，则会抑制所有的叠层规则（overlay specifications）。大多数实例说明这会产生预期的结果。

```
\documentclass[trans]{beamer}
```

创建使用 `trans` 叠层规则的胶片版本。

在某些情况下，我们可能想要更复杂的行为（complex behavior），例如，使用多个 `\only` 命令创建动画（animation）。既然这样，抑制所有叠层规则不是个好主意，因为这样会在同一时间显示动画的每一步。在某些情况下，也不期望这样。而且，可能希望抑制一些 `\alert` 命令，这些命令与讲义中指定幻灯片有关。

为精细控制在讲义中显示什么，我们可以使用 模式规则（*mode specifications*）。它指定在特定的版本（special version）如讲义版本中显示帧的哪些幻灯片。下如第 9.2 节中讲述的那样，模式规则（mode specification）写在紧靠普通叠层规则（normal overlay specification）的尖括号中，模式规则通过竖条（vertical bar）和一个空格与普通规则分开。例如：

```
\only<1-3,5-9| trans:2-3,5>{Text}
```

该规则声明：“通常情况下（在 `beamer` 模式中），在第 1-3 和第 5-9 张幻灯片中插入文本。在胶片版本中，在第 2、3、5 张幻灯片中插入文本”。如果没有为 `trans` 模式给出特定的模式规则，则“总是（always）”使用默认的模式规则。如果我们没有指定任何东西，这会产生预期的结果，会为讲义有效地抑制叠层规则。

下面的叠层规则特别有用：

```
\only<3| trans:0>{Not shown on transparencies.}
```

因为没有第 0 张幻灯片，所以不会显示文本。同理，`\alert<3| trans:0>{Text}` 不会在胶片中显示提醒文本（alert the text）。

我们也可以为 `{frame}` 环境的叠层规则使用模式规则，如下例所示：

```
\begin{frame}<1-| trans:0>
  Text...
\end{frame}
```

这会抑制胶片版本中的帧。而且，我们可以限定演示稿，这样就可以在讲义中只显示帧的特定幻灯片：

```
\begin{frame}<1-| trans:4-5>
  Text...
\end{frame}
```

可能只会给出一个替代叠层规则（alternate overlay specification），例如，`\alert<trans:0>{...}` 会使文本在整个演示稿中一直高亮显示，但在胶片版本中从不显示。同理，`\frame<trans:0>{...}` 会抑制讲义中的帧。

最后，应注意可能会给出不止一个替代叠层规则，它们的顺序不定。例如，下面的规则表明：对于演示稿（presentation）版本，在前面的三张幻灯片中插入文本；对于胶片（transparency）版本，在前面两张幻灯片中插入文本；对于讲义（Handout）版本，根本就不插入文本。

```
\only<trans:1-2| 1-3| handout:0>{Text}
```

如果我们想在所有版本中给出相同的规则，则可以将版本指定为 `all:`，如下所示：

```
\frame<all:1-2>{blah...}
```

必须确保在所有版本中帧有两张幻灯片。

21 创建讲义和演讲记录

在演讲过程中，观众如有一份演讲稿的讲义 (*handout*) 或演讲记录 (*lecture notes*)，他们会感到称心如意 (*desirable*)。一份讲义可以让每一位观众返回去解决其各自的不懂的问题。

通常应尽可能早地提供讲义，甚至是演讲前的几个星期。不要等到演讲结束了才将讲义拿出来。

BEAMER 提供了两种不同的方法创建讲义，下面对它们进行讨论。第一种更简单，只需添加 `handout` 选项，该选项以讲义模式 (`handout`) 排版文档。这“看起来像”演示稿 (`presentation`)，但更容易打印 [因为叠层已被“弄平 (`flattened`)”]。第二种更复杂，却是更强大的方法，它可以创建演示稿的独立的“论文 (`article`)”版本。该版本共存于我们的主文件 (`main file`)。

21.1 使用讲义模式创建讲义

创建讲义容易的（但不是称心的）方法是使用 `handout` 选项。该选项的所作所为与 `trans` 选项相像。在 `beamerexample1.tex` 文件中有演示稿的不同叠层规则的详尽的例子、讲义、胶片 (`transparencies`) 等。

```
\documentclass[handout]{beamer}
```

创建使用 `handout` 叠层规则的版本。

我们可能为讲义选用不同的颜色和/或演示稿主题。

当打印以此方法创建的讲义时，我们可能希望在一个页面上打印至少两张甚至四项幻灯片 (`slides`)，这种容易的方法可能和使用 `pgfpages` 有关，如下所示：

```
\usepackage{pgfpages}
\pgfpagesuselayout{2 on 1}[a4paper,border shrink=5mm]
```

用 `4 on 1` 替换 `2 on 1`（但必须将 `landscape` 添加到选项列表），我们就可使用 `letterpaper` 代替 `a4paper`。

21.2 使用论文模式创建讲义

下面，演示稿的“论文版本 (`article version`)”涉及普通的 $\text{T}_\text{E}\text{X}$ 文本，这些普通的 $\text{T}_\text{E}\text{X}$ 文本使用了文档类 `article` 或 `llncs` 或相似的文档类进行排版。演示稿的论文版本遵循不同的排版规则，甚至有不同的结构。然而，我们可能希望演示稿和其论文版本共存于一个文件，或者希望演示稿共享其论文版本的一部分（如图形和公式）。

一般而言，演讲的论文版本和讲义更相配，而和使用简单讲义模式 (`handout mode`) 创建的讲义更不相配，因为论文版本更经济并能包含更多更深的信息。

21.2.1 开始论文模式

要进入演示稿的论文模式，只需将 `beamer` 文档类替换成 `article` 或 `book` 或其它文档类，然后加载 `beamerarticle` 宏包。

`beamerarticle` 宏包定义了 BEAMER 所有的命令，这些命令在论文模式中能被感知 (`sensible`)。而且，一旦加载了 `beamerarticle` 宏包，叠层规则 (`overlay specifications`) 就可以添加给象 `\textbf` 或 `\item` 这样的命令。注意，除 `\item` 外，这些叠层规则有这样的作用：在论文版本 (`article version`) 中，`\section<presentation>{Name}` 会抑制这个节命令 (`section command`)。要了解论文模式中叠层规则的更确切的功能，请参阅我们使用的命令的说明 (`descriptions`)。

```
\usepackage[<options>]{beamerarticle}
```

即:

```
\usepackage[<选项>]{beamerarticle}
```

使 BEAMER 的大部分命令在其它文档类中可用。

可能会给出下面的 *<options>*:

- **activeospeccharacters** 由其它宏包指定尖括号 (pointed brackets) 的字符代码 (character code)。通常, BEAMER 会关闭 `<` 和 `>` 这两个字符的特定行为 (special behavior)。使用该选项, 我们可以在论文模式中重新安装初始行为 (original behavior), 但当使用叠层规则时这可能带来问题。
- **noamssymb** 会抑制自动加载 `amssymb` 宏包。通常, BEAMER 会加载该宏包, 因为许多主题 (theme) 使用 AMS 符号。在论文模式 (article mode) 中该选项我们从这个行为中退出 (opt-out), 这样, 可以避免与一些类和字体宏包产生冲突, 这些类和字体宏包与 `amssymb` 冲突。注意, 如果使用了该选项, 在使用了各自 (respective) 的符号时, 我们必须加载 `amssymb` 或可选的 (alternative) 宏包。
- **noamsthm** 会抑制加载 `amsthm` 宏包。因此不会定义定理 (theorems)。
- **notheorem** 会抑制像 `theorem` 这样的标准环境的定义 (definition), 但仍会加载 `amsthm` 宏包, 而且 `\newtheorem` 命令使已定义的环境可以接受叠层规则 (overlay-specificationaware)。使用该选项允许我们以我们喜欢的任何方式定义标准环境, 但会保留 `amsthm` 宏包的扩展能力 (the power of the extensions to `amsthm`)。
- **envcountsect** 用每一节的编号给定理 (theorem)、定义 (definitions) 和类似物 (the like) 编号。因此, Theorem 1 会变成 Theorem 1.1。推荐使用该选项。
- **noxcolor** 会抑制加载 `xcolor` 宏包。因此不会定义颜色 (colors)。

举例:

```
\documentclass{article}
\usepackage{beamerarticle}
\begin{document}
\begin{frame}
  \frametitle{A frame title}
  \begin{itemize}
\item<1-> You can use overlay specifications.
\item<2-> This is useful.
  \end{itemize}
\end{frame}
\end{document}
```

这会留下一个问题: 虽然论文版本可以轻而易举地 \TeX 整个文件, 甚至是在像 `\frame<2>` 这样的命令面前, 但我们也不想原始的 BEAMER 演示稿中插入特定的论文文本 (the special article text)。这意味着, 我们不希望帧之间的所有文本被抑制。更精确地讲, 我们希望抑制除像 `\section` 这样的命令外的所有文本。这个行为可以通过在演示稿版本 (presentation version) 中指定 `ignorenonframetext` 选项而被强制实施。该选项会在演示稿的开始处插入一个 `\mode*`。

下面的例子显示了论文模式的一个简单用法:

```

\documentclass[a4paper]{article}
\usepackage{beamerarticle}
%\documentclass[ignorenonframetext,red]{beamer}

\mode<article>{\usepackage{fullpage}}
\mode<presentation>{\usetheme{Berlin}}

% everyone:
\usepackage[english]{babel}
\usepackage{pgf}

\pgfdeclareimage[height=1cm]{myimage}{filename}

\begin{document}

\section{Introduction}

This is the introduction text. This text is not shown in the
presentation, but will be part of the article.

\begin{frame}
  \begin{figure}
    % In the article, this is a floating figure,
    % In the presentation, this figure is shown in the first frame
    \pgfuseimage{myimage}
  \end{figure}
\end{frame}

This text is once more not shown in the presentation.

\section{Main Part}

While this text is not shown in the presentation, the section command
also applies to the presentation.

We can add a subsection that is only part of the article like this:

\subsection<article>{Article-Only Section}

With some more text.

\begin{frame}
  This text is part both of the article and of the presentation.
  \begin{itemize}
\item This stuff is also shown in both version.

```



```

\item This too.
  \only<article>{\item This particular item is only part
    of the article version.}
\item<presentation:only@0> This text is also only part of the article.
  \end{itemize}
\end{frame}
\end{document}

```

换行符（line break）`\\` 的行为在论文模式中有点特别。在论文模式中，除非出现了一个叠层规则，否则帧内的换行符无效。在演示稿（presentation）中我们可能会插入大量的 `\\` 命令以控制换行，但在论文模式中，这样的换行符大多数是多余的（superfluous）。如果我们想在所有版本中使换行符有效，请声明 `\\<all>`。注意，某些环境常会重定义 `\\` 命令，因此 `\\` 命令就不一定可以添加叠层规则了。这种情况下，我们不得不写下这样的语句：`\only<presentation>{\\}`。

21.2.2 工作流程

下面的工作流程（workflow）是可选的，但它们可以简化论文版本的创建过程。

- 在主文件 `main.tex` 中，删除第一行，该行用于设置文档类。
- 创建一个名为 `main.beamer.tex` 的文件，该文件包含下面的内容：

```

\documentclass[ignorenonframetext]{beamer}
\input{main.tex}

```

- 创建一个名为 `main.article.tex` 的额外文件（extra file），该文件包含下面的内容：

```

\documentclass{article}
\usepackage{beamerarticle}
\setjobnamebeamerversion{main.beamer}
\input{main.tex}

```

- 现在可以对上述两个文件 `main.beamer.tex` 和 `main.article.tex` 执行 `pdflatex` 或 `latex`。

`\setjobnamebeamerversion` 命令告诉论文版本在哪里可以找到演示稿版本。如果我们希望将来自于演示稿版本中的幻灯片以图片（figures）的形式插入到论文（article）中，则该命令是必需的。

```

\setjobnamebeamerversion{\filename without extension}

```

即：

```

\setjobnamebeamerversion{无扩展名的文件名}

```

告诉 BEAMER 文档类在哪里可以找到当前文件的演示稿版本。

21.2.3 在论文版本中包含来自演示稿的幻灯片

如果我们使用了 `beamerarticle` 宏包，在论文模式中就可以调用 `\frame` 命令。通过调整帧模板 (frame template)，在我们的论文中就可以“模拟 (mimic)”用 BEAMER 排版的帧的外观。然而，有时我们可能希望在论文版本中插入“真正的幻灯片”，即来自演示稿幻灯片的精确“截屏 (screenshot)”。下面的命令有助于我们实现上述目标。

要在我们的论文版本中包含来自演示稿的幻灯片，必须做两件事：第一件，用 `\label` 命令在幻灯片中放置一个标准的 L^AT_EX 标签。因为该命令可以添加叠层规则，所以我们就可以选择帧的特定幻灯片。而且，通过给帧添加 `label=<name>` 选项，帧的每一张幻灯片就会自动添加 `<name><slide number>` 标签。

第二件，一旦我们标签了 (have labeled) 一张幻灯片，就可以用下面的命令在我们的论文版本中插入该幻灯片：

```
\includeslide[<options>]{<label name>}
```

即：

```
\includeslide[<选项>]{<标签名>}
```

该命令为一个文件调用带有 `<options>` 的 `\pgfimage` 命令，这个文件由下面的命令指定：

```
\setjobnamebeamerversion<filename> 即：\setjobnamebeamerversion(文件名)
```

此外，`page=<page of label name>` 选项会传递给 `\pgfimage`，在这里，`<page of label name>` 由后台读取自 `<filename>.snm` 文件。

举例：

```
\article
\begin{figure}
  \begin{center}
    \includeslide[height=5cm]{slide1}
  \end{center}
  \caption{The first slide (height 5cm). Note the partly covered second item.}
\end{figure}
\begin{figure}
  \begin{center}
    \includeslide{slide2}
  \end{center}
  \caption{The second slide (original size). Now the second item is also shown.}
\end{figure}
```

有关传递 `page=<page of label name>` 选项给 `\pgfimage` 命令的确切作用请参阅 `pgf` 的文档。实际上会发生下面的事情：

- 对于旧版本的 `pdflatex` 和任何版本的 `latex` 连同 `dvips`，`pgf` 宏包会查找一个文件，其名为：

`<filename>.page<page of label name>.<extension>` 即： `<文件名>.page<标签名的页>.<扩展名>`

对于以这种方式被包含的 .pdf 或 .ps 文件的每一页，我们必须手工创建这样一个文件。例如，如果演示稿版本的 PostScript 文件名为 main.beamer.ps，要包含 main.beamer.ps 文件的第 2、第 3 页（即第 2、第 3 张幻灯片），则必须“手工”（或通过脚本）创建（单页的）main.beamer.page2.ps 和 main.beamer.page3.ps 文件，如果 pgf 找不到这样的单页文件，它是会抱怨（complain）的。

- 对于旧版本的 pdf_latex，pdf_latex 也会根据上述的命名方法（naming scheme）查找相应的文件。然而，如果查找不成功（因为我们没有创建这样的文件），就会用特定的机制从 main.beamer.pdf 这个演示稿文件中直接解压从而得到所需的单个页面。

21.3 模式的相关细节

这一小节阐述模式（modes）是如何工作的、使用 \mode 命令我们怎样才能控制哪部分文本属于哪种模式。当 BEAMER 进行排版时，它总是使用下列五种模式之一：

- **beamer** 是默认的模式。
- **second** 当为可选的第二屏（second screen）排版一张幻灯片时使用该模式。
- **handout** 当创建讲义（Handout）时使用该模式。
- **trans** 当创建胶片（transparencies）时使用该模式。
- **article** 当控制器（control）切换（transfer to）成其它文档类如 article.cls 时使用该模式。注意，如果控制器切换成 book.cls 时，模式仍为 article。

除上述模式外，BEAMER 能识别出下列名称的模式设置（modes sets）：

- **all** 适应于所有模式。
- **presentation** 适应于前面的四种模式，也就是说，适应于除 article 模式的其它四种模式。

依据当前的模式，我们可希望只在该模式中插入特定的文本。例如，我们可能希望在论文版本中忽略（leave out of）特定的帧或特定的表格（table）。某些情况下，我们可以使用 \only 命令达到该目的。然而，下述的 \mode 命令比 \only 命令强悍多了。

\mode 命令有三种“风味（flavors）”，它们的语法略有不同。第一种，也是最简单的一种，只带有一个参数（argument）。它的所作所为与 \only 命令相同。

```
\mode<<mode specification>>{<text>}
```

即：

```
\mode<<模式规则>>{<文本>}
```

只为指定的模式插入 <text>。一个 <mode specification> 仅仅是一个不提及（mention）幻灯片的叠层规则（overlay specification）。

<text> 不可以做花俏的（fancy）事情如模式切换（mode switches）或包含其它文件。尤其我们不能在 <text> 里面放置 \include 命令。但可以在 <text> 里面使用参数自由（argument-free）的命令。

举例：

```
\mode<article>{Extra detail mentioned only in the article version.}
```

```
\mode  
<beamer|trans>  
{\frameof{tableofcontents[currentsection]}}
```

第二种风味的 `\mode` 命令不带参数 (No argument)。“不带参数”意味着其后不跟有左大括号 (opening brace)，但跟有其它符号。

```
\mode<(mode specification)>
```

即：

```
\mode<(模式规则)>
```

在指定的模式中，该命令其实没有什么作用 (effect)。有趣的部分是在非指定的 (nonspecified) 模式的作用：在这些模式中，该命令使 `TEX` 进入一种“忽略 (gobbling)”状态。它会忽略后面所有的行，直到下一行出现下列命令之一：`\mode`、`\mode*`、`\begin{document}`、`\end{document}`。即使这一行的注释都会使 `TEX` 跳过它。注意，带有使 `TEX` 结束忽略状态的特定命令不应直接跟在这样的行（在这一行开始忽略状态）的后面。相反 (Rather)，在这样的特定的命令之前必须要有一个非空的行 (non-empty line) 或两个空行 (empty lines)。

当 `TEX` 遇到单个的 `\mode` 命令时，它会执行该命令。如果 `\mode` 命令是第一种风味的，`TEX` 将在插入 (或不插入) `\mode` 命令的参数后收回 (resume) 其“忽略 (gobbling)”状态。如果 `\mode` 命令是第二种风味的，则 `TEX` 继续 (takes over) 其“忽略 (gobbling)”状态。

使用第二种风味的 `\mode` 命令不如第一种风味的便利 (convenient)，但以下的原因让我们使用第二种风味的 `\mode` 命令：

- 第二种风味的行智能忽略 (line-wise gobbling) 比第三种风味的忽略 (gobble) 更快 (faster)，请参考下面的解释。
- 第一种风味的 `\mode` 命令会完全地 (completely) 读入其参数。这意味着，它不能包含含有不匹配大括号 (unbalanced braces) 的逐字显示的文本 (Verbatim Text)。
- 第一风味的 `\mode` 命令不能处理含有 `\include` 的参数。
- 如果文本 (text) 主要属于一种模式，该模式插入有来自其它模式的少量文本，那么，第二种风味的命令就很值得使用。

Note: 当 `TEX` 进行行智能查找 (searching line-wise) 以让 `\mode` 命令摆脱忽略状态，如果同一行跟有一个模式规则 (mode specification)，`TEX` 将无法识别 `\mode` 命令。因此，这样一个模式规则必须在下一行给出。

Note: 当一个 `TEX` 文件结束时，不一定处于忽略状态。可以在一行中使用 `\mode` 命令并在下一行使用 `<all>`。

举例：

```
\mode<article>
```

This text is typeset only in |article| mode.

```

\verb!verbatim text is ok {!

\mode
<presentation>
{ % this text is inserted only in presentation mode
\frame{\tableofcontents[currentsection]}}

```

Here we are back to article mode stuff. This text is not inserted in presentation mode

```

\mode
<presentation>

```

This text is only inserted in presentation mode.

第三种风味的 `\mode` 命令的行为有很大的不同。

`\mode*`

该模式的作用是忽略 `presentation` 模式中帧外的所有文本。在 `article` 模式中该命令无作用。

该模式只进入（be entered）帧外。一旦进入，如果当前模式是 `presentation` 模式，`TEX` 将进入一种忽略状态，这种忽略状态与第二种“风味”的 `\mode` 命令的忽略状态相似。不同之处是这时的文本是以标记智能（token-wise）读取的，而非行智能（line-wise）读取。会按标记（token）忽略文本，直到出现下列标记之一：`\mode`、`\frame`、`\againframe`、`\part`、`\section`、`\subsection`、`\appendix`、`\note`、`\begin{frame}`、`\end{document}`（最后两个是真正的标记，无论如何它们都能被认出）

一旦进入了这些命令之一，则会停止忽略（gobbling）并执行该命令。无论如何，当启动这些命令时，所有这些命令恢复（restore）了有效的模式（mode that was in effect）。这样，一旦这些命令结束，`TEX` 返回到其忽略状态。

通常，`\mode*` 知道我们想让 `TEX` 在帧外做什么：忽略所有东西，除 `presentation` 模式中帧外的上述提及的命令。然而，下面这些场合我们必须用第二种风味的 `\mode` 命令代替：含有上述命令之一的逐字显示文本（Verbatim Text）、帧外很长的文本、一些帧外文本（如一个定义）也能在 `presentation` 模式中被执行。

文档类选项 `ignorenonframetext` 会在文档开始处开启 `\mode*`。

举例：

```

\begin{document}
\mode*

```

This text is not shown in the presentation.

```

\begin{frame}
  This text is shown both in article and presentation mode.
\end{frame}

```

this text is not shown in the presentation again.

```
\section{This command also has effect in presentation mode}
```

Back to article stuff again.

```
\frame<presentation>
{ this frame is shown only in the presentation. }
\end{document}
```

举例：下面的例子显示了如何在一个主文件（main file）中包含其它文件。一个 `main.tex` 的内容如下：

```
\documentclass[ignorenonframetext]{beamer}
\begin{document}
This is star mode stuff.
```

Let's include files:

```
\mode<all>
\include{a}
\include{b}
\mode*
```

Back to star mode

```
\end{document}
```

和 `a.tex`（和同样的 `b.tex`）：

```
\mode*
\section{First section}
Extra text in article version.
\begin{frame}
  Some text.
\end{frame}
\mode<all>
```

22 多屏显示的好处

本节阐述由 BEAMER 提供的选项，使用该选项后可以使电脑（computers）有多个视频输出（video output），这些视频输出具有不同的内容。对于这样一个系统，第一个视频输出对应于一个投影机（projector），在此投影机上显示主演示稿（main presentation）；第二个视频输出对应于一个小的额外的显示器（a small extra monitor）（或者就显示于电脑的屏幕上），在这里显示例如我们自己用的笔记（notes）。当然，两个视频输出也可以对应于两个不同的投影机，第一个用于显示主演示稿；第二个用于显示例如目录（table of contents），或者显示翻译成其它语言的翻译版本。或者第二个投影机显示“前一”幻灯片。或者...— 我们能想到的在将来更有用的东西。

BEAMER 支持两个视频输出背后的基本思想是：使用特殊的选项让 BEAMER 创建的 PDF-文件的“页面（pages）”具有不寻常地的宽度或高度。不寻常是指页面的高度仍为 128mm，但宽度为 192mm（为普通宽度 96mm 的两倍）。这些“超宽（superwide）”页面会在左侧显示主演示稿的幻灯片，而在右侧显示辅助内容（auxilliary material），这种情形可以使用恰当的选项来切换，但超链接（hyperlinks）只有在左侧显示演示稿而右侧显示第二个屏幕时才能正常工作。

对于附加（attach）了两个屏幕的演示稿，视窗系统（windowing system）认为屏幕的宽度是其真实宽度的两倍。视窗系统将放置于虚拟的（virtual）大屏幕左侧半的内容重定向（redirect）到第一个视频输出，将放置于虚拟的大屏幕右侧半的内容重定向到第二个视频输出。

当演示稿程序（presentation program）显示专门准备的超宽 BEAMER-演示稿时，屏幕的左侧半将填以主演示稿，屏幕的右侧半将填以辅助内容 —就这样（voilà）。并非所有的演示稿程序具有该功能，例如，在 MacOS X 系统中，Acrobat Reader 6.0.2 在全屏模式下只使用一个屏幕。而名为 PDF Presenter 的演示稿程序就支持显示双屏演示稿（dual-screen presentations）。通常，必须查清楚我们的显示程序（display program）和系统（system）是否支持显示双屏的超宽演示稿。

BEAMER 使用 `pgfpages` 宏包排版双屏的演示稿。因此，创建双屏演示稿的第一步就是包含（include）该宏包：

```
\documentclass{beamer}
\usepackage{pgfpages}
```

下一步是选择一个恰当的选项，该选项能让某些特别的内容显示于第二屏幕上。下一节会阐述这些选项。

这些选项会做的事情之一是建立一个特定的适合双屏演示稿的 `pgfpages`-层。然而，以后我们仍可以任意地（arbitrarily）改变这个 `pgfpages`-层，例如，我们可能增大（enlarge）虚拟的（virtual）页面。详细内容请参阅 `pgfpages` 宏包的文档。

22.1 在第二个屏幕上显示笔记

使用双屏幕的第一种情形是在主屏幕中显示演示稿，在第二屏幕中显示笔记（notes）。这时可用 `show notes on second screen` 选项，请参阅第 247 页。

22.2 在第二个屏幕上显示第二模式的材料

使用双屏幕的第二种情形是在主屏幕中显示演示稿，在第二屏幕中显示演示稿的“一个不同版本”。这个不同版本可能是一个翻译版本或当前目录（table of contents）。

要指定在第二屏幕中显示什么，我们可以使用名为 `second` 的特定 BEAMER-模式。该模式的行为和 `handout` 或 `beamer` 等模式的行为类似，但该模式的作用据所使用的选项不同而不同：

```
\setbeameroption{second mode text on second screen=<location>}
```

该选项让第二屏幕显示 `second` 模式的材料 (material)。第二屏幕的 `<location>` 可以是 `left`、`right`、`bottom`、`top`。

具体会发生以下事情：当排版一个新帧时，BEAMER 会检查是否给定了特定的 `typeset second` 选项。如果没有给定，则按常规排版帧，并将幻灯片放在主演示稿屏幕上（更精确地讲，放在 `pgfpages`-页的逻辑页码为 0 的页面上）。在第二屏幕（逻辑页码为 1）上显示在排版帧之前会显示的任何内容。

如果给定了特定的 `typeset second` 帧选项，则在帧的每一张幻灯片之后还会再一次排版帧的内容 (frame contents)，但这次是为 `second` 模式排版的。这会生成 (result in) 另一张幻灯片，这张幻灯片用于第二屏幕（逻辑页码为 1）。然后离开 (ship out) 整个页面。

`second` 模式的行为和其它模式相比更象 `beamer` 模式：用于 `beamer` 的任何叠层规则也适应于 `second` 模式，除非给定了一个明确的 (explicit) `second` 模式规则。特别是，`\only<1-2>{Text}` 的含义是：在 `second` 模式中，只会在第 1 和第 2 张幻灯片中显示；而在 `handout` 模式或 `trans` 模式中，只会在第 1 张幻灯片中显示。

举例：

```
\documentclass{beamer}
\usepackage{pgfpages}
\setbeameroption{second mode text on second screen}
\begin{document}
\begin{frame}[typeset second]
  This text is shown on the left and on the right.
  \only<second>{This text is only shown on the right.}
  \only<second:0>{This text is only shown on the left.}
\end{frame}
\begin{frame}
  This text is shown on the left. The right shows the same as for the
  previous frame.
\end{frame}
\begin{frame}[typeset second]
  \alt<second>{The \string\alt command is useful for second
    mode. Let's show the table of contents, here: \tableofcontents}
  {Here comes some normal text for the first slide.}
\end{frame}
\end{document}
```

举例：下面的例子显示了如何以一种自由自在的 (comfortable) 方式添加翻译。

```
\documentclass{beamer}
\usepackage{pgfpages}
\setbeameroption{second mode text on second screen}
\DeclareRobustCommand\translation[1]{\mytranslation#1\relax}
\long\def\mytranslation#1|#2\relax{\alt<second>{#2}{#1}}
\title{\translation{Preparing Presentations|Vortr"age vorbereiten}}
\author{Till Tantau}
\begin{document}
```



```

\begin{frame}[typeset second]
  \titlepage
\end{frame}
\begin{frame}[typeset second]
  \frametitle{\translation{This is the frame title.|Dies ist der Titel des Rahmens.}}
  \begin{itemize}
    \item<1-> \translation{First|Erstens}.
    \item<2-> \translation{Second|Zweitens}.
    \item<3-> \translation{Third|Drittens}.
  \end{itemize}
  \translation{Do not use line-by-line uncovering.|Man sollte Text nicht
  Zeile f\"ur Zeile aufdecken.}
\end{frame}
\end{document}

```

在上述的最后一个例子中，为每一帧添加 `typeset second` 选项有点令人讨厌。下面的选项会全局地（globally）设置该选项：

```
\setbeameroption{always typeset second mode=<true or false>}
```

当该选项设为 `true` 时，随后的每一帧会将 `typeset second` 选项设为 `true`。

22.3 在第二个屏幕上显示以前的幻灯片

```
\setbeameroption{previous slide on second screen=<location>}
```

该选项使第二屏幕显示前一张已排版好的幻灯片，除非它被一个带有 `[typeset second]` 选项设置的帧覆盖。想法是这样的，如果我们有二个投影机，则可以总是同时呈现（present）“最后两张”幻灯片并讨论它们。

使用该选项将关闭内部文件如目录（table of contents）的更新。

部分 V

如何做

这部分包含对如何做（通常的术语是 *howtos*）的解释。这些解释不是“BEAMER 内核”的一部分。它们说明了如何用 BEAMER 取得特定的效果或如何完成特定的事情。

如何做的第一部分与一些复杂情形有关。

如何做的第二部分说明了怎样导入用其它 L^AT_EX 演示稿文档类如 PROSPER 创建的（部分）演示稿。

如何做的第三部分讨论 TRANSLATOR，一个用于翻译简单字串的 BEAMER 宏包。

23 如何分段显示

23.1 分段显示排序列表

叠层 (overlays) 的一个用途是用于分段 (piecewise) 显示一个排序列表 (enumeration) 中的每一点。按下面的方法能很容易也很灵活地做到这一点:

```
\begin{itemize}
\item<1-> First point.
\item<2-> Second point.
\item<3-> Third point.
\end{itemize}
```

该方法 (approach) 的益处是可以保持所有条目 (item) 的显示顺序。通过更改叠层规则如将最后的一个改为 <2->, 我们就可以让最后的两点 (points) 同时显示。

该方法的弊端是如果我们添加了一个新条目, 则必须记住所有的东西。这通常不是一个大问题, 却令人讨厌。

要自动显示 (uncover), 可以使用下面的命令:

```
\begin{itemize}[<+>->]
\item First point.
\item Second point.
\item Third point.
\end{itemize}
```

[<+>->] 的作用是安装默认的叠层规则 (*default overlay specification*), 详细内容请参阅 `itemize` 的定义。

现在, 假定要使第二点及第三点同时显示。我们可以通过给第二个或第三个 `\item` 命令指定规则 <2-> 来实现。然而, 如果在开始处添加了一个新条目, 那么我们仍需记住一些东西。一个好的办法是临时使用一个不同的叠层规则和点记号 (dot-notation):

```
\begin{itemize}[<+>->]
\item First point.
\item[<.->] Second point.
\item Third point.
\end{itemize}
```

也许我们希望建立自己的基于上述想法的宏包 (macros) (就像 `itemstep` 环境或 `\itemlikeprevious` 命令一样)。

23.2 高亮显示排序列表中的当前条目

如果我们分段 (piecewise) 显示 (uncover) 一个排序列表 (enumeration), 加亮 (highlight) 最后显示的条目以吸引观众的注意力是个好主意。下面是最好的实现的方法:

```
\begin{itemize}
\item<1-| alert@1> First point.
\item<2-| alert@2> Second point.
\item<3-| alert@3> Third point.
\end{itemize}
```

或者

```
\begin{itemize}[<+-| alert@+>]
\item First point.
\item Second point.
\item Third point.
\end{itemize}
```

注意，条目前的记号（item symbol）也会显示为红色。

23.3 改变排序列表中的前导符

当显示一个任务（tasks）或问题（problems）列表时，我们可能希望最后显示的记号前面的记号为选票 X（ballot X），而前一条目的记号为勾号（check mark）（可以在 Dingbats 字体中找到这些字符）。

最好的实现办法是执行（implement）一个新行为环境。如果激活该行为，它会将条目记号模板（item symbol template）临时更改为其它的记号：

```
\newenvironment{ballotenv}
{\only{%
  \setbeamertemplate{itemize item}{code for showing a ballot}%
  \setbeamertemplate{itemize subitem}{code for showing a smaller ballot}%
  \setbeamertemplate{itemize subsubitem}{code for showing a smaller ballot}}}
{}

\setbeamertemplate{itemize item}{code for showing a check mark}
\setbeamertemplate{itemize subitem}{code for showing a smaller check mark}
\setbeamertemplate{itemize subsubitem}{code for showing a smaller check mark}
```

上述代码的作用是安装一个 check mark 作为默认的模板。如果一些条目请求（request）了 ballot 行为，则该模板临时被 ballot 模板取代。

注意，ballotenv 和指定给行为的叠层规则（行为直接跟在叠层规则的后面）一起被调用。这会导致只为指定的叠层调用 \only 命令。

下面是该用法的举例：

```
\begin{itemize}
\item<1-| ballot@1> First point.
\item<2-| ballot@2> Second point.
\item<3-| ballot@3> Third point.
\end{itemize}
```

和

```
\begin{itemize}[<+-| ballot@+>]
\item First point.
\item Second point.
\item Third point.
\end{itemize}
```

在下面的例子中，从一张幻灯片到另一张幻灯片越来越多的条目变成“checked”：

```
\begin{itemize}[<ballot@+-| visible@1-,+(1)>]
\item First point.
\item Second point.
\item Third point.
\end{itemize}
```

重要的一点是 `ballot@+.` `visible@1-,+(1)` 很有趣，其作用如下：虽然它对显示什么没有作用（毕竟它适应于所有幻灯片），但确保了在排序列表中会提到（mention）幻灯片序号 4。这样，会有一张幻灯片，其中的三点都被选取（are checked）。

23.4 分段显示标记的公式

假定我们有一个三行的公式，如下所示：

```
\begin{align}
A &= B \\
&= C \\
&= D
\end{align}
```

要逐行显示该公式有点困难（tricky）。第一种方法是使用 `\pause` 或 `\onslide` 命令。不幸的是，这些命令无法工作，因为在后台，`align` 会多次再加工（reprocess）其输入，这弄乱（mess）这些命令精细的内部构件（delicate internals）。第二种方法如下，它工作得更顺一点：

```
\begin{align}
A &= B \\
\uncover<2->{&= C} \\
\uncover<3->{&= D}
\end{align}
```

不幸的是，在出现标签（tags）时工作不正常（而对于 `align*` 环境却工作正常）。会发生的事情是最后一行的标签会显示于全部幻灯片。这里的问题是，当遭遇到 `\\` 或 `\end{align}` 时会创建标签。在最后一行，这些已经处于 `\uncover` “之后”。

为解决该问题，我们可以添加一个没有标签的空行，然后插入一个负的垂直跳跃（negative vertical skip）以取消（undo）最后一行：

```
\begin{align}
A &= B \\
\uncover<2->{&= C} \\
\uncover<3->{&= D} \\
\notag \\
\end{align}
\vskip-1.5em
```

23.5 逐行显示表格

当我们想逐行 (line-by-line) 显示一个表格时, 如果在表格中有垂直线 (vertical lines) 和水平线 (horizontal lines) 时会遭遇 (run into) 各种各样的问题。原因是, 在读取行 (line) 之前会绘制 (draw) 最左侧的第一条垂直线 (因此, 在读取行之前, 也可以读取任一 `\onslide` 命令)。然而, 在行 (line) 的结束处先放置一个 `\pause` 命令或 `\uncover` 命令并没有益处, 因为这样做会抑制最后显示的行 (line) 下面的水平线。⁶⁶

解决该问题的一个可能的办法是不使用水平线或垂直线。而使用 `colortbl` 宏包给行 (lines) 着色是一个好的构造表格的替代办法。下面是一个养眼的 (optically pleasing) 例子, 这个表格的显示是行智能 (line-wise) 的:

```
\rowcolors[] {1}{blue!20}{blue!10}
\begin{tabular}{l!{\vrule}cccc}
  Class & A & B & C & D \\ \hline
  X      & 1 & 2 & 3 & 4 \pause \\
  Y      & 3 & 4 & 5 & 6 \pause \\
  Z      & 5 & 6 & 7 & 8
\end{tabular}
```

用 `\onslide` 替换 `\pause`, 我们能更精细地 (fine-grained) 控制在哪一张幻灯片中显示哪一行。

23.6 逐列显示表格

列智能 (columnwise) 显示表格会出现与行智能 (linewise) 显示表格相同的问题。

再一次, 可以使用 `colortbl` 宏包来解决该问题。在下面的例子中, 在下面的例子中, `tabular` header 用于插入 `\onslide` 命令, 一条 `\onslide` 命令对应一列 (column), 从某张幻灯片开始, 一条 `\onslide` 命令盖住 (cover) 相应列中的条目。在最后一列的结束处, 不带规则 (specification) 的 `\onslide` 命令确保了下一行的第一列再一次正常显示。

插入一条水平线是很棘手的, 因为该水平线会伸出 (protrude) 到表格之外, 最好的办法是使用水平条 (horizontal bars)。

```
\rowcolors[] {1}{blue!20}{blue!10}
\begin{tabular}{l!{\vrule}c<{\onslide<2->}c<{\onslide<3->}c<{\onslide<4->}c<{\onslide}c}
  Class & A & B & C & D \\
  X      & 1 & 2 & 3 & 4 \\
  Y      & 3 & 4 & 5 & 6 \\
  Z      & 5 & 6 & 7 & 8
\end{tabular}
```

⁶⁶这段的原文是: When you wish to uncover a table line-by-line, you will run into all sorts of problems if there are vertical and horizontal lines in the table. The reason is that the first vertical line at the left end is drawn before the line is even read (and thus, in particular, before any `\onslide` command can be read). However, placing a `\pause` or `\uncover` at the end of the line before is also not helpful since it will then suppress the horizontal line below the last uncovered line.

24 如何导入基于其它宏包和文档类的演示稿

BEAMER 文档类带有许多用于文档类或宏包的仿真层 (emulation layers)，这些仿真层不直接支持 BEAMER，例如，`beamerseminar` 宏包映射 (map) SEMINAR 文档类的一些 (不是所有) 命令到 BEAMER 的命令中。这样，使用 SEMINAR 文档类创建的演示稿，其个别幻灯片或幻灯片的整体设置 (whole sets) 就可以在 BEAMER 中使用了，这是个简单的过程。

没有任何一个仿真层可以完美地替代其原物 (original) (硬件仿真也不能)，在将来也不敢期望。如果我们想/需要/希望使用另一文档类的功能 (features)，请用该文档类准备我们的演示稿。这些仿真层的目的只是加快创建 BEAMER 演示稿的速度，该 BEAMER 演示稿使用了部分以前的演示稿 (old presentations)。我们只需简单地复制以前演示稿的这些部分，而无需当心语法上的细微差异 (subtle differences)。

使用仿真层的有益之处是，当使用另一文档类的语法时我们能够利用 BEAMER 的所有功能，例如，使用 `article` 模式创建 PROSPER 演讲的漂亮的论文版本。

24.1 Prosper、HA-Prosper、Powerdot

`beamerprosper` 宏包映射 PROSPER 宏包的命令到 BEAMER 的命令，PROSPER 宏包由 Frédéric Goualard 开发。而且，HA-PROSPER 宏包和 POWERDOT 宏包的一些命令也映射到 BEAMER 的命令，HA-PROSPER 宏包和 POWERDOT 宏包由亨德里·安德瑞恩 (Hendri Adriaens) 开发。这些映射不能完美地仿真全部 *Prosper!* 更确切地，当移植 (port) 用 PROSPER 创建的演示稿的一部分到 BEAMER 时，这些映射只是一个辅助。没有哪个样式 (*styles*) 能执行那个模仿的 *Prosper styles*。相反，必须使用普通的 BEAMER 主题 (虽然可以执行一个能模仿现有的 PROSPER 样式的 BEAMER 主题，我们没有这样做，也不想这样做)。

使用 PROSPER 代码创建 BEAMER 演示稿的工作流程如下：

1. 使用 `beamer` 文档类而不是 `prosper`。传递给 `prosper` 的大部分选项不适合 `beamer` 并被忽略。
2. 添加 `\usepackage{beamerprosper}` 开始仿真。
3. 如果添加了一张依赖 HA-PROSPER 的幻灯片，那我们希望给 `beamerprosper` 添加 `framesassubsections` 选项，虽然我们不推荐这样做 (用普通的 `\subsection` 命令代替；它能进行更精细地控制)。
4. 如果我们复制了标题 (title) 命令，则必须调整像 `\title` 或 `\author` 这样的命令的内容。注意，在 PROSPER 中，`\email` 命令只能在 `\author` 命令之外给出，然而，在 BEAMER 和 HA-PROSPER 中，`\email` 命令在 `\author` 命令之内给出
5. 当复制包含 `\includegraphics` 命令的幻灯片时，我们殆必 (almost surely) 调整该命令的用法。如果我们使用了 pdfL^AT_EX 排版演示稿，那就不能包含 (include) PostScript 文件。我们必须将它们转换成 `.pdf` 或 `.png`，并相应地调整 `\includegraphics` 的用法。
6. 当事情开始变化时，我们可以使用 BEAMER 的全部命令，甚至和 PROSPER 的命令相混 (mix)。

`beamerexample-prosper.tex`⁶⁷ 是一个示例文件。

不幸的是，在多个地方我们可能遭遇到问题：

- 在 BEAMER 中，`\PDForPS` 命令的功能正如其名称所示：当运行 `pdflatex` 时，插入第一个参数，当运行 `latex` 时，插入第二个参数。然而，在 PROSPER 中，插入到 PDF 实例 (case) 中的代码实际上就是

⁶⁷在装有 CTEX 套装的 Windows 平台中，该文件位于如 `D:\CTEX\MiKTeX\tex\latex\beamer\base\emulation\examples` 中。

PostScript 代码，PostScript 代码后来由某个外部程序转换成 PDF。我们必须调整 (adjust) 该 PostScript 代码使它能遵守 (work with) pdf_latex (这不一定是)。

- 如果我们使用了精细的 (fine-grained) 间距命令 (spacing commands)，如在这里添加了一个小水平跳跃，在那里添加了一个垂直跳跃，则文本的排版可能很糟糕。好的办法是移除这些间距命令。
- 如果使用了 pstricks 命令，那么我们将不得不坚持用 latex 和 dvips，或不得不使用如 pgf 这样的东西以绕开它们。如果我们能改用 pdf_latex，so be warned (因此被警告)，则移植 (Porting) 大量的 pstricks 代码必定很困难。更多相关图形的内容请阅读第 13 节。
- 如果因为没有执行某 PROSPER 命令而不能编译 (compile)，则我们将不得不删除该命令并使用 BEAMER 的命令模仿其行为。

```
\usepackage{beamerprosper}
```

在 beamer 演示稿中包含 (Include) 该宏包以获得访问 PROSPER 命令的权限。使用 beamer 作为文档类而不是 prosper。传递给 prosper 文档类的大部分选项在 beamer 中毫无意义，所以可以删除它们。

该宏包带有下列选项：

- **framesassubsections** 让每一帧创建它自己的小节 (subsection)，并用帧标题 (frame title) 作为小节名 (subsection name)。这个行为模仿了 HA-PROSPER 的行为。在一个长的演讲 (talk) 中这会创建太多的小节。

ARTICLE 在 article 模式中，framesassubsections 选项没有作用。

举例：

```
\documentclass[notes]{beamer}

\usepackage[framesassubsections]{beamerprosper}

\title{A Beamer Presentation Using (HA-)Prosper Commands}
\subtitle{Subtitles Are Also Supported}
\author{Till Tantau}
\institution{The Institution is Mapped To Institute}

\begin{document}

\maketitle

\tsectionandpart{Introduction}

\overlays{2}{
\begin{slide}{About this file}
\begin{itemstep}
\item
This is a beamer presentation.
\item

```



```

    You can use the prosper and the HA-prosper syntax.
\item
    This is done by mapping prosper and HA-prosper commands to beamer
    commands.
\item
    The emulation is by no means perfect.
\end{itemstep}
\end{slide}
}

\section{Second Section}
\subsection{A subsection}
\begin{frame}
    \frametitle{A frame created using the \texttt{frame} environment.}

    \begin{itemize}[<+>]
    \item You can still use the original beamer syntax.
    \item The emulation is intended only to make recycling slides
        easier, not to install a whole new syntax for beamer.
    \end{itemize}
\end{frame}

\begin{notes}{Notes for these slides}
My notes for these slides.
\end{notes}
\end{document}

```

我们可以运行 pdfL^AT_EX 以获得一个带叠层的演示稿。添加 `notes` 选项后可以显示笔记 (notes)。会忽略特定的命令如 `\LeftFoot`。我们可以使用通常的命令 (usual commands) 更改主题。也可以在文件中使用所有普通的 BEAMER 命令和概念 (concepts) 如叠层规则。我们还可以用 `article` 文档类并包含 `beamerarticle` 宏包以创建一个版本。

下面罗列了在 BEAMER 中 PROSPER 命令的作用。

`\email{<text>}`

即:

`\email{<文本>}`

以打字机文本 (typewriter text) 排版其参数。因此, 必须在 `\author` 命令之内 (*inside*) 给出该命令。

`\institution{<text>}`

即:

`\institution{<文本>}`

如果在 `\author` 命令之外给出该命令, 则该命令映射到 BEAMER 的 `\institute` 命令, 而且用小字体 (smaller font) 排版其参数。

`\Logo(<x>,<y>){<logo text>}`

即：

`\Logo(<x>,<y>){<徽标文本>}`

该命令映射到 `\logo{<logo text>}` 命令。会忽略坐标 (coordinates)。

```
\begin{slides}[<options>]{<frame title>}
  <environment contents>
\end{slides}
```

即：

```
\begin{slides}[<选项>]{<帧标题>}
  <environment contents>
\end{slides}
```

插入一个带有 `fragile=singleslide` 选项设置的帧。将<帧标题 (Frame Title) > 封入一个 `\frametitle` 命令。

可能会给出下面的 *<options>*：

- `trans=<prosper transition>` 安装指定的 *<prosper transition>* 作为显示幻灯片时的过渡效果。
- *<prosper transition>* 和 `trans=<prosper transition>` 具有相同的效果。
- `toc=<entry>` 用 *<entry>* 覆盖由该幻灯片创建的小节目录条目 (subsection table of contents entry)。注意，只有指定了 `framesassubsections` 选项时才会为幻灯片创建一个小节条目 (subsection entry)。
- `template=<text>` 会被忽略。

举例：下面的两部分代码具有相同的效果：

```
\begin{slide}[trans=Glitter,toc=short]{A Title}
  Hi!
\end{slide}
```

和

```
\subsection{short} % omitted, if framesassubsections is not specified
\begin{frame}[fragile=singleslide]
  \transglitter
  \frametitle{A Title}
  Hi!
\end{frame}
```

`\overlays{<number>}{<slide environment>}`

即：

`\overlays{<序号>}{<幻灯片环境>}`

在一个不带 `fragile` 选项的帧内放置 `\slide environment`，该帧从此（hence）可以包含叠层文本（overlaid text）。会忽略 `\number`，因为所需的叠层序号（number of necessary overlays）由 BEAMER 自动计算出来。

举例：下面的两部分代码具有相同的效果：

```
\overlays{2}{  
\begin{slide}{A Title}  
  \begin{itemstep}  
    \item Hi!  
    \item Ho!  
  \end{itemstep}  
\end{slide}}
```

和

```
\subsection{A Title} % omitted, if framesassubsections is not specified  
\begin{frame}  
  \frametitle{A Title}  
  \begin{itemstep}  
    \item Hi!  
    \item Ho!  
  \end{itemstep}  
\end{frame}
```

`\fromSlide{<slide number>}{<text>}`

该命令映射到 `\uncover<slide number>->{<text>}`。

`\fromSlide*{<slide number>}{<text>}`

该命令映射到 `\only<slide number>->{<text>}`。

`\onlySlide{<slide number>}{<text>}`

该命令映射到 `\uncover<slide number>>{<text>}`。

`\onlySlide*{<slide number>}{<text>}`

该命令映射到 `\only<slide number>>{<text>}`。

`\untilSlide{<slide number>}{<text>}`

该命令映射到 `\uncover<-<slide number>>{<text>}`。

`\untilsSlide*{<slide number>}{<text>}`

该命令映射到 `\only<-<slide number>>{<text>}`。

`\FromSlide{<slide number>}`

该命令映射到 `\onslide<slide number>->`。

`\OnlySlide{<slide number>}`

该命令映射到 `\onslide<slide number>>`。

`\UntilSlide{<slide number>}`

该命令映射到 `\onslide<-<slide number>`。

`\slideCaption{<text>}`

该命令映射到 `\date{<text>}`。

`\fontTitle{<text>}`

只插入 `<text>`。

`\fontText{<text>}`

只插入 `<text>`。

`\PDFtransition{<prosper transition>}`

映射 `<prosper transition>` 到一个相称的 `\transxxxx` 命令。

`\begin{Itemize}`

`<environment contents>`

`\end{Itemize}`

该命令映射到 `itemize`。

`\begin{itemstep}`

`<environment contents>`

`\end{itemstep}`

该命令映射到带有选项 `[<+>]` 的 `itemize`。

`\begin{enumstep}`

`<environment contents>`

`\end{enumstep}`

该命令映射到带有选项 `[<+>]` 的 `enumerate`。

`\hiddenitem`

该命令映射到 `\addtocounter{beamerpauses}{1}`。

`\prosperpart [<options>]{<text>}`

该命令的效果和 PROSPER 的 `\part` 命令的效果相同。BEAMER 的普通 (normal) 命令 `\part` 保留了它的普通语义 (normal semantics)。因此, 我们希望用 `\prosperpart` 替换所有的 `\part`。

`\tsection*{<section name>}`

创建一个名为 `<section name>` 的节。如果出现了星号 (star) 则该星号被忽略。

`\tsectionandpart*{<part text>}`

映射到后跟 `\prosperpart` 命令的 `\section` 命令。

ARTICLE 在 `article` 模式中, 不添加部分页 (part page)。

`\dualslide[⟨x⟩][⟨y⟩][⟨z⟩]{⟨options⟩}{⟨left column⟩}{⟨right column⟩}`

该命令映射到 `columns` 环境。⟨left column⟩ 文本显示于左栏，⟨right column⟩ 文本显示右栏。会忽略 ⟨x⟩、⟨y⟩、⟨z⟩ 选项。而且，会忽略除 `lcolwidth=` 和 `rcolwidth=` 外的所有选项 ⟨options⟩。 `lcolwidth=` 和 `rcolwidth=` 选项分别设置左和右栏的宽度。

`\PDForPS{⟨PostScript text⟩}{⟨PDF text⟩}`

据使用 `latex` 或 `pdflatex` 不同，插入 ⟨PostScript text⟩ 或 ⟨PDF text⟩。当移植 (porting) 时，⟨PDF text⟩ 最可能出错 (incorrect)，因为在 PROSPER 中，⟨PDF text⟩ 实际是 PostScript 文本，该 PostScript 文本后来由某外部程序转换成 PDF。

如果 ⟨PDF text⟩ 包含一个 `\includegraphics` 命令 (这是它的常见用法)，我们则必须将包含的图形文件名更改为 `.pdf`、`.png`、`.jpg`。通常，我们必须将图形转换成上述格式。

`\onlyInPDF{⟨PDF text⟩}`

如果使用了 `pdflatex`，则只会包含 ⟨PDF text⟩。在这儿也可以用 `\PDForPS` 命令，效果相同。

`\onlyInPS{⟨PS text⟩}`

如果使用了 `latex` 则只会包含 ⟨PS text⟩。

`\begin{notes}{⟨title⟩}`

⟨environment contents⟩

`\end{notes}`

映射到 `\note{\textbf{⟨title⟩}⟨environment contents⟩}` (或多或少)。

下面的命令由 BEAMER 解析 (parse)，但没有作用 (effect)：

- `\myitem`,
- `\FontTitle`,
- `\FontText`,
- `\ColorFoot`,
- `\DefaultTransition`,
- `\NoFrenchBabelItemize`,
- `\TitleSlideNav`,
- `\NormalSlideNav`,
- `\HAPsetup`,
- `\LeftFoot`, and
- `\RightFoot`.

24.2 Seminar

`beamerseminar` 宏包映射 SEMINAR 宏包的命令子集 (subset) 到 BEAMER。对于 PROSPER，该仿真功能不理想。例如，不支持直印幻灯片 (portrait slides)、不会自动分页、不会仿真幻灯片的帧。不幸的是，对包含叠层的所有帧 (slide 环境)，我们不得不“手工”在 `frame` 环境中放置环境 (environment)，并通过关闭幻灯片或打开新的幻灯片 (然后将这些放入 `frame` 环境中) 来移除环境内的所有 `\newslide` 命令。

移植 (migration) 的工作流程如下：

1. 使用 `beamer` 文档类，而不是 `seminar`。传递给 `seminar` 的大部分选项不适应于 `beamer` 并被忽略。
2. 如果我们拷贝了混有普通文本（normal text）的演示稿的一部分，请添加 `ignorenonframetext` 选项并在 `frame` 内放置每一 `slide` 环境，因为 BEAMER 无法把 `\begin{slide}` 认作帧的开始。
3. 添加一个 `\usepackage{beamerseminar}` 开始仿真。如果我们想创建一个支持录像投影机（video projector）的演示稿，请添加 `accumulate` 选项。
4. 可能要添加命令以安装主题和模板。
5. 在导言区不应放置与页面（page）和幻灯片风格（slide styles）有关的命令。它们不适应于 `beamer`。
6. 如果在一个包含叠层（overlay）的 `slide`（或类似 `slide*` 的）环境中使用了 `\newslide` 命令，则我们必须用一个关的 `\end{slide}` 和一个开的 `\begin{slide}` 替换该命令。
7. 接下来，对于每一个包含叠层的 `slide` 或 `slide*` 环境，必须在其周围放置一个 `frame` 环境。我们可以移除 `slide` 环境（和从此用 `frame` 有效地替换它），除非我们使用了 `accumulate` 选项。
8. 如果我们在幻灯片内（inside slides）使用了 `\section` 或 `\subsection` 命令，则不得不在帧外（*outside the frames*）移除它们。为幻灯片添加一个 `\frametitle` 命令是必须的。
9. 如果我们使用了 pdfL^AT_EX 排版演示稿，则不能包含（include）PostScript 文件。必须将它们转换成 `.pdf` 或 `.png`，并相应地调整 `\includegraphics` 的用法。
10. 当开始改变一些东西时，我们就可以使用 BEAMER 的所有命令，甚至可以和 SEMINAR 的命令混合使用。

示例文件为 `beamerexample-seminar.tex`⁶⁸。

不幸的是，在多个地方我们可能遭遇到如下的问题：

- `seminar` 的笔记管理（note management）和 `beamer` 有很大的不同，这样，我们不得不“手工”编辑笔记。特别是，像 `\ifslidesonly` 和 `\ifslide` 这样的命令所起的作用可能与你期望的不一样。
- 如果我们使用 `pstricks` 命令，则不得不坚持使用 `latex` 和 `dvips`，或不得不使用例如 `pgf` 绕开（work around）它们。如果我们希望切换到 `pdflatex`，则输出大量的 `pstricks` 代码应该是很困难的，并因此受到警告。
- 如果因没有执行 SEMINAR 命令而无法编译，则我们不得不删除这些命令，并用相应的 BEAMER 命令仿真其行为。

`\usepackage{beamerseminar}`

在一个 `beamer` 演示稿中包含该宏包以获得访问 SEMINAR 命令的权限。使用 `beamer` 作为文档类而不是 `seminar`。传递给 `seminar` 文档类的大部分选项对 `beamer` 没有作用，因此可以删除它们。

该宏包可以带有下列选项：

- `accumulate` 仿真叠层。SEM_INAR 宏包的初始行为是在每一叠层内，只有真正的叠层的“新（new）”部分才会显示。这意味着，如果我们真正以透明方式（on transparencies）印出（print out）叠层，那么，才会真正将层叠放在一起（really stack overlays on top of each other）。对于一个支持录像投影机（video projector）的演示稿，我们宁可呈现（present）一个叠层的“仿真”版本。这是该选项的所作所为：当显示 *i*-th 叠层的新内容（new material）时，也会显示所有以前叠层的内容。

⁶⁸在装有 C_TE_X 套装的 Windows 平台中，该文件位于如 `D:\CTEX\MiKTeX\tex\latex\beamer\base\emulation\examples` 中。

举例：下面的例子提取自 `beamerexample-seminar.tex`⁶⁹：

```
\documentclass[ignorenonframetext]{beamer}
\usepackage[accumulated]{beamerseminar}
\usepackage{beamerthemeclassic}

\title{A beamer presentation using seminar commands}
\author{Till Tantau}

\let\heading=\frametitle

\begin{document}

\begin{frame}
  \maketitle
\end{frame}

This is some text outside any frame. It will only be shown in the
article version.

\begin{frame}
  \begin{slide}
    \heading{This is a frame title.}

    \begin{enumerate}
      {\overlay1
      \item Overlays are a little tricky in seminar.
      {\overlay2
      \item But it is possible to use them in beamer.
      }
      }
    \end{enumerate}
  \end{slide}
\end{frame}
\end{document}
```

我们可以在文件中使用 BEAMER 的所有普通命令和概念，如叠层规则。也可以用 `article` 文档类或包含 `beamerarticle` 宏包以创建一个 `article` 版本。

下面罗列了 BEAMER 中的 SEMINAR 命令的作用：

`\overlay{<number>}`

只在编号为 $\langle number \rangle + 1$ 的叠层中或如果给定了 `accumulate` 选项则从那个叠层开始显示内容 (material)，一直到当前 TeX group 的结束处。该命令可以嵌套（像在 SEMINAR 中）。如果一个 `\overlay` 命令用在另一

⁶⁹在装有 CTEX 套装的 Windows 平台中，该文件位于如 `D:\CTEX\MiKTeX\tex\latex\beamer\base\emulation\examples` 中。

个 `\overlay` 命令内部，则内层的命令会“临时”覆盖外层的命令，如下例所示，假设给定了 `accumulate` 选项。

举例：

```
\begin{frame}
  \begin{slide}
    This is shown from the first slide on.
    {\overlay{2}
      This is shown from the third slide on.
      {\overlay{1}
        This is shown from the second slide on.
      }
    }
    This is shown once more from the third slide on.
  }
\end{slide}
\end{frame}
```

```
\begin{slide}*
  <environment contents>
\end{slide}
```

在 `<environment contents>` 周围安装 `\overlay{0}`。如果给定了 `accumulate` 选项，它不起作用，但它会使幻灯片的主体文本（main text）只在第一张幻灯片中显示。如果我们真的希望在每一张幻灯片之上物理放置（physically place）幻灯片，则该选项很有用。

带星号的版本的所作所为与不带星号的版本相同。

如果在一个 `\frame` 内没有出现该命令，则建立一个带 `fragile=singleframe` 选项设置的帧。这样，该帧只包含一张幻灯片。

举例：

```
\begin{slide}
  Some text.
\end{slide}

\frame{
  \begin{slide}
    Some text. And an {\overlay{1} overlay}.
  \end{slide}
}
```

`\red`

映射到 `\color{red}`。

`\blue`

映射到 `\color{blue}`。

`\green`

映射到 `\color{green}`。

`\ifslide`

在 `presentation` 模式中是正确的，在 `article` 模式中是错误的。

`\ifslidesonly`

作用和 `\ifslide` 相同。

`\ifarticle`

在 `presentation` 模式中是错误的，在 `article` 模式中是正确的。

`\ifportrait`

常常是错误的。

下面的命令由 BEAMER 解析，但没有作用：

- `\ptsize`.

24.3 Foil \TeX

`beamerfoils` 宏包映射 FOILS 宏包的命令的子集 (subset) 到 BEAMER 宏包。因为 `beamerfoils` 宏包只定义了少数几个非标准的 (non-standard) \TeX 命令，而 BEAMER 却执行所有标准命令 (standard commands)，因此仿真层 (emulation layer) 相当地简单。

版权提示：Foil \TeX 宏包有一个限制许可证 (restricted license)。正因如此，在 BEAMER 文档类中没有包含来自 FOILS 宏包的例子。在仿真时不能使用 FOILS 宏包的代码 (只映射 FOILS 的命令到 BEAMER 的命令)。因此，我们的理解是，由 BEAMER 文档类提供的仿真 (emulation) 是“自由的 (free)”和合法的 (legally)。IBM 拥有 FOILS 文档类的版权，但在该文档类的命令功效上却无所作为 (effect)⁷⁰。(至少，这就是我们的知道的情况)

移植 (migration) 的工作流程如下：

1. 使用 `beamer` 文档类而不是 `foils`。
2. 添加 `\usepackage{beamerfoils}` 开始仿真。
3. 可能会添加命令安装主题和模板。
4. 如果在一个 `\frame` 命令或 `frame` 环境内部使用了 `\foilhead` 命令，则它的行为和 `\frametitle` 相像。如果在帧外使用了该命令，它将新开一个帧 (带有 `allowframebreaks` 选项，因此不允许使用叠层)。该帧会一直持续到下一 `\foilhead` 或一个新 `\endfoil` 命令出现时。注意，`\frame` 命令不会结束一个用 `\foilhead` 开始的帧。
5. 如果依靠 `\foilhead` 自动创建帧，则必须在文档结束之前插入一个 `\endfoil` 以结束最后的一个帧。
6. 如果使用了 pdf \LaTeX 排版演示稿，则不能包含 (include) PostScript 文件。我们必须将它们转换成 `.pdf` 或 `.png` 并调整 `\includegraphics` 的一些用法。
7. 在 BEAMER 中对象的尺寸不相同，因为由浏览器 (viewer) 而不是文档类进行缩放 (scaling)。因此，一个 6 英寸的 `framebox` 在 BEAMER 演示稿中会显得有太大。我们不得不手工调整 foil \TeX 演示稿的尺寸。

⁷⁰该句的原文是：IBM has a copyright on the foils class, not on the effect the commands of this class have.

```
\usepackage{beamerfoils}
```

在 `beamer` 演示稿中包含该宏包以获得访问 `FOILS` 命令的权限。用 `beamer` 作为文档类而不是 `foils`。

举例：在下面的例子中，会自动创建帧。结束处的 `\endfoil` 是必须的，它关闭（close）最后一个帧。

```
\documentclass{beamer}
\usepackage{beamerfoils}

\begin{document}

\maketitle

\foilhead{First Frame}

This is on the first frame.
\pagebreak
This is on the second frame, which is a continuation of the first.

\foilhead{Third Frame}

This is on the third frame.

\endfoil
\end{document}
```

举例：在下面的例子中，手工插入帧。`\endfoil` 不是必须的。

```
\documentclass{beamer}
\usepackage{beamerfoils}

\begin{document}

\frame{\maketitle}

\frame{
\foilhead{First Frame}
This is on the first frame.
}

\frame{
\foilhead{Second Frame}
This is on the second frame.
}
\end{document}
```

下面罗列了在 `BEAMER` 中 `FOILS` 命令的作用：

```
\MyLogo{{logo text}}
```

该命令映射到 `\logo`，即使内部存储了徽标 (Logo)，因此，可以用 `\LogoOn` 和 `\LogoOff` 打开和关闭徽标。

`\LogoOn`

使徽标 (Logo) 可见。

`\LogoOff`

使徽标 (Logo) 不可见。

`\foilhead[⟨dimension⟩]{⟨frame title⟩}`

如果在一个 `\frame` 命令或 `frame` 环境内部使用了该命令，则该命令映射到 `\frametitle{⟨frame title⟩}`。如果在帧外使用了该命令，则新开一个带有 `allowframebreaks` 选项的帧。如果以前用该命令开始了一个帧，则该帧会在下一帧开始之前关闭。会忽略 `⟨dimension⟩`。

`\rotatefoilhead[⟨dimension⟩]{⟨frame title⟩}`

该命令的作用和 `\foilhead` 相同。

`\endfoil`

在 FOILS 中该命令不可用。在 BEAMER 中，该命令可用于结束用 `\foilhead` 自动创建的帧。如果最后的一帧是由 `\foilhead` 打开的，则必须在文档的结束之前给出 `\endfoil`。

`\begin{boldequation}*`

`⟨environment contents⟩`

`\end{boldequation}`

该命令映射到 `equation` 或 `equation*` 环境，并打开 `\boldmath`。

`\FoilTeX`

像在 FOILS 宏包中一样排版 `foilTEX` 这个单词。

`\bm{⟨text⟩}`

执行方式和在 FOILS 宏包中的执行方式相同。

`\bmstyle{⟨text⟩}{⟨more text⟩}`

执行方式和在 FOILS 宏包中的执行方式相同。

下面附加的类似定理的 (theorem-like) 环境是预定义好的：

- `Theorem*`,
- `Lemma*`,
- `Corollary*`,
- `Proposition*`, and
- `Definition*`.

例如，第一个是用 `\newtheorem*{Theorem*}{Theorem}` 预定义的。

下面的命令由 BEAMER 解析，但没有作用：

- `\leftheaderr`,
- `\rightheaderr`,
- `\leftfooterr`,
- `\rightfooterr`,
- `\Restriction`, 和
- `\marginpar`.

24.4 T_EXPower

`beamertexpower` 宏包映射 TEXPOWER 宏包的命令的子集 (subset) 到 BEAMER。TEXPOWER 宏包由史蒂芬·雷曼克 (Stephan Lehmk) 创建。目前该子集还很小，对它进行大量的改编 (adaptions) 是必要的。注意，TEXPOWER 自身不是一个完整的文档类，而是一个依赖于其它文档类如 `seminar` 或 `prosper` 才能完成排版的宏包，而且还必须另外加载一个仿真层。事实上，可能会在 BEAMER 中直接使用 TEXPOWER，但我们没有这样试过，在将来这应该是可行的。

目前，`beamertexpower` 宏包主要映射 `\stepwise` 和相关的命令到 BEAMER 相应的命令。`\pause` 命令无需映射，因为无论如何，BEAMER 都会直接执行该命令。

移植 (migration) 的工作流程如下：

1. 用 `beamer` 替换文档类。如果文档类是 `seminar` 或 `prosper`，则可以使用上述的仿真层，也就是说，我们可以包含 (include) `beamerseminar` 或 `beamerprosper` 文件以仿真文档类。

如何仿真 SEMINAR 或 PROSPER 的注意事项在这里也适应。

2. 另外，添加 `\usepackage{beamertexpower}` 开始仿真。

`\usepackage{beamertexpower}`

在 `beamer` 演示稿中包含该宏包以获得访问 TEXPOWER 的和 `\stepwise` 有关的命令的权限。

使用 `\pause` 命令的一个注意事项是：BEAMER 和 TEXPOWER 都可执行该命令并具有相同的语义 (semantics)，因此没有必要映射该命令到 `beamertexpower` 中的不同的命令。然而，不同的一点是，在 BEAMER 中几乎可以在任何地方使用 `\pause` 命令，而在 TEXPOWER 中，却只能在非嵌套的 (non-nested) 情形中使用该命令。因为 BEAMER 比 TEXPOWER 更灵活 (flexible)，当输出 (port) 时不会出现问题。

下面罗列了在 BEAMER 中 TEXPOWER 命令的作用：

`\stepwise{<text>}`

像在 TEXPOWER 中一样，该命令启动 (initiates) `text`，在 `text` 中，可能会给出像 `\step` 或 `\switch` 这样的命令。包含在 `\switch` 命令中的文本 (Text) 可能被围入一个带有叠层规则 `<+(1)->` 的 `\only` 命令中。这意味着从第二张幻灯片向前 (onward) 插入第一个 `\step` 的文本，从第三张幻灯片向前 (onward) 插入第二个 `\step` 的文本。等等。

`\parstepwise{<text>}`

和 `\stepwise` 相同，当映射 `\step` 命令时，只是用 `\uncover` 替代 `\only`。

`\liststepwise{<text>}`

和 `\stepwise` 相同，只是在 `<text>` 之前插入一条不可见的水平线。推测这有利于解决 `TEXPOWER` 中和垂直间距（vertical spacing）有关的问题。

`\step{<text>}`

该命令映射到 `\only<+(1)-><text>` 或 `\uncover<+(1)-><text>`，据该命令用于 `\stepwise` 环境内 `\parstepwise` 环境内不同而不同。

`\steponce{<text>}`

该命令映射到 `\only<+(1)><text>` 或 `\uncover<+(1)><text>`，据该命令用于 `\stepwise` 环境内或 `\parstepwise` 环境内不同而不同。

`\switch{<alternate text>}{<text>}`

该命令映射到 `\alt<+(1)->{<text>}{<alternate text>}`。注意会交换（swap）参数。

`\bstep{<text>}`

该命令常常映射到 `\uncover<+(1)-><text>`。

`\dstep`

该命令只是将计数器（counter）`beamerpauses` 增加 1。它没有其它作用。

`\vstep`

和 `\dstep` 相同。

`\restep{<text>}`

和 `\step` 相同，但 `<text>` 显示于和前一 `\step` 命令相同的幻灯片上。这通过在调用 `\step` 之前将计数器 `beamerpauses` 减 1 来完成。

`\reswitch{<alternate text>}{<text>}`

和 `\restep` 相似，只用于 `\switch` 命令。

`\rebstep<text>`

和 `\restep` 相似，只用于 `\bstep` 命令。

`\redstep`

该命令没有作用。

`\revstep`

该命令没有作用。

`\boxedsteps`

如果在 `\stepwise` 环境中使用该命令，则临时（为当前 `TEX` group）改变 `\step` 的作用以启动（issue）一个 `\uncover`。

`\nonboxedsteps`

如果在 `\parstepwise` 环境中使用该命令，则临时（为当前 `TEX` group）改变 `\step` 的作用以启动（issue）一个 `\only`。

`\code{<text>}`

用粗体的打字机字体（boldface typewriter font）排版参数（argument）。

`\codeswitch`

切换到粗体的打字机字体（boldface typewriter font）。

25 翻译字串

25.1 介绍

25.1.1 该宏包的概述

TRANSLATOR 宏包是一个 L^AT_EX 宏包，它提供了将个别单词 (individual words) 翻译不同语言的灵活机制，例如，将 “figure” 翻译成德语的 “Abbildung”。当某个宏包的作者想本地化 (localize) 该宏包，这样文本就能正确地翻译成用户喜欢的语言，这时上述的翻译机制就很有用。TRANSLATOR 宏包不是用于自动翻译多个单词的。

我们可能会奇怪 TRANSLATOR 宏包是否真的是必须的，因为在 L^AT_EX 中还可应用更好的 babel 宏包。该宏包已经提供了单词 “figure” 的翻译。不幸的是，babel 宏包的结构 (architecture) 是以这样的一种方式设计的：没有办法在 (很短的) 翻译列表中添加一个新单词的翻译，该翻译列表直接建入 babel 中 (directly built into babel)。

TRANSLATOR 宏包很容易扩展词汇 (vocabulary)。它既可以添加新词也可以添加这些新词的翻译。

TRANSLATOR 宏包可以和 babel 一起使用。这样，babel 用于像特殊引用记号 (special quotation marks) 和输入捷径 (input shortcuts) 这样的语言规则的 (language-specific) 东西，而 TRANSLATOR 则用于翻译单词。

25.1.2 如何看懂这一节

这一节讲述了 TRANSLATOR 宏包的命令及其用法。TRANSLATOR 宏包提供的 “全局 (public)” 命令和环境的描述见于下文 (text) 的各个地方。在每一个这样的描述中，被描述的命令、环境、选项以红色表示，显示为绿色的文本是可选的并可删去。

下面，首先讲述安装，然后讲述基本的概念，最后讲述宏包的用法。

25.1.3 贡献

因为该宏包是关于国际化的 (internationalization)，它所需的输入 (input) 来自人们的翻译 (翻译其母语)。

为了登录 (submit) 词典，请按下述步骤执行：

1. 阅读该手册并确保理解了基本的概念。
2. 查看翻译 (translation) 是 TRANSLATOR 宏包还是其它宏包的一部分。通常，向 TRANSLATOR 项目 (project) 提出 (submit) 翻译和新的关键词 (new keys)，如果它们是公众感兴趣的。
例如，像 figure 这样关键词的翻译就可以发送到 TRANSLATOR 项目。特定宏包的关键词的翻译应发送到该宏包的作者。
3. 如果我们确定翻译可以进入 TRANSLATOR 宏包，则可以创建一个具有正确名称的字典 (dictionary) (请再次阅读该文档)。
4. 最后，使用开发网站 (development site) 上正确的论坛 (forum) 登录词典。[该句的原文是：Finally, submit the dictionary using the correct forum on the development site]

25.1.4 安装

这个宏包与 BEAMER 一起分发。通常，如果安装了 BEAMER，则我们的系统中就已安装了该宏包。如果没有，请查看第 2 节的关于如何安装 BEAMER 的指令 (instructions)。

25.2 基本概念

25.2.1 关键词

TRANSLATOR 宏包的主要目的是提供关键词 (*keys*) 的翻译。通常, 一个关键词就是一个像 **Figure** 这样的英语单词, 以及该英语单词的德文翻译 “Abbildung”。

对于一个像 “figures” 这样的概念, 单个的关键词 (a single key) 是不够的:

1. 常常需要将像 “Table of figures” 这样的一组单词作为一个整体来翻译。虽然在英语中这组单词有三个, 但其德文翻译却只有一个单词: “Abbildungsverzeichnis”。
2. 大写字母 (Uppercase) 和小写字母 (lowercase) 可能会带来问题。假设我们提供了关键词 **Figure** 的翻译, 当我们将 **Figure** (其首字母为小写) 用于普通文本 (normal text) 中时会发生什么呢? 我们可能会使用 T_EX 的将翻译转变成小写的功能, 但这样做对于德文的翻译 “Abbildung” (其首字母常常为大写) 则会出错。
3. 复数 (Plurals) 也可能导致问题。如果我们知道 “Figure” 的翻译, 并不意味着我们知道 “Figures” (德语为 “Abbildungen”) 的翻译。

因为上述问题, 简单的 “figures” 对应的关键词有: **Figure**、**figure**、**Figures**、**figures**。第一个关键词用于顶部导航区 (headline) 处使用的单数的 (in singular) “figure” 的翻译。最后一个关键词用于普通文本 (normal text) 中使用的复数的 (in plural) “figure” 的翻译。

一个关键词可以包含空隔 (spaces), 因此 **Table of figures** 是许可的 (permissible) 关键词。

关键词 (Keys) 常常是英文文本, 其翻译同样是关键词, 但不总是这样。理论上, 关键词可以是任何东西。However, since the key is used as a last fallback when no 翻译 (Translation) whatsoever is available, a key should be readable by itself.

25.2.2 语言名

TRANSLATOR 宏包使用的语言名 (names for languages) 不同于像 babel 这样的宏包所使用语言名。其原因是, babel 宏包使用的名称有点乱, 因此, 提尔 (Till) 教授决定将 TRANSLATOR 宏包清理干净。然而, 已提供了从 babel 名称到 TRANSLATOR 名称的映射 (mappings)。

TRANSLATOR 宏包使用的名称通常是英文名称, 英文名称常用于各种语言。因此, 英语 (English language) 的英文名称是 **English**, 德语 (German) 的英文名称是 **German**。

语言的变种 (Variants) 有其自己的名称: 英语的不列颠 (British) 版本名称是 **BritishEnglish**, 英语的美国 (US) 版本名称是 **AmericanEnglish**。

对于德语, 有一个特殊的问题, 就是 1998 年以前的拼写和现在的拼写相反 (oppose)。German 反应的是当前的官方拼写, 但 **German1997** 这样的拼写用于 1997 年。

25.2.3 语言路径

当我们请求一个关键词的翻译时, TRANSLATOR 宏包会试图提供当前语言的 (current language) 翻译版本。语言的例子是德语 (German) 或英语 (English)。

当 TRANSLATOR 宏包查找给定关键词的当前语言的翻译时, 可能会失败。这样, TRANSLATOR 宏包将尝试一种撤退策略 (fallback strategy): 跟踪语言路径 (*language path*) 并在此路径继续查找各种语言的翻译。

语言路径不仅对于撤退 (fallbacks) 有益, 对于使用另一语言的变种的情形也有益。例如, 当 TRANSLATOR 宏包查找一个奥地利语 (Austrian) 的关键词的翻译时, 将从语言路径的奥地利语 (Austrian) 开始, 接着是德

语 (German)。Then, 一个奥地利语的字典只需提供这些关键词的翻译, [这句的原文是: Then, a dictionary for Austrian only needs to provide 翻译 (Translation) s for those keys where Austrian differs from German.]

25.2.4 字典

关键词的翻译通常由字典 (*dictionaries*) 提供。一个字典包含关键词集 (set of keys) 的特定语言的翻译, 例如, 一个字典可能包含月名称 (names of months) 的德语翻译, 另一个字典可能包含数字 (numbers) 的法语翻译。

25.3 用法

25.3.1 基本用法

下面是一个如何使用该宏包的典型例子:

```
\documentclass[german]{article}

\usepackage{babel}
\usepackage{some-package-that-uses-translator}

\begin{document}
...
\end{document}
```

As can be seen, things really happen behind the scenes, so, 通常, 我们无需做任何事情。其它宏包要做的事只是加载 TRANSLATOR 宏包、加载字典 (dictionaries)、翻译关键词 (keys)。

25.3.2 提供翻译

有多个命令可以告诉 TRANSLATOR 宏包所给的关键词的翻译是什么。如前所述, 作为一个普通的作者, 我们无需提供这样明确的 (explicitly) 翻译, 会自动加载这些翻译。然而, 下列两种情况需要我们提供翻译:

1. 我们不喜欢现有的翻译, 想提供新的翻译。
2. 我们正在编写一部字典。

可以使用下面的命令提供一个翻译:

```
\newtranslation[<options>]{<key>}{<translation>}
```

即:

```
\newtranslation[<选项>]{<关键词>}{<翻译>}
```

该命令定义了 *<key>* 的翻译为 *<key>*, 该翻译的语言由 *<options>* 指定。

如果翻译真是“新的 (new)”即没有给定相应语言的关键词的翻译, 这时, 我们才可以使用该命令。如果已经有了一个翻译, 则会出现一条错误信息。

可能会给出下面的 *<options>*:

- `[to]=⟨language⟩` 该选项告诉 TRANSLATOR 宏包, `⟨keys⟩` 的翻译 `⟨translation⟩` 适应于 `⟨language⟩` 语言。在一个字典文件内 (请参阅第 25.3.3 节), 该选项被自动设为字典的语言。

举例: `\newtranslation[to=German]{figure}{Abbildung}`

举例: `\newtranslation[to=German]{Figures}{Abbildungen}`

`\renewtranslation[⟨options⟩]{⟨key⟩}{⟨translation⟩}`

即:

`\renewtranslation[⟨选项⟩]{⟨关键词⟩}{⟨翻译⟩}`

该命令的作用和 `\newtranslation` 相似, 不同的是, 该命令会重定义 (redefine) 一个现有的翻译。

`\providetranslation[⟨options⟩]{⟨key⟩}{⟨translation⟩}`

即:

`\providetranslation[⟨选项⟩]{⟨关键词⟩}{⟨翻译⟩}`

该命令的作用和 `\newtranslation` 相似, 但如果已经有了一个翻译, 不会出现一条错误信息。这时, 不会改变现有的翻译。

该命令由字典的作者使用, 因为他们的翻译不会废弃 (overrule) 任何由文档的作者或其它字典作者提供的翻译。

`\deftranslation[⟨options⟩]{⟨key⟩}{⟨translation⟩}`

即:

`\deftranslation[⟨选项⟩]{⟨关键词⟩}{⟨翻译⟩}`

“无论如何” 该命令都会定义翻译。并覆盖 (overwritten) 现有的翻译。

该命令由字典的作者用于安装他们喜欢的翻译。

举例: `\deftranslation[to=German]{figure}{Figur}`

下面这个例子有一个由文档的作者提供的翻译:

```
\documentclass[ngerman]{article}

\usepackage{babel}
\usepackage{some-package-that-uses-translator}

\deftranslation[to=German]{Sketch of proof}{Beweisskizze}

\begin{document}
...
\end{document}
```

25.3.3 创建和使用字典

两种人会创建字典 (*dictionaries*)。第一种, 宏包的作者会创建包含用于宏包的 (新) 关键词翻译字典。第二种, 文档的作者会创建他们自己的私人字典, 该字典会覆盖来自其它字典的设置或, 或者该字典会提供 missing 翻译。

不仅每一种语言有一个相应的字典, 而且, 当 TRANSLATOR 宏包试图查找一个翻译时, 它可能会使用多个不同的字典。添加新翻译是很容易的: 宏包的作者可能只添加一个新字典, 该新字典包含宏包所需的关键词, 而不是更改 TRANSLATOR 的主字典 (main dictionaries) (它包含新版 translator 宏包的发行版)。

字典的命名规则如下: 字典的名称以其类型 (*kind*) 开头。类型告诉 TRANSLATOR 宏包字典包含哪一类的关键词。例如, translator-months-dictionary 这一类的字典包含像 January (注意, 这是一个关键词, 而不是翻译) 这样的关键词。类型的后面跟有一个破折号 (dash)。然后是字典文件提供的翻译的语言。最后, 字典文件的后缀名是 .dict。

上述月字典 (month dictionary) 的德语版名称是:

```
translator-months-dictionary-German.dict
```

其内容如下:

```
\ProvidesDictionary{translator-months-dictionary}{German}
```

```
\providetranslation{January}{Januar}  
\providetranslation{February}{Februar}  
\providetranslation{March}{M\ "arz}  
\providetranslation{April}{April}  
\providetranslation{May}{Mai}  
\providetranslation{June}{Juni}  
\providetranslation{July}{Juli}  
\providetranslation{August}{August}  
\providetranslation{September}{September}  
\providetranslation{October}{Oktober}  
\providetranslation{November}{November}  
\providetranslation{December}{Dezember}
```

注意, \providetranslation 命令无需 [to=German] 选项。在一个字典文件内部, TRANSLATOR 常常将默认的翻译语言 (default translation language) 设置成字典提供的语言。然而, 如果我们愿意也可以指定成其它语言。

目前, \ProvidesDictionary 命令只在 log-文件中打印一条信息。

```
\ProvidesDictionary{<kind>}{<language>}[<version>]
```

即:

```
\ProvidesDictionary{<类型>}{<语言>}[<版本>]
```

目前, 该命令只在 log-文件中打印一条信息。格式和 L^AT_EX 的 \ProvidesPackage 命令相同。

字典以一种分散的方式 (decentralized manner) 存储: 某个宏包的特定字典存储于与该宏包有密切关系的某个地方。因此, 必须告诉 TRANSLATOR 宏包需要加载哪一类型 (*kinds*) 的字典, 需要使用哪一种语言 (*languages*)。可以使用下面的命令实现这一点:

`\usedictionary{<kind>}`

即:

`\usedictionary{<类型>}`

该命令告诉 `translator` 宏包，在文档的起始处应该加载 `<kind>` 类型的全部字典，这些字典的语言是文档使用的语言。注意，不会立即加载字典，只会在文档的起始处加载字典。

如果不存在一种语言的给定 `kind` 类型的字典，则 `nothing bad happens`。

可以累积 (accumulate) 调用该命令，也就是说，我们可以为不同的字典多次调用该命令。

`\uselanguage{<list of languages>}`

即:

`\uselanguage{<语言列表>}`

该命令告诉 `translator` 宏包应该加载 `<list of languages>` 列举的所有语言的字典。在文档的起始处加载这些字典。

下面的例子显示了所有的工作：假设我们想创建一个画图的 (drawing) 新宏包，如棋盘 (chess boards)。让我们调用 `chess` 宏包。在 `chess.sty` 文件中我们可写入：

```
// This is chess.sty

\RequirePackage{translator}
\usedictionary{chess}

...

\newcommand\MoveKnight[2]{%
  ...
  \translate{knight}
  ...
}
```

现在我们可以像下面这样创建字典：

```
// This is chess-German.dict
\ProvidesDictionary{chess}{German}

\providetranslation{chess}{Schach}
\providetranslation{knight}{Springer}
\providetranslation{bishop}{L\ "aufer}
...
```

和

```
// This is chess-English.dict
\ProvidesDictionary{chess}{English}
```

```

\providetranslation{chess}{chass}
\providetranslation{knight}{knight}
\providetranslation{bishop}{bishop}
...

```

下面是一些注意事项:

- `chess.sty` 宏包不使用 `\uselanguage` 命令。毕竟, 宏包不知道 (或不关心) 最终文档所使用的语言。只需告诉 `TRANSLATOR` 宏包应使用 `chess` 字典即可。
- 我们可能想知道为什么需要一个英文字典。毕竟, 如果没有其它可用的翻译, 则最终返回关键词自身。该问题的答案是, 首先, 可以像处理其它语言一样处理英语。第二, 在某些场合, 有比关键词自身“更好”的英语翻译。如下例所示。
- 我们选择的关键词可能不是最佳的 (optimal)。如果某些关于中世纪建筑 (medieval architecture) 的宏包需要 knights (骑士) 和 bishops (主教) 的翻译, 这时会发生什么呢? 然而, 在不同的语境 (context) 中, knight 和 bishop 的翻译完全不同, 即 Ritter 和 Bischof。

这样, 为关键词添加某些东西以使“chess bishop”的含义更清晰可能是个好主意:

```

// This is chess-German.dict
\providetranslation{knight (chess)}{Springer}
\providetranslation{bishop (chess)}{L\ "aufer}

// This is chess-English.dict
\providetranslation{knight (chess)}{knight}
\providetranslation{bishop (chess)}{bishop}

```

25.3.4 创建用户字典

建立翻译的个人设置有两种方式。第一种, 可以在我们个人的宏文件 (macro files) 中简单地添加如下的命令:

```

\deftranslation[to=German]{figure}{Figur}

```

第二种, 我们可以像下面这样创建个人的字典文件: 在我们的文档中声明:

```

\documentclass[ngerman]{article}

\usepackage{translator}
\usedictionary{my-personal-dictionary}

```

然后, 在 T_EX 能找到的地方创建如下的文件:

```

// This is file my-personal-dictionary-German.dict
\ProvidesDictionary{my-personal-dictionary}{German}

\deftranslation{figure}{Figur}

```

25.3.5 翻译关键词

一旦设置好了字典和语言，我们就可以使用下面的命令翻译关键词：

```
\translate[<options>]{<key>}
```

即：

```
\translate[<选项>]{<关键词>}
```

该命令会在当前位置将 *<key>* 的翻译插入到文本 (text) 中。该命令是稳健的 (robust)。

<key> 的翻译过程如下：TRANSLATOR 宏包在当前的语言路径 (*language path*) (请参阅第 25.3.6 节) 中迭代 (iterate) 所有语言。对于语言路径中的每一种语言，TRANSLATOR 宏包会查找是否有可用的翻译用于 *<key>*，如果有，则使用该翻译。如果语言路径的任何一种语言都没有可用的翻译，则使用 (*<关键词 (key)>*) 自身作为其翻译。

举例：`\caption{\translate{Figure}~2.}`

可能会给出下面的选项：

- `[to]=<language>` 该选项会废弃 (overrides) 语言路径的设置并安装 *<language>* 作为目标语言 (target language)，该目标语言就是 TRANSLATOR 试图查找的翻译的语言。

```
\translatelet[<options>]{<macro>}{<key>}
```

即：

```
\translatelet[<选项>]{<宏>}{<关键词>}
```

该命令的工作方式和 `\translate` 命令相像，不同的仅仅是不会在文本 (text) 中插入翻译，但会为 `\translate` 命令查找到的翻译设置 *<macro>* 宏。

举例：`\translatelet\localfigure{figure}`

25.3.6 语言路径和语言替代

```
\languagepath{<language path>}
```

即：

```
\languagepath{<语言路径>}
```

该命令设置当 TRANSLATOR 查找关键词时所使用的语言路径 (*language path*)。

语言路径的默认值是 `\language`，`English`。`\language` 是标准的 T_EX 宏，它扩展了当前语言 (*current language*)。通常，该值正是我们所需要的，而无需更改该默认的语言路径。

在 `\language` 宏中使用名称存在问题。这些名称如 `ngerman`，不是 TRANSLATOR 使用的名称，我们不得不设法告诉 TRANSLATOR 隐含语言名 (*cryptic language names*) 如 `ngerman` 的别名 (*alias*)，这可以通过以下的命令实现：

```
\languagealias{<name>}{<language list>}
```

即:

```
\languagealias{<名称>}{<语言列表>}
```

该命令告诉 TRANSLATOR，必须用 *<language list>* 中的语言替代语言 *<name>*。

举例: `\languagealias{ngerman}{German}`

举例: `\languagealias{german}{German1997,German}`

对于 babel 宏包使用的语言，会自动建立其别名 (alias)，因此，我们无需调用 `\languagepath` 或 `\languagealias`。

25.3.7 宏包加载过程

TRANSLATOR 宏包的加载过程是“分阶段的 (in stages)”:

1. 首先，某些宏包或文档的作者请求 (request) 加载 TRANSLATOR 宏包。
2. TRANSLATOR 宏包允许给定像 `ngerman` 这样的选项。这些选项要求必需的别名 (alias) 和正确的 TRANSLATOR 语言。
3. 在导言区，宏包和文档的作者请求 (request) 创建要使用的字典类型 (dictionary kinds) 和特定的语言。
4. 在文档的起始处 (`\begin{document}`) 加载请求的 `dictionary-language-pairs`。

首先要做的事是加载宏包。通常，这由其它宏包自动完成，但我们可能希望直接包含 (include) 宏包:

```
\usepackage{translator}
```

当加载宏包时，我们可能 (多次) 指定 babel 语言作为 *<options>*。给定这样一个选项的作用如下: 为使翻译的 babel 语言名适应 TRANSLATOR 的语言名，让 TRANSLATOR 宏包调用 `\uselanguage`。也会调用 `\languagealias`。

索引

很抱歉，该索引只包含自动产生的条目（entry）。一份好的索引应包含精心挑选的词汇（keywords）。

<professional font package> package, 13

10pt class option, 238

11pt class option, 238

12pt class option, 238

14pt class option, 238

17pt class option, 238

20pt class option, 238

8pt class option, 238

9pt class option, 238

abstract color/font, 156

abstract environment, 156

abstract begin template, 157

abstract end template, 157

abstract title template/color/font, 156

actionenv environment, 99

albatross color theme, 214

AlDraTeX package, 9

alertblock environment, 141

alerted text color/font, 139

alerted text begin template, 139

alerted text end template, 139

alertenv environment, 139

alltt package, 9

altenv environment, 92

always typeset second mode beamer option, 262

amsthm package, 9

AnnArbor presentation theme, 182

Antibes presentation theme, 184

aspectratio=149 class option, 82

aspectratio=1610 class option, 82

aspectratio=169 class option, 82

aspectratio=32 class option, 82

aspectratio=43 class option, 82

aspectratio=54 class option, 82

babel package, 9, 10

background template/color/font, 81

background canvas template/color/font, 80

Beamer colors

 abstract, 156

 abstract title, 156

 alerted text, 139

 background, 81

 background canvas, 80

 bibliography entry author, 117

 bibliography entry location, 117

 bibliography entry note, 118

 bibliography entry title, 117

 bibliography item, 118

 block body, 141

 block body alerted, 142

 block body example, 143

 block title, 141

 block title alerted, 142

 block title example, 143

 button, 121

 button border, 121

 caption, 152

 caption name, 153

 description item, 135

 description 条目, 135

 enumerate item, 133

 enumerate mini template, 133

 enumerate subitem, 133

 enumerate subsubitem, 133

 enumerate 列表条目, 133

 enumerate 小小条目, 133

 enumerate 小条目, 133

 enumerate 小模板, 133

 example text, 230

 footline, 67

 footnote, 160

 footnote mark, 160

 framesubtitle, 79

 frametitle, 78

 frametitle continuation, 60

 headline, 65

 item, 135

 item projected, 136

 itemize item, 131

 itemize subitem, 131

 itemize subsubitem, 131

itemize 列表小小条目, 131
 itemize 列表小条目, 131
 itemize 列表条目, 131
 local structure, 137
 logo, 77
 lower separation line foot, 231
 lower separation line head, 231
 math text, 227
 math text displayed, 227
 math text inlined, 227
 middle separation line foot, 231
 middle separation line head, 230
 mini frame, 72
 navigation symbols, 76
 normal text, 229
 normal text in math text, 228
 note page, 245
 page number in head/foot, 68
 palette primary, 228
 palette quaternary, 229
 palette secondary, 228
 palette sidebar primary, 229
 palette sidebar quaternary, 229
 palette sidebar secondary, 229
 palette sidebar tertiary, 229
 palette tertiary, 228
 part page, 111
 qed symbol, 146
 qed 符号, 146
 quotation, 158
 quote, 159
 section in head/foot, 72
 section in sidebar, 73
 section in sidebar shaded, 73
 section in toc, 106
 section in toc shaded, 106
 separation line, 230
 sidebar left, 69
 sidebar right, 69
 structure, 137
 subitem, 136
 subitem projected, 136
 subsection in head/foot, 74
 subsection in sidebar, 74
 subsection in toc, 107
 subsection in toc shaded, 107
 subsubitem, 136
 subsubitem projected, 136
 subsection in head/foot, 74
 subsection in sidebar, 75
 subsection in toc, 108
 subsection in toc shaded, 108
 title page, 103
 titlelike, 230
 upper separation line foot, 231
 upper separation line head, 230
 verse, 157
 侧栏中带阴影的节, 73
 侧栏中的小小节, 75
 侧栏中的小节, 74
 侧栏中的节, 73
 分隔线, 230
 参考书目条目, 118
 参考书目条目作者, 117
 参考书目条日期刊, 117
 参考书目条目标题, 117
 参考书目条目笔录, 118
 右侧栏, 69
 块标题, 141
 块正文, 141
 导航符, 76
 封面, 103
 小小条目, 136
 小条目, 136
 局部结构, 137
 左侧栏, 69
 已显示的数学文本, 227
 帧子标题, 80
 帧标题, 78
 帧标题连续, 60
 底部导航区, 68
 底部导航区分隔线的中间, 231
 底部导航区分隔线的最上面, 231
 底部导航区分隔线的最下面, 231
 微帧, 72
 徽标, 77
 按钮, 121
 按钮边界, 121
 提醒块标题, 142
 提醒块正文, 142

提醒文本, 139

摘要, 156

摘要标题, 156

数学文本, 227

数学文本内的普通文本, 228

普通文本, 230

条目, 136

标题, 152

标题名, 153

标题类, 230

目录中的小小节, 108

目录中的小节, 107

目录中的带阴影的小小节, 108

目录中的带阴影的小节, 107

目录中的带阴影的节, 107

目录中的节, 106

示例块标题, 143

示例块正文, 143

示例文本, 230

笔记页, 245

结构, 137

背景, 81

背景画布, 80

脚注, 160

脚注符号, 160

行内数学文本, 227

被投影的小小条目, 136

被投影的小条目, 136

被投影的条目, 136

诗歌, 157

调色板侧栏第一, 229

调色板侧栏第三, 229

调色板侧栏第二, 229

调色板侧栏第四, 229

调色板第一, 228

调色板第三, 229

调色板第二, 228

调色板第四, 229

部分页, 111

顶部/底部中的小小节, 74

顶部/底部中的小节, 74

顶部/底部中的节, 72

顶部/底部的页码, 68

顶部导航区, 65

顶部导航区分隔线的上面, 230

顶部导航区分隔线的中间, 231

顶部导航区分隔线的最下面, 231

首行不缩进的引用, 159

首行缩进的引用, 158

Beamer elements, *see* Beamer templates, colors, and fonts

Beamer fonts

- abstract, 156
- abstract title, 156
- alerted text, 139
- background, 81
- background canvas, 80
- bibliography entry author, 117
- bibliography entry location, 117
- bibliography entry note, 118
- bibliography entry title, 117
- bibliography item, 118
- block body, 141
- block body alerted, 142
- block body example, 143
- block title, 141
- block title alerted, 142
- block title example, 143
- button, 121
- caption, 152
- caption name, 153
- description item, 135
- description 条目, 135
- enumerate item, 133
- enumerate mini template, 133
- enumerate subitem, 133
- enumerate subsubitem, 133
- enumerate 列表条目, 133
- enumerate 小小条目, 133
- enumerate 小条目, 133
- enumerate 小模板, 133
- example text, 230
- footline, 67
- footnote, 160
- footnote mark, 160
- framesubtitle, 79
- frametitle, 78
- frametitle continuation, 60
- headline, 65
- item, 135
- item projected, 136

itemize item, 131
 itemize subitem, 131
 itemize subsubitem, 131
 itemize 列表小小条目, 131
 itemize 列表小条目, 131
 itemize 列表条目, 131
 logo, 77
 mini frame, 72
 navigation symbols, 76
 normal text, 229
 note page, 245
 page number in head/foot, 68
 part page, 111
 qed symbol, 146
 qed 符号, 146
 quotation, 158
 quote, 159
 section in head/foot, 72
 section in sidebar, 73
 section in toc, 106
 section in toc shaded, 106
 sidebar left, 69
 sidebar right, 69
 structure, 137
 subitem, 136
 subitem projected, 136
 subsection in head/foot, 74
 subsection in sidebar, 74
 subsection in toc, 107
 subsection in toc shaded, 107
 subsubitem, 136
 subsubitem projected, 136
 subsubsection in head/foot, 74
 subsubsection in sidebar, 75
 subsubsection in toc, 108
 subsubsection in toc shaded, 108
 tiny structure, 138
 title page, 103
 titlelike, 230
 verse, 157
 侧栏中的小小节, 75
 侧栏中的小节, 74
 侧栏中的节, 73
 参考书目条目, 118
 参考书目条目作者, 117
 参考书目条目期刊, 117
 参考书目条目标题, 117
 参考书目条目笔录, 118
 右侧栏, 69
 块标题, 141
 块正文, 141
 导航符, 76
 封面, 103
 小小条目, 136
 小条目, 136
 左侧栏, 69
 帧子标题, 80
 帧标题, 78
 帧标题连续, 60
 底部导航区, 68
 微帧, 72
 微结构, 138
 徽标, 77
 按钮, 121
 提醒块标题, 142
 提醒块正文, 142
 提醒文本, 139
 摘要, 156
 摘要标题, 156
 普通文本, 230
 条目, 136
 标题, 152
 标题名, 153
 标题类, 230
 目录中的小小节, 108
 目录中的小节, 107
 目录中的带阴影的小小节, 108
 目录中的带阴影的小节, 107
 目录中的带阴影的节, 107
 目录中的节, 106
 示例块标题, 143
 示例块正文, 143
 示例文本, 230
 笔记页, 245
 结构, 137
 背景, 81
 背景画布, 80
 脚注, 160
 脚注符号, 160
 被投影的小小条目, 136

- 被投影的小条目, 136
- 被投影的条目, 136
- 诗歌, 157
- 部分页, 111
- 顶部/底部中的小小节, 74
- 顶部/底部中的小节, 74
- 顶部/底部中的节, 72
- 顶部/底部的页码, 68
- 顶部导航区, 65
- 首行不缩进的引用, 159
- 首行缩进的引用, 158

Beamer options

- `always typeset second mode`, 262
- `hide notes`, 247
- `previous slide on second screen`, 262
- `second mode text on second screen`, 261
- `show notes`, 247
- `show notes on second screen`, 247
- `show only notes`, 248

Beamer templates

- `abstract begin`, 157
- `abstract end`, 157
- `abstract title`, 156
- `alerted text begin`, 139
- `alerted text end`, 139
- `background`, 81
- `background canvas`, 80
- `bibliography entry author`, 117
- `bibliography entry location`, 117
- `bibliography entry note`, 118
- `bibliography entry title`, 117
- `bibliography item`, 118
- `block alerted begin`, 142
- `block alerted end`, 142
- `block begin`, 141
- `block end`, 141
- `block example begin`, 143
- `block example end`, 143
- `button`, 121
- `caption`, 152
- `description item`, 135
- `description` 条目, 135
- `enumerate item`, 133
- `enumerate mini template`, 133
- `enumerate subitem`, 133
- `enumerate subsubitem`, 133
- `enumerate` 列表条目, 133
- `enumerate` 小小条目, 133
- `enumerate` 小条目, 133
- `enumerate` 小模板, 133
- `footline`, 67
- `footnote`, 160
- `frame begin`, 64
- `frame end`, 64
- `frametitle`, 78
- `frametitle continuation`, 60
- `headline`, 65
- `itemize item`, 131
- `itemize subitem`, 131
- `itemize subsubitem`, 131
- `itemize` 列表小小条目, 131
- `itemize` 列表小条目, 131
- `itemize` 列表条目, 131
- `itemize/enumerate body begin`, 134
- `itemize/enumerate body end`, 134
- `itemize/enumerate` 主体开始, 134
- `itemize/enumerate` 主体结束, 134
- `logo`, 77
- `mini frame`, 72
- `mini frame in current subsection`, 72
- `mini frame in other subsection`, 72
- `navigation symbols`, 76
- `note page`, 245
- `part page`, 111
- `qed symbol`, 146
- `qed` 符号, 146
- `quotation begin`, 158
- `quotation end`, 158
- `quote begin`, 159
- `quote end`, 159
- `section in head/foot`, 72
- `section in head/foot shaded`, 73
- `section in sidebar`, 73
- `section in sidebar shaded`, 73
- `section in toc`, 106
- `section in toc shaded`, 106
- `sidebar canvas left`, 69
- `sidebar canvas right`, 70
- `sidebar left`, 69
- `sidebar right`, 69

structure begin, 138
structure end, 138
subsection in head/foot, 74
subsection in head/foot shaded, 74
subsection in sidebar, 74
subsection in sidebar shaded, 74
subsection in toc, 107
subsection in toc shaded, 107
subsubsection in head/foot, 74
subsubsection in head/foot shaded, 75
subsubsection in sidebar, 75
subsubsection in sidebar shaded, 75
subsubsection in toc, 108
subsubsection in toc shaded, 108
theorem begin, 147
theorem end, 148
title page, 103
verse begin, 157
verse end, 158
 侧栏中带阴影的节, 73
 侧栏中的小小节, 75
 侧栏中的小节, 74
 侧栏中的带阴影的小小节, 75
 侧栏中的带阴影的小节, 74
 侧栏中的节, 73
 其它小节的微帧, 72
 参考书目条目, 118
 参考书目条目作者, 117
 参考书目条目标题, 117
 参考书目条目目录, 118
 右侧栏, 69
 右侧栏画布, 70
 块开始, 141
 块结束, 141
 定理开始, 147
 定理结束, 149
 导航符, 76
 封面, 103
 左侧栏, 69
 左侧栏画布, 69
 帧开始, 64
 帧标题, 78
 帧标题连续, 60
 帧结束, 64
 底部导航区, 68
 当前小节的微帧, 72
 微帧, 72
 徽标, 77
 按钮, 121
 提醒块开始, 142
 提醒块结束, 142
 提醒文本开始, 139
 提醒文本结束, 139
 摘要开始, 157
 摘要标题, 156
 摘要结束, 157
 标题, 152
 目录中的小小节, 108
 目录中的小节, 107
 目录中的带阴影的小小节, 108
 目录中的带阴影的小节, 107
 目录中的带阴影的节, 107
 目录中的节, 106
 示例块开始, 143
 示例块结束, 143
 笔记页, 245
 结构开始, 138
 结构结束, 138
 背景, 81
 背景画布, 80
 脚注, 160
 诗歌开始, 158
 诗歌结束, 158
 部分页, 111
 顶部/底部中的小小节, 74
 顶部/底部中的小节, 74
 顶部/底部中的带阴影的小小节, 75
 顶部/底部中的带阴影的节, 73
 顶部/底部中的带阴影的小节, 74
 顶部/底部中的节, 72
 顶部导航区, 65
 首行不缩进的引用开始, 159
 首行不缩进的引用结束, 159
 首行缩进的引用开始, 158
 首行缩进的引用结束, 158
beamerarticle package, 252
beamerboxesrounded environment, 151
beamercolorbox environment, 149
beamerfoils package, 279

- beamerprosper package, 269
- beamerseminar package, 275
- beamertexpower package, 281
- beaver color theme, 218
- beetle color theme, 215
- Bergen presentation theme, 180
- Berkeley presentation theme, 185
- Berlin presentation theme, 188
- bibliography entry author template/color/font, 117
- bibliography entry location template/color/font, 117
- bibliography entry note template/color/font, 118
- bibliography entry title template/color/font, 117
- bibliography item template/color/font, 118
- bigger class option, 238
- block environment, 140
- block alerted begin template, 142
- block alerted end template, 142
- block begin template, 141
- block body color/font, 141
- block body alerted color/font, 142
- block body example color/font, 143
- block end template, 141
- block example begin template, 143
- block example end template, 143
- block title color/font, 141
- block title alerted color/font, 142
- block title example color/font, 143
- blocks parent template, 140
- Boadilla presentation theme, 181
- boldequation environment, 280
- boxes presentation theme, 179
- button template/color/font, 121
- button border color, 121

- c class option, 80
- CambridgeUS presentation theme, 182
- caption template/color/font, 152
- caption name color/font, 153
- circles inner theme, 196
- CJK package, 10
- Class options for BEAMER
 - 10pt, 238
 - 11pt, 238
 - 12pt, 238
 - 14pt, 238
 - 17pt, 238
 - 20pt, 238
 - 8pt, 238
 - 9pt, 238
 - aspectratio=149, 82
 - aspectratio=1610, 82
 - aspectratio=169, 82
 - aspectratio=32, 82
 - aspectratio=43, 82
 - aspectratio=54, 82
 - bigger, 238
 - c, 80
 - color=*(list of options)*, 10
 - compress, 71
 - draft, 26
 - envcountsect, 145
 - handout, 251
 - hyperref=*(list of options)*, 11
 - noamssymb, 147
 - noamsthm, 147
 - notheorems, 147
 - smaller, 238
 - t, 80
 - trans, 249
 - ucs, 11
 - usepdftitle=false, 106
 - utf8, 11
 - utf8x, 11
 - xcolor=*(list of options)*, 13
- Classes
 - foils, 10
 - prosper, 13
 - seminar, 13
- color package, 10
- Color themes
 - albatross, 214
 - beaver, 218
 - beetle, 215
 - crane, 216
 - default, 212
 - dolphin, 222
 - dove, 216
 - fly, 217
 - lily, 219
 - monarca, 217
 - orchid, 220

- rose, 220
- seagull, 217
- seahorse, 222
- sidebartab, 213
- spruce, 219
- structure, 213
- whale, 221
- wolverine, 218
- color=*<list of options>* class option, 10
- Colors, *see* Beamer colors
- colortbl package, 10
- column environment, 154
- columns environment, 153
- compress class option, 71
- conference-talks/conference-ornate-20min solution, 41
- Copenhagen presentation theme, 192
- crane color theme, 216

- Darmstadt presentation theme, 190
- default color theme, 212
- default font theme, 234
- default inner theme, 195
- default outer theme, 198
- default presentation theme, 179
- definition environment, 145
- deluxetable package, 10
- description environment, 134
- description item template/color/font, 135
- description 条目 template/color/font, 135
- dolphin color theme, 222
- dove color theme, 216
- draft class option, 26
- DraTex package, 10
- Dresden presentation theme, 189

- EastLansing presentation theme, 182
- Elements, *see* Beamer templates, colors, and fonts
- enumerate environment, 131
- enumerate package, 10
- enumerate item template/color/font, 133
- enumerate items parent template, 132
- enumerate mini template template/color/font, 133
- enumerate subitem template/color/font, 133
- enumerate subsubitem template/color/font, 133
- enumerate 列表条目 parent template, 132
- enumerate 列表条目 template/color/font, 133
- enumerate 小小条目 template/color/font, 133
- enumerate 小条目 template/color/font, 133
- enumerate 小模板 template/color/font, 133
- enumstep environment, 273
- envcountsect class option, 145
- Environments
 - abstract, 156
 - actionenv, 99
 - alertblock, 141
 - alertenv, 139
 - altenv, 92
 - beamerboxesrounded, 151
 - beamercolorbox, 149
 - block, 140
 - boldequation, 280
 - column, 154
 - columns, 153
 - definition, 145
 - description, 134
 - enumerate, 131
 - enumstep, 273
 - example, 145
 - exampleblock, 142
 - frame, 58
 - Itemize, 273
 - itemize, 129
 - itemstep, 273
 - notes, 274
 - onlyenv, 91
 - overlayarea, 93
 - overprint, 93
 - proof, 146
 - quotation, 158
 - quote, 158, 159
 - semiverbatim, 156
 - slide, 277
 - slides, 271
 - structureenv, 138
 - thebibliography, 116
 - theorem, 144
 - verse, 157
- example environment, 145
- example text color/font, 230
- exampleblock environment, 142
- fly color theme, 217

foils class, 10
 Font themes
 default, 234
 professionalfonts, 234
 serif, 235
 structurebold, 236
 structureitalicserif, 236
 structuresmallcapserif, 237
 fontenc package, 10
 Fonts, *see* Beamer fonts
 footline template/color/font, 67
 footnote template/color/font, 160
 footnote mark color/font, 160
 fourier package, 11
 frame environment, 58
 frame begin template, 64
 frame end template, 64
 framesubtitle color/font, 79
 frametitle template/color/font, 78
 frametitle continuation template/color/font, 60
 Frankfurt presentation theme, 190

 generic-talks/generic-ornate-15min-45min solution, 41
 Goettingen presentation theme, 187

 HA-prosper package, 11
 handout class option, 251
 Hannover presentation theme, 188
 headline template/color/font, 65
 hide notes beamer option, 247
 hyperref package, 11
 hyperref=*(list of options)* class option, 11

 Ilmenau presentation theme, 189
 infolines outer theme, 199
 inmargin inner theme, 197
 Inner themes
 circles, 196
 default, 195
 inmargin, 197
 rectangles, 196
 rounded, 197
 inputenc package, 11
 item color/font, 135
 item projected color/font, 136
 Itemize environment, 273
 itemize environment, 129
 itemize item template/color/font, 131
 itemize items parent template, 130
 itemize subitem template/color/font, 131
 itemize subsubitem template/color/font, 131
 itemize 列表小小条目 template/color/font, 131
 itemize 列表小条目 template/color/font, 131
 itemize 列表条目 parent template, 130
 itemize 列表条目 template/color/font, 131
 itemize/enumerate body begin template, 134
 itemize/enumerate body end template, 134
 itemize/enumerate 主体开始 template, 134
 itemize/enumerate 主体结束 template, 134
 items parent template, 134
 itemstep environment, 273

 JuanLesPins presentation theme, 185

 lily color theme, 219
 listings package, 12
 local structure color, 137
 logo template/color/font, 77
 lower separation line foot color, 231
 lower separation line head color, 231
 Luebeck presentation theme, 192

 Madrid presentation theme, 181
 Malmoe presentation theme, 193
 Marburg presentation theme, 187
 math text color, 227
 math text displayed color, 227
 math text inlined color, 227
 middle separation line foot color, 231
 middle separation line head color, 230
 mini frame template/color/font, 72
 mini frame in current subsection template, 72
 mini frame in other subsection template, 72
 mini frames parent template, 71
 miniframes outer theme, 199
 monarca color theme, 217
 Montpellier presentation theme, 185
 msc package, 12
 multimedia package, 164
 musixtex package, 12

 navigation symbols template/color/font, 76
 noamssymb class option, 147
 noamsthm class option, 147

- normal text color/font, 229
- normal text in math text color, 228
- note page template/color/font, 245
- notes environment, 274
- notheorems class option, 147

- onlyenv environment, 91
- orchid color theme, 220
- Outer themes
 - default, 198
 - infolines, 199
 - miniframes, 199
 - shadow, 203
 - sidebar, 201
 - smoothbars, 200
 - smoothtree, 203
 - split, 202
 - tree, 203
- overlayarea environment, 93
- overprint environment, 93

- Packages
 - (professional font package)*, 13
 - AlDraTex, 9
 - alltt, 9
 - amsthm, 9
 - babel, 9, 10
 - beamerarticle, 252
 - beamerfoils, 279
 - beamerprosper, 269
 - beamerseminar, 275
 - beamertexpower, 281
 - CJK, 10
 - color, 10
 - colortbl, 10
 - deluxetable, 10
 - DraTex, 10
 - enumerate, 10
 - fontenc, 10
 - fourier, 11
 - HA-prosper, 11
 - hyperref, 11
 - inputenc, 11
 - listings, 12
 - msc, 12
 - multimedia, 164
 - musixtex, 12
 - pdfpages, 12
 - pstricks, 13
 - texpower, 13
 - textpos, 13
 - translator, 292
 - ucs, 13
 - xcolor, 13
 - xmpmulti, 169
- page number in head/foot color/font, 68
- palette primary color, 228
- palette quaternary color, 229
- palette secondary color, 228
- palette sidebar primary color, 229
- palette sidebar quaternary color, 229
- palette sidebar secondary color, 229
- palette sidebar tertiary color, 229
- palette tertiary color, 228
- PaloAlto presentation theme, 186
- part page template/color/font, 111
- pdfpages package, 12
- Pittsburgh presentation theme, 183
- Presentation themes
 - AnnArbor, 182
 - Antibes, 184
 - Bergen, 180
 - Berkeley, 185
 - Berlin, 188
 - Boadilla, 181
 - boxes, 179
 - CambridgeUS, 182
 - Copenhagen, 192
 - Darmstadt, 190
 - default, 179
 - Dresden, 189
 - EastLansing, 182
 - Frankfurt, 190
 - Goettingen, 187
 - Hannover, 188
 - Ilmenau, 189
 - JuanLesPins, 185
 - Luebeck, 192
 - Madrid, 181
 - Malmoe, 193
 - Marburg, 187

- Montpellier, 185
- PaloAlto, 186
- Pittsburgh, 183
- Rochester, 183
- Singapore, 191
- Szeged, 191
- Warsaw, 193
- previous slide on second screen beamer option, 262
- professionalfonts font theme, 234
- proof environment, 146
- prosper class, 13
- pstricks package, 13
- qed symbol template/color/font, 146
- qed 符号 template/color/font, 146
- quotation color/font, 158
- quotation environment, 158
- quotation begin template, 158
- quotation end template, 158
- quote color/font, 159
- quote environment, 158, 159
- quote begin template, 159
- quote end template, 159
- rectangles inner theme, 196
- Rochester presentation theme, 183
- rose color theme, 220
- rounded inner theme, 197
- seagull color theme, 217
- seahorse color theme, 222
- second mode text on second screen beamer option, 261
- section in head/foot template/color/font, 72
- section in head/foot shaded template, 73
- section in sidebar template/color/font, 73
- section in sidebar shaded template/color, 73
- section in toc template/color/font, 106
- section in toc shaded template/color/font, 106
- section/subsection in toc parent template, 115
- section/subsection in toc shaded parent template, 115
- seminar class, 13
- semiverbatim environment, 156
- separation line color, 230
- serif font theme, 235
- shadow outer theme, 203
- short-talks/speaker_introduction-ornate-2min solution, 41
- show notes beamer option, 247
- show notes on second screen beamer option, 247
- show only notes beamer option, 248
- sidebar outer theme, 201
- sidebar canvas left template, 69
- sidebar canvas right template, 70
- sidebar left template/color/font, 69
- sidebar right template/color/font, 69
- sidebartab color theme, 213
- Singapore presentation theme, 191
- slide environment, 277
- slides environment, 271
- smaller class option, 238
- smoothbars outer theme, 200
- smoothtree outer theme, 203
- Solutions
 - conference-talks/conference-ornate-20min, 41
 - generic-talks/generic-ornate-15min-45min, 41
 - short-talks/speaker_introduction-ornate-2min, 41
- split outer theme, 202
- spruce color theme, 219
- structure color theme, 213
- structure color/font, 137
- structure begin template, 138
- structure end template, 138
- structurebold font theme, 236
- structureenv environment, 138
- structureitalicserif font theme, 236
- structuresmallcapserif font theme, 237
- subitem color/font, 136
- subitem projected color/font, 136
- subsection in head/foot template/color/font, 74
- subsection in head/foot shaded template, 74
- subsection in sidebar template/color/font, 74
- subsection in sidebar shaded template, 74
- subsection in toc template/color/font, 107
- subsection in toc shaded template/color/font, 107
- subsubitem color/font, 136
- subsubitem projected color/font, 136
- subsubsection in head/foot template/color/font, 74
- subsubsection in head/foot shaded template, 75
- subsubsection in sidebar template/color/font, 75
- subsubsection in sidebar shaded template, 75
- subsubsection in toc template/color/font, 108
- subsubsection in toc shaded template/color/font, 108

Szeged presentation theme, 191

t class option, 80

Template inserts, *see* Inserts

Templates, *see* Beamer templates

texpower package, 13

textpos package, 13

thebibliography environment, 116

Themes, *see* Presentation themes

theorem environment, 144

theorem begin template, 147

theorem end template, 148

theorems parent template, 147

tiny structure font, 138

title page template/color/font, 103

titlelike color/font, 230

trans class option, 249

translator package, 292

tree outer theme, 203

ucs class option, 11

ucs package, 13

upper separation line foot color, 231

upper separation line head color, 230

usepdftitle=false class option, 106

utf8 class option, 11

utf8x class option, 11

verse color/font, 157

verse environment, 157

verse begin template, 157

verse end template, 158

Warsaw presentation theme, 193

whale color theme, 221

wolverine color theme, 218

xcolor package, 13

xcolor=*<list of options>* class option, 13

xmpmulti package, 169

侧栏中带阴影的节 template/color, 73

侧栏中的小小节 template/color/font, 75

侧栏中的小节 template/color/font, 74

侧栏中的带阴影的小小节 template, 75

侧栏中的带阴影的小节 template, 74

侧栏中的节 template/color/font, 73

其它小节的微帧 template, 72

分隔线 color, 230

参考书目条目 template/color/font, 118

参考书目条目作者 template/color/font, 117

参考书目条日期刊 template/color/font, 117

参考书目条目标题 template/color/font, 117

参考书目条目笔录 template/color/font, 118

右侧栏 template/color/font, 69

右侧栏画布 template, 70

块 parent template, 140

块开始 template, 141

块标题 color/font, 141

块正文 color/font, 141

块结束 template, 141

定理 s parent template, 147

定理开始 template, 147

定理结束 template, 149

导航符 template/color/font, 76

封面 template/color/font, 103

小小条目 color/font, 136

小条目 color/font, 136

局部结构 color, 137

左侧栏 template/color/font, 69

左侧栏画布 template, 69

已显示的数学文本 color, 227

帧子标题 color/font, 80

帧开始 template, 64

帧标题 template/color/font, 78

帧标题连续 template/color/font, 60

帧结束 template, 64

底部导航区 template/color/font, 68

底部导航区分隔线的中间 color, 231

底部导航区分隔线的最上面 color, 231

底部导航区分隔线的最下面 color, 231

当前小节的微帧 template, 72

微帧 parent template, 71

微帧 template/color/font, 72

微结构 font, 138

徽标 template/color/font, 77

按钮 template/color/font, 121

按钮边界 color, 121

提醒块开始 template, 142

提醒块标题 color/font, 142

提醒块正文 color/font, 142

提醒块结束 template, 142

提醒文本 color/font, 139

提醒文本开始 template, 139

提醒文本结束 template, 139

摘要 color/font, 156

摘要开始 template, 157

摘要标题 template/color/font, 156

摘要结束 template, 157

数学文本 color, 227

数学文本内的普通文本 color, 228

普通文本 color/font, 230

条目 color/font, 136

条目 parent template, 134

标题 template/color/font, 152

标题名 color/font, 153

标题类 color/font, 230

目录中的小小节 template/color/font, 108

目录中的小节 template/color/font, 107

目录中的带阴影的小小节 template/color/font, 108

目录中的带阴影的小节 template/color/font, 107

目录中的带阴影的节 template/color/font, 107

目录中的带阴影的节/小节 parent template, 115

目录中的节 template/color/font, 106

目录中的节/小节 parent template, 115

示例块开始 template, 143

示例块标题 color/font, 143

示例块正文 color/font, 143

示例块结束 template, 143

示例文本 color/font, 230

笔记页 template/color/font, 245

结构 color/font, 137

结构开始 template, 138

结构结束 template, 138

背景 template/color/font, 81

背景画布 template/color/font, 80

脚注 template/color/font, 160

脚注符号 color/font, 160

行内数学文本 color, 227

被投影的小小条目 color/font, 136

被投影的小条目 color/font, 136

被投影的条目 color/font, 136

诗歌 color/font, 157

诗歌开始 template, 158

诗歌结束 template, 158

调色板侧栏第一 color, 229

调色板侧栏第三 color, 229

调色板侧栏第二 color, 229

调色板侧栏第四 color, 229

调色板第一 color, 228

调色板第三 color, 229

调色板第二 color, 228

调色板第四 color, 229

部分页 template/color/font, 111

顶部/底部中的小小节 template/color/font, 74

顶部/底部中的小节 template/color/font, 74

顶部/底部中的带阴影的小小节 template, 75

顶部/底部中的带阴影的节 template, 73

顶部/底部中的带阴暗影的小节 template, 74

顶部/底部中的节 template/color/font, 72

顶部/底部的页码 color/font, 68

顶部导航区 template/color/font, 65

顶部导航区分隔线的上面 color, 230

顶部导航区分隔线的中间 color, 231

顶部导航区分隔线的最下面 color, 231

首行不缩进的引用 color/font, 159

首行不缩进的引用开始 template, 159

首行不缩进的引用结束 template, 159

首行缩进的引用 color/font, 158

首行缩进的引用开始 template, 158

首行缩进的引用结束 template, 158