通配符

使用 **?** 代表任意单个字符。例如 ???lo ， 表示 lo 前有三个字符，它可以匹配 Hello

使用 * 代表随意几个任意字符。例如 *.iso ， 代表所有 iso 格式的文件。

您可以将遍历补全和通配符结合使用，以提高效率。例如：

```
cd */   则遍历补全只补全文件夹chmview *.chm 则遍历补全只补全 chm 文件
```

任务管理

&

在命令的末尾加上一个 & 符号，表示背景任务，例如：

```
wget http://www.download.net/xxx/mp3 &
```

;

使用 ; 将多个命令连结起来，则表示任务按顺序执行

&&

使用 && 将多个命令连结起来，则表示只有前面的命令执行成功，后面的命令才能得以执行

``

`(命令)` ，如果一个命令中包含以 `` （Esc键 下方的按键）括起来的子命令，那么子命令将被优先执行，执行结果被代入上一级命令继续执行，例如创建一个以当前时间命名的文件：

```
touch `date +%m.%d_%H:%M:%S`
```

touch 命令能够创建一个文件，它的操作对象，为 date +%m%d%H%M%S 命令的输出 06.06_06:06:60 这样，我们创建了一个名为 06.06_06:06:60 的文件（六月六日六时六分刚过六十秒-_-!）

Ctrl+z

将当前 Shell 中的任务挂起,这个时候任务的状态为

```
[1]+  Stopped   xxx
```

bg

将挂起的任务背景运行。这时它的状态为

```
[1]+ xxx &
```

fg

将背景任务调到前台执行

jobs

查看背景任务，方括号中的数字为命令的任务编号

如果后台运行多个任务，您可以在 bg 或者 fg 后跟任务编号，作为操作对象，例如：

```
bg 2
```

管道、重定向

>

重定向符号，它的作用是将命令的输出重定向到一个文件中。比如我们想把命令 **ls** 的结果保存为 FileList 文件，作一个清单，我们可以使用重定向符号来完成它：

```
ls -l > FileList
```

>>

作用与 > 基本相同，不同点在于， >> 以追加的方式，将命令的输出写入文件的末尾。

<

是从文件到命令的重定向，将文件的内容作为命令的输入。

|

为管道符号，它的作用是将前一个命令的输出，作为下一个命令的输入。假设一个目录下的文件太多，使用 ls 命令不能够在屏幕中完全显示，这个时候您可以将 ls 命令的输出，通过管道符号，作为浏览器 less 的输入。就可以使用浏览器的功能翻页、查找：

```
ls -al | less
```

> 提示
> less 浏览器的键绑定几乎与 man 相同，请参阅"帮助系统"一节

脱字符

Shell 中的一些功能是通过特殊符号作为控制字符来实现的，上面已经介绍了很多了。这产生一个问题，如果一个文件名中，刚好包含了这些字符，比如 ; ，就很难对它进行操作。使用 less 浏览这个文件

```
less ;xxx
```

less 会很快返回一个错误信息，因为并没有一个文件名作为操作对象。接着，Shell 会报告，系统中没有 xxx 这个命令。

这是因为 Shell 将文件名中的 ; 解析为按顺序执行命令。

或者您的文件名以空白起始，而在 Shell 中，无论多少个空格，都将被解析为一个分隔符。您甚至不能使用命令重命名此文件。

这个时候就要用到脱字符 \ 了，它能够将一个具有特殊涵义的字符转换普通字符。上面的两个任务，可以在文件名中每个特殊字符前加一个 \ ，像这样

```
less \;xxxless \ \xxxless \;\ \&\xxx
```

> 提示
> 也可以使用 " 将文件名括起来，例如 **less "; &xxx"** ，在很多情况下，这样甚至更方便。

脱字符在 Shell 中也可以作为换行符，在一个命令的末尾添加一个 \ ，然后回车，在下一行继续输入命令剩余的部分，将一个命令拆分为多行且不影响它的执行（如果执行一个很长的命令，请将它拆分为多行以便于阅读）

事实上换行符也符合脱字符的定义。回车键有两个涵义，一个是 执行 （Enter），另一个 换行 （折线箭头）。在 Shell 中它作为控制字符 执行 ，使用脱字符后，它便代表排版字符 换行 了。