作为其上下文的一部分，每个进程都有一个当前工作目录(current working directory)来确定其在目录层次结构中的当前位置。你可以用 cd 命令来修改 shell 中的当前工作目录。

目录层次结构中的位置用路径名(pathname)来指定。路径名是一个字符串，包括一个可选斜杠，其后紧跟一系列的文件名，文件名之间用斜杠分隔。路径名有两种形式：

- 绝对路径名(absolute pathname)以一个斜杠开始，表示从根节点开始的路径。例如，在图 10-1 中，hello.c 的绝对路径名为/home/droh/hello.c。
- 相对路径名(relative pathname)以文件名开始，表示从当前工作目录开始的路径。例如，在图 10-1 中，如果/home/droh 是当前工作目录，那么 hello.c 的相对路径名就是./hello.c。反之，如果/home/bryant 是当前工作目录，那么相对路径名就是../home/droh/hello.c。

10.3　打开和关闭文件

进程是通过调用 open 函数来打开一个已存在的文件或者创建一个新文件的：

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(char *filename, int flags, mode_t mode);
                                    返回：若成功则为新文件描述符，若出错为-1。
```

open 函数将 filename 转换为一个文件描述符，并且返回描述符数字。返回的描述符总是在进程中当前没有打开的最小描述符。flags 参数指明了进程打算如何访问这个文件：

- O_RDONLY：只读。
- O_WRONLY：只写。
- O_RDWR：可读可写。

例如，下面的代码说明如何以读的方式打开一个已存在的文件：

```
fd = Open("foo.txt", O_RDONLY, 0);
```

flags 参数也可以是一个或者更多位掩码的或，为写提供给一些额外的指示：

- O_CREAT：如果文件不存在，就创建它的一个截断的(truncated)(空)文件。
- O_TRUNC：如果文件已经存在，就截断它。
- O_APPEND：在每次写操作前，设置文件位置到文件的结尾处。

例如，下面的代码说明的是如何打开一个已存在文件，并在后面添加一些数据：

```
fd = Open("foo.txt", O_WRONLY|O_APPEND, 0);
```

mode 参数指定了新文件的访问权限位。这些位的符号名字如图 10-2 所示。

作为上下文的一部分，每个进程都有一个 umask，它是通过调用 umask 函数来设置的。当进程通过带某个 mode 参数的 open 函数调用来创建一个新文件时，文件的访问权限位被设置为 mode & ~umask。例如，假设我们给定下面的 mode 和 umask 默认值：

```
#define DEF_MODE   S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH
#define DEF_UMASK  S_IWGRP|S_IWOTH
```

接下来，下面的代码片段创建一个新文件，文件的拥有者有读写权限，而所有其他的