目錄

前言

关于本书

Linux是一个很伟大的系统，除了桌面系统占有率不高以外，Linux在各个平台发挥着巨大的作用。了解Linux对工作学习生产都有很大帮助。现在我大中国的Linux爱好者也越来越多，大家也了解了很多Linux系统管理员(Linux Sys Admin)的相关知识，所以本书从Linux的系统编程为切入点了解Linux，编写Linux下的程序。注意的是本书不是讲解Linux内核的编程，而是Linux提供的系统接口的编程。

此外本内容也是为《LINUX系统编程》视频的教程文字版，旨在补充视频教程。本书的全部内容都在Linux环境下完成，logo、书、视频等也是使用Linux下的原生软件创作。

本书的中出现的代码可在这获得：LINUX系统编程源代码

本书对象

本书面向的对象不限，但前提是有一定的Linux系统的使用经历，并且有一定的C语言基础。你可以使用自己喜欢的Linux发行版，我主要推荐使用Fedora、Ubuntu或Archlinux作为学习开发的环境。

- Fedora

  > 虽然Fedora上是Redhat试用新技术的地方，但还是十分稳定的，安装方便，Linux之父Linus也是使用Fedora的。

- Ubuntu

  > 在Linux用户占有一定数量，相关学习资源多，而且是大多开发商开发的平台，当然Google的Goobuntu也是它的衍生版。

- Archlinux

  > 在众多发行版中不得不说Archlinux是wiki最丰富的，也有很好的包管理软件和aur的补充，及时的软件更新，用来学习开发十分不错。

关于作者

作者为Wyatt Jee，昵称为煎鱼大魔王，

Linux历程：由于破解wifi的原因开始第一次接触了Linux，接触的发行版为cdlinux和backtrack(现在改名为kali linux)，后来因为compiz的炫酷特性开始接触了Ubuntu发行版，也眞正开始去了解Linux的相关知识，实在难以想象当初的我一个系统可以折腾一天。和老一辈Linux爱好者不同，他们都是从centos开始的，我想是当时运维大环境下的关系吧。也正是我从Debain系的Ubuntu开始，所以所以习惯了Debain系以及GNOME体验。后来由于Ubuntu的一些问题以及想要对redhat系尝试，开始使用Fedora，体验也还不错，rpm和dnf也很好用。由于系统版本更新的方式使我开始走向滚动更新系，也开始使用了archlinux发行版，也是现在正在使用的主发行版，体验了各大DE(桌面环境)和WM(窗口管理器)，都很不错。现在的我开始制作Linux相关的学习资料，旨在为Linux和开源界带来新的活力，让更多的人了解学习Linux。

关于版权

Wyatt Jee (煎鱼大魔王)

2016年8月9日，中国浙江

Chapter 01 标准化

20世纪80年代以来，UNIX出现各种版本，程序的可移植性很差，从而恶化了生态圈。在这历史原因下，C语言和UNIX类系统开始了标准化。

1.1 C语言

C语言的标准是独立与操作系统和硬件，这一特性让C可以跨硬件，电脑cpu还是微型cpu都可以用C来编程，早期最常见的是ANSI C标准，也就是C89或者说ISO C90，而编写程序的习惯还是取决与编译器，因为标准为定义部分如何实现取决与编译器本身。所以高校C语言课程有些地方让人哭笑不得。随后有出来了C95，C99，C11标准。

1.2 POSIX标准

术语“POSIX”是可移植操作系统接口（Portable Operating System Interface)缩写，是IEEE为要在各种UNIX操作系统上运行软件，而定义API的一系列互相关联的标准的总称，其正式称呼为IEEE Std 1003，而国际标准名称为ISO/IEC 9945。此标准源于一个大约开始于1985年的项目。POSIX这个名称是由理查德·斯托曼应IEEE的要求而提议的一个易于记忆的名称。它基本上是Portable Operating System Interface (可移植操作系统接口)的缩写，而X则表明其对Unix API的传承。Linux基本上逐步实现了POSIX兼容，但并没有参加正式的POSIX认证。

1.3 SUS标准

术语“SUS”是单一UNIX规范 (Single UNIX Specification) 的缩写

1.4 LSB标准

术语LSB是Linux标准规范 (Linux Standard Base) 的缩写，LSB是一个在Linux基金会结构下对Linux发行版的联合项目，其目标使Linux操作系统匹配软件系统架构，或文件系统架构标准的规范及标准。LSB基于POSIX，统一UNIX规范及其他开放标准，在某些领域扩展它们。

1.5 可移植性

系统调用和库函数API是受标准约束的，我们可以通过宏来加入。

_POSIX_SOURCE

_POSIX_C_SOURCE

_XOPEN_SOURCE

_BSD_SOURCE

_SVID_SOURCE

_GNU_SOURCE

可以在代码前定义

```
#define _GNU_SOURCE
```

或者在编译的时候使用 -D选项

```
$ gcc -D_GNU_SOURCE program.c
```

Chapter 02 文件IO

文件I\/O是Linux下十分常见的系统调用，使用率也很高。而文件I\/O中最常见的是以下系统调用

在介绍这些系统调用时需要提到术语文件描述符**(file descriptors)**这个概念，这个不是Linux特有的概念，类UNIX都是采用这种方式描述文件。而所谓的文件描述符其实是从用非负整数来表述打开的文件，它是我们对文件操作的入口。

另外，shell启动后会占用3个特殊文件描述符，这也是POSIX.1标准定义的。

- 文件描述符0为标准输入(standard input)，POSIX名称为STDIN_FILENO；
- 文件描述符1为标准输出(standard output)，POSIX名称为STDOUT_FILENO；
- 文件描述符2为标准输错误(standard error)，POSIX名称为STDERR_FILENO；

在程序使用这三个文件描述符时，直接数字是可行的，但不推荐，使用POSIX标准定义的名字更加合理，通过导入<unistd.h>来使用。

有过shell输入输出重定向使用经历的人因该很熟悉这三个数字,我们可以查看\/dev目录下的链接情况得到间接验证。

```
$ ll /dev/std*
lrwxrwxrwx 1 root root 15 Aug 9 09:19 /dev/stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 Aug 9 09:19 /dev/stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 Aug 9 09:19 /dev/stdout -> /proc/self/fd/1
```

值得一体的是C语言库函数stdio也有I\/O函数，比如fopen()，fclose()，scanf()，fgets()，fputs()等。它们和我们这里简单I\/O调用的区别是，stdio的函数构建在系统I\/O调用的上方。

2.1 打开或同时创建文件

open相关的函数如下面所示

```
NAME
       open, openat, creat - open and possibly create a file

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

       int open(const char *pathname, int flags);
       int open(const char *pathname, int flags, mode_t mode);

       int creat(const char *pathname, mode_t mode);

       int openat(int dirfd, const char *pathname, int flags);
       int openat(int dirfd, const char *pathname, int flags, mode_t mode);

RETURN VALUE
       open(), openat(), and creat() return the new file descriptor, or -1  if
       an error occurred (in which case, errno is set appropriately).
```

- 参数**pathname**——标识要打开的文件，如果pathname是一个符号链接则会对其进行解引用。如果调用成功返回文件描述符，如果错误，返回-1，并将errno设置相应的错误标识。
- 参数**flags**——为位掩码，主要用于指定文件的访问模式：
    - O_RDONLY
    - O_WRONLY
    - O_RDWR
- 参数**mode**——为位掩码，主要用于指定文件权限:
    - S_IRWXU
    - S_IRUSR
    - S_IWUSR
    - S_IXUSR

以上 `flags` 和 `mode` 仅为部分参数，其中 `mode` 只有在 `flags` 有 `O_CREAT` 时才需使用，具体详情可通过 `man 2 open` 查手册。

以下 `open()` 的实例

> 程序代码 **2-1 chapter02/open-file.c**

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

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv)
{
    int fd;

    fd = open("file", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

以上程序没有输出，如果想可视化fd，自行打印出来即可，也可以更改 `mode` 掩码，看不同掩码对文件权限的变化。

这个是最为常规的打开文件的模式，调用完后要马上进行错误检查以及错误处理。

由于历史原因，以前的 `open()` 只有打开的功能，是不能创建新文件，而用来实现创建的使用 `creat()` 调用来实现。

```c
int creat(const char *pathname, mode_t mode);
```

所以为来向前兼容， `creat()` 还是保留来下来，但如今使用并不多见，也不推荐。而实现 `creat()` 效果的可由 `open()` 调用通过如下方式实现。

```c
fd = open(pathname, O_WRONLY | O_CREAT | O_TRUNC, mode);
```

虽然这样这样的代码量比使用 `creat()` 多，但 `open()` 更符合规范，而且对于新建的文件控制性更好，只需使用需要的 `flags` 的位掩码就行。

2.2 关闭文件

close()调用如下

```
NAME
       close - close a file descriptor

SYNOPSIS
       #include <unistd.h>

       int close(int fd);

RETURN VALUE
       close()  returns  zero on success.  On error, -1 is returned, and errno
       is set appropriately.
```

及时显示关闭我们不需要的fd很重要，文件描述符数量是有限的，绝不可犯这种系统级别的错误。

一般最大打开文件数会是系统内存的10%（以KB来计算）（称之为系统级限制），系统最多可以使用的数量可以通过以下方式查看。

```
cat /proc/sys/fs/file-max
```

内核为了不让某一个进程消耗掉所有的文件资源，其也会对单个进程最大打开文件数做默认值处理（称之为用户级限制），默认值一般是1024，使用`ulimit -n`命令可以查看。

2.3 读取文件

read()函数如下

```
NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

RETURN VALUE
       On success, the number of bytes read is returned (zero indicates end of
       file), and the file position is advanced by this number.  It is not  an
       error  if  this  number  is smaller than the number of bytes requested;
       this may happen for example because fewer bytes are actually  available
       right  now  (maybe  because we were close to end-of-file, or because we
       are reading from a pipe, or from a terminal),  or  because  read()  was
       interrupted by a signal.  See also NOTES.

       On  error,  -1  is  returned,  and errno is set appropriately.  In this
       case, it is  left  unspecified  whether  the  file  position  (if  any)
       changes.
```

`read()` 从 `fd` 对应的文件里读取 `count` 字节到 `buf` 所指向的缓冲区。如果调用成功，返回实际读取的字节数，如果遇到EOF则返回0，错误为-1。如果 `count` 比 `SSIZE_MAX` 大，结果是未指定的，要避免。

下面是 `read()` 从标准输入读取的实例

> 程序代码 **2-2 chapter02/read-stdin.c**

```c
#include <unistd.h>

#include <stdlib.h>
#include <stdio.h>

#define MAX_READ 4096

int main(int argc, char *argv[])
{
    char buffer[MAX_READ+1];
    ssize_t num_read;

    num_read = read(STDIN_FILENO, buffer, MAX_READ);
    if (num_read == -1) {
        perror("read");
        exit(EXIT_FAILURE);
    }

    buffer[num_read] = '\0';

    printf("The data you input: %s\n", buffer);

    exit(EXIT_SUCCESS);
}
```

下面是 `read()` 从文件中读取的实例

> 程序代码 **2-3 chapter02/read-file.c**

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

#include <stdlib.h>
#include <stdio.h>

#define MAX_READ 4096

int main(int argc, char *argv[])
{
    // variable for read()
    char buffer[MAX_READ+1];
    ssize_t num_read;

    // variable for open()
    int fd;

    fd = open("open-file.c", O_RDONLY);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    num_read = read(fd, buffer, MAX_READ);
    if (num_read == -1) {
        perror("read");
        exit(EXIT_FAILURE);
    }

    buffer[num_read] = '\0';
    printf("\n%s\n", buffer);

    exit(EXIT_SUCCESS);
}
```

2.4 写入文件

```
NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);


RETURN VALUE
       On  success,  the  number  of bytes written is returned (zero indicates
       nothing was written).  It is not an error if  this  number  is  smaller
       than the number of bytes requested; this may happen for example because
       the disk device was filled.  See also NOTES.

       On error, -1 is returned, and errno is set appropriately.

       If count is zero and fd refers to a  regular  file,  then  write()  may
       return  a failure status if one of the errors below is detected.  If no
       errors are detected, or error detection is not  performed,  0  will  be
       returned  without  causing  any  other effect.  If count is zero and fd
       refers to a file other than a regular file, the results are not  speci-
       fied.
```

`write()` 调用和 `read()` 类似， `buffer` 为要写入的数据的地址， `count` 为要写入的数量。

如果 `write()` 调用成功，返回实习写入的字节数；错误则返回-1。实际写入的字节数有可能小于 `count` 。

2.5 文件偏移量

文件偏移量指定是下一次 `read()` 或 `write()` 操作的起始位置，文件的第一个字节的偏移量是0。通过 `lseek()` 可以查看和改变文件偏移量的位置。

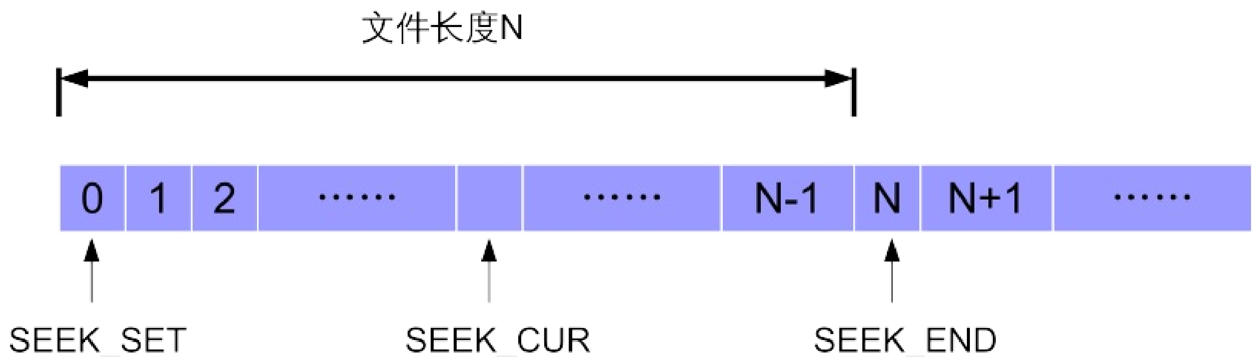
> 指定一提的是 `lseek()` 名称是历史遗留问题，早期UNIX存在 `int seek()` 和 `long lseek()` ，现在一些人认知的int和long大小都是4字节是错误的，其实C语言标准没有明确规定类型的大小，而是范围，具体大小由编译器自行处理，只不过现在int基本都为4字节了。而老一辈的人应该接触过int为2字节情况。现在在32位系统下，基本是int为4字节，long为4字节；64位系统下，int为4字节，long为8字节，如果编译为32位程序那和前面一样都为4字节。最后 `lseek()` 名字继承了下来，现在SUSv3规定为 `off_t lseek()` ，提高了可移植性。

```
NAME
       lseek — move the read/write file offset

SYNOPSIS
       #include <unistd.h>

       off_t lseek(int fildes, off_t offset, int whence);

RETURN VALUE
       Upon successful completion, the resulting offset, as measured in  bytes
       from  the beginning of the file, shall be returned. Otherwise, −1 shall
       be returned, errno shall be set to indicate the  error,  and  the  file
       offset shall remain unchanged.
```

`offset` 为相对于参考基准的偏移量，`whence` 为参考基准，为下面三种之一：

- SEEK_SET
  - 文件开头为参考基准
- SEEK_CUR
  - 以现在文件偏移量为基准
- SEEK_END
  - 以文件末尾为基准，为文件最后一个字节后面的一个字节

逻辑位置如下图所示



`lseek()` 调用成功返回新的偏移量；失败则返回-1。

2.6 文件空洞

我们可以通过 `lseek()` 到达文件之后的空间，我们还是可以进行I/O操作，比如我们在N+M的位置开始写入数据，那么N到N+M-1之间我们称之为文件空洞。这个文件空洞不会占用磁盘空间，直到后面写入以后才占用实际的磁盘空间。这个特性需要文件系统支持，比如Microsoft的vfat就不能，那么它们会把空洞部分以0写入磁盘中。

> 需要说明的是，磁盘分配空间时都是以一个块为单位分配，块的大小也是和文件系统相关的。常见的方式是1024字节、2048字节或4096字节为一个块。

具体的细节要到文件系统的i-node，才容易讲。

2.7 综合运用

现在我们有 `open()` 、 `close()` 、 `read()` 、 `write()` 和 `lseek()` 系统调用了，所以我们用这几个系统调用写一小段实例。

> 程序代码 **2-4 chapter02/basic-io.c**

```
/*
 * =====================================================================================
 *
 *       Filename:  basic-io.c
 *
 *    Description:  basic io using open() close() write() read() lseek()
 *
 *          Usage:  ./basic-io [optrion] file
 *                      -h, --help         display this help and exit
 *                      -o, --offset       set offset
 *                      -r, --read         read part of file
 *                      -w, --write        write data to file
 *
 *        Version:  1.0
 *        Created:  08/13/2016 01:10:38 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =====================================================================================
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <getopt.h>
#include <ctype.h>

static void usage_error(char *progname);

int main(int argc, char *argv[])
```

```c
{
    int fd, opt, j;
    off_t offset = 0;
    char *buf;
    ssize_t num_read, num_write;
    size_t opt_read;
    int flag_lseek = 0;
    int flag_read = 0;
    int flag_write = 0;
    const char shortopts[] = "ho:r:w:";
    const struct option longopts[] = {
        {"help",    no_argument,       0, 'h'},
        {"offset",  required_argument, 0, 'o'},
        {"read",    required_argument, 0, 'r'},
        {"write",   required_argument, 0, 'w'},
        {0, 0, 0, 0},
    };


    if (argc < 2) {
        usage_error(argv[0]);
    }

    while ((opt = getopt_long(argc, argv, shortopts, longopts, NULL)) != -1) {
        switch (opt) {
        case 'h':
            usage_error(argv[0]);
        case 'o':
            flag_lseek = 1;
            offset = (off_t) strtol(optarg, NULL, 10);
            break;
        case 'r':
            flag_read = 1;
            opt_read = (ssize_t) strtol(optarg, NULL, 10);
            break;
        case 'w':
            flag_write = 1;
            buf = optarg;
            break;

        case '?':
            break;

        default:
            fprintf(stderr, "Unexpected error\n");
            exit(EXIT_FAILURE);
        }
    }

    if ( flag_write && flag_read) {
        fprintf(stderr, "don't using read and write at same time\n");
        usage_error(argv[0]);
    }
```

```
    if (argv[optind] == NULL) {
        printf("Choose a file to open\n\n");
        usage_error(argv[0]);
    }

    fd = open(argv[optind], O_RDWR | O_CREAT,
            S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH);
    /* rw-rw-rw- */

    if (flag_lseek && lseek(fd, offset, SEEK_SET) == -1) {
        perror("lseek");
        exit(EXIT_FAILURE);
    }

    if (flag_write == 1) {
        if ((num_write = write(fd, buf, strlen(buf))) == -1) {
            perror("write");
            exit(EXIT_FAILURE);
        }

        printf("write succeeded\n");

    }

    if (flag_read == 1) {
        if ((buf = malloc(opt_read)) == NULL) {
            perror("malloc");
            exit(EXIT_FAILURE);
        }

        if ((num_read = read(fd, buf, opt_read)) == -1) {
            perror("read");
            exit(EXIT_FAILURE);
        }

        printf("[file]: %s [start position]: %ld\n", argv[optind], offset);
        for (j = 0; j < num_read; j++) {
            printf("%c", isprint(buf[j]) ? buf[j] : '~');
        }
        printf("\n");

        free(buf);
    }

    close(fd);
    exit(EXIT_SUCCESS);
}


/*
 * ===  FUNCTION  ==============================================================
 *         Name:  usage_error
```

```
 *  Description:  print usage and exit
 * =====================================================================================
 */

static void usage_error(char *progname)
{
    fprintf(stderr, "Usage: %s [optrion] file\n", progname);
    fprintf(stderr, "  -h, --help        display this help and exit\n");
    fprintf(stderr, "  -o, --offset      set offset\n");
    fprintf(stderr, "  -r, --read        read part of file\n");
    fprintf(stderr, "  -w, --write       write data to file\n");

    exit(EXIT_FAILURE);
}
```

Linux下处理命令行参数的需求是一定的，所以这里使用了 `getopt_long()` 来处理命令行参数，这个调用可以同时处理 `-h` 参数和 `--help` 参数，而 `getopt()` 只能不能处理 `--help` 大家也可以自行处理命令行的解析。这个系统调用相关如下：

```
NAME
       getopt, getopt_long, getopt_long_only, optarg, optind, opterr, optopt -
       Parse command-line options

SYNOPSIS
       #include <unistd.h>

       int getopt(int argc, char * const argv[],
                  const char *optstring);

       extern char *optarg;
       extern int optind, opterr, optopt;

       #include <getopt.h>

       int getopt_long(int argc, char * const argv[],
                  const char *optstring,
                  const struct option *longopts, int *longindex);

       int getopt_long_only(int argc, char * const argv[],
                  const char *optstring,
                  const struct option *longopts, int *longindex);

RETURN VALUE
       If an option was successfully found, then getopt() returns  the  option
       character.  If all command-line options have been parsed, then getopt()
       returns -1.  If getopt() encounters an option character that was not in
       optstring, then '?' is returned.  If getopt() encounters an option with
       a missing argument, then the return value depends on the first  charac-
       ter  in optstring: if it is ':', then ':' is returned; otherwise '?' is
       returned.

       getopt_long() and getopt_long_only() also return the  option  character
       when  a short option is recognized.  For a long option, they return val
       if flag is NULL, and 0 otherwise.  Error and -1 returns are the same as
       for  getopt(), plus '?' for an ambiguous match or an extraneous parame-
       ter.
```

大家也可以试试编写 `cp` 和 `cat` 等程序，毕竟有了上面的基础后，可以简单的实现它们的功能。

Chapter 03 文件IO+

前面我们讲解了最为基本的文件调用，这一章我们讲一些高级一点的调用以及一些逻辑上的概念。

3.1 文件控制

`fcntl()` 系统调用可以打开的文件进行更加细致的操作。

```
NAME
       fcntl — file control

SYNOPSIS
       #include <fcntl.h>

       int fcntl(int fildes, int cmd, ...);

RETURN VALUE
       Upon  successful  completion, the value returned shall depend on cmd as
       follows:

       F_DUPFD     A new file descriptor.

       F_DUPFD_CLOEXEC
                   A new file descriptor.

       F_GETFD     Value of flags defined  in  <fcntl.h>.   The  return  value
                   shall not be negative.

       F_SETFD     Value other than −1.

       F_GETFL     Value  of  file  status  flags and access modes. The return
                   value is not negative.

       F_SETFL     Value other than −1.

       F_GETLK     Value other than −1.

       F_SETLK     Value other than −1.

       F_SETLKW    Value other than −1.

       F_GETOWN    Value of the socket owner process or  process  group;  this
                   will not be −1.

       F_SETOWN    Value other than −1.

       Otherwise, −1 shall be returned and errno set to indicate the error.
```

`cmd` 参数有很多，用的最多的是

- F_GETFL，获得参数
- F_SETFL，设置参数

下面的例子演示 `fcntl()` 的基本使用

> 程序代码 **3-1 chapter03/set-flags.c**

```c
/*
 * =====================================================================================
 *
 *       Filename:  set-flags.c
 *
 *    Description:  setting the close-on-exec flag
 *
 *        Version:  1.0
 *        Created:  08/15/2016 03:01:26 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =====================================================================================
 */

#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int fd;
    int flags;

    if (open("testfile", O_RDWR) == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    if ((flags = fcntl(fd, F_GETFL)) == -1) {
        perror("fcntl get flags");
        exit(EXIT_FAILURE);
    }

    flags |= FD_CLOEXEC;
    if (fcntl(fd, F_SETFL, flags) == -1) {
        perror("fcntl set flags");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

下面再介绍一个实用的 `cmd` 参数 `F_SETLK` ，它可以用来禁止写入或读取，或者说是写入读取保护。与之匹配的参数是 `struct flock` 的结构：

- [锁的类型] `l_type`

- `F_WRLCK`
- `F_RDLCK`
- `F_UNLCK`

- [参考位置] `l_whence`
    - `SEEK_SET`
    - `SEEK_CUR`
    - `SEEK_END`
- [起始位置] `l_start`
- [锁区长度] `l_len`
- [进程身份] `l_pid`

如果 `l_whence` 为 `SEEK_SET` ， `l_start` 和 `l_len` 都为 `0` 的话，整个文件锁上。不过这里的锁是建议性锁，就是提过一个锁的标识，如果你要读取或者写入也不会阻止你。

> 程序代码 **3-2 chapter03/set-lock.c**

```c
/*
 * =============================================================================
 *
 *       Filename:  set-lock.c
 *
 *    Description:  set write lock
 *
 *        Version:  1.0
 *        Created:  08/15/2016 03:39:46 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =============================================================================
 */

#include <sys/stat.h>
#include <sys/types.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int fd;
    struct flock fl;
```

```
    if (argc < 3 || strcmp(argv[1], "--help") == 0) {
        fprintf(stderr, "Usage: %s file [option]\n", argv[0]);
        fprintf(stderr, "       -w    lock write\n");
        fprintf(stderr, "       -r    lock read\n");
        fprintf(stderr, "       -u    unlock\n");
        exit(EXIT_FAILURE);
    }

    if ((fd = open(argv[1], O_RDWR)) == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    switch (argv[2][1]) {
    case 'w':
        fl.l_type = F_WRLCK;
        fl.l_whence = SEEK_SET;
        fl.l_start = 0;
        fl.l_len = 0;
        break;
    case 'r':
        fl.l_type = F_RDLCK;
        fl.l_whence = SEEK_SET;
        fl.l_start = 0;
        fl.l_len = 0;
        break;
    case 'u':
        fl.l_type = F_RDLCK;
        fl.l_whence = SEEK_SET;
        fl.l_start = 0;
        fl.l_len = 0;
        break;
    }

    if (fcntl(fd, F_SETLK, &fl) == -1) {
        perror("fcntl set lock");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

这一次，命令行选项我们是自己解析的，没有使用 `getopt` 家族的调用，也算是另一种演示。如果要实现复杂的命令行选项，那么就会复杂，建议使用 `getopt` 家族调用更佳。

3.2 文件的三个数据结构

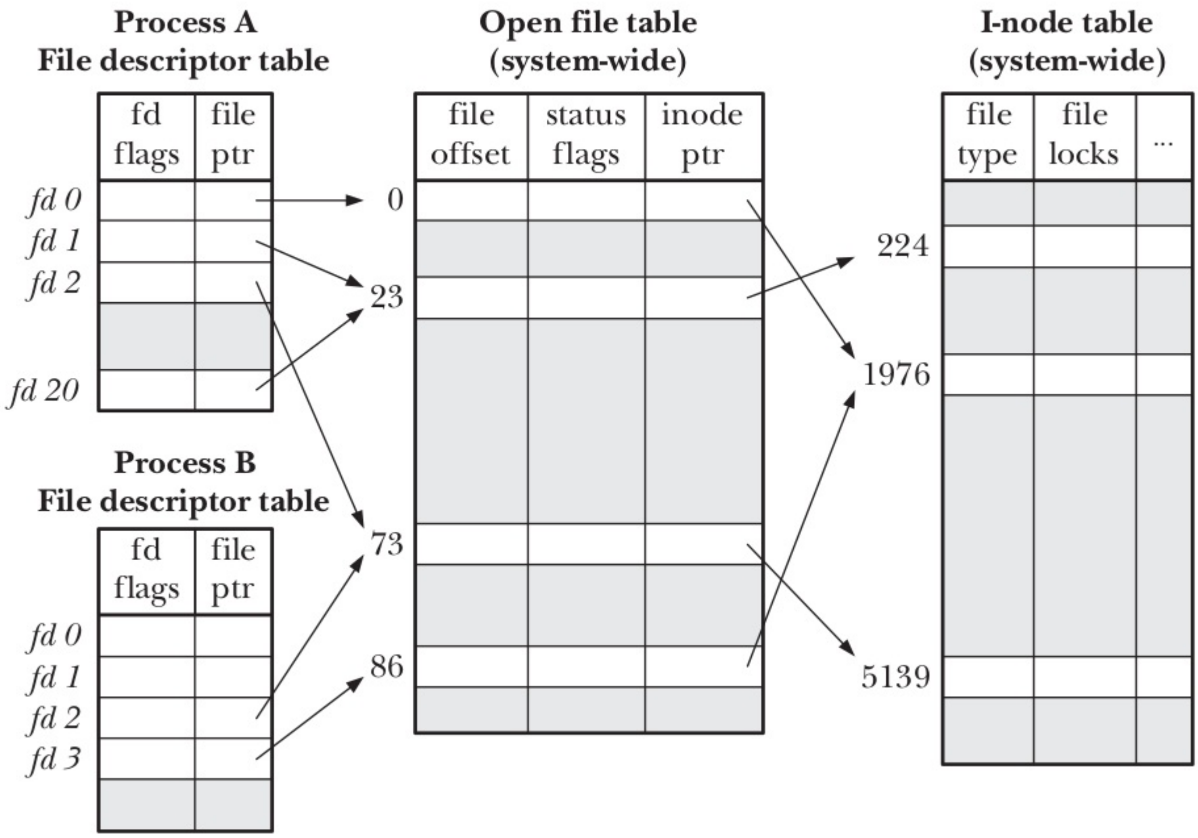
在之前，我们简单的建立起文件描述符和文件的直接对应关系，如果深入细节的话，我们发现更加奇妙的设计也就是文件的三个数据结构的表：

- 文件描述符表
    - 文件描述符flags (目前只有close-on-exec一个flag)
    - 指向打开文件指针
- 打开文件表
    - 当前文件偏移量 (offset, 可用lseek()修改)
    - 状态flags (如，open()的中的flags参数)
    - 文件访问mode (如，O_RDONLY, O_WRONLY, O_RDWR)
    - 信号驱动I/O设置
    - 指向i-node的指针
- i-node表
    - 文件类型 (常规文件，目录，符号链接等)
    - 文件属主 (用户ID，即UID)
    - 文件属组 (组ID，即GId)
    - 访问权限 (USR, GRP和OTH的权限)
    - 时间戳
        - 最后访问时间 (`$ ls -lu`)
        - 最后修改时间 (`$ ls -l`)
        - 最后改变时间 (`$ ls -lc`)
    - 指向文件的硬链数量
    - 文件大小
    - 实际分配文件块数量
    - 指向文件数据结构的指针

> 注： 一般文件系统的组成如下

引导块	超级块	**i-node**	数据块
总是作为文件系统的首块，包含用来引导操作系统的信息	在引导块之后，包含文件系统相关信息	文件系统中每个文件目录在i-node有对应的记录	存放数据以构建上层的文件目录

三个数据结构表的关系如下图所示



- 进程A的 `fd1` 和 `fd20` 都指向同一个文件引用(标号23)，这可能通过 `dup()` 家族或 `fcntl()` 形成的。
- 进程A的 `fd2` 和进程B的 `fd2` 都指向文件引用(标号73)，这可能通过 `fork()` 后形成的。
- 进程A的fd0和进程B的 `fd3` 指向不同的文件引用，但它们都指向同一个i-node结构(1976)，这可能是同一个文件进行李多次 `open()` 调用。

> 传统的UNIX既有v-node结构也有i-node结构，v-node的数据结构中包含了i-node信息。但在Linux中没有使用v-node，而使用了通用i-node。实现方式虽不同，但在逻辑概念上是一致的。

3.3 文件描述符拷贝

相关系统调用如下

```
NAME
       dup, dup2 — duplicate an open file descriptor

SYNOPSIS
       #include <unistd.h>

       int dup(int fildes);
       int dup2(int fildes, int fildes2);

RETURN VALUE
       Upon successful completion a  non-negative  integer,  namely  the  file
       descriptor,  shall  be  returned;  otherwise,  −1 shall be returned and
       errno set to indicate the error.
```

> 注：`dup(fildes);` 相当与 `fcntl(fildes, F_DUPFD, 0);`

`dup(fildes)` 会复制一个创建一个fildes的副本，文件描述符编号和fildes不一样，但两者指向同一个文件引用。

由于新建的fd都会从用可用最小的我么可用通过下面让标准错误从定向到标准输出。

```
close(2);
fd = dup(1);
```

而dup2()调用实现了上面的功能，还可以指定新的文件描述符的编号。

```
dup2(1,2);
```

下面是Linux特有的系统调用

```
SYNOPSIS
       #define _GNU_SOURCE
       #include <fcntl.h>
       #include <unistd.h>

       int dup3(int oldfd, int newfd, int flags);

RETURN VALUE
       On success, these system calls return  the  new  file  descriptor.   On
       error, -1 is returned, and errno is set appropriately.
```

`dup3()` 和 `dup2()` 一样，多了一个 `flags` 参数目前只有 `O_CLOEXEC` 一个参数。

3.3 指定偏移处IO

```
NAME
       pread, pwrite - read from or write to a file descriptor at a given off-
       set

SYNOPSIS
       #include <unistd.h>

       ssize_t pread(int fd, void *buf, size_t count, off_t offset);

       ssize_t pwrite(int fd, const void *buf, size_t count, off_t offset);

RETURN VALUE
       On success, pread() returns the number of bytes read (a return of  zero
       indicates  end  of file) and pwrite() returns the number of bytes writ-
       ten.

       Note that is not an error for a successful call to transfer fewer bytes
       than requested (see read(2) and write(2)).

       On  error, -1 is returned and errno is set to indicate the cause of the
       error.
```

`pwrite()` 和 `pread()` 功能和 `write()` 和 `read()` 基本一致，这是这两个调用可在指定 `offset` 处读写，且不改变文件的当前偏移量。在多线程对同一文件进行读写就不会出现竞争关系。

3.4 分散输入集中输出

```
NAME
       readv,  writev, preadv, pwritev - read or write data into multiple buf-
       fers

SYNOPSIS
       #include <sys/uio.h>

       ssize_t readv(int fd, const struct iovec *iov, int iovcnt);

       ssize_t writev(int fd, const struct iovec *iov, int iovcnt);

       ssize_t preadv(int fd, const struct iovec *iov, int iovcnt,
                      off_t offset);

       ssize_t pwritev(int fd, const struct iovec *iov, int iovcnt,
                       off_t offset);

       ssize_t preadv2(int fd, const struct iovec *iov, int iovcnt,
                       off_t offset, int flags);

       ssize_t pwritev2(int fd, const struct iovec *iov, int iovcnt,
                        off_t offset, int flags);


RETURN VALUE
       On success, readv(), preadv() and preadv2() return the number of  bytes
       read;  writev(),  pwritev()  and  pwritev2() return the number of bytes
       written.

       Note that is not an error for a successful call to transfer fewer bytes
       than requested (see read(2) and write(2)).

       On error, -1 is returned, and errno is set appropriately.
```

其中的 `struct iovec` 如下

```
struct iovec {
    void  *iov_base;    /* Starting address */
    size_t iov_len;     /* Number of bytes to transfer */
};
```

> 程序代码 **3-3 chapter03/scatter-gather-io.c**

以下是对两个调用的实例，其中把写入的文件默认为标准输出(STDOUT_FILENO)，也可写入特定的文件。

> 为了方便，从这里开始，我们写了通用的头文件，并建立的静态共享库(即**.a**文件，为**.o**文件的打包文件，而动态库是***.so**文件**)**。相关文件在：通用头文件，这里我们在**“Appendix** 甲**”**列出相关文件。

```
/*
 * =====================================================================================
 *
 *       Filename:  scatter-gather-io.c
 *
 *    Description:  a demonstrate of scatter-gathert I/O
 *
 *        Version:  1.0
 *        Created:  08/18/2016 03:55:28 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =====================================================================================
 */

#include "common.h"

#include <sys/uio.h>
#include <fcntl.h>
#include <getopt.h>

int main(int argc, char *argv[])
{
    int opt;
    int fd_read;
    int fd_write = STDOUT_FILENO;
    ssize_t num_read;
    static char buf0[10];
    static char buf1[20];
    static char buf2[30];

    int iovcnt;
    struct iovec iov[3];

    static struct option long_option[] = {
        {"help",     no_argument,        0, 'h'},
        {"read",    required_argument,    0, 'r'},
        {"write",    required_argument,    0, 'w'},
        {0, 0, 0, 0}
    };

    const struct help help[] = {
        {'h', "help", "show the usage and exit"},
        {'r', "readv", "readv file"},
        {'w', "writev", "writev file"},
        {0, 0, 0},
    };

    while ((opt = getopt_long(argc, argv, "hr:w:", long_option, NULL)) != -1) {
```

```
        switch (opt) {
            case 'r':
                if ((fd_read = open(optarg, O_RDONLY)) == -1)
                    err_exit("open file for reading");
                break;
            case 'w':
                if ((fd_write = open(optarg, O_RDWR)) == -1)
                    err_exit("open file for writing");
                break;
            case 'h':
                /* break through */
            default:
                usage(help, "Usage: %s -r[file] -w[file]\n", argv[0]);
        }
    }

    iov[0].iov_base = buf0;
    iov[0].iov_len = sizeof(buf0);
    iov[1].iov_base = buf1;
    iov[1].iov_len = sizeof(buf1);
    iov[2].iov_base = buf2;
    iov[2].iov_len = sizeof(buf2);

    iovcnt = sizeof(iov) / sizeof(struct iovec);

    if ((num_read = readv(fd_read, iov, iovcnt)) == -1)
        err_exit("readv");

    if ((num_read = writev(fd_write, iov, iovcnt)) == -1)
        err_exit("writev");

    exit(EXIT_SUCCESS);
}
```

3.5 临时文件

临时文件是经常需要使用到的，为此提供了相对于的系统调用。

```
NAME
       mkstemp, mkostemp, mkstemps, mkostemps - create a unique temporary file

SYNOPSIS
       #include <stdlib.h>

       int mkstemp(char *template);

       int mkostemp(char *template, int flags);

       int mkstemps(char *template, int suffixlen);

       int mkostemps(char *template, int suffixlen, int flags);


RETURN VALUE
       On success, these functions return the file descriptor of the temporary
       file.  On error, -1 is returned, and errno is set appropriately.
```

这里就简单介绍一最后下 `mkstemp()` 这个调用，其参数 `template` 说指向的文件最后的6个字符必须是XXXXXX。这6个字符会被替换，以保证文件名的唯一性。创建的文件权限为0600，即只供拥有者进行读写。

3.6 大文件IO

这个就目前为止，已经没有什么意义了。因为 `off_t` 为有符号长整型，在32位系统下long为4字节，所以文件限制与$$2^{31}-1$$个字节，也就是2GB。而64位系统为$$2^{63}-1$$个字节，也就是8EB(EB = 1024PB = 1024*1024 TB)所以64位系统就现在根本没有压力，而且Linux系统已经慢慢放弃32位系统了，32位系统已为少数。

但还是说以下32位系统下如何使用大文件，LFS的支持，同样和文件系统有很大的关系，对于微软的vfat格式就然并软了。Linux下原生的文件系统都是支持的。

启用大文件支持我们需用启用宏 `_LARGEFILE64_SOURCE` ，之后系统调用和基本的一样，就多了64，如 `open64(), off64_t`

这一部分也就不写演示代码了，用处实在不大。

> 就此和文件IO相关的章节也就结束，当然还有还多没有讲到的部分，文件的各种flags有很多特性，这些大家可以自行查看man page里的介绍。
>
> 归档与2016年8月21日

Chapter 04 文件属性

这一章我们主要来讨论Linux下的文件属性，毕竟文件在我们平时操作接触中占很大一个部分，文件属性最为直观的就是时间戳以及文件权限

4.1 获取文件信息

我们可以使用 `stat()` 家族的系统调用获取文件信息

```
NAME
       stat, fstat, lstat, fstatat - get file status

SYNOPSIS
       #include <sys/types.h>
       #include <sys/stat.h>
       #include <unistd.h>

       int stat(const char *pathname, struct stat *buf);
       int fstat(int fd, struct stat *buf);
       int lstat(const char *pathname, struct stat *buf);

       #include <fcntl.h>           /* Definition of AT_* constants */
       #include <sys/stat.h>

       int fstatat(int dirfd, const char *pathname, struct stat *buf,
                   int flags);

RETURN VALUE
       On  success,  zero is returned.  On error, -1 is returned, and errno is
       set appropriately.
```

以上的调用返回stat结构

```c
struct stat {
        dev_t     st_dev;         /* ID of device containing file */
        ino_t     st_ino;         /* inode number */
        mode_t    st_mode;        /* file type and mode */
        nlink_t   st_nlink;       /* number of hard links */
        uid_t     st_uid;         /* user ID of owner */
        gid_t     st_gid;         /* group ID of owner */
        dev_t     st_rdev;        /* device ID (if special file) */
        off_t     st_size;        /* total size, in bytes */
        blksize_t st_blksize;     /* blocksize for filesystem I/O */
        blkcnt_t  st_blocks;      /* number of 512B blocks allocated */

        /* Since Linux 2.6, the kernel supports nanosecond
           precision for the following timestamp fields.
           For the details before Linux 2.6, see NOTES. */

        struct timespec st_atim;  /* time of last access */
        struct timespec st_mtim;  /* time of last modification */
        struct timespec st_ctim;  /* time of last status change */

#define st_atime st_atim.tv_sec      /* Backward compatibility */
#define st_mtime st_mtim.tv_sec
#define st_ctime st_ctim.tv_sec
};
```

- st_dev
    - 文件所驻留带设备
- st_rdev
    - 特殊设备号码
- st_size
    - 常规文件，对应的文件字节数；符号链接，表示链接所指路径名的长度；共享内存对象，表示对象大小
- st_blocks
    - 分配给文件的总块数，以512B为单位量度。
- st_blksize
    - 针对文件系统上文件进行IO操作的最优块大小。一般为4096。
- st_atime
    - 文件最后被访问的时间
- st_mtime
    - 文件内容最后被修改的时间
- st_ctime

    - 文件状态改变时间
- st_mode
    - 该字段里面内含位掩码，起标识文件类型和指定文件权限的双重作用。与常量S_IFMT相与（&），可以析出文件类型。

常量	数值	文件类型
S_IFMT	0170000	位掩码
S_IFSOCK	0140000	套接字
S_IFLNK	0120000	符号链接
S_IFREG	0100000	常规文件
S_IFBLK	0060000	块设备
S_IFDIR	0040000	目录
S_IFCHR	0020000	字符设备
S_IFIFO	0010000	FIFO

> `lstat()` 和 `stat()` 类似，但如果文件类型为符号链接，返回是链接本身，而不是其指向的文件。
>
> `fstat()` 是使用文件描述符参数来获取文件属性。

下面我们简单的展示一下用法

> 程序代码 **4-1 chapter04\/stat.c**

```
/*
 * =============================================================================
 *
 *       Filename:  stat.c
 *
 *    Description:  a demostration of stat()
 *
 *        Version:  1.0
 *        Created:  08/24/2016 01:41:34 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =============================================================================
 */
#define _DEFAULT_SOURCE

#include "common.h"
```

```
#include <time.h>

static void show_stat( const struct stat *sp );

int main(int argc, char *argv[])
{
    struct stat status;

    if (argc != 2 || strcmp(argv[1], "--help") == 0) {
        fprintf(stderr, "Usage: %s file\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    if (stat(argv[1], &status) == -1) {
        err_exit("stat\n");
    }

    show_stat(&status);

    exit(EXIT_SUCCESS);
}

static void show_stat( const struct stat *sp )
{
    fprintf(stdout, "File Type:\t\t");

    switch (sp->st_mode & S_IFMT) {
    case S_IFREG:
        fprintf(stdout, "regular file\n");
        break;
    case S_IFDIR:
        fprintf(stdout, "directory\n");
        break;
    case S_IFCHR:
        fprintf(stdout, "character device\n");
        break;
    case S_IFBLK:
        fprintf(stdout, "block device\n");
        break;
    case S_IFLNK:
        fprintf(stdout, "symbolic link\n");
        break;
    case S_IFIFO:
        fprintf(stdout, "FIFO or pipe\n");
        break;
    case S_IFSOCK:
        fprintf(stdout, "socket\n");
        break;
    default:
        fprintf(stdout, "unknown file type\n");
        break;
    }
```

```
    fprintf(stdout, "I-node Number:\t\t%ld\n", (long) sp->st_ino);
    fprintf(stdout, "Mode:\t\t\t%lo\n", (unsigned long)sp->st_mode);
    fprintf(stdout, "Number of link:\t\t%ld\n", (long) sp->st_nlink);
    fprintf(stdout, "Oweership:\t\tUID=%ld GID=%ld\n",
            (long) sp->st_uid, (long) sp->st_gid);
    fprintf(stdout, "File size:\t\t%lld Byte\n", (long long)sp->st_size);
    fprintf(stdout, "Optimal IO block size:\t%ld Byte\n",
            (long)sp->st_blksize);
    fprintf(stdout, "number of block:\t%ld\n", (long) sp->st_blocks);
    fprintf(stdout, "Last time access:\t%s", ctime(&sp->st_atime));
    fprintf(stdout, "Last time modify:\t%s", ctime(&sp->st_mtime));
    fprintf(stdout, "Last time chage:\t%s", ctime(&sp->st_ctime));
}
```

以上是简单的使用 `stat()` 获取的信息，当然有些信息不是人可读化的，比如权限，我们更习惯“-rw-r--r--”或“0644"这样的方式，我们可以对其通过为掩码进行解析，再以可读化方式输出，这些都是后话了。

> 这一次我使用 `fprintf()` 代替 `printf()` 是因为 `printf()` 是不安全有漏洞的，相关安全问题大家可以自行Google就知道原因了。当然使用 `printf()` 不是不可以，有安全的尽量使用安全的，并养成不用 `printf()` 的习惯。

4.2 文件权限

有前面我们可以知道在 `stat` 结构里面的 `st_mode` 为位掩码，在Linux四位用来标识文件类型，12位标识权限位。布局如下图所示

在权限位中，前三位专用位图中的U、G、T位分别表示 `set-user-ID` 位、 `set-group-ID` 位、 `sticky` 位。在后面9位就是最为常见的用户、组、其他的权限位。

- User： 授予文件属主的权限
- Group： 授予文件属组成员的权限
- Other： 授予其他用户的权限

其中上面三者都有三个权限类型，也就是最常见的 `rwx`

- Read： 可读
- Write： 可写
- Execute： 可执行

我们可以通过 `ls -l` 来查看文件权限及其所有权等信息。如下面显示

```
$ ls -l
drwxr-xr-x 2 wyatt users  4096 Aug 19 09:33 assets
drwxr-xr-x 2 wyatt users  4096 Aug 17 16:45 chapter02
drwxr-xr-x 2 wyatt users  4096 Aug 20 20:48 chapter03
drwxr-xr-x 2 wyatt users  4096 Aug 28 17:08 chapter04
drwxr-xr-x 2 wyatt users  4096 Aug 21 15:24 lib
-rw-r--r-- 1 wyatt users 35147 Aug 17 18:19 LICENSES
-rw-r--r-- 1 wyatt users   186 Aug 17 16:25 makefile.inc
-rw-r--r-- 1 wyatt users  3401 Aug 20 21:50 README.md
$
```

第一位用来表示文件类型， `-` 表示是普通文件。如果是文件夹，就是 `drwxr-xr-x` 这样以d字母开头，后面9位也是对应的权限 `-` 表示无权限。下面列出各个掩码对应的值和权限位。

常量	数值	权限位
S_ISUID	04000	set-user-ID bit
S_ISGID	02000	set-group-ID bit (see below)
S_ISVTX	01000	sticky bit (see below)
S_IRWXU	00700	owner has read, write, and execute permission
S_IRUSR	00400	owner has read permission
S_IWUSR	00200	owner has write permission
S_IXUSR	00100	owner has execute permission
S_IRWXG	00070	group has read, write, and execute permission
S_IRGRP	00040	group has read permission
S_IWGRP	00020	group has write permission
S_IXGRP	00010	group has execute permission
S_IRWXO	00007	others (not in group) have read, write, and execute permission
S_IROTH	00004	others have read permission
S_IWOTH	00002	others have write permission
S_IXOTH	00001	others have execute permission

特别注意，这是在linux下对应的数值，在其他系统下实现的数值不同，所以直接用数值是很愚蠢的。

Chapter 05 进程

这一章我们主要来讨论Linux下的进程以及虚拟内存相关内容。由于进程涉及到的东西计较多，所以好分多章节来介绍，但不保证是连续的，因为有些内容需要在了解了其他内容的基础上才方便讲解。

5.1 什么是进程

狭义定义：
进程是正在运行的程序的实例（an instance of a computer program that is being executed）。

广义定义：
进程是一个具有一定独立功能的程序关于某个数据集合的一次运行活动。它是操作系统动态执行的基本单元，在传统的操作系统中，进程既是基本的分配单元，也是基本的执行单元。

5.2 较程序和线程的区别

- 程序
    - 程序是指令和数据的有序集合，其本身没有任何运行的含义，是一个静态的概念,而进程是程序在处理机上的一次执行过程，它是一个动态的概念。
    - 程序可以作为一种软件资料长期存在，而进程是有一定生命期的。程序是永久的，进程是暂时的。
    - 进程更能真实地描述并发，而程序不能；
    - 进程是由进程控制块、程序段、数据段三部分组成;
    - 进程具有创建其他进程的功能，而程序没有。
    - 同一程序同时运行于若干个数据集合上，它将属于若干个不同的进程，也就是说同一程序可以对应多个进程。
    - 在传统的操作系统中，程序并不能独立运行，作为资源分配和独立运行的基本单元都是进程。
- 线程
    - 通常在一个进程中可以包含若干个线程，它们可以利用进程所拥有的资源，
    - 在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位，
    - 由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高系统内多个程序间并发执行的程度。

- 当下推出的通用操作系统都引入了线程，以便进一步提高系统的并发性，并把它视为现代操作系统的一个重要指标。

5.3 程序信息

程序里是包含一些信息文件的，其内容如下。

- 二进制格式标识： 每个程序文件都包含用于描述可执行文件格式的元信息(metainformation)。以前常用的有两种格式，为a.out(汇编程序输出，也是gcc默认)和COFF(通用对象文件格式)。现在多为ELF(可执行连接格式)。
- 机械语言指令： 对程序算法进行编码
- 程序入口地址： 标识程序开始执行时的起始指令位置
- 数据： 程序文件包含的变量初始值和程序使用的字面常量(literal constant)值。
- 符号表及重定位表： 描述程序中函数和变量的位置和名称。
- 共享库和动态链接信息： 程序文件所包含的一些字段，列出绿程序运行时需要使用的共享库，以及加载共享库的动态链接器的路径名。
- 其他信息： 其他信息用以描述如何创建进程。

从内核中看，进程有用户内存空间(user-space memory)和一系列内核数据结构组成，其中用户内存空间包含绿程序代码及代码所使用的变量，而内核数据结构则用于维护进程状态信息。记录在内核数据结构中的信息包括许多与进程相关的标识号(IDs)、虚拟内存表、打开文件的描述表、信号传递及处理的有关信息、进程资源使用及限制、当前工作目录和大量的其他信息。

5.4 进程内存布局

在Linux内存里面一个进程划分了4个逻辑区域：text(文本段)，data(数据段)，heap(堆)，stack(栈)。

- 文本段：包含绿进程运行的程序机器语音指令。文本段具有只读属性，以防止进程意外修改指令。
- 数据段：在文本段之上，有初始化数据段和未初始数据段，更加专业的说法是用户初始化数据段**(user-initialized data segment)**和零初始化数据段**(zero-initialized data segment)**
    - 初始化数据段：包含显式初始化的全局变量和静态变量。当程序加载到内存时，从可执行文件中读取数据。
    - 未初始化数据段：包含未进行显式初始化的全局变量和静态变量。程序启动之前，这段为0。出于历史原因，这段也叫bss段，由于老版本的汇编语言助记符“block started by symbol”。
- 堆：在运行时动态分配内存的地方。堆顶端为program break.

- 栈：是动态增长和收缩的段，有栈帧(stack frames)组成。系统为每一个函数分配栈帧，用于保存函数的局部变量、实参、返回值。位于一个进程的高地址处，在x86_64 Linux上是0x7fffffffffff(131TB),也就是48bits中47bits为1。

size命令可以显示二进制可执行文件的文本段、初始化数据段、未初始化数据段的大小。

```
Here is an example of the Berkeley (default) format of output from
size:

       $ size --format=Berkeley ranlib size
       text    data    bss     dec     hex     filename
       294880  81920   11592   388392  5ed28   ranlib
       294880  81920   11888   388688  5ee50   size

This is the same data, but displayed closer to System V conventions:

       $ size --format=SysV ranlib size
       ranlib  :
       section         size         addr
       .text         294880         8192
       .data          81920       303104
       .bss           11592       385024
       Total         388392

       size  :
       section         size         addr
       .text         294880         8192
       .data          81920       303104
       .bss           11888       385024
       Total         388688
```

程序代码 **5-1 chapter05/process_segments.c**

```
/*
 * ==========================================================================================
==
 *
 *       Filename:  process_segments.c
 *
 *    Description:  demonstrate the mapping between C variable
 *                  and the segment of process.
 *
 *        Version:  1.0
 *        Created:  07/29/2017 08:28:43 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
```

```
 * ==================================================================================
==
 */

#include <stdio.h>
#include <stdlib.h>

char buffer[2048];                  /* Uninitialized data segment */
int position[] = { 3, 4, 5 };       /* Initialized data segment */

static int cube(int num)            /* Allocated in frame for cube() */
{
    int result;                     /* Allocated in frame for cube() */

    result = num * num * num;
    return result;                  /* return value passed via register */
}

static void do_cube(int num)        /* Allocated in frame for do_cube() */
{
    if (num > 10 || num < 1) {
        fprintf(stderr, "ERROR: Out of Range\n");
    }
    else {
        for(; num > 0; num--) {
            fprintf(stdout, "The cube of %2d is %4d\n", num, cube(num));
        }
    }
}

int main(int argc, char *argv[])    /* Allocated in frame for main() */
{
    static int num = 6;             /* Initialized data segment */
    char *ptr;                      /* Allocated in frame for main() */

    ptr = malloc(1024);             /* Points to memory in heap segment */

    do_cube(num);

    free(ptr);                      /* Always remember to free */

    exit(EXIT_SUCCESS);
}
```

虽然SUSv3没有规定，但在大多数的UNIX和Linux里提供了3个全局符号: `etext` 、 `edata` 、 `end` ，可以在程序内使用这些符号以获取相应程序的文本段、初始化数据段以及非初始化数据段结尾处下一个字节的地址。为了能够使用这个特性，需要进行显示的声明：

```
extern char etext, edata, end;
```

下面的程序简单来实验这三个符号

> 程序代码 **5-2 chapter05/segments_address.c**

```
/*
 * =====================================================================================
==
 *
 *       Filename:  segments_address.c
 *
 *    Description:  demonstrate the symbol: etext, edata, end
 *
 *        Version:  1.0
 *        Created:  07/29/2017 09:46:55 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =====================================================================================
==
 */

#include <stdio.h>
#include <stdlib.h>

extern char etext, edata, end;

int main(int argc, char *argv[])
{
    fprintf(stdout, "First address past:\n");
    fprintf(stdout, "    program text (stdout, etext)   %10p\n", &etext);
    fprintf(stdout, "    initialized data (edata)       %10p\n", &edata);
    fprintf(stdout, "    uninitialized data (end)       %10p\n", &end);

    exit(EXIT_SUCCESS);
}
```

由于我编译的是64位程序，所以运行输出如下

```
$ ./segments_address
First address past:
    program text (stdout, etext)   0xc87430f86d
    initialized data (edata)       0xc874510040
    uninitialized data (end)       0xc874510050
```

5.5 虚拟内存管理

进程中各个内存段都建立在虚拟内存上，而虚拟内存管理技术现在多数内核都是采用的。它使得应用程序认为它拥有连续可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。与没有使用虚拟内存技术的系统相比，使用这种技术的系统使得大型程序的编写变得更容易，对真正的物理内存（例如RAM）的使用也更有效率。这个技术利用了大多数程序的一个特点——访问局部性(locality of reference)，主要为两大局限性。

- 空间局限性**(Spatial locality)**：程序倾向于访问最近访问过的内存地址附近的地址(由于指令顺序执行)
- 时间局限性**(Temporal locality)**：程序倾向于在不久的将来再次访问最近刚访问过的内存地址(由于循环)

虚拟内存的理念是将每个程序使用的内存分割为小型且固定大小的页单元(page)，将RAM划分为一系列同虚拟页大小相同的的页帧(page frame)。无论何时，每个程序仅有部分分页需要驻留在物理内存上的页帧中。这些页构成了所谓的驻留集(resident set)，程序未使用的页拷贝保存到交换分区(swap area)中如若进程要访问的页面尚不在物理内存中，将会发生页面错误(page fault)，内核将进程挂起(suspend)，同时从磁盘中将该页面载入内存。

页面大小通常由处理器架构决定，在x86和x86_64中，分页大小为4KiB(4096bytes)。

架构	最小页大小	较大页大小
x86	4 KiB	4 MiB in PSE mode, 2 MiB in PAE mode
x86-64	4 KiB	2 MiB, 1 GiB (only when the CPU has PDPE1GB flag)
IA-64	4 KiB	8 KiB, 64 KiB, 256 KiB, 1 MiB, 4 MiB, 16 MiB, 256 MiB
Power Architecture	4 KiB	64 KiB, 16 MiB, 16 GiB
SPARC v8 with SPARC Reference MMU	4 KiB	256 KiB, 16 MiB
UltraSPARC Architecture 200	8 KiB	64 KiB, 512 KiB (optional), 4 MiB, 32 MiB (optional), 256 MiB (optional), 2 GiB (optional), 16 GiB (optional)
ARMv7	4 KiB	64 KiB, 1 MiB ("section"), 16 MiB ("supersection") (defined by a particular implementation)

对于基于Unix和POSIX的操作系统可以使用系统函数 `sysconf()` 来获取页大小

```
#include <stdio.h>
#include <unistd.h> /* sysconf(3) */

int main(void) {
    printf("The page size for this system is %ld bytes.\n",
           sysconf(_SC_PAGESIZE)); /* _SC_PAGE_SIZE is OK too. */

    return 0;
}
```

当然也可以使用命令行工具 `getconf` 来获取

```
$ getconf PAGE_SIZE
4096

# 或者

$ getconf PAGESIZE
4096
```

为了支持这一组织方式，内核需要为每一个进程维护分页表(page table)。这个分页表描述了每页在进程虚拟地址空间(virtual address space)中的位置。页表中的每一个条目要么指出一个虚拟分页在RAM中所在的位置，要么表明其当前驻留在磁盘上。

在进程虚拟内存空间中，并非所以的地址范围都需要页表条目。通常情况下，由于可能存在大段的虚拟地址空间并为投入使用，故没有必要为其维护页表条目。若进程试图访问的地址并没有页表条目与之相对应，那么进程将收到一个SIGSEGV信号。

由于内核能够为进程分配和释放页（和页表条目），所以进程的有效虚拟地址范围在其生命周期内可以发生变化，主要为以下几个场景：

- 由于栈向下增长超出之前曾达到的位置。
- 当在堆中分配或释放内存，通过调用 `brk()` 、 `sbrk()` 和 `malloc函数族` 来提升了program break的位置。
- 当调用 `shmat()` 连接System V共享内存区的时候，或者当调用 `shmat()` 脱离共享内存区的时候。
- 当调用 `mmap()` 创建内存映射的时候，或者调用 `munmap()` 解除内存映射的时候。

虚拟内存管理技术带来的好处：

- 进程和进程、进程与内核互相隔离，所以一个进程不能读取或修改另一个进程或内核的内存。
- 适当情况下，两个或多个进程能够共享内存。这是由于内核可以使用不同进程的页表条目指向相同的RAM页
    - 执行同一个程序的多个进程，可共享一份（只读）程序代码副本。当多个程序执行

  相同的程序文件（或加载相同的共享库），会隐式地实现这一类型的共享。
  - 进程可以调用 `shmat()` 和 `mmap()` 系统调用显式地请求与其他进程共享内存区，这么做是为了进程间通讯考虑的。
- 便于实现内存保护机制
- 程序员和编译器、链接器之类的工具不需要考虑程序在RAM中的物理布局。
- 由于需要驻留在内存的只是程序的一小部分，因此程序可以加载和运行的更快。而且占用内存的大小可以超过RAM的容量。
- 由于每个进程使用的RAM减少了，RAM中就能同时容纳更多的进程数。

5.6 栈与栈帧

在计算机科学中，调用堆栈是堆栈数据结构，其存储关于计算机程序的活动子程序的信息。这种堆栈也被称为执行堆栈，程序堆栈，控制堆栈，运行时堆栈或机器堆栈，并且通常被简化为“堆栈”。虽然调用堆栈的维护对于大多数软件的正常运行很重要，但是细节通常在高级编程语言中是隐藏的和自动的。

函数的调用与返回是栈的增长和收缩是线性的。在Linux下，栈驻留在内存高处并向下增长（朝堆的方向）。专用寄存器——栈指针(stack pointer)，用于跟踪当前栈顶。每次调用函数时，会在栈上分配一帧，每当函数返回，再移除。 每个栈包括如下内容：

- 函数实参和局部变量：由于这些变量都是在调用函数时自动创建的，因此在C语言中称为自动变量。函数返回时自动销毁这些变量
- 函数调用的链接信息：每个函数都会到一些CPU的寄存器，比如程序计数器，其指向下一条要执行的机器语言指令。每当一函数调用另一个函数的时，会在被调用函数的栈帧里面保存这些寄存器的副本，以便函数返回时能为函数调用者把寄存器恢复原状。

5.7 命令行参数

每个C语言程序都必须有一个 `main()` 作为程序的入口。当执行程序时，命令行参数通过两个入参提供给 `main()` 函数。第一个为 `int argc` 为命令行参数个数；第二个为 `char *argv[]` 为指向命令行参数的指针数组，每一个参数都是以空字符 `null` (\0)结尾的字符串。第一个字符串为 `argv[0]` ，且通常为程序名 `argv` 中的指针列表以 `NULL` 直接结尾（argv[argc] = NULL）。

Appendix 甲

为了以后代码方便，常用的代码和导入库，我们放到了一个通用头文件 `common.h` 里，并把其实现生成静态库 `libcommon.a` 用于以后编译，同时通过makefile文件方便管理。

通用头文件

> 代码清单 **lib/common.h**

```
/*
 * =====================================================================================
 *
 *       Filename:  common.h
 *
 *    Description:  common include file
 *
 *        Version:  1.0
 *        Created:  08/17/2016 05:14:45 PM
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *   Organization:  JianYuChuPing
 *
 * =====================================================================================
 */

#ifndef COMMON_H
#define COMMON_H

#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>

typedef enum { FALSE, TRUE } boolen;

struct help {
    char short_opt;
    const char *long_opt;
    const char *description;
};

void err_exit(char *s);
void usage(const struct help *help, const char *format, ...);

#endif
```

> 代码清单 **lib/common.c**

```
/*
 * =====================================================================================
 *
 *       Filename:  common.c
 *
 *    Description:  some common function
 *
```

```
 *         Version:  1.0
 *         Created:  08/17/2016 07:13:24 PM
 *        Revision:  none
 *        Compiler:  gcc
 *
 *          Author:  Wyatt Jee (WJ), bluesorrow221@gmail.com
 *    Organization:  JianYuChuPing
 *
 * =============================================================================
 */

#include "common.h"

void err_exit(char *s)
{
    perror(s);
    exit(EXIT_FAILURE);
}

void usage(const struct help *help, const char *format, ...)
{
    int j;
    va_list ap;

    fflush(stdout);
    va_start(ap, format);
    vfprintf(stderr, format, ap);
    va_end(ap);

    if (help[0].long_opt == NULL) {
        for (j = 0; help[j].short_opt != 0; j++) {
            fprintf(stderr, " -%c\t%s\n", help[j].short_opt,
                    help[j].description);
        }
    }
    else {
        for (j = 0; help[j].short_opt != 0; j++) {
            fprintf(stderr, " -%c, --%s\t%s\n", help[j].short_opt,
                    help[j].long_opt, help[j].description);
        }
    }

    fflush(stderr);

    exit(EXIT_FAILURE);
}
```

代码清单 **makefile.inc**

```
# This is the common definitions using by subdir makefiles

ROOT = ..
LIB_DIR = ${ROOT}/lib
LIB = ${LIB_DIR}/libcommon.a

CFLAGS = -std=c99 \
         -g -I${LIB_DIR} \
         -Wall \

RM = rm -f
```

> 代码清单 **lib\/makefile**

```
include ../makefile.inc

all : ${LIB}

${LIB} : *.c
    ${CC} -c -g ${CFLAGS} *.c
    ${AR} rs ${LIB} *.o

clean :
    ${RM} *.o ${LIB}
```

各个章节目录里的makefile

> 代码清单 **chapter02/makefile**

```
include ../makefile.inc

EXE = basic-io \
      open-file \
      read-file \
      read-stdin

all : ${EXE}

clean :
    ${RM} ${EXE}
```

> 代码清单 **chapter03/makefile**

```
include ../makefile.inc

EXE = set-flags \
      set-lock \
      scatter-gather-io

all : ${EXE}

${EXE} : ${LIB}

clean :
    ${RM} ${EXE}
```

> 代码清单 **chapter04/makefile**

```
include ../makefile.inc

EXE = stat

all : ${EXE}

${EXE} : ${LIB}

clean :
    ${RM} ${EXE}
```

> 代码清单 **chapter05/makefile**

```
include ../makefile.inc

EXE = process_segments \
      segments_address

all : ${EXE}

${EXE} : ${LIB}

clean :
    ${RM} ${EXE}
```

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical

distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally 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.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.