第二章 Unix 系统概貌

同济大学计算机系操作系统课程讲稿　邓蓉　　2022-9-10

UNIX 不仅仅是一个操作系统，更是一种生活方式。经过几十年的发展，UNIX 在技术上日臻成熟的过程中，它独特的设计哲学和美学也深深地吸引了一大批技术人员，他们维护、开发、使用 UNIX，同时 UNIX 也深深影响了他们的思维方式和看待世界的角度。UNIX 重要的设计原则：

- 简洁至上（KISS 原则）
- 提供机制而非策略

Part 1　Unix 系统 和 系统中的软硬件资源

UNIX 操作系统，是一个强大的多用户、多任务操作系统，支持多种处理器架构，按照操作系统的分类，属于分时操作系统。肯·汤姆森和丹尼斯·里奇在 1974 年 7 月号上的《ACM 通信》上的一篇论文“The UNIX Time Sharing System”正式将 unix 操作系统介绍给世人。

图 1、Unix 的作者 肯·汤普逊、丹尼斯·里奇在（AT&T 的贝尔实验室）

肯·汤普逊是 Unix 之父，丹尼斯·里奇为 Unix 设计了 C 语言并用 C 语言改写了 Unix 内核

UNIX 第六版（英语：Version 6 Unix，简称 V6），由贝尔实验室于 1975 年 5 月发布，是第一个对外公开的 UNIX 版本，主要运行在 DEC PDP-11 系列的小型计算机上。

澳大利亚新南威尔士大学 John Lions 教授，注释了 Unix V6 内核源代码，出版著作《UNIX OPERATING SYSTEM SOURCE CODE LEVEL SIX》。这便是上海交大尤晋元老师《莱昂式 UNIX 源代码分析》译文的原著。

图 2、John Lions 教授和他的 Unix 源代码分析

我校操作系统组在 i386 平台上重新实现了这份源代码。没有这本书就没有我们手上 Unix V6++系统

一、多用户 UNIX 系统的硬件

UNIX 系统是一个多用户分时系统，可以同时为很多用户服务。其硬件配置包括一台主机和许多终端。主机的硬件资源为所有用户提供计算、存储和网络数据传输能力，包括：CPU，内存，硬盘，GPU，网络 和 所有其它外部设备。

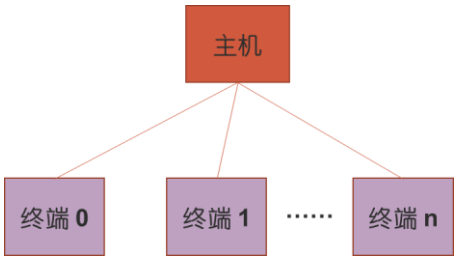

图 3、Unix 系统硬件构成（概念图）

系统为每个用户提供一个终端。每个终端带一个键盘和一个屏幕，是人机接口。用户使用键盘输入命令行、提交作业，在屏幕上查看作业的运行结果。Unix 用户使用终端与系统交互，过程如下：

- 登录（login）系统。
- 提交作业（submit job），观察作业运行结果，还可以用 ctrl+c 等等组合键向自己提交的作业发信号，对其运行过程进行控制。
- 离开系统前 logout。

主机通常是服务器，包括传统的大、中、小型机，也可以是刀片、塔式 或 机架式服务器。也可以是 PC（笔记本，台式机或工作站）。

Unix 服务器的终端，可以是 tty 物理终端，通过 modem 物理线路与主机相连，还可以是运行在客户机上的 ssh 程序，telnet 程序，这些应用程序通过网络链接（socket）向服务器提交作业，拿回作业运行结果。个人计算机上运行的 Unix 系统，如果运行在字符模式下，PC 的键盘和显示屏是其物理终端。运行在 GUI 模式下，桌面上的 terminal 窗口是伪终端。

二、Unix 系统的软件

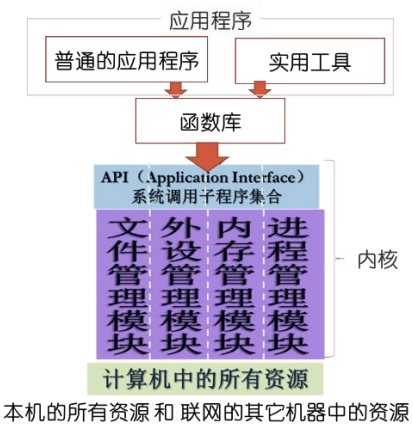

图 4、Unix 系统软件构成

Unix 系统的软件包括一个内核(kernel)、一组函数库，实用程序(Utility)和普通应用程序。

1、内核是 UNIX 操作系统自己。负责管理本机的所有资源，能够帮助应用程序访问联网的其它机器中的资源。

2、函数库。使用库函数可以减小程序员编码的工作量，提高应用程序的质量。Unix 系统一定会装标准 C 库 glibc，g 表示 GNU，lib 是 library，c 表示 C 语言。搭建开发应用系统时，还要装其它库，比如通用 GPU 编程需要装 CUDA，玩计算机视觉要装 openCV，做计算机图形学作业要装 openGL。

3、实用程序（Utility）是发行版 Unix/Linux 系统自带的标准工具。程序员编程会用到它们。包括

- 命令解释器 shell，图形用户界面 x_windows
- UNIX 系统的标准命令：ls、cat、echo、chmod、date……
- 编译链接工具 gcc、g++、ld、make……
  调试程序的工具 gdb、gprof……
  编辑器 VSCode，emacs……
- 和所有独立于应用的程序，如：JVM、DBMS

4、普通应用系统面向应用，包括

- python3
- Hadoop
- TensorFlow, PyTorch，ROS ……

Part 2　Unix 系统的 进程、用户 和 文件

进程、文件和用户是 Unix 系统最主要的概念。进程有生命，建模系统中的活动。一切皆文件，指的是进程处理的数据来自文件，运算的结果送给文件。用户标识，用来实现文件系统的访问权限。

一、用户

1、登录过程 和 花名册文件

用户使用 Unix 系统要登录。系统要求你提供合法的用户名和口令字，如图：

login：
Passwd：

图 5、Unix 系统，使用前用户需要提供用户名和口令字

登录时，系统需要使用花名册文件 /etc/passwd 验证用户身份。

在花名册文件中，每个合法用户有一条记录，记录该用户的用户名，口令字，用户标识符，初始工作目录，用户环境变量等等信息。下图是我们后续要使用的示例系统中的花名册文件。

user name (in ASCII) (string)	encrypted password	numerical user ID	numerical group ID	PATH	initial working directory	initial shell (UI program)
root	X	0	0	*******	/root	/bin/sh
……						
John	X	12	1	*******	/home/John	/bin/sh
Mary	X	13	1	*******	/home/Mary	/bin/sh
……						
Tony	X	58	1	*******	/home/Tony	/bin/x_windows

得到用户输入的用户名和口令字后，系统搜索花名册文件，寻找用户名、口令字匹配的记录。如果成功，这是一个合法用户。

2、会话（session）

会话（session）是指用户的一次上机过程，从 login 开始至 logout 结束。下图是用户 John，使用终端 tty9 的一次会话。

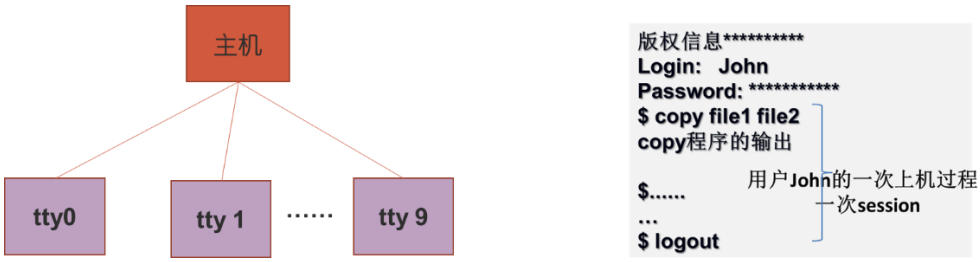

图 6、用户 John，使用终端 tty9 的一次会话

所有会话经由用户登录。用户名、口令字被验证通过后，花名册文件中该用户的 record 会被系统记录下来，作为此次会话的属性。

- uid 用户标识；例：John 用户 uid==12，Mary 用户 uid==13
- 用户家目录；例：John 用户 /home/John，Mary 用户 /home/Mary。这是会话的默认当前工作目录。cd 命令用来更改会话当前工作目录。

会话的其它重要属性还包括：会话使用的终端 tty_p。

每次会话，系统会派生一个新进程为这次会话提供字符界面服务。这个进程执行 shell 程序，命名其为 shell 进程。

- shell 进程输出命令行提示符等待用户输入命令行。
- 用户输入命令行，回车提交作业。系统无条件接纳作业，创建足够多的新进程。每个进程执行一个应用程序。

补充：命令行、作业、进程和程序的关系请参考文献《实验二 背景知识，Linux 命令行，前台作业，后台作业》。

回到图 6，记 session9 是 9 号终端正在进行的会话。会话持续期间，可能会执行很多应用程序。每执行一次应用程序，系统就会派发一个新进程。程序执行完毕，进程终止。所有这些进程拥有相同的静态属性：

- p_uid = 12 //进程的用户号
- p_ttyp = tty9 //进程的终端号
- u_cdir 是 /home/John //进程的当前工作目录

下面，我们介绍应用程序的标准输入输出。应用程序的标准输入是进程的 0#文件，标准输出是进程的 1#文件。上一章我们介绍过，内核为每个终端配一对缓存。输入缓存用来存放用户键盘输入的字符，输出缓存暂存应用程序输出至显示屏的字符串。

- 输入缓存（键盘）是 shell 和 所有应用程序的标准输入。程序执行 scanf 时，从这里读取用户输入字符。
- 输出缓存（屏幕）是 shell 和 所有应用程序的标准输出。程序执行 printf 时，会把要写到屏幕的字符先存到这里。
- 进程的 2#文件是标准错误输出文件，应用程序向该文件写入的报错信息，红字写屏幕。

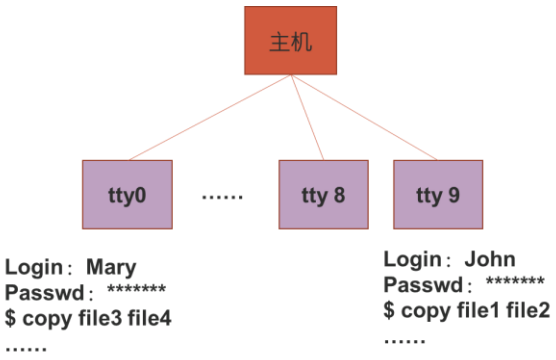

图 7、示例 Linux 系统

这是一个拥有 10 个终端的单核 UNIX 系统，并发 2 个 session，Mary 使用 tty0，John 使用 tty9。在这个时刻有 2 个 copy 进程并发，竞争使用 CPU 和磁盘。每个进程有一个 PID，是该进程系统内部的标识符。Mary 的 copy 进程，pid==100，printf/scanf 使用 tty0。John 的 copy 进程，pid==566，printf/scanf 使用 tty9。

二、进程

在多道系统中，用应用程序来描述系统正在进行的活动是不足够的。应用程序有执行它的用户，提交它的终端，还有它使用相对路径名引用文件时，目录搜索的起点。

进程是应用程序具体的一次执行过程。除了程序的代码和数据，还包括**进程控制块 PCB（Process Control Block）**。这是最重要的内核数据结构，用来登记每个进程的属性，包括以下静态字段。

- p_pid，内部引用名（进程对象的编号）
- comm，进程执行的应用程序（字符串格式的可执行文件名）
- p_uid，进程的用户名
- p_ttyp，进程使用的终端
- u_cdir，进程的当前工作目录

…… 其余属性，我们以后再介绍。

三、文件

1、文件树

UNIX/Linux 文件系统采用树型结构，只有一个树根。系统能够访问到的所有文件挂在这棵树上。下图是 Linux 文件树根目录的结构。

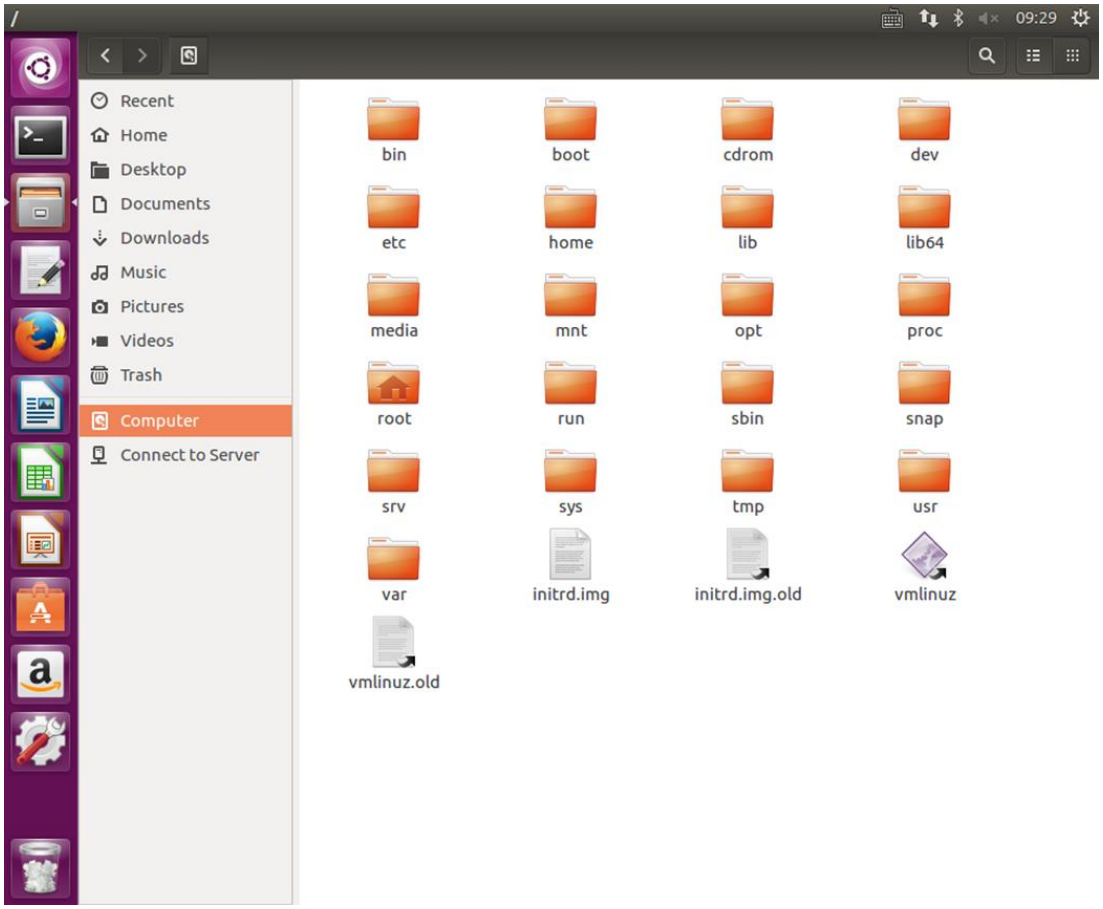

图 8、Linux 标准根目录结构

其中最关键的几个子目录的用法如下：

- /bin： 用来存放 shell（terminal）中可以使用的常用命令。这里存放的应用程序所有用户都可以用。比如，ls，cat，date，echo ……。
- /dev： 所有硬件（包括 CPU）是这个目录下的一个特殊文件。比如，图 7 所示系统有 10 个终端，对应 tty 子目录下的 10 个特殊文件/dev/tty/tty0， …… ，/dev/tty/tty9。
- /etc： 用来存放系统配置文件。比如，用户登录时，系统进行身份认证时使用的花名册文件/etc/passwd 就是重要的系统配置文件。
- /home：每个合法用户的家目录是 home 目录的一个子目录，以自己的用户名命名。比如，John 用户的家目录是/home/John。

- /lib：用来存放 Linux 函数库。这是标准 C 库：/lib/x86_64-linux-gnu/libc-2.23。
- /sbin： 管理员用户 root 维护系统使用的工具。比如，用来检查 ext2 文件系统的 fsck，就存放在这个子目录下。
- /usr： usr 是 unix software resource 的缩写。这个子目录中存放的是 Linux 源代码
- /proc： 系统中正在执行的所有进程在 proc 目录中有一个以 PID 命名的子目录。这个子目录，以只读文本文件的形式向用户/应用程序暴露进程控制块中的字段，方便用户读取。

2、文件 和 文件的引用名

文件是文件树上的节点。普通文件，是叶子节点。可能是数据文件（磁盘），可执行文件（磁盘），还可能是一个外部设备硬件。其余的节点，是目录（文件夹）。

进程引用文件给出的字符串是文件的路径名。文件路径名有 2 种。绝对路径名是基于根目录的文件名。以 '/' 开头。从树根到每个节点有唯一路径，所以，任何文件，绝对路径名唯一。

每个进程的 PCB 中登记着进程的当前工作目录 u_cdir，它是文件树上的一个节点。从这个节点出发，树上的每个节点有唯一路径。相对路径名是基于进程当前工作目录的文件名，最开始的字符不是'/'。给定具体进程，所有文件相对路径名唯一。

特殊文件名 ".." 指父目录。支持相对路径名的文件系统必须支持这个特殊文件名。沿文件树搜索文件，这个文件名可以让搜索过程向上，访问其它子树中的文件和目录。

另一个特殊文件名是 "."，指当前目录。Unix 系统执行当前目录下的应用程序，需要"./"前缀。参见实验二背景知识中执行 cpu 程序的命令行 ./cpu A。

例题：参考图 7 中的示例 Linux 系统。John 用户和 Mary 用户会话当前状态如图 8 所示。请回答以下问题：

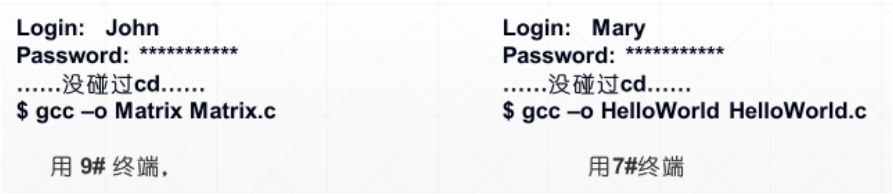

图 9、John 用户、Mary 用户会话的当前状态

1、在目录树上标出可执行文件 Matrix 和 HelloWorld 的位置，写绝对路径名

2、gcc 进程的 p_uid 和 u_cdir？p_ttyp?

3、明天，John 重新编译 Matrix.c 程序。为之服务的进程，p_uid 和 u_cdir？p_ttyp?

参考答案如下：

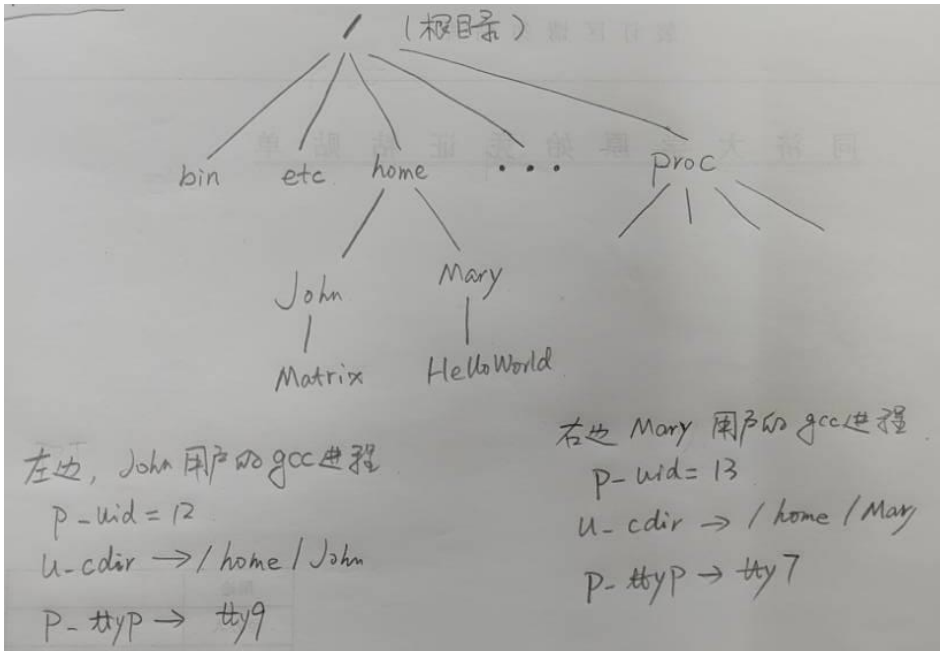

图 10、例题的参考答案

明天，John 重新编译 Matrix.c 程序。为之服务的 gcc 进程，p_uid 和 u_cdir 不变。p_ttyp 待定。因为不知道他会用哪台终端呀。

3、文件控制块

所有文件有且仅有一个**文件控制块（FCB，File Control Block）**。普通文件、目录，甚至外设特殊文件和进程的 proc 文件都有唯一的绝对路径名和文件控制块。Windows 的文件控制块是目录项，Unix/Linux 的文件控制块是 inode。

文件控制块中登记文件属性，是另一个最重要的内核数据结构。没有第 3 个了。FCB 中最重要的字段包括：

- uid，文件 owner。文件主的 uid。
- gid。文件所在的组。
- 文件的访问权限。标准的 Unix 文件系统中，文件的访问权限是 9 个比特 RWXRWXRWX。分 3 组。红的是文件主（owner）的权限，蓝的是同组用户（group）的权限，黑的是其它用户（nobody）的权限。如果这是一个普通文件，R，读权限；W，写权限；X，执行权限。如果这是一个目录，R，ls 权限；W，在这个目录下创建/删除 文件/子目录的权限；X，表示目录搜索（文件检索）可不可以过本目录。

3.2 creat 系统调用确定文件控制块中的 uid 和访问权限字段

creat 系统调用确定文件的 uid 和文件的访问权限。看示例程序：

```
#include <fcntl.h>
char buffer[2048];
int version = 1;

int main(argc,argv)
        int argc
        char *argv[];
{
        int fdold,fdnew;
        fdold = open(argv[1], O_RDONLY);
        if(fdold == -1)
        {
                printf("can not open file %s\n",argv[1]);
                exit(1);
        }
        fdnew = creat(argv[2], 0666);
        copyOperation (fdold, fdnew);
L:      printf ("This is process %d:  \n", getpid() );
        printf("Copy Done!From %s To %s\n",
                        argv[1],argv[2]);
        exit(0);
}

copyOperation (old,new)
        int old,new;
{
        int count;
N:      while ((count=read(old,buffer,sizeof(buffer)))>0)
                write(new, buffer, count);
}
```

图 11、示例程序 copy

$ copy oldFile newFile

这个程序为已有文件 oldFile 建立一个复本 newFile。创建 newFile 文件的是 copy 进程。文件的 owner 就是这个进程的 uid。看源代码，创建 newFile 的系统调用是 creat。它的第二个参数是文件的 RWXRWXRWX，前缀 0 表示 666 是八进制数，8 进制数与 RWX 3 个 bit 搭配，可读性非常好。无论那个用户执行 copy 程序，newFile 的权限是文件主、同组用户、所有其它用户对文件拥有读写权限，没有执行权限。

creat 系统调用的执行过程如下：

1、为新文件分配一个 FCB

2、FCB uid = 进程 PCB 中的 uid

3、FCB gid = 进程 PCB 中的 gid

4、把 creat 系统调用的第 2 个参数，写进 FCB 的 RWXRWXRWX

例题：John 用户执行命令$ copy file1 file2，创建文件 file2。求 file2 文件的绝对路径名，FCB 中的 uid 字段和访问权限字段。

答：图 9 John 用户没有执行过 cd 命令，故 copy 进程创建的 file2 文件的绝对路径名是

/home/John/file2。

file2 文件 uid=12，访问权限是 rw- rw- rw-。

另外，如果 Mary 用户执行$ copy file3 file4。该 copy 进程创建的 file4 文件，uid=13，访问权限也是 rw- rw- rw-。

creat 系统调用确定的 owner 和访问权限是可以改的。用 chown 和 chmod 命令。chown 改变文件主, chmod 改变文件访问权限。owner 有权修改访问控制权限。超级用户可以换掉文件的 owner。执行 sudo chown ……命令试试。

装系统的时候，这两个命令很有用。

3.3 open 系统调用使用 FCB 中的字段判断进程对文件的访问权限

文件，使用前要用 open 系统调用打开。

看图 11，open 系统调用。它有 2 个参数，第一个参数是已有文件的文件名，open 用它目录搜索找到要打开的文件，取用它的 FCB；第二个参数 mode，表示进程要对文件实施的访问，是只读还是会写。mode 是 3 个 bit，RWX，读访问，R 置 1；写访问，W 置 1。更新（又读又写），R 和 W 都置 1。

open，用进程的 uid 识别访问文件的用户，确定其对文件的访问权限。具体过程如下:

1、从文件 FCB 中读出 uid，gid 和 RWXRWXRWX

2、从执行 open 系统调用的 进程的 PCB 中读出进程的 uid 和 gid

3、确定进程的身份。uid 相等吗？Y，用红的 RWX。否则，gid 相等吗？

Y，用蓝的 RWX。否则，用黑的 RWX。(FCB 中的 RWX)

4、确定读写权限。匹配 open 系统调用的 mode 和 FCB 中的 RWX。

对应 bit 该是 1。

回到图 11。open 系统调用的第 2 个参数是 O_RDONLY, 它是二进制数 100。对 oldFile，进程只读，不会修改它的内容。

例题：13#用户 Mary 是 John 同组用户。执行命令

copy ../john/file2 file3

问：1、成功吗？ 2、file3 的绝对路径名？文件 uid？访问权限？

*

*

*

答：1、成功。因为../john/file2 文件的访问权限是 0666，同组用户有读写权限。

2、file3 的绝对路径名是/home/Mary/file3。文件 uid=13。访问权限 0666。

权限检测通过, 进程会得到一个文件描述符 fd。随后, 它会用文件描述符 fd 访问文件。执行 read 系统调用获得新数据，执行 write 系统调用把处理结果持久化至磁盘 或 传给本地/远端的其它进程。我们说，fd 是进程引用文件的内部标识符，文件一经打开，进程将不再使用其外部引用名（文件名字符串），用 fd 识别所有数据源。“一切皆文件”。

四、进程对文件的访问

1、 进程的打开文件表

进程的打开文件表是一个指针数组。

- 进程正在使用的每一个文件对应打开文件表中的一项，这个数据结构记录读写文件需要的全部信息。包括存放文件每个数据块的扇区号，文件长度，文件读写指针，打开方式 mode，存放文件内容的缓存块。。。
- Unix V6++系统，进程的打开文件表有 15 项。表示一个进程可以同时使用 15 个文件。Linux 系统，65535 项。

打开文件表中有 3 个文件是系统帮进程打开的。进程无需执行 open 系统调用就可以访问它们。这 3 个文件是:

- 0#文件，进程用它获取键盘输入，是标准输入文件，STDIN。 scanf，getc read 0#文件。
- 1#文件，进程用它写屏幕，是标准输出文件，STDOUT。 printf 写 1#文件。
- 2#文件，进程用它向屏幕输出告警信息。是标准错误输出文件，STDERR。

磁盘文件或进程用来和其它进程交换数据的 socket/PIPE，先打开再使用。系统调用成功返回后，系统会在打开文件表中为其分配一个表项，返回其下标，这就是文件描述符 fd。

进程执行 open 系统调用时，会从打开文件表的 0#表项开始线性搜索，找第一个空表项，将其分配给想要访问的文件。所以，应用程序打开的第一个磁盘文件，fd 会是 3。。。

完善 open 系统调用逻辑:

- 打开文件表 openFileTable 分配空闲表项 item
  - 从打开文件表 0# 表项开始，线性搜索为 null 的表项
- 文件系统里找到要打开的文件
  - 没找着，报错，此文件不存在。open 失败
- 检查文件访问权限
  - 不允许访问，报错，权限不够。open 失败
- 创建（或申请）用来管理这个打开文件的数据结构。 openFileTable[item]指向它。
  - 文件读写指针 f_offset 赋 0。
  - 记下来 open 的第 2 个参数 mode。
- 返回 item。 // item 下标是文件描述符 fd。

在进一步学习之前强调一点，只要应用程序访问文件读写数据，最终都要执行 read/write 系统调用。顺序读写，随机读写皆如此。

read 系统调用的用法:

```
count = read( fd, &array, num );
```

进程读其打开文件表中 fd 管理的文件。从读写指针的当前位置开始，连续读 num 个字节，存入起始地址是 array 的内存单元。array 通常会是一个长度至少 num 字节的数组。num 为 1 时，array 也可能是一个字节变量。read 读操作的具体步骤:

- 从读写指针的当前位置连续读 n 个字节
- 读写指针的值加 n

write 系统调用的用法：

count = write(fd, &array, num);

进程将起始地址是 array 的连续内存单元写入其打开文件表中 fd 管理的磁盘文件，从文件当前位置开始，连续写 num 个字节。write 写操作的具体步骤：

- 以读写指针的当前位置为起点，向文件连续写入 n 个字节
- 读写指针 += n
- 若修改后的读写指针的值大于文件长度，修改文件长度

2、文件的顺序读写操作

图 11，copy 程序对文件执行的是顺序读写操作。每次 read 从上次 read 操作结束的位置开始。我们观察 while 循环，文件描述符 old 引用的磁盘文件。第一次 read 读文件的 0~2047 字节，结束后读写指针的值是 2048。第二次 read，读入 2048~4095 字节。。。这是文件的顺序读操作，从文件的某个位置开始，顺序读入后续连续的大块文件数据。

例题：下面是一段顺序读文件的代码。写出每个系统调用完成后，文件 fd 读写指针的值。

```
fd = open（）;          // 0
read（fd，…，X）;       // X
read（fd，…，Y）;       // X + Y
read（fd，…，Z）;       // X + Y + Z
……
close（fd）;
```

3、文件的随机读写操作

程序执行 lseek 系统调用调整读写指针，实现随机读写。

例题：下面是一段随机读文件的代码。写出每次 read 读入文件的内容，以及 read 完成后，文件读写指针的值。

```
fd = open（）;          // 0
read（fd，…，X）;       // X，[ 0, X )
read（fd，…，Y）;       // X + Y，[ X, X+Y )
lseek（fd，SEEK_SET, 1000）; // 1000，只调读写指针，没有读文件
read（fd，…，Z）;       // X + Y + Z，[ 1000, 1000+Z )
……
close（fd）;
```

红色的是读入的文件内容，黑色的是 read 结束后的读写指针。看，第 3 次 read 的起始位置是文件的第 1000#字节，并不是第二次 read 完成时，读写指针的值 X + Y。

数据库访问时随机文件读写。

注，系统调用 lseek(fd, mode, num)的语义，调整文件的读写指针。从 mode 指示的位置开始，向后移动 num 个字节。mode 的取值可以是 3 个常量，

- SEEK_SET　0：文件头部（0 字节）
- SEEK_CUR　1：读写指针当前位置
- SEEK_END　2：文件尾部（最后一个字节）

4、close（fd）

文件使用完毕要关闭。系统会

- 打开文件表中释放 fd 相关的数据结构。
- 打开文件表 fd 所在位置置空：openFileTable[fd] = null。

打开文件表和相关数据结构是进程使用文件系统的现场。记录了正在使用的每个文件，现在都读到哪儿了，还有哪些暂存有这个文件数据的磁盘缓存块。

这些数据结构会消耗系统资源，所以不用的文件应该尽快关掉。

习题　　参考文献《深入理解计算机系统》

一、

1、fd1，fd2 的值是几？

2、下面程序的输出是什么？解释程序的输出。

```
int main()
{
    int fd1, fd2;

    fd1 = Open("foo.txt", O_RDONLY, 0);
    Close(fd1);
    fd2 = Open("baz.txt", O_RDONLY, 0);
    printf("fd2 = %d\n", fd2);
    exit(0);
}
```

二、下面程序的输出是什么？解释程序的输出。

假设 foobar.txt 文件的内容是字符串“1234567890”

```
int main()
{
    int fd1, fd2;
    char c;

    fd1 = Open("foobar.txt", O_RDONLY, 0);
    fd2 = Open("foobar.txt", O_RDONLY, 0);
    Read(fd1, &c, 1);
    Read(fd2, &c, 1);
    printf("c = %c\n", c);
    exit(0);
}
```