杨劲松
yjs@oldhand.org
133 0122 6268

今天讲课的内容：
http://192.168.1.108/bsechr-linux-io-course-yjs-2011.08.22.pdf

远程桌面： 192.168.1.101

$ vncviewer 192.168.1.101

没有 vncviewer 的同学，请使用如下的方式安装：
$ wget http://192.168.1.108/xtightvncviewer_1.3.9-6_i386.deb
$ sudo dpkg -i xtightvncviewer_1.3.9-6_i386.deb

- Linux 使用基础
  - http://192.168.1.108/bsec-linux-basic-2011.08.22.pdf
- 今天的代码：
  - http://192.168.1.108/examples-2011.08.22.tar.gz
- 今天的笔记：
  - http://192.168.1.108/2011.08.22.pdf

- 使用一个函数的时候，需要弄清除资源由谁负责分配，谁来负责释放，这个到后边会讲到重入问题，跟重入问题有关系。
  - 第一种情况：调用者分配，调用释放，可重入
  - 第二种情况：被调用者分配，调用者释放，可重入
  - 第三种情况：静态分配，不需要释放，不可重入

我们已经遇到的进程的环境

- Resource limit
  - 进程可以使用 getrlimit()/setrlimt() 取得 / 修改进程的 resource limit
  - 可以使用 ulimit 命令改变 shell 的 resource limit， Resource limit 值会被 shell 的子进程继承
- umask
  - 进程可以继承 umask
    - 进程继承的 umask 从哪儿来？从父进程来
    - 在命令行下设置的 umask 是干什么的？在命令行下设置 umask，会改变当前 shell 的 umask，那么通过当前 shell 启动的子进程，将继承该 umask 值
    - umask 命令是一个 shell 内置的命令 (built-in command)
  - 进程可以主动设置自己的 umask
  - 进程的 umask 会被其子进程继承
- 当前工作目录 ($PWD)
  - 进程的当前工作目录是继承而来的，如果进程是由 shell 启动的，那么 shell 当前所在的目录就会被进程作为当前工作目录
  - 进程可以使用 chdir()/fchdir() 改变当前工作目录

操作系统

- 汤子瀛，西安电子
- Http://www.youku.com 搜索“操作系统原理”，电子科大

lseek() 用途

- 创建空洞文件
- 随机的文件访问，比如实现一个数据库，在数据库检索的时候，需要做 lseek() 移动文件的读写指针位置

文件指针位置被谁改变？

- open 时
  - 如果设定了 O_APPEND，设置为末尾，并且以后在每次写文件时，指针位置都被自动原子的移动到文件尾
  - 如果未设置 O_APPEND，文件指针设置为文件开头位置（绝对偏移为 0 的位置）
- close() 时
  - close() 调用与文件指针移动无关
- read()
  - 成功时，文件指针位置向文件末尾方向移动实际读取的字节数
- write()
  - 成功时，文件指针位置向文件末尾方向移动实际写的字节数
- lseek()
  - 根据需要显式的移动文件读写指针的位置

文件描述符的复制

- 使用 fcntl(F_DUPFD)
  - 得到的描述符不小于传入的 arg 参数值（结果是大于或者等于 arg 参数的可用的最小描述符）
- 使用 dup()/dup2()
  - dup() 得到的描述符是顺序分配的可用的最小的描述符，跟 open() 调用得到的描述符的分配机制是相同的
  - dup2() 得到的是指定的描述符
- 通过 fork() 实现描述符的复制
  - 在子进程中，得到的与父进程值相等的描述符

共享的问题

- File descriptor flags 在复制的文件描述符之间不共享，也就是说对某个描述符修改了该标志，另一个描述符不受影响
- File status flags 在复制的文件描述符之间共享，其本质是复制的文件描述符指向同一个文件描述信息 (open file description)
- 文件的读写位置指针（ File offset， position) 在复制的文件描述符之间是共享的

vim 替换

- 范围 s/ 查找目标 / 替换为 / 指示符
- %s/abc/ABC/g
  - % 表示全文
  - s 为替换的指令
  - 查找 abc 串
  - 替换为 ABC
  - g 表示 global，一行之内的所有的 abc 都将被替换，如果不给 g 参数的话，只替换第一个
- 范围可以用行号表示
  - 1,10s/abc/ABC/g，在第一行到第十行之间范围内替换
- 查找目标可以使用正则表达式 (正则表达式可以理解为符合一定规律的字符序列)

任务

- 写程序验证一下通过文件描述符复制方式得到的描述符，假定原来的为 oldfd ，新获得的复制描述符为 newfd
  - 如果在 oldfd 上做 lseek 操作，是否会影响到 newfd
  - 如果在 newfd 上做写操作，是否会影响到 oldfd
  - 如果在 oldfd 上做读操作，是否会影响到 newfd

2011.08.23 上午的代码和笔记

- 大家可以使用 wget 下载，命令行：
  - $ wget URL
- 上午的笔记
  - http://192.168.1.108/memo-2011.08.23-morning.pdf
- 上午的代码：
  - http://192.168.1.108/examples-2011.08.23-morning.tar.gz

文件和目录

- 文件和目录
  - http://192.168.1.108/bsec-linux-file-course-yjs-2011.08.23.pdf
- 全部的代码：
  - http://192.168.1.108/examples-2011.08.23.tar.gz
- 笔记
  - http://192.168.1.108/memo-2011.08.23.pdf

stat()/fstat()/lstat() 规律

- stat() 也就是不是以“*f*”或者“*l*”开头的函数，使用的时文件名作为第一个参数，如果 stat() 传入的第一个参数是一个符号连接，那么 stat() 最终处理的是符号连接最终指向的文件
  - stat() 会跟随符号连接
- fstat()，也就是以“*f*”开头的函数，使用的是文件描述符作为第一个参数
- lstat()，也就是以“*l*”开头的函数，处理的是符号连接本身
  - lstat() 不跟随符号连接
- 符合该规律的还有如下调用
  - chmod()/fchmod()
  - chown()/fchown()/lchown()
  - truncate()/ftruncate()
  - chdir()/fchdir()

字符串处理是基本功

- 字符串连接(把两个字符串接起来)
  - strcat()
- 字符串拷贝
  - strcpy()/strncpy()/memcpy()
  - strdup()可以用于字符串复制
- 字符串分割
  - strtok(),
  - strsep()
  - 还可以使用状态机编程思想处理字符串)
- 字符串内查找子串
  - strstr()
  - 最好了解一下正则表达式
  - 如何想办法提高效率？
- 字符串解析，把需要的内容提取出来，如何提高效率？
- 字符串格式化
  - snprintf()

文件系统相关的命令

- 使用 fdisk 对磁盘进行分区
- 使用 mkfs 创建文件系统 (类似于 Windows/Dos 下的格式化 format)
  - mkfs 的参数 -t 指定文件系统类型
- e2label 可以对文件系统设置 label
- dumpe2fs 可以查看文件系统信息
- debugfs 可以不通过内核来对文件系统操作
- fsck 可以对磁盘进行检查

系统的文件组织

- 系统存储文件时，文件名和文件的实体(inode)是分开存储的
- 文件名存储在所在目录的目录文件中(没错：目录也是一个文件)，通常给目录分配一个块，如果一个块不够用，则继续分配一个块，所以目录的大小是块大小的整数倍。
- 文件的实体(实质上就是inode)存在其他的地方，存储在文件系统中，比如EXT2文件系统，在用户空间一般来说，无法通过inode编号访问到一个文件，只能通过文件名找到对应的inode，然后访问该inode，获取或者修改文件内容和文件属性。
- 文件名和inode之间的映射关系就是一个硬连接，存储在目录文件中
- 可以有多个文件名指向同一个inode这种情况，那么这些文件名之间互为硬连接
- 如果有兴趣的话，可以去看EXT2文件系统的组织和VFS，其中通过文件名找对应的inode的过程是使用dcache来实现的。

权限？

- 中文的“权限”是有歧义的
  - 可以表示为许可权 (permission)
  - 可以表示为优先权 (privilege)
  - 可以表示为权利 (right)
- 在我们进行文件许可权检查的时候，权限的意思是 permission，最好翻译为“许可权”

系统如何做权限检查？

- 每个进程都有real uid, real gid, effective uid, effective gid, 添加组ID
- 添加组ID怎么来？
  - 某个用户可以属于多个组，其中有一个是默认组，用户所属的默认组在/etc/passwd中设置
  - 在/etc/group文件里设置的是用户所属的其他组
  - 该用户启动一个程序的时候，其他组就变成了进程的添加组
- 进程的实际用户ID从哪儿来？谁启动程序，谁就是进程的实际用户ID
- 进程的实际组ID从哪儿来？谁启动程序，该用户所属的组ID就是进程的实际组ID
- 通常来说，进程的有效用户ID等于进程的实际用户ID，进程的有效组ID等于进程的实际组ID，进程的添加组ID从用户所属的组中取得。
- 但是，如果程序本身是set-user-id的，那么该程序启动为进程之后，就将进程的有效用户ID变更为程序文件的所有者ID
- 如果程序本身是set-group-id的，那么该程序启动为进程之后，就将进程的有效组ID变更为程序文件的组ID
- 当进程欲操作一个文件时，按照如下的规则去检查permission
  - Step 1, 检查进程的有效用户ID是否等于被操作文件的owner id，如果等于的话，进程就拥有owner的permission(rwx------)，后续的检查不做了
  - Step 2, 如果进程的有效用户ID不等于被操作文件的owner id，那么检查进程的有效组ID，如果进程的有效组ID等于文件的group id，那么，进程将拥有文件group的permission
  - Step 3, 如果不符合step 1, 也不符合step 2，那么依次检查进程的添加组ID，如果某个添加组(一个进程可以有0个或多个添加组ID)ID等于文件的group，则进程拥有文件group的permission
  - 最后，上述都不满足时，进程将拥有文件的other的permission

- UNIX 的权限机制比较初级，比较高级的权限机制是 ACL(access control list)
- sudo 就使用类似 ACL 的机制

- 如果有兴趣的话，对目录设置为 set-gid-bit 之后，在该目录下创建文件或者子目录会有什么效果。
- $ chmod 2755 <directory name>

变量命名

- 命名规则：
  - 作用域 _ 单元名 _ 变量用途
  - int gobal_net_xxx;
  - C 语言缺少 namespace
  - 如果使用 C++ ，你的代码可以用 namespace 封装起来。
  - 局部作用域的变量，可以短一些，比如循环变量
  - 如果局部变量使用范围跨越多个几十行代码，也要有描述性

chmod() 执行的条件

- 如果进程的有效用户 ID 等于文件的 owner，才可以做 chmod()，否则就不可以做 chmod()
- 如果被操作文件的 group 不等于进程的有效组 ID，也不等于进程的任何一个添加组 ID，并且进程是非特权的，那么试图去对文件设置 set-group-id 标志，则该标志会被屏蔽掉（也就是说在这种情况下该标志不允许设置）。

chown() 执行的条件

- 只有特权进程才可以改变文件的 owner，通常来说时 root 用户进程改变文件的 owner。
- 文件的 owner 所启动的进程 (确切的应该是进程的有效用户 ID 等于文件的 owner 时) 可以改变文件的组所有者关系，将文件的组所有者在 owner 所属的组之间改变。
- 假设 x 用户属于以下组：
  - stuff
  - manager
  - hr
- 那么如果某个一个 x 用户所拥有的文件 (即该文件的 owner=x)，可以将该文件的组在 stuff/manager/hr 之间变换。
- 特权进程可以改变文件的组为任意的组 ID。
- 如果非特权用户改变了文件的 owner 或者 group，那么文件原有的 set-uid-bit 和 set-gid-bit 会被清除掉。

任务

- 建议大家去找 binutils 介绍相关的资料看看：
  - 了解链接的一些过程
  - 了解静态库、共享库的情况
  - 了解如何制作静态库 (.a)
  - 了解如何制作共享库 (.so)
  - 了解如何链接静态库
  - 了解如何链接共享库
  - 了解如何动态的加载共享库，并调用共享库的函数
  - 了解共享库的路径文件
    - 配置文件 /etc/ld.so.conf
    - ldconfig 命令
    - 环境变量 LD_LIBRARY_PATH

任务：实现一个 mv 命令

- 移动和改名的区别？
  - 移动是将源文件转移到目标目录，但是名字不变，所以目标可以是一个路径名，实际上，构造出目标完整的路径名就可以调用 rename()
  - 改名是将文件名字改变为目标的名字，可以直接调用 rename() 实现
- 支持在同一文件系统内移动和改名
  - 用 rename() 实现
- 支持跨文件系统的移动和改名
  - 先拷贝源文件到目标文件 (或者目录)
  - 源文件的属性 (用 stat(2) 获得) 要尽可能的复制过去
    - 如何复制属性？
  - 删除源文件

任务：实现一个 rm

- 功能
  - 可以删除文件
  - 可以删除目录
  - 可以删除非空的目录
  - 如果可以支持一次删除多个文件和目录就更好了
- 通过命令行参数：
  - 支持请求用户确认删除功能 (-i)
  - 支持强制删除功能 (-f)
  - 支持删除目录选项 (-r)
- 命令行参数如何解析？
  - 可以使用 getopt()/getopt_long() 解析命令行参数
  - 可以使用 GNU gengetopt 工具去生成命令行解析的代码

实现 ls

- 实现系统的 ls -l 功能

- 今天的笔记：
  - http://192.168.1.108/memo-2011.08.25.pdf
- 今天的代码：
  - http://192.168.1.108/examples-2011.08.25.tar.gz