

# Linux 进程管理实验

## 实验报告

### 一. 实验目的

熟练掌握 Linux 操作系统的使用，掌握 Linux 的系统的进程管理相关内容，掌握进程之间的通信方式。

进程是操作系统中最重要的概念，贯穿始终，也是学习现代操作系统的核心。通过本次实验，要求理解进程的实质和进程管理的机制。在 Linux 系统下实现进程从创建到终止的全过程，从中体会进程的创建过程、父进程和子进程的关系、进程状态的变化、进程之间的同步机制、进程调度的原理和以信号和管道为代表的进程间通信方式的实现。

### 二. 实验内容

- 1、 在命令行新建多个普通用户，如 tux, bob, Alice, lily 等，给每个用户创建密码，并将这几个用户分到同一个组 xjtuse 中。再新建两个组 coding 和 testing，使得某些用户也分别为其组用户。在 root 用户和新建用户之间切换，验证用户创建成功与否。（给出相关命令运行结果）
- 2、 实现 *sudo* 委托管理任务，给上述某一指定的普通用户赋予创建用户的权限。（给出相关配置文件和命令运行结果）
- 3、 备份数据是系统应该定期执行的任务，请利用 cron 计划作业在每周五下午 6:10 对某用户（如 tux）主目录下的文件进行备

份（可使用 tar 命令）。给出相关运行结果和邮件记录。

#### 4、 编制实现软中断通信的程序

使用系统调用 fork() 创建两个子进程，再用系统调用 signal() 让父进程捕捉键盘上发出的中断信号（即按 delete 键），当父进程接收到这两个软中断的某一个后，父进程用系统调用 kill() 向两个子进程分别发出整数值为 16 和 17 软中断信号，子进程获得对应软中断信号，然后分别输出下列信息后终止：

**Child process 1 is killed by parent !!**

**Child process 2 is killed by parent !!**

父进程调用 wait() 函数等待两个子进程终止后，输入以下信息，结束进程执行：

**Parent process is killed!!**

多运行几次编写的程序，简略分析出现不同结果的原因。

#### 5、 编制实现进程的管道通信的程序

使用系统调用 pipe() 建立一条管道线，两个子进程分别向管道写一句话：

**Child process 1 is sending a message!**

**Child process 2 is sending a message!**

而父进程则从管道中读出来自于两个子进程的信息，显示在屏幕上。

要求：父进程先接收子进程 P1 发来的消息，然后再接收子进程 P2 发来的消息。

### 三. 题目分析及基本设计过程分析

1. 使用 `useradd` 指令添加用户, 使用 `groupadd` 指令添加组, 使用 `usermod` 指令将用户加入到组, 使用 `passwd` 指令修改密码。调用 `su` 进行用户切换, 并利用 `id` 等指令查看用户所在的组。
2. 修改 `/etc/sudoers` 给指定用户赋予相应权限。该题目应向 `/etc/sudoers` 文件中添加 `xhwang ALL=/sbin/useradd` 来赋予 `xhwang` 添加用户的权限。
3. 使用 `/etc/crontab` 文件控制系统作业。应当在该文件中添加这样的作业: “10 18 \* \* 5 root tar tux” 来在每周五的 18 时 10 分备份用户 `tux` 的数据。
4. 使用 Linux 系统中的 `fork`、`wait`、`exit`、`getpid`、`kill` 和 `signal` 等系统调用来实现进程控制。使用 `fork` 来创建子进程, 并根据其返回值判断在子进程还是父进程中。在父进程中, 利用 `signal` 接收软中断信号, 若未接收到信号则在 `while` 循环中等待。子进程中利用 `signal` 接收父进程发送的信号, 若未接收到信号也在 `while` 循环中等待。父进程接收到软中断信号后, 向子进程发送终止信号, 并使用 `wait` 等待子进程终止, 之后将自己终止。
5. 使用 `pipe (int pipeid[2])` 来创建一个管道。使用 `fork` 来创建和区分子进程。在子进程中, 使用 `write (pipeid[1], buf, size)` 向管道写入数据。由于管道是共享的, 使用 `lockf` 将 `pipeid[1]` 保护起来, 写入后释放。在父进程中, 等待两个子进程完成写入, 并使用 `read (pipeid[0], buf, size)` 从管道读取数据。

## 四. 运行截图和相关说明

1. 新建用户 tux、bob、Alice、lily，组 xjtuse、coding、testing。

设置用户密码并将用户加入各个组。利用 `usermod -g` 设置主组，`-G` 设置附加组：

```
[xhwang@bogon root]$ su
Password:
[root@bogon ~]# useradd tux
[root@bogon ~]# useradd bob
[root@bogon ~]# useradd Alice
[root@bogon ~]# useradd lily
[root@bogon ~]# groupadd xjtuse
[root@bogon ~]# groupadd coding
[root@bogon ~]# groupadd testing
[root@bogon ~]# usermod -g xjtuse tux
[root@bogon ~]# usermod -g xjtuse bob
[root@bogon ~]# usermod -g xjtuse Alice
[root@bogon ~]# usermod -g xjtuse lily
[root@bogon ~]# usermod -G coding tux
[root@bogon ~]# usermod -G coding bob
[root@bogon ~]# usermod -G testing Alice
[root@bogon ~]# usermod -G testing lily
[root@bogon ~]# passwd tux
Changing password for user tux.
New UNIX password:
BAD PASSWORD: it is based on a dictionary word
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@bogon ~]# su tux
[tux@bogon root]$ id
uid=501(tux) gid=505(xjtuse) groups=505(xjtuse),506(coding) context=user_u:system_r:unconfined_t

[tux@bogon root]$ su
Password:
[root@bogon ~]# su Alice
[Alice@bogon root]$ id
uid=503(Alice) gid=505(xjtuse) groups=505(xjtuse),507(testing) context=user_u:system_r:unconfined_t
```

利用 `su` 切换用户，检查用户是否添加成功，并调用 `id` 查看用户是否加入上述编写的组中：

```
[root@bogon ~]# su tux
[tux@bogon root]$ id
uid=501(tux) gid=505(xjtuse) groups=505(xjtuse),506(coding) context=user_u:system_r:unconfined_t
[tux@bogon root]$ su bob
Password:
[Bob@bogon root]$ id
uid=502(bob) gid=505(xjtuse) groups=505(xjtuse),506(coding) context=user_u:system_r:unconfined_t
[Bob@bogon root]$ su Alice
Password:
[Alice@bogon root]$ id
uid=503(Alice) gid=505(xjtuse) groups=505(xjtuse),507(testing) context=user_u:system_r:unconfined_t
[Alice@bogon root]$ su lily
Password:
su: incorrect password
[Alice@bogon root]$ su lily
Password:
[lily@bogon root]$ id
uid=504(lily) gid=505(xjtuse) groups=505(xjtuse),507(testing) context=user_u:system_r:unconfined_t
```

2. 修改/etc/sudoers，赋予 xhwang 添加用户的权限：

```
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
xhwang  ALL=(ALL) ALL
```

尝试添加用户 abc：

```
[xhwang@bogon root]$ sudo useradd abc
Password:
[xhwang@bogon root]$ su abc
Password:
su: incorrect password
[xhwang@bogon root]$ su
Password:
[root@bogon ~]# su abc
[abc@bogon root]$ id
uid=505(abc) gid=508(abc) groups=508(abc) context=user_u:system_r:unconfined_t
```

3. 输入 nano /etc/crontab 进入/etc/crontab 文件的修改界面，并输入 10 18 \* \* 5 root tar tux，使得每周五的 18 时 10 分备份用户 tux 的数据：

```
[xhwang@bogon ~]$ su -
Password:
[root@bogon ~]# nano /etc/crontab

SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
10 18 * * 5 root tar tux
```

使用 cat /var/log/cron 命令查看 root 接受的邮件，发现存在备份 tux 用户文件的邮件：

```
root@bogon:~  
File Edit View Terminal Tabs Help  
-bash: systemctl: command not found  
[root@bogon ~]# grep CRON /var/log/cron  
[root@bogon ~]# ls /var/log  
acpid          cron           messages       secure.1  
anaconda.log   cron.1         messages.1     spooler  
anaconda.syslog cups           pm             spooler.1  
anaconda.xlog  dmesg         ppp            tallylog  
audit          faillog       prelink        vmware-tools-upgrader.log  
boot.log       gdm           rpmpkgs        wtmp  
boot.log.1     lastlog       rpmpkgs.1      Xorg.0.log  
btmp           mail          samba          Xorg.0.log.old  
conman         maillog       scrollkeeper.log yum.log  
conman.old     maillog.1     secure  
[root@bogon ~]# cat /var/log/cron  
Oct 18 10:19:10 bogon anacron[3276]: Job `cron.daily' terminated  
Oct 18 10:23:34 bogon anacron[3276]: Job `cron.weekly' started  
Oct 18 10:24:12 bogon anacron[3276]: Job `cron.weekly' terminated  
Oct 18 10:24:12 bogon anacron[3276]: Normal exit (2 jobs run)  
Oct 18 10:36:01 bogon crond[3228]: (*system*) RELOAD (/etc/crontab)  
Oct 18 11:01:02 bogon crond[8407]: (root) CMD (run-parts /etc/cron.hourly)  
Oct 18 18:01:02 bogon crond[8857]: (root) CMD (run-parts /etc/cron.hourly)  
Oct 18 18:10:05 bogon crond[9067]: (root) CMD (tar tux)  
Oct 18 18:24:01 bogon crond[3228]: (*system*) RELOAD (/etc/crontab)  
[root@bogon ~]#
```

4. 编制实现软中断通信的程序的代码如下:

```
# include<stdio.h>  
  
# include<signal.h>  
  
# include<unistd.h>  
  
# include<sys/wait.h>  
  
#include<stdlib.h>  
  
int wait_mark;  
  
void waiting()  
{  
    while(wait_mark!=0);  
}  
  
void stopwaiting()  
{  
    wait_mark=0;  
}  
  
int main()  
{
```

```

int pid1,pid2;

while((pid1=fork())!=-1);

if(pid1>0)
{
    while((pid2=fork())!=-1);

    //Parent process

    if(pid2>0)
    {
        wait_mark=1;

        signal(SIGINT,stopwaiting);

        waiting();

        kill(pid2,16);

        kill(pid1,17);

        wait(0);

        wait(0);

        printf("parent process is killed!!\n");

        exit(0);
    }

    //Child process 2

    else

    {
        wait_mark=1;

        signal(SIGINT,SIG_IGN);

        signal(16,stopwaiting);

        waiting();

        printf("child process 2 is killed!!\n");

        exit(0);
    }
}

//Child process 1

```

```

else
{
    wait_mark=1;

    signal(SIGINT,SIG_IGN);

    signal(17,stopwaiting);

    waiting();

    printf("child process 1 is killed!!\n");

    exit(0);

}
}

```

运行结果如下：

```

[xhwang@bogon ~]$ gcc assignment4.c -o assignment4
[xhwang@bogon ~]$ ./assignment4
child process 1 is killed!
child process 2 is killed!
parent process is killed!
[xhwang@bogon ~]$
[xhwang@bogon ~]$ gcc assignment4.c -o assignment4
[xhwang@bogon ~]$ ./assignment4
child process 2 is killed!
child process 1 is killed!
parent process is killed!
[xhwang@bogon ~]$ gcc assignment4.c -o assignment4
[xhwang@bogon ~]$ ./assignment4
child process 1 is killed!
child process 2 is killed!
parent process is killed!

```

可以看出，三次运行中，子进程 1 和子进程 2 结束的先后顺序可能存在差异。这是因为 Linux 系统中操作系统负责管理多个进程的调度，父进程通过 `kill()` 向两个子进程发送信号 16 和 17，这两个信号发出后，子进程具体何时收到信号并执行是由操作系统的调度机制决定的。由于调度是异步的，在某些情况下，子进程 1 可能比子进程 2 先被调度执行，或者子进程 2 比子进程 1 先执行，这会导致输出顺序的不同。但由于父进程调用了 `wait` 而被阻塞，父进程不会在它们结束之前中止。



5. 管道通信的程序代码如下：

```
#include<stdio.h>

#include<unistd.h>

#include<signal.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

    int pid1,pid2;

    int fd[2];

    char data[100];

    //创建管道

    pipe(fd);

    //创建子进程

    while((pid1 = fork()) == -1);

    //子进程 1

    if(pid1 == 0)

    {

        lockf(fd[1],1,0);

        sprintf(data,"Child process 1 is sending a message!");

        write(fd[1],data,50);

        lockf(fd[1],0,0);

        sleep(1);

        exit(0);

    }

    else

    {

        //创建子进程

        while((pid2 = fork()) == -1);
```

```

//子进程 2

if(pid2 == 0){

lockf(fd[1],1,0);

sprintf(data,"Child process 2 is sending a message!");

write(fd[1],data,50);

lockf(fd[1],0,0);

sleep(1);

exit(0);

}

//父进程

else{

wait(0);

read(fd[0],data,50);

printf("%s\n",data);

read(fd[0],data,50);

printf("%s\n",data);

close(0);

exit(0);

}

}

return 0;

}

```

子进程和父进程成功实现了管道通信，并满足题目先接收子进程 1 的消息的要求，运行结果如下：

```

[xhwang@bogon ~]$ gcc assignment5.c -o assignment5
[xhwang@bogon ~]$ ./assignment5
Child process 1 is sending a message!
Child process 2 is sending a message!
[xhwang@bogon ~]$ gcc assignment5.c -o assignment5
[xhwang@bogon ~]$ ./assignment5
Child process 1 is sending a message!
Child process 2 is sending a message!
[xhwang@bogon ~]$ gcc assignment5.c -o assignment5
[xhwang@bogon ~]$ ./assignment5
Child process 1 is sending a message!
Child process 2 is sending a message!

```

## 五. 实验中出现的问题和解决

### 1. 登录 Linux 系统时卡在此页面无法进入系统:

```
Booting 'Red Hat Enterprise Linux Server (2.6.18-164.el5)'
root (hd0,0)
Filesystem type is ext2fs, partition type 0x03
kernel /vmlinuz-2.6.18-164.el5 ro root=LABEL=/ rhgb quiet
[Linux-bzImage, setup=0x1e00, size=0x1c31d4]
initrd /initrd-2.6.18-164.el5.img
[Linux-initrd @ 0x37d71000, 0x27e6d1 bytes]

Memory for crash kernel (0x0 to 0x0) not within permissible range
```

解决:

进入 GRUB 菜单, 选择启动项, 按 e 进入编辑模式, 找到内核参数那一行, 删除其中的 rhgb 和 quiet 选项, 等待几分钟后即可进入系统。

### 2. 给用户赋予了添加用户的权限但仍然无法使用该权限:

```
[lily@bogon root]$ su
Password:
[root@bogon ~]# visudo
[root@bogon ~]# su tux
[tux@bogon root]$ sudo useradd testadduser

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

Password:
Sorry, user tux is not allowed to execute '/usr/sbin/useradd testadduser' as root on bogon.
[tux@bogon root]$

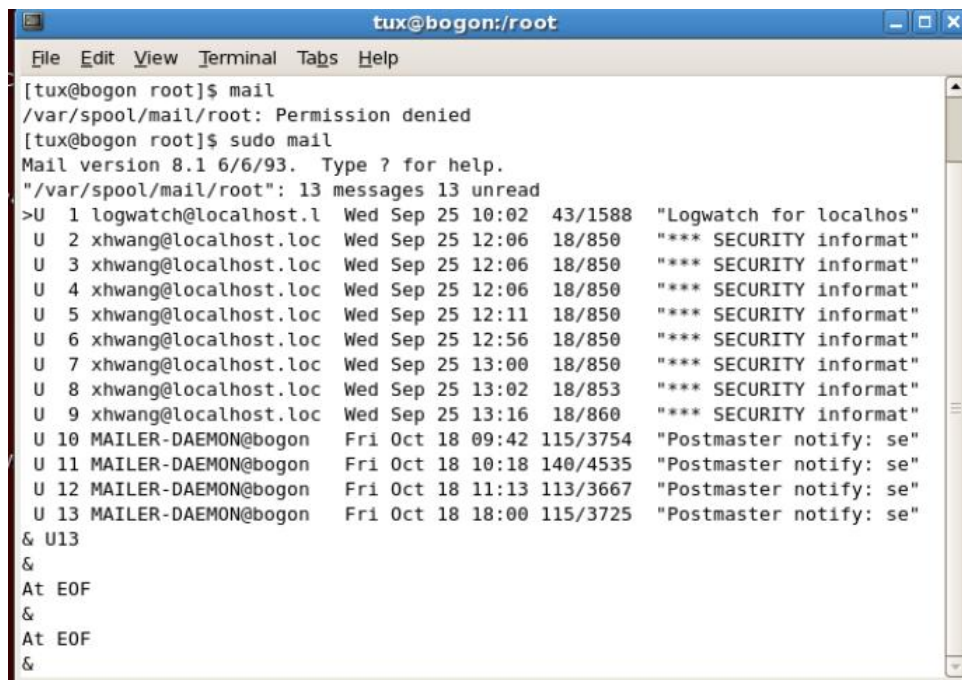
[root@bogon ~]# su - tux
[tux@bogon ~]$ sudo useradd testadd user
Password:
sudo: useradd: command not found
[tux@bogon ~]$
```

解决:

给用户赋予更多的权限:

```
## Allow root to run any commands anywhere
root    ALL=(ALL)          ALL
xhwang  ALL=(ALL) ALL
```

3. 找不到备份 tux 的相关邮件:

A terminal window titled 'tux@bogon:/root' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The user runs 'mail' and receives a 'Permission denied' error for '/var/spool/mail/root'. Then, the user runs 'sudo mail', which shows 'Mail version 8.1 6/6/93. Type ? for help.' and lists 13 unread messages. The messages include a logwatch report and several security notifications from xhwang@localhost.loc, followed by postmaster notifications from MAILER-DAEMON@bogon.

```
tux@bogon:/root
File Edit View Terminal Tabs Help
[tux@bogon root]$ mail
/var/spool/mail/root: Permission denied
[tux@bogon root]$ sudo mail
Mail version 8.1 6/6/93. Type ? for help.
"/var/spool/mail/root": 13 messages 13 unread
>U 1 logwatch@localhost.l Wed Sep 25 10:02 43/1588 "Logwatch for localhos"
U 2 xhwang@localhost.loc Wed Sep 25 12:06 18/850  "**** SECURITY informat"
U 3 xhwang@localhost.loc Wed Sep 25 12:06 18/850  "**** SECURITY informat"
U 4 xhwang@localhost.loc Wed Sep 25 12:06 18/850  "**** SECURITY informat"
U 5 xhwang@localhost.loc Wed Sep 25 12:11 18/850  "**** SECURITY informat"
U 6 xhwang@localhost.loc Wed Sep 25 12:56 18/850  "**** SECURITY informat"
U 7 xhwang@localhost.loc Wed Sep 25 13:00 18/850  "**** SECURITY informat"
U 8 xhwang@localhost.loc Wed Sep 25 13:02 18/853  "**** SECURITY informat"
U 9 xhwang@localhost.loc Wed Sep 25 13:16 18/860  "**** SECURITY informat"
U 10 MAILER-DAEMON@bogon Fri Oct 18 09:42 115/3754 "Postmaster notify: se"
U 11 MAILER-DAEMON@bogon Fri Oct 18 10:18 140/4535 "Postmaster notify: se"
U 12 MAILER-DAEMON@bogon Fri Oct 18 11:13 113/3667 "Postmaster notify: se"
U 13 MAILER-DAEMON@bogon Fri Oct 18 18:00 115/3725 "Postmaster notify: se"
& U13
&
At EOF
&
At EOF
&
```

解决: 发现是在编辑 cron 文件时输入错误, 修改 cron 文件后得以解决:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
10 18 * * 5 root tar -czf /backup/tux_backup.tar.gz /home/tux
```

## 六. 实验总结

通过本次实验，我学习了 Linux 系统中进程的相关概念、用户及用户组的创建与管理、定时任务、进程管理中 fork、wait、signal、pipe 等指令。实验中遇到了很多意料之外的问题（如无法登入 Linux 系统），但最终都通过查阅资料等方式将问题解决并独立完成了本次实验。本次实验增进了我对 Linux 系统的了解与对命令行的运用，也增加了我的操作系统知识储备。