



中山大學

SUN YAT-SEN UNIVERSITY

实 验 报 告

课程名称：____ 操作系统 ____

姓 名：____ 孙广岩 ____

学 号：____ 20354242 ____

专业班级：____ 智能科学与技术专业 5 班 ____

任课教师：____ 吴贺俊 ____

____ 2022 年 10 月 9 日 ____

实验报告成绩评定表

评定项目	内 容	满 分	评 分	总 分
实验态度	态度端正、遵守纪律、出勤情况	10		
实验过程	按要求完成算法设计、代码书写、 注释清晰、运行结果正确	30		
实验记录	展示讲解清楚、任务解决良好、实 验结果准确	20		
报告撰写	报告书写规范、内容条理清楚、表 达准确规范、上交及时、无抄袭， 抄袭记 0 分，提供报告供抄袭者扣 分。	40		

评语：

指导老师签字：

年 月 日

实验 3.1 进程创建实验

一、实验目的

1. 了解计算机系统结构，理解计算机功能和设计，掌握操作系统的位置和功能：应用程序/工具软件/用户与计算机硬件之间的系统软件。深度了解操作系统的内部机制。
2. 创建一个进程，掌握创建进程的方法，理解进程和程序的区别。
3. 调试跟踪进程创建的执行过程，了解进程的创建过程，理解进程是资源分配的单位。
4. 加深对进程概念的理解，进一步认识并发执行的实质。
5. 掌握 Linux 操作系统中进程的创建和终止操作。
6. 掌握在 Linux 操作系统中创建子进程并加载新映像的操作。

二、实验内容

1. 任务描述

1) Linux 的文件访问权限设置

- 准备工作
 - ◆ 启动上次安装的虚拟机和 Linux
 - ◆ 在用户主目录下创建目录 test, 进入 test 目录，用 vi/vim 创建文件 file1，并输入任意的文字内容。
- 文件权限查看、理解和文件权限的设置
- 输入输出重定向和管道操作

2) 创建进程的系统调用

- 进程创建—fork()
- 进程标识符管理
- 加载新的进程映像—exec 函数族

3) 编写一个 C 程序编写一个 C 程序，并使用系统调用 fork() 创建一个子进程。

要求如下：

- ①在子进程中分别输出当前进程为子进程的提示、当前进程的 PID 和父进程的 PID、根据用户输入确定当前进程的返回值、退出提示等信息。
- ②在父进程中分别输出当前进程为父进程的提示、当前进程的 PID 和子进

程的 PID、等待子进程退出后获得的返回值、退出提示等信息。

- 4) 编写另一个 C 程序，使用系统调用 `fork()` 以创建一个子进程，并使用这个子进程调用 `exec` 函数族以执行系统命令 `ls`。

2. 实验方案

这里主要对两个 C 程序进行一些说明。

C 程序 1 `fork1.c`，使用 `fork()` 来创建一个子进程，然后 `fork()` 返回 0 的话表示在子进程中，这是会输出当前进程为子进程，当前进程的 PID 和父进程的 PID，之后等待子进程退出后的返回值（用户输入）以及退出提示；在子进程中时会输出当前进程为子进程的提示，在当前进程的 PID 和父进程的 PID，根据用户的输入当前进程的返回值、退出提示等信息。

C 程序 2 `fork2.c`，使用 `fork()` 来创建子进程，然后用 `execlp()` 来执行系统命令 `ls`。

三、实验记录

1. Linux 的文件访问权限设置

准备工作：

```
george@LAPTOP-NCJJCQ1J: ~$ mkdir test
george@LAPTOP-NCJJCQ1J: ~$ cd test
george@LAPTOP-NCJJCQ1J: ~/test$ vim file1

george@LAPTOP-NCJJCQ1J: ~/test$ cat file1
hello world
my name is george
```

文件权限的查看：

```
george@LAPTOP-NCJJCQ1J: ~/test$ ls -l
total 4
-rw-r--r-- 1 george george 30 Oct  9 16:15 file1
george@LAPTOP-NCJJCQ1J: ~/test$ chmod o+w file1
george@LAPTOP-NCJJCQ1J: ~/test$ ls -l
total 4
-rw-r--rw- 1 george george 30 Oct  9 16:15 file1
george@LAPTOP-NCJJCQ1J: ~/test$ chmod g-r file1
george@LAPTOP-NCJJCQ1J: ~/test$ ls -l
total 4
-rw----rw- 1 george george 30 Oct  9 16:15 file1
george@LAPTOP-NCJJCQ1J: ~/test$ chmod 755 file1
george@LAPTOP-NCJJCQ1J: ~/test$ ls -l
total 4
-rwxr-xr-x 1 george george 30 Oct  9 16:15 file1
```

第一列就是文件属性字段：

(1) 第 1 个字母为文件类型：

d：目录文件，

-：普通文件

p：管理文件

- l: 链接文件
- b: 块设备文件
- c: 字符设备文件
- s: 套接字文件

(2) 文件权限: r 表示读权限, w 表示写权限, x 表示可执行权限, - 表示无权限

第 2 个字母-第 4 个字母: 表示所有者权限

第 5 个字母-第 7 个字母: 表示组用户权限

第 8 个字母-第 10 个字母: 表示其他用户组的权限

重定向:

```
george@LAPTOP-NCJJCQ1J: ~$ ls -l
total 4
-rwxr-xr-x 1 george george 30 Oct  9 16:15 file1
george@LAPTOP-NCJJCQ1J:~/test$ ls -l > list
george@LAPTOP-NCJJCQ1J:~/test$ cat list
total 4
-rwxr-xr-x 1 george george 30 Oct  9 16:15 file1
-rw-r--r-- 1 george george  0 Oct  9 16:17 list
george@LAPTOP-NCJJCQ1J:~/test$ ls -l >> list
george@LAPTOP-NCJJCQ1J:~/test$ cat list
total 4
-rwxr-xr-x 1 george george 30 Oct  9 16:15 file1
-rw-r--r-- 1 george george  0 Oct  9 16:17 list
total 8
-rwxr-xr-x 1 george george  30 Oct  9 16:15 file1
-rw-r--r-- 1 george george 105 Oct  9 16:17 list
george@LAPTOP-NCJJCQ1J:~/test$ ls -l > list
george@LAPTOP-NCJJCQ1J:~/test$ cat list
total 4
-rwxr-xr-x 1 george george 30 Oct  9 16:15 file1
-rw-r--r-- 1 george george  0 Oct  9 16:18 list
```

可以看到>是会覆盖目标文件的, >>则是追加内容。

管道:

由于我的 ws12 使用 who 之后没有反应, 我换了一个服务器来展示命令:

who | grep sunguanyan

```
(base) sunguanyan@ubuntu1:~$ who
luliucun pts/6      2022-09-08 10:16 (tmux(11523)).%0
xiezhenyu pts/9      2022-09-09 06:37 (tmux(1193652)).%0
zhengjun pts/18     2022-09-09 08:19 (tmux(1335307)).%0
zhengjun pts/24     2022-10-02 13:31 (tmux(1335307)).%4
sunguanyan pts/13    2022-10-09 09:29 (120.236.174.179)
luliucun pts/12     2022-10-09 02:58 (tmux(11523)).%1
zhengjun pts/21     2022-09-15 08:37 (tmux(1335307)).%1
zhengjun pts/22     2022-10-02 13:28 (tmux(1335307)).%3
zhengjun pts/10     2022-09-17 00:38 (tmux(1335307)).%2
zhengjun pts/8      2022-10-05 11:10 (tmux(1335307)).%6
(base) sunguanyan@ubuntu1:~$ who | grep sunguanyan
sunguanyan pts/13    2022-10-09 09:29 (120.236.174.179)
```

who 命令用于显示系统中有哪些使用者正在上面, grep 命令用于查找文件里符合条件的字符串

```
george@LAPTOP-NCJJCQ1J: ~$ ls -l | wc -l
3
george@LAPTOP-NCJJCQ1J:~/test$ ls -l
total 8
-rwxr-xr-x 1 george george  30 Oct  9 16:15 file1
-rw-r--r-- 1 george george 105 Oct  9 16:18 list
```

wc -l 统计有多少行，可以看到确实有三行

2. C 程序 1

直接使用课堂提供的代码会有些 warning，在一开始加#include <stdio.h>就可以避免

```
george@LAPTOP-NCJJCQ1J: ~  
george@LAPTOP-NCJJCQ1J:~/os/exp3$ gcc fork1.c -o fork1  
george@LAPTOP-NCJJCQ1J:~/os/exp3$ ./fork1  
PARENT: I am the parent process!  
PARENT: Here's my PID: 120  
PARENT: The value of my child's PID is: 121  
PARENT: I will now wait for my child to exit.  
CHILD: I am the child process!  
CHILD: Here's my PID: 121  
CHILD: My parent's PID is: 120  
CHILD: The value of fork return is: 0  
CHILD: Sleep for 1 second...  
CHILD: Enter an exit value (0~255): 128  
CHILD: Goodbye!  
PARENT: Child's exit code is: 128  
PARENT: Goodbye!
```

接下来讲解一下代码过程。

第一行为输出当前进程为父进程的提示；

第二行为当前进程的 PID；

第三行为子进程的 PID；

第四行为等待子进程停止提示；

第五行为当前进程为子进程的提示；

第六行为当前进程的 PID；

第七行为父进程的 PID；

第八行为 fork() 的返回值；

第九行为等待 1s；

第十行为输入当前进程的返回值；

第十一行为子进程的退出提示；

第十二行为父进程得到子进程推出后的返回值；

第十三行为父进程的退出提示，程序结束。

3. C 程序 2

直接使用课堂提供的代码会有些 warning，在一开始加#include <sys/wait.h>就可以避免

```
george@LAPTOP-NCJJCQ1J: ~  
george@LAPTOP-NCJJCQ1J:~/os/exp3$ gcc fork2.c -o fork2  
george@LAPTOP-NCJJCQ1J:~/os/exp3$ ./fork2  
fork1 fork1.c fork2 fork2.c
```

可以看到返回了当前目录下的全部文件。

四、总结与讨论

实验结果分析和问题思考：

1. 文件 backup.tar 的权限如下：

```
-rw-r--r-- 1 root root 19274 Jul 14 11:00 backup.tar
```

请问-rw-r--r-- 的含义是什么？

前面已经说明了每个字母代表的含义，那么这里就是首先该文件为一个普通文件，所有者有读写权限，组用户和其他用户租只有读权限。

2. 文件 backup.tar 的所有者添加执行权限的命令是什么？

```
chmod u+x backup.tar
```

u 表示所有者 user，x 为可执行权限

3. 赋予所有用户读和写 backup.tar 文件权限的命令是？

```
chmod a+rw backup.tar
```

4. 如何把一个 backup 文件夹打包成 backup.tar？

```
tar -cvf backup.tar bacakup
```

将 backup 文件夹打包为 backup.tar，并显示打包过程

实验 3.2 进程调度实验

一、实验目的

1. 加深对进程概念的理解，明确进程和程序的区别
2. 深入理解系统如何组织进程
3. 理解常用进程调度算法的具体实现

二、实验内容

1. 任务描述

编写 C 程序模拟实现单处理机系统中的进程调度算法，实现对多个进程的调度模拟，要求采用常见进程调度算法（如先来先服务、时间片轮转和优先级调度等算法）进行模拟调度。

2. 实验方案

数据结构设计：PCB 结构体主要需要以下几个部分：进程名称，进程状态（就绪态/运行态），优先级，需要运行的时间，已经运行的时间，时间片信息。

算法一共进行了三个实验，分别是先来先服务算法，时间片轮转算法，最高优先级优先调度算法。

3. 实验说明

(1) 先来先服务（FCFS）调度算法

先来先服务调度算法是最简单的调度方法。其基本原则是，按照进程进入就绪队列的先后次序进行选择。对于进程调度来说，一旦一个进程得到处理机，它就一直运行下去，直到该进程完成任务或者因等待某事件而不能继续运行，才会让出处理机。先来先服务调度算法属于非剥夺方式。

(2) 时间片轮转算法

时间片轮转调度算法也多用于进程调度。采用此算法的系统，其进程就绪队列往往按进程到达的时间来排序。进程调度程序总是选择就绪队列中的第一个进程，也就是说，按照先来先服务原则进行调度，但进程仅占用处理机一个时间片。在使用完一个时间片后，即使进程还没有完成其运行，它也必须让出（被剥夺）处理机给下一个就绪的进程。而被

剥夺的进程返回就绪队列的末尾重新排队，等候再次运行。时间片轮转调度算法特别适合分时系统使用。当多个进程驻留主存时，在进程间转接的开销一般不是很大。

(3) 最高优先级优先调度算法

按照进程的优先级高低来进行调度，使高优先级进程优先得到处理机的调度算法称为优先级调度算法。进程的优先级可以由操作系统按一定原则赋予。

三. 实验记录

1. 实施步骤

- (1) 先来先服务算法
- (2) 时间片轮转算法
- (3) 优先级调度算法

2. 实验结果

(2) 先来先服务算法

```
george@LAPTOP-NCJJCQ1J: ~  
george@LAPTOP-NCJJCQ1J:~/os/exp4$ ./schedule_f  
选择算法:P/R/FC优先数算法/时间片轮转算法/先来先服务算法)  
F  
输入进程数:  
3  
输入进程号和运行时间:  
p1 20  
p2 15  
p3 5  
先来先服务算法输出信息:  
*****  
进程号 所需时间 状态  
p3 5 W  
p2 15 W  
p1 20 W  
进程号 所需时间 状态  
p2 15 R  
p1 20 W  
p3 0 F  
进程号 所需时间 状态  
p1 20 R  
p2 0 F  
p3 0 F  
进程号 所需时间 状态  
p1 0 F  
p2 0 F  
p3 0 F  
按任意键退出
```

可以看到顾名思义，先来先服务算法就是按顺序执行。

(3) 时间片轮转算法

```
2560 x 1390
george@LAPTOP-NCJJCQ1J: ~$ ./schedule_f
选择算法:P/R/F(优先数算法/时间片轮转算法/先来先服务算法)
R
输入进程数:
3
请输入时间片: 1
输入进程号和运行时间:
p1 4
p2 3
p3 2
时间片轮转算法输出信息:
*****
进程号  cpu时间  所需时间  记数  时间片  状态
p1      0      4      0      1      W
p2      0      3      0      1      W
p3      0      2      0      1      W
进程号  cpu时间  所需时间  记数  时间片  状态
p2      0      3      0      1      R
p3      0      2      0      1      W
p1      1      3      1      1      W

进程号  cpu时间  所需时间  记数  时间片  状态
p3      0      2      0      1      R
p1      1      3      1      1      W
p2      1      2      1      1      W

进程号  cpu时间  所需时间  记数  时间片  状态
p1      1      3      1      1      R
p2      1      2      1      1      W
p3      1      1      1      1      W

进程号  cpu时间  所需时间  记数  时间片  状态
p2      1      2      1      1      R
p3      1      1      1      1      W
p1      2      2      2      1      W

进程号  cpu时间  所需时间  记数  时间片  状态
p3      1      1      1      1      R
p1      2      2      2      1      W
p2      2      1      2      1      W

进程号  cpu时间  所需时间  记数  时间片  状态
p1      2      2      2      1      R
p2      2      1      2      1      W
p3      2      0      2      1      F

进程号  cpu时间  所需时间  记数  时间片  状态
p2      2      1      2      1      R
p1      3      1      3      1      W
p3      2      0      2      1      F

进程号  cpu时间  所需时间  记数  时间片  状态
p1      3      1      3      1      R
p2      3      0      3      1      F
p3      2      0      2      1      F

进程号  cpu时间  所需时间  记数  时间片  状态
p1      4      0      4      1      F
p2      3      0      3      1      F
p3      2      0      2      1      F
按任意键退出
```

可以看到，就是按照顺序每个进程会运行时间片大小长度的时间，然后直到全部程序运行成功。

(4) 优先级调度算法

也是顾名思义，每次会选择优先度最高的进程运行，一个时间之后进程的优先数会减 1。

```
george@LAPTOP-NCJJCQ1J: ~ X + v
george@LAPTOP-NCJJCQ1J:~/os/exp4$ ./schedule

请输入被调度的进程数目：3

进程号No.0:
输入进程名:p1
输入进程优先数:1
输入进程运行时间:2

进程号No.1:
输入进程名:p2
输入进程优先数:4
输入进程运行时间:1

进程号No.2:
输入进程名:p3
输入进程优先数:4
输入进程运行时间:3

The execute number:1

**** 当前正在运行的进程是:p2
qname    state    nice    ndtime    runtime
p2       R       4       1         0

****当前就绪队列状态为:
qname    state    nice    ndtime    runtime
p1       W       1       2         0

qname    state    nice    ndtime    runtime
p3       W       4       3         0
进程 [p2] 已完成.

按任一键继续.....

The execute number:2

**** 当前正在运行的进程是:p1
qname    state    nice    ndtime    runtime
p1       R       1       2         0

****当前就绪队列状态为:
qname    state    nice    ndtime    runtime
p3       W       4       3         0
```

```
george@LAPTOP-NCJJCQ1J: ~  
The execute number:3  
**** 当前正在运行的进程是:p3  
qname    state    nice    ndtime    runtime  
p3       R        4       3         0  
****当前就绪队列状态为:  
qname    state    nice    ndtime    runtime  
p1       W        0       2         1  
按任一键继续.....  
The execute number:4  
**** 当前正在运行的进程是:p3  
qname    state    nice    ndtime    runtime  
p3       R        3       3         1  
****当前就绪队列状态为:  
qname    state    nice    ndtime    runtime  
p1       W        0       2         1  
按任一键继续.....  
The execute number:5  
**** 当前正在运行的进程是:p3  
qname    state    nice    ndtime    runtime  
p3       R        2       3         2  
****当前就绪队列状态为:  
qname    state    nice    ndtime    runtime  
p1       W        0       2         1  
进程 [p3] 已完成.  
按任一键继续.....  
The execute number:6  
**** 当前正在运行的进程是:p1  
qname    state    nice    ndtime    runtime  
p1       R        0       2         1  
****当前就绪队列状态为: 空  
进程 [p1] 已完成.  
按任一键继续.....  
所有进程已经运行完成!
```

五、总结与讨论

本次实验比较简单，主要加深了对进程概念的理解，也了解了常用进程调度算法是如何具体实现的。