山东大学 计算机科学与技术 学院

操作系统 课程实验报告

学号：202100130022	姓名：郭家宁	班级：21 数据
实验题目：实验一 进程控制		
实验学时：2	实验日期：2023-4-10	

实验目的：

-加深对于进程并发执行概念的理解。

实践并发进/线程的创建和控制方法。

观察和体验进程的动态特性。

进一步理解进程生命期期间创建、变换、撤销状态变换的过程。

掌握进程控制的方法，了解父子进程间的控制和协作关系。

练习 Linux 中有关的系统调用的编程和调试技术。

实验结果：

实验样例的实验结果：

执行缺省的指令情况：

父进程 4720，建立子进程 4721，然后暂停子进程 4721，然后父进程并没有等待子进程结束

“waitpid(pid, &status, 0); //等待子进程结束”

就自行结束了。

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe1/exe1_example$ ./pctl
SIGINT: Success

I am Parent process 4720
I am Child process 4721
My father is 4720
4720 Wakeup 4721 child.
4720 don't Wait for child done.

4721 Process continue
4721 child will Running:
执行缺省的命令
/bin/ls -a
jianing@jianing-virtual-machine:~/桌面/os_ex/exe1/exe1_example$ .  ..  Makefile  os_1  pctl  pctl.cpp  pctl.h  pctl.o
```

如果有命令行指定的执行：

子进程仍然被挂起，而父进程则在等待子进程的完成。为了检测父子进程是否都在并发执行，输入 ctrl+z 将当前进程放入后台

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe1/exe1_example$ ./pctl /bin/ls -l
SIGINT: Success

I am Parent process 4957
4957 Waiting for child done.

I am Child process 4958
My father is 4957
^Z
[1]+  已停止               ./pctl /bin/ls -l
```

输 入 ps 命令查看当前系统进程信息

fg 指令回到前台后，使用 ctrl + c，让父子进程继续执行，执行指定的命令 ls -l

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe1/exe1_example$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    4361    4335  0  80   0 -  4991 do_wai pts/0    00:00:00 bash
0 T  1000    4957    4361  0  80   0 -   693 do_sig pts/0    00:00:00 pctl
1 T  1000    4958    4957  0  80   0 -   693 do_sig pts/0    00:00:00 pctl
0 R  1000    4960    4361  0  80   0 -  5335 -      pts/0    00:00:00 ps
jianing@jianing-virtual-machine:~/桌面/os_ex/exe1/exe1_example$ fg
./pctl /bin/ls -l
^C4958 Process continue
4958 child will Running:
命令行参数指定了子进程要执行的命令
/bin/ls -l
4957 Process continue
total 84
-rw-rw-r-- 1 jianing jianing   168 Apr 10 17:00 Makefile
-rwxrwxr-x 1 jianing jianing 31128 Apr 10 16:35 os_1
-rwxrwxr-x 1 jianing jianing 21632 Apr 16 15:02 pctl
-rw-rw-r-- 1 jianing jianing  3123 Apr 11 21:38 pctl.cpp
-rw-rw-r-- 1 jianing jianing   272 Apr 10 17:45 pctl.h
-rw-rw-r-- 1 jianing jianing 12776 Apr 16 15:02 pctl.o

My child exit! status = 0
```

2. 编写一个多进程并发执行程序。父进程每隔 3 秒重复建立两个子进程,首先创建的子进程让其执行 s 命令,之后创建的子进程让其执行 ps 命令，并控制 ps 命令总在 ls 命令之前执行。

思路：先由使用系统调用 fork()，让父进程创建子进程 1，然后，再使用 pause（），阻塞子进程 1，然后进入父进程，继续创建子进程 2，，

然后使用，execvp()系统调用，执行 ps 命令，然后子进程 2 执行结束后，使用 kill()系统调用唤醒子进程 1，继续执行子进程 1，然后执行 ls 命令，等待所有进程结束，然后套用一个 while 循环并且 sleep(3)重复上述过程，即可完成实验要求。

实验现象如下：

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe1/exe1_test$ ./pctl
Start a new Program!
SIGINT: Success
I am father process 4538
I am Child2 process 4540
My father is 4538
执行的命令: /bin/ps
I am Child1 process 4539
My father is 4538
pasue child process one : 4539
    PID TTY          TIME CMD
   4361 pts/0    00:00:00 bash
   4538 pts/0    00:00:00 pctl
   4539 pts/0    00:00:00 pctl
   4540 pts/0    00:00:00 ps

My child2 exit! status = 0

4539 Process continue
执行的命令: /bin/ls
Makefile  pctl  pctl.cpp  pctl.h  pctl.o

My child1 exit! status = 0

ALL process have completed
```

可以看出：

父进程 4538，创建了子进程 1:4539 和子进程 2:4540，然后阻塞子进程 4539，然后执行子进程 4540 的命令 ps，然后子进程 2 结束后，唤醒子进程 4539，继续执行 4539，然后执行命令 ls，最后所有子进程结束。

第二次建立两个子进程：

```
Start a new Program!
SIGINT: Success
I am father process 4538
I am Child1 process 4541
My father is 4538
pasue child process one : 4541
I am Child2 process 4542
My father is 4538
执行的命令: /bin/ps
    PID TTY          TIME CMD
   4361 pts/0    00:00:00 bash
   4538 pts/0    00:00:00 pctl
   4541 pts/0    00:00:00 pctl
   4542 pts/0    00:00:00 ps

My child2 exit! status = 0

4541 Process continue
执行的命令: /bin/ls
Makefile  pctl  pctl.cpp  pctl.h  pctl.o

My child1 exit! status = 0

ALL process have completed
```

可以看出：父进程 4538，又建立两个子进程：4541, 4542，然后继续上述操作。

- 实验反映出的特征和功能：

动态性：每次创建两个子进程的先后顺序是动态变化的

并发性：一个父进程可以创建两个并发执行的子进程，提高系统的资源利用率

独立性：每个进程都有自己的独立的内存空间和资源

生命周期：进程在创建、执行、等待输入输出、终止等过程中，具有一系列的生命周期状态。

- 在真实的操作 系统中它是怎样实现和反映出教材中讲解的进程的生命期、进程的实体和进程状态 控制的？

在操作系统中，进程的生命周期是通过进程控制块 PCB 来实现的。每个进程在创建时会被分配一个唯一的进程标识符 PID，并与一个 PCB 相关联。有时还会包含一些身份信息（如进程名、进程号）、统计信息（如当前正在运行 的线程数、总计内存大小）、线程信息（当前的线程列表，内（含指向各个 TCB 的指针）等。它在一般是 C 语言的一个结构体。

- 对于进程概念和并发概念有哪些新的理解和认识？

进程是由程序创造的实体，并发是操作系统能够同时执行多个任务的能力，可以多线程并发实现，使用多个程序实现。

- 子进程是如何创建和执行新程序的?信号的机理是什么?怎样利用信号实现进程控制?

子进程是由父进程派生出来的，它们共享一些父进程的属性，但也可以有自己独立的属性和资源。使用系统调用 execvp() 系统调用实现了子程序的执行。

信号机制是一种进程间通信的方法，通过向进程发送信号的方式，来实现异步事件的处理和进程状态的控制。 进程可以接收和处理多种信号，例如 SIGINT 信号是中断信号，用于终止进程；SIGKILL 信号用于强制终止进程等。

通过向进程发送信号，进程可以响应不同类型的事件，例如 Ctrl-C 组合键产生了中断信号通过向进程发送信号，进程可以响应不同类型的事件，例如 Ctrl-C 组合键产生了中断信号

问题及收获：

对线程的并发执行以及对线程的特征和功能有了跟加深刻的理解与体会。