实验报告

实验名称：shell 编程

实验时间：2018 年 3 月 17 日

实验人员：李子强（姓名）11510352（学号）15（年级）

实验目的：学习 shell 编程

实验环境：Linux

实验步骤：

1. 阅读关于 fork,exec,wait,exit，pipe 系统调用的 man 帮助手册
2. 编译程序 fork.c 并运行，观察结果，观察进程
3. 编译程序 pipe.c 并运行，观察结果
4. 阅读关于函数 sigaction,tcsetpgrp 和 setpgid 的 man 帮助手册
5. 编译程序 signal.c 并运行，观察结果，观察进程
6. 编译程序 process.c 并运行，观察结果，观察进程
7. 写实验总结

实验陈述：

1、基础知识：

- ✧ 什么是系统调用：指运行在使用者空间的程序向操作系统内核请求需要更高权限运行的服务。系统调用提供用户程序与操作系统之间的接口。大多数系统交互式操作需求在内核态执行。如设备 IO 操作或者进程间通信。
- ✧ 简述 fork 调用：fork 是一种创建自身进程副本的操作。fork 操作会为子进程创建一个单独的地址空间。子进程拥有父进程所有内存段的精确副本。
- ✧ 如何实现进程间的通信：管道（pipe），流管道(s_pipe)和有名管道（FIFO）；信号（signal）；消息队列；共享内存；信号量；套接字（socket)
- ✧ 如何实现进程间的连接：

2、 写出下列函数的原型

fork: pid_t fork(void);

signal： typedef void (*sighandler_t)(int);
sighandler_t signal(int signum, sighandler_t handler);

pipe： int pipe(int pipefd[2]);
int pipe2(int pipefd[2], int flags);

tcsetpgrp: pid_t tcgetpgrp(int fd);
int tcsetpgrp(int fd, pid_t pgrp);

3、运行和观察结果

- ✧ fork.c
  - ➢ 简述结果（不是执行结果）：在终端输出 ls -l / 相同的结果，程序运行到 fork()生成 2 个进程，子进程进行系统调用”ls -l /”，父进程一直等待直到子进程结束。
  - ➢ 程序中如何区分父进程和子进程：通过 fork()返回值区分，程序中返回值存储在名为 pid 的变量中。Pid 为 0 是子进程，pid 不为 0 是父进程，代表子进程的 pid

号。

- pipe.c
  - 简述结果（不是执行结果）：　在终端输出 ls -l /etc/ | more 相同的结果。
  - execvp(prog2_argv[0],prog2_argv)（第 56 行）是否执行，如果没有执行是什么原因：　有执行。

- signal.c
  - 简述结果（不是执行结果）：　子进程和父进程交替输出 pid 号，给子进程一个退出信号，子进程输出提示并退出，父进程继续运行。
  - 怎样让函数 ChildHandler 执行？　用 kill 子进程，子进程给父进程传递信号。

- process.c
  - 简述结果（不是执行结果）：显示父进程和子进程的 pid 号，子进程运行 vi，并使子进程成为前台程序
  - 进程列表中有几个./process，区别在哪里：1 个，另一个相关的是 vi。
  - 杀死主进程后，出现什么情况：　子进程也结束了。

实验总结：

学习了 fork，process，signal，pipe 的使用。