

深圳大学实验报告

课程名称: Linux 系统编程

实验项目名称: 进程控制和守护进程的创建

学 院: 计算机与软件学院

专 业: 数学与应用数学, 计算机科学与技术

指导老师: 老师姓名

学生姓名: 报告人姓名

学 号: 2011XXXXXX

实验时间: 2014 年 3 月 24 日

报告提交时间: 2014 年 4 月 25 日

Contents

实验目标	2
实验环境与工件	2
实验内容与步骤	2
问题一	2
问题二	4

实验目标

1. 掌握 fork() 系统调用及进程的相关概念
2. 掌握 wait() 和 waitpid() 系统调用
3. 掌握进程组，会话进程等概念和 setsid() 系统调用
4. 掌握文件重定向的技巧
5. 掌握创建守护进程的步骤及其实现

实验环境与工件

1. 湖边 Linux 实验室
2. Fedora 13

实验内容与步骤

下面的程序会用到如下程序段：从命令行获取数字参数，参考实现见下图：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]){
    pid_t childpid = 0;
    int i,n;

    if (argc!=2){
        fprintf(stderr,"Usage: %s processes \n",argv[0]); return 1;
    }

    n=atoi(argv[1]);
```

问题一

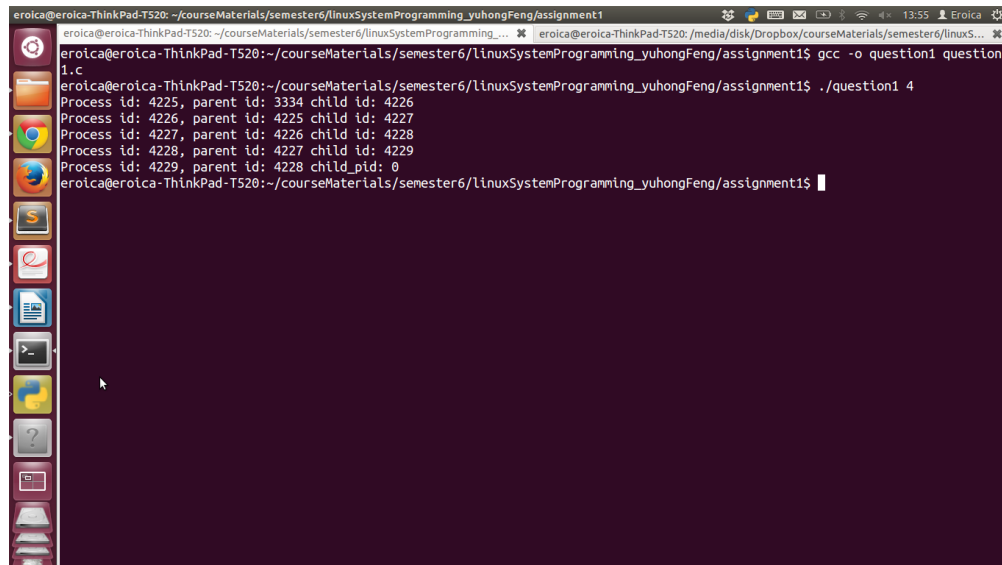
编例实现创建 n 个子进程 P_1, P_2, \dots, P_n ，其中，各进程之间的关系是： P_1 是调用进程的子进程， $P_{(k+1)}$ 是 P_k 的子进程。请打印各进程本身的进程号、父进程号，子进程号。参考运行结果如下。要求：（1）每个父进程都要等待子进程退出后才能退出；（2） n 通过命令行参数传入；（3）附上源代码截图和运行结果截图。（20 分）

相关代码：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[]){
6     pid_t child_pid = 0;
7     int i, n;
8
```

```
9      if (argc != 2){
10          fprintf(stderr, "Usage: %s _process_\n", argv[0]);
11          return 1;
12      }
13
14      n = atoi(argv[1]);
15
16      for (i = 0; i < n; i++){
17          child_pid = fork();
18          if (child_pid > 0){
19              printf("Process_id: %d, _parent_id: %d_\n", \
20                  getpid(), getppid());
21              printf("child_id: %d\n", child_pid);
22              break;
23          } else if (child_pid < 0) {
24              printf("fork_error!\n");
25              exit(-1);
26          }
27      }
28
29      if (i == n){
30          printf("Process_id: %d, _parent_id: %d_\n", getpid(), getppid());
31          printf("child_pid: %d\n", child_pid);
32      }
33
34      if (child_pid > 0){
35          waitpid(child_pid, NULL, 0);
36      }
37      return 0;
38  }
```

实验结果:



```
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$ gcc -o question1 question1.c
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$ ./question1 4
Process id: 4225, parent id: 3334 child id: 4226
Process id: 4226, parent id: 4225 child id: 4227
Process id: 4227, parent id: 4226 child id: 4228
Process id: 4228, parent id: 4227 child id: 4229
Process id: 4229, parent id: 4228 child_pid: 0
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$
```

问题二

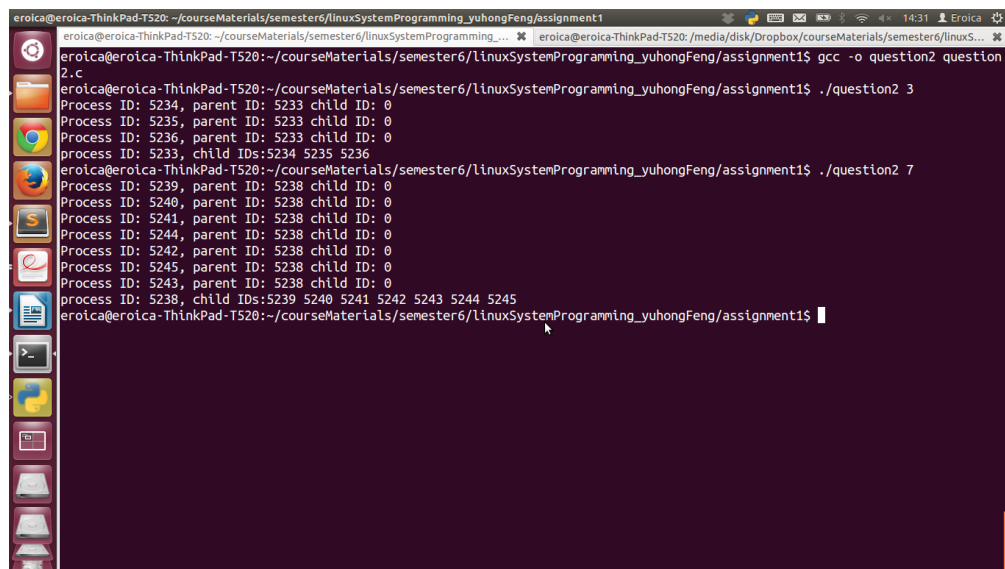
编例实现创建 n 个子进程 P_1, P_2, \dots, P_n , 其中, 各进程之间的关系是: P_1, \dots, P_n 都是调用进程的子进程。请打印各进程本身的进程号、父进程号, 子进程号。参考运行结果如下。要求: (1) 每个父进程都要等待子进程退出后才能退出; (2) n 通过命令行参数传入; (3) 附上源代码截图和运行结果截图。(20 分)

相关代码:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[]){
6     pid_t child_pid = 0;
7     int i, j, n;
8     int child_IDS[200];
9
10    if (argc != 2){
11        fprintf(stderr, "Usage: %s process\n", argv[0]);
12        return 1;
13    }
14
15    n = atoi(argv[1]);
16
17    for (i = 0; i < n; i++){
18        child_pid = fork();
19        if (child_pid > 0){
20            child_IDS[i] = child_pid;
21        }
22        if (child_pid == 0){
23            printf("Process_ID: %d, parent_ID: %d\n", \
24                getpid(), getppid());
25            printf("child_ID: %d\n", child_pid);
26            exit(1);
27        } else if (child_pid < 0) {
28            printf("fork_error!\n");
29            exit(-1);
30        }
31    }
32
33    if (child_pid > 0){
34        waitpid(child_pid, NULL, 0);
35    }
36
37    if (i == n){
38        printf("process_ID: %d, child_IDS:", getpid());
39        for (j = 0; j < n; j++){
40            printf("%d ", child_IDS[j]);
41        }
42        printf("\n");
```

```
43     }  
44  
45     return 0;  
46 }
```

实验结果：



```
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1  
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$ gcc -o question2 question2.c  
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$ ./question2 3  
Process ID: 5234, parent ID: 5233 child ID: 0  
Process ID: 5235, parent ID: 5233 child ID: 0  
Process ID: 5236, parent ID: 5233 child ID: 0  
Process ID: 5233, child IDs: 5234 5235 5236  
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$ ./question2 7  
Process ID: 5239, parent ID: 5238 child ID: 0  
Process ID: 5240, parent ID: 5238 child ID: 0  
Process ID: 5241, parent ID: 5238 child ID: 0  
Process ID: 5242, parent ID: 5238 child ID: 0  
Process ID: 5243, parent ID: 5238 child ID: 0  
Process ID: 5244, parent ID: 5238 child ID: 0  
Process ID: 5245, parent ID: 5238 child ID: 0  
Process ID: 5243, parent ID: 5238 child ID: 0  
Process ID: 5238, child IDs: 5239 5240 5241 5242 5243 5244 5245  
eroica@eroica-ThinkPad-T520: ~/courseMaterials/semester6/linuxSystemProgramming_yuhongFeng/assignment1$
```