《Linux 操作系统设计实践》实验一：进程管理

实验内容：

运行代码：

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

int father\_process\_waiting\_flag = 1; //父进程等待标志

int child\_process\_id = 0; //子进程ID

void father\_process\_wait() //父进程等待

{

while(father\_process\_waiting\_flag) {

printf("Father Process Waiting\n");

sleep(1);

}

}

void child\_process\_wait() //子进程等待

{

while(1) {

printf("Child Process Waiting\n");

sleep(2);

}

}

void father\_process\_stop() //父进程中止

{

if(child\_process\_id) { //若子进程仍在运行，则中止子进程

kill(child\_process\_id, SIGUSR1); //父进程发送用户信号给子进程

wait(0); //等待子进程执行退出

printf("Child Process Exited\n");

child\_process\_id = 0;

} else { //若子进程不存在，则父进程退出

printf("Father Process Exiting\n");

father\_process\_waiting\_flag = 0;

}

}

void child\_process\_stop() //子进程中止

{

printf("Child Process Exiting\n");

exit(0);

}

int main()

{

child\_process\_id = fork();

if(child\_process\_id < 0) { //fork失败时的处理

perror("");

exit(child\_process\_id);

}

if(child\_process\_id) { //父进程分支

signal(SIGINT, father\_process\_stop); //绑定键盘^C信号到对应程序

father\_process\_wait(); //进入循环等待

} else { //子进程分支

signal(SIGINT, SIG\_IGN); //忽略键盘^C信号

signal(SIGUSR1, child\_process\_stop); //绑定用户信号到对应程序

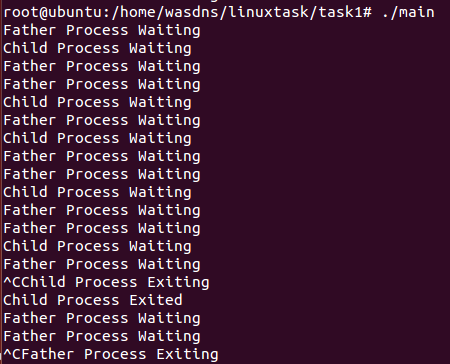
child\_process\_wait(); //进入循环等待

}

return 0;

}

运行结果：



实验总结：

利用fork()函数，创建父子进程，同时利用signal()函数对SIGINT信号进行管控，使得子进程存在时中止字进程，若子进程不存在则中止父进程。

kill()用于父进程发送用户信号SIGUSR1给子进程，子进程以此执行中止函数。