**s实验4 信号机制**

实验目的

1．熟悉信号的基本概念。

2．掌握Linux系统中进程之间利用信号机制进行软中断通信的基本原理。

实验内容

要求：所有程序保存在自己学号命名的文件夹中。

1．运行以下程序，分析程序结果。

/\*signal1.c\*/

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

#include<stdlib.h>

int i;

void intfun( )

{

i=0;

}

int main( )

{

int k,j,pid;

j=1;

while((pid=fork( ))==-1);

if (pid>0)

{

for(k=1;k<3;k++)

{

printf("how are you!\n");

sleep(1);

}

kill(pid,17); /\*发送软中断信号给子进程\*/

wait(0); /\*等待子进程终止\*/

printf("Parent exited\n");

exit(0);

}

else

{

signal(17,intfun); /\*预置软中断信号17\*/

i=1;

while(i==1) /\*循环显示并等待父进程发软中断信号\*/

{

printf("I am child:%d\n",j++);

sleep(1);

}

printf("Child exited\n");

exit(0);

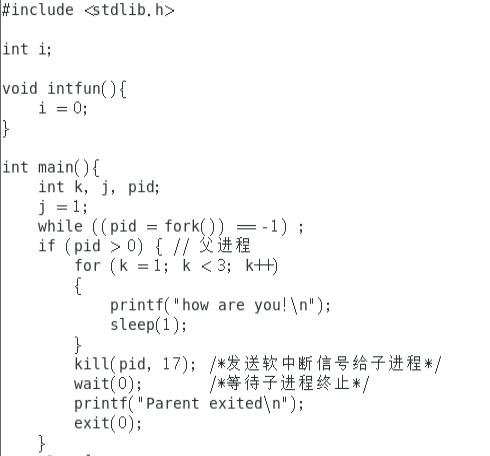
}

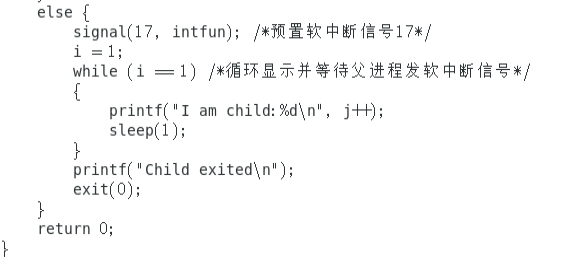
return 0;

}

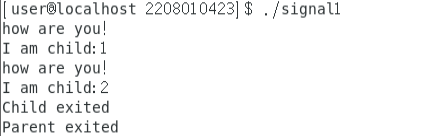
要求：1.运行该程序，运行结果截图，并简要分析程序实现的功能。

1. 总结有信号的进程如何执行
2. 代码





结果截图



分析程序实现功能 :

1. 创建父子进程
2. 父进程的行为
3. 父进程打印 how are you, 每秒一次
4. 父进程向子进程发送 17 信号
5. 父进程发送信号之后会等待子进程执行完毕
6. 子进程执行完毕之后，父进程打印 “Parent exited”，然后退出
7. 子进程的行为
8. 首先设置软中断信号 17，执行逻辑是设置 i = 0
9. 初始化 i = 1, 只要 i 等于 1，每秒打印 I am child j, 其中 j 每次加 1。

（那么当收到信号的时候，i = 0， 此时就会退出循环）

1. 退出循环，打印 child exited, 然后 eixt(0)

2, 有信号的程序如何执行 :

当信号到达时，操作系统会打断当前进程的执行（如果没有被阻塞），并调用相应的信号处理函数。

信号处理程序执行后，控制权返回到被打断的程序，继续执行。

2．运行以下程序，分析程序运行结果。

/\*signal2.c\*/

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

#include <stdlib.h>

void waiting( ),stop( );

int wait\_mark;

void waiting( )

{

while(wait\_mark!=0);

}

void stop( )

{

wait\_mark=0;

}

main( )

{

int p1,p2;

signal(SIGINT,SIG\_IGN); /\*设置忽略键盘中断信号，创建的子进程与父进程相同设置\*/

while((p1=fork( ))==-1); /\*创建子进程p1\*/

if (p1>0)

{

while((p2=fork( ))==-1); /\*创建子进程p2\*/

if(p2>0)

{

printf("请按Ctrl+c键终止程序!\n");

wait\_mark=1;

signal(SIGINT,stop); /\*接收到^c信号，转stop\*/

waiting( );

kill(p1,10); /\*向p1发软中断信号10\*/

kill(p2,12); /\*向p2发软中断信号12\*/

wait(0); /\*同步\*/

wait(0);

printf("Parent process is killed!\n");

exit(0);

}

else

{

wait\_mark=1;

signal(12,stop); /\*接收到软中断信号12，转stop\*/

waiting( );

lockf(1,1,0); /\*加锁，第一个参数为stdout（标准输出设备的描述符）\*/

printf("Child process 2 is killed by parent!\n");

lockf(1,0,0); /\*解锁，第一个参数为stdout（标准输出设备的描述符）\*/

exit(0);

}

}

else

{

wait\_mark=1;

signal(10,stop); /\*接收到软中断信号10，转stop\*/

waiting( );

lockf(1,1,0);

printf("Child process 1 is killed by parent!\n");

lockf(1,0,0);

exit(0);

}

}

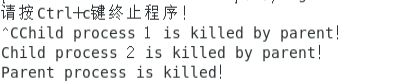
注意： 按Ctrl + C 的时候，会向**所有的前台进程**发送 SIGINT 信号。

1. 运行编译后的程序，查看输出；按Ctrl+c键再查看程序执行结果。

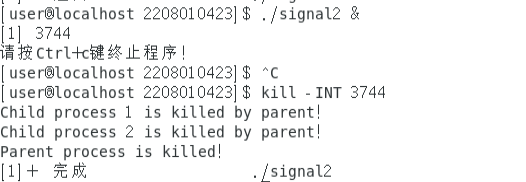
不按 ctrl + c ，结果如下 , 说明三个进程都在执行 waiting 函数等待。



按 ctrl + c 之后: 父进程停止等待，向两个子进程发送信号，两个子进程收到信号之后，执行 stop 函数，停止等待，并执行相关打印。

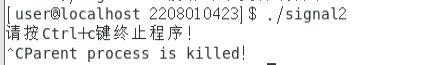


1. 将编译后的程序后台执行，查看输出；按Ctrl+c键再查看程序执行结果；在shell提示符下输入kill –INT pid，查看程序执行结果，其中，pid是该程序后台执行后显示的父进程号。



后台执行，先打印 “请按ctrl+c键终止程序”。按下ctrl+c之后，由于ctrl+c 只向所有前台发送信号，所以进程不会被终止。执行 kill -INT 3744 时，会发生与按 ctrl+c 时相同的行为。

1. 将第一个signal(SIGINT,SIG\_IGN)注释，看一下运行结果，分析原因。



由于子进程不再忽略 ctrl + c 信号，收到 ctrl + c 时，会直接终止。所以最终只会打印”"请按Ctrl+c键终止程序!”.

要求：写出三种情况的运行结果截图及分析**（写出程序结果的分析，不要翻译程序，30字之内）**

3．（选做）设计一个程序，父进程创建一个子进程，子进程睡眠5秒后给父进程发送一个信号，然后终止；父进程等待子进程的信号，收到信号后执行一个函数，输出系统的日期（执行语句：system("date"))后结束。

提示：父进程等信号时可以执行无限循环（保证能收到信号）或者执行pause()函数，子进程的发送的信号使用SIGUSR1或SIGUSR2，父进程定义自己的信号捕捉函数。

要求：程序源代码和运行结果截图



