**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);

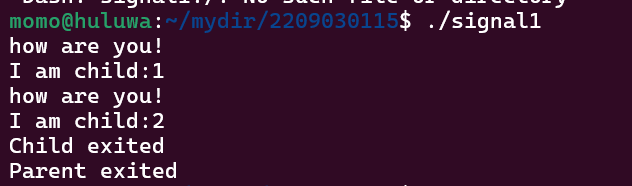
}

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。

2.总结有信号的进程如何执行



该程序创建了一个父进程和一个子进程。

父进程在创建子进程后，会打印两次 “how are you!” 信息，每次打印后休眠1秒。

父进程在打印完毕后，向子进程发送信号17（SIGCHLD），这是一个通常用于通知父进程子进程状态变化的信号。

子进程收到信号后，会调用信号处理函数 intfun，该函数将全局变量 i 设置为0，这将导致子进程退出其循环并终止。

父进程在发送信号后，会调用 wait 函数等待子进程终止。

父进程和子进程在终止前都会打印一条退出信息。

总结：

当一个进程设置了一个信号处理函数（通过 signal 或 sigaction 函数），它就为特定的信号指定了一个处理程序。

如果该进程收到了一个信号，操作系统会打断当前执行的代码，并调用信号处理函数。

信号处理函数执行完毕后，进程通常会继续执行它在信号到达时被打断的代码。

如果信号处理函数中调用了 exit 或 \_exit 函数，进程会立即终止，不会返回到被中断的代码。

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);

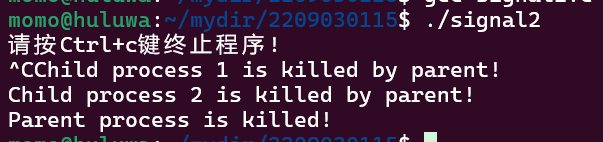
exit(0);

}

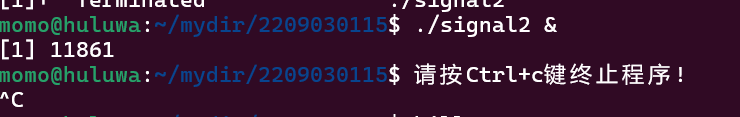
}

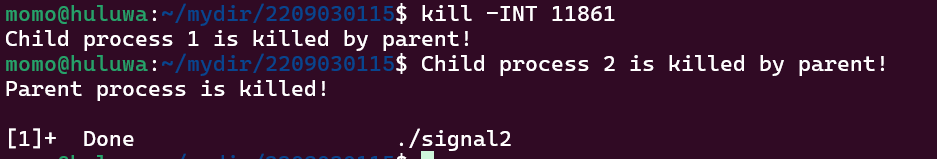
注意： 按Ctrl + C 的时候，会向**所有的前台进程**发送 SIGINT 信号。

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。



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。





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

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）**

1. **按Ctrl+C后，父进程发信号，子进程和父进程依次终止，因为SIGINT的默认行为是终止进程。**
2. **后台执行，Ctrl+c无效。**
3. **未忽略SIGINT，程序启动后立即因Ctrl+C终止。**



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

