实验五　进程间通信

UNIX/LINUX 系统的进程间通信机构（IPC）允许在任意进程间大批量地交换数据。本实验的目的是了解和熟悉 LINUX 支持的信号量机制、管道机制、消息通信机制及共享存储区机制。

（一）信号量机制实验

【实验目的】

1、了解什么是信号。
2、熟悉 LINUX 系统中进程之间软中断通信的基本原理。

【实验内容】

1、编写一段程序，使用系统调用 fork()创建两个子进程，再用系统调用 signal()让父进程捕捉键盘上来的中断信号（即按 ctrl+c 键），当捕捉到中断信号后，父进程用系统调用 kill( )向两个子进程发出信号，子进程捕捉到信号后，分别输出下列信息后终止：

Child process 1 is killed by parent!
Child process 2 is killed by parent!

父进程等待两个子进程终止后，输出以下信息后终止：

Parent process is killed!

<参考程序>

```
# include<stdio.h>
# include<signal.h>
# include<unistd.h>
# include<stdlib.h>
void waiting();
void stop();
int wait_mark;
main()
{ int  p1, p2;
signal(SIGINT,stop);
while((p1=fork())==-1);
   if(p1>0)
{ ①
while((p2=fork())==-1);
   If(p2>0)
    { ②
      wait_mark=1;
      waiting(0);
      kill(p1,10);
```

```
      kill(p2,12);
      wait();
      wait();
      printf(“parent process is killed!\n”);
      exit(0);
    }
    else
    {
wait_mark=1;
signal(12,stop);
waiting();
lockf(1,0,0);
exit(0);
}
}
else
{
   wait_mark=1;
   signal(10,stop);
   waiting();
   lockf(1,1,0);
   printf(“child process 1 is killed by parent!\n”);
   lockf(1,0,0);
   exit(0);
}
   }
void waiting()
{
   while(wait_mark!=0);
}
void stop()
{
   wait_mark=0;
}
```

实验要求：

⑴、运行程序并分析结果。
⑵、如果把 signal(SIGINT,stop)放在①号和②号位置，结果会怎样并分析原因。
⑶、该程序段前面部分用了两个 wait(0),为什么？
⑷、该程序段中每个进程退出时都用了语句 exit(0),为什么？

2、司机售票员问题（选做题）

编程用 fork()创建一个子进程代表售票员，司机在父进程中，再用系统调用 signal()让父进程（司机）捕捉来自子进程（售票员）发出的中断信号，让子进程（售票员）捕捉来自（司机）发出的中断信号，以实现进程间的同步运行。

【实验报告】

1、列出调试通过程序的清单，分析运行结果。
2、给出必要的程序设计思路和方法（或列出流程图）。
3、总结上机调试过程中所遇到的问题和解决方法及感想。

【实验相关资料】

一、信号

1. 信号的基本概念

每个信号都对应一个正整数常量(称为 signal number,即信号编号。定义在系统头文件<signal.h>中)，代表同一用户的诸进程之间传送事先约定的信息的类型，用于通知某进程发生了某异常事件。每个进程在运行时，都要通过信号机制来检查是否有信号到达。若有，便中断正在执行的程序，转向与该信号相对应的处理程序，以完成对该事件的处理；处理结束后再返回到原来的断点继续执行。实质上，信号机制是对中断机制的一种模拟，故在早期的 UNIX 版本中又把它称为软中断。

⑴ 信号与中断的相似点：

①采用了相同的异步通信方式；
②当检测出有信号或中断请求时，都暂停正在执行的程序而转去执行相应的处理程序；
③都在处理完毕后返回到原来的断点；
④对信号或中断都可进行屏蔽。

⑵ 信号与中断的区别：

①中断有优先级，而信号没有优先级，所有的信号都是平等的；
②信号处理程序是在用户态下运行的，而中断处理程序是在核心态下运行；
③中断响应是及时的，而信号响应通常都有较大的时间延迟。

⑶ 信号机制具有以下三方面的功能：

①发送信号。发送信号的程序用系统调用 kill( )实现；
②预置对信号的处理方式。接收信号的程序用 signal( )来实现对处理方式的预置；
③收受信号的进程按事先的规定完成对相应事件的处理。

2、信号的发送

信号的发送，是指由发送进程把信号送到指定进程的信号域的某一位上。如果目标进程正在一个可被中断的优先级上睡眠，核心便将它唤醒，发送进程就此结束。一个进程可能在其信号域中有多个位被置位，代表有多种类型的信号到达，但对于一类信号，进程却只能记

住其中的某一个。进程用 kill( )向一个进程或一组进程发送一个信号。

3、对信号的处理

当一个进程要进入或退出一个低优先级睡眠状态时，或一个进程即将从核心态返回用户态时，核心都要检查该进程是否已收到软中断。当进程处于核心态时，即使收到软中断也不予理睬；只有当它返回到用户态后，才处理软中断信号。对软中断信号的处理分三种情况进行：

①如果进程收到的软中断是一个已决定要忽略的信号（function=1），进程不做任何处理便立即返回；

②进程收到软中断后便退出（function=0）；

③执行用户设置的软中断处理程序。

二、所涉及的中断调用

1、kill( )

系统调用格式

int kill(pid,sig)

参数定义

int pid,sig;

其中，pid 是一个或一组进程的标识符，参数 sig 是要发送的软中断信号。

（1）pid>0 时，核心将信号发送给进程 pid。

（2）pid=0 时，核心将信号发送给与发送进程同组的所有进程。

（3）pid=-1 时，核心将信号发送给所有用户标识符真正等于发送进程的有效用户标识号的进程。

2、signal( )

预置对信号的处理方式，允许调用进程控制软中断信号。

系统调用格式

signal(sig,function)

头文件为

#include <signal.h>

参数定义

signal(sig,function)

int sig;

void (*func) ( )

其中 sig 用于指定信号的类型，sig 为 0 则表示没有收到任何信号，余者如下表：

值	名　字	说　　明
01	SIGHUP	挂起（hangup）
02	SIGINT	中断，当用户从键盘按^c 键或^break 键时
03	SIGQUIT	退出，当用户从键盘按 quit 键时
04	SIGILL	非法指令
05	SIGTRAP	跟踪陷阱（trace trap），启动进程，跟踪代码的执行

06	SIGIOT	IOT 指令
07	SIGEMT	EMT 指令
08	SIGFPE	浮点运算溢出
09	SIGKILL	杀死、终止进程
10	SIGBUS	总线错误
11	SIGSEGV	段违例（segmentation violation），进程试图去访问其虚地址空间以外的位置
12	SIGSYS	系统调用中参数错，如系统调用号非法
13	SIGPIPE	向某个非读管道中写入数据
14	SIGALRM	闹钟。当某进程希望在某时间后接收信号时发此信号
15	SIGTERM	软件终止（software termination）
16	SIGUSR1	用户自定义信号 1
17	SIGUSR2	用户自定义信号 2
18	SIGCLD	某个子进程死
19	SIGPWR	电源故障

function：在该进程中的一个函数地址，在核心返回用户态时，它以软中断信号的序号作为参数调用该函数，对除了信号 SIGKILL，SIGTRAP 和 SIGPWR 以外的信号，核心自动地重新设置软中断信号处理程序的值为 SIG_DFL，一个进程不能捕获 SIGKILL 信号。

function 的解释如下：

（1）function=1 时，进程对 sig 类信号不予理睬，亦即屏蔽了该类信号；

（2）function=0 时，缺省值，进程在收到 sig 信号后应终止自己；

（3）function 为非 0，非 1 类整数时，function 的值即作为信号处理程序的指针。