实验五 进程间通信——信号

**一、实验目的**

1、了解采用信号机制进行进程间通信的原理。

2、掌握通过信号进行进程间通信的方法。

**二、实验原理**

1、信号是软件层次上的中断模拟，是一种异步通信方式。

2、信号机制的使用流程如下：

信号产生🡪信号注册/注销🡪信号处理。

3、信号处理函数：signal()/sigaction()

4、信号集函数组的三个模块：

（1）创建信号集合：

sigemptyset()：初始化信号集合为空。

sigfillset()：初始化信号集合为所有信号的集合。

sigaddset()：将指定集合加到信号集中去。

（2）登记信号处理器：登记信号处理器用于决定进程如何处理信号。

sigprocmask()：判断检测或更改信号屏蔽字。

sigaction()：设定进程接收到特定信号后的行为。

（3）测试信号sigpending()：读取当前进程的未决信号集。

**三、实验内容**

编写sigaction.c源程序，实现如下功能：

1、a.捕获“Ctrl-C”信号时，输出”I have get SIGINT.”

b.捕获“Ctrl-\”信号时， 输出”I have get SIGQUIT.”

2、a.设置进程屏蔽字，本例中屏蔽信号SIGINT，并测试SIGINT是否被屏蔽，如果被 屏蔽，输出“The SIGINT signal has been ignored”。

b.从信号SIGINT所在的信号集中删除SIGINT，同时取消对SIGINT的屏蔽并挂起进程。接着再次按下“Ctrl-C”，输出”I have get SIGINT.”。

#### 四．代码

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<signal.h>

volatile int quit = 0;

void sigint\_handler(int num){

quit = 1;

printf("I have get SIGINT.\n");

}

void sigquit\_handler(int num){

printf("I have get SIGQUIT.\n");

}

int main(int argc, char \*argv[]){

// set sigint handler

struct sigaction sigint\_act;

sigint\_act.sa\_handler = sigint\_handler;

sigemptyset(&sigint\_act.sa\_mask);

sigint\_act.sa\_flags = 0;

sigaction(SIGINT, &sigint\_act, NULL);

// set sigquit handler

struct sigaction sigquit\_act;

sigquit\_act.sa\_handler = sigquit\_handler;

sigemptyset(&sigquit\_act.sa\_mask);

sigquit\_act.sa\_flags = 0;

sigaction(SIGQUIT, &sigquit\_act, NULL);

printf("Test: send current process SIGINT and SIGQUIT.\n");

kill(getpid(), SIGINT);

kill(getpid(), SIGQUIT);

printf("block sigint....\n");

sigset\_t set, prev\_set;

sigemptyset(&set);

sigaddset(&set, SIGINT);

sigprocmask(SIG\_SETMASK, &set, &prev\_set);

quit = 0;

kill(getpid(), SIGINT); // sigint will be deal until UNBLOCK SIGINT.

if(sigismember(&set, SIGINT) == 1 && quit == 0){

printf("The SIGINT signal has been ignored\n");

}

printf("unblock sigint....\n");

sigprocmask(SIG\_SETMASK, &prev\_set, NULL);

return 0;

}

