

**计算机学院实验报告**

班 级 网络工程1801 姓 名 吴斌 学 号 18408020129

时 间 2020.11.1 评 分 教师签名

课程名称  **网络程序设计**

**1、实验名称；** 网络编程基础技术

**2、实验目的；**

1.理解多进程的概念并掌握其编程应用

2.理解多线程的概念并掌握其编程应用

3.理解信号机制的原理并掌握其编程应用

4.掌握间隔定时器的编程应用

**3、实验原理；**

1.进程ID，创建进程，进程善后

2.线程ID，创建线程，

3.信号，信号集和信号掩码，信号处理

**4、实验内容与结果；**

**场景1（1）forkprocs.c.**

**#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
int main()  
{  
pid\_t child\_pid;  
int i=1;  
printf("tha main program processs id is %d\n",getpid());  
child\_pid=fork();  
if (child\_pid!=0)  
{  
printf("this is the parent process id,with id =%d and i=%d\n ",(int)getpid(),i);  
printf("this child  id is %d\n",(int)child\_pid);  
}  
else  
{  
sleep(5);  
i++;  
printf("this is child process ,with id =%d and i =%d\n",(int)getpid(),i);  
printf("this  parent id is %d\n",(int)getppid());  
  
}  
return 0;  
  
}**

**场景1（2）forkprocs.c.**

**#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
int main()  
{  
pid\_t child\_pid;  
int i=1;  
printf("tha main program processs id is %d\n",getpid());  
child\_pid=fork();  
if (child\_pid!=0)  
{  
printf("this is the parent process id,with id =%d and i=%d\n ",(int)getpid(),i);  
printf("this child  id is %d\n",(int)child\_pid);  
sleep(5);  
}  
else  
{  
  
i++;  
printf("this is child process ,with id = %d and i=%d\n",(int)getpid(),i);  
printf("this  parent id is %d\n",(int)getppid());  
  
}  
return 0;  
  
}**

**场景2 catchctrl.c**

**#include <stdio.h>  
#include <signal.h>  
#include <stdlib.h>  
#include <unistd.h>  
struct sigaction act;  
  
static int count;  
void catchctrlc(int signo){  
count++;  
char msg[]="ctrl+c entered!\n";  
int msglen=sizeof(msg);  
write(STDERR\_FILENO,msg,msglen);  
if(count==2){  
char errmsg[]="Failed to restore SIGINT\n";  
int meslen=sizeof(errmsg);  
act.sa\_handler=SIG\_DFL;  
if(sigaction(SIGINT,&act,NULL)==-1)  
{  
write(STDERR\_FILENO,errmsg,msglen);}  
}  
}  
int main(){  
  
act.sa\_handler=catchctrlc;  
act.sa\_flags=0;  
if(sigemptyset(&act.sa\_mask)==-1||(sigaction(SIGINT,&act,NULL)==-1))  
{  
perror("failef to set SIGINT to handle Ctrl-c");  
}**

**for(;;);  
return 0;  
}**

**场景3 ignoresig.c**

**#include <stdio.h>  
#include <signal.h>  
#include <stdlib.h>  
#include <string.h>  
struct sigaction act;  
int main(int argc,char \*argv[]){  
act.sa\_flags=0;  
act.sa\_handler=SIG\_IGN;  
printf("SIGTSTP is ignored\n");  
if(sigaction (SIGTSTP,&act,NULL)==-1)  
perror("failef to set handler for SIGSTP");  
sleep(10);  
printf("SIGSTP handler is restored\n");  
act.sa\_handler=SIG\_DFL;**

**if(sigaction (SIGTSTP,&act,NULL)==-1)  
perror("failef to restore handler for SIGSTP");  
for(;;);  
return 0;  
  
}**

**场景4 interval.c**

**#include <stdio.h>  
#include <signal.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/time.h>  
struct itimerval val\_alarm;  
void sig\_handler(int sig)  
{  
printf("timer has expired,elapsed %d seconds\n",(int)val\_alarm.it\_value.tv\_sec);  
val\_alarm.it\_value.tv\_sec++;  
if(setitimer(ITIMER\_REAL,&val\_alarm,NULL)==-1)  
perror("ailed to call setitimer");  
}  
int main()  
{  
val\_alarm.it\_interval.tv\_sec=0;  
val\_alarm.it\_interval.tv\_usec=0;  
val\_alarm.it\_value.tv\_sec=1;  
val\_alarm.it\_value.tv\_usec=0;  
struct sigaction sa;  
memset(&sa,0,sizeof(sa));  
sa.sa\_handler=&sig\_handler;  
if(sigaction(SIGALRM,&sa,NULL)==-1)  
perror("failed to call sigcation");  
if(setitimer(ITIMER\_REAL,&val\_alarm,NULL)==-1)  
perror("ailed to call setitimer");  
while(1)  
;  
return 0;  
}**

**场景5（1） zombie.c**

**#include <stdlib.h>  
#include <sys/types.h>  
#include <unistd.h>  
int main(){  
pid\_t child\_d;  
child\_d=fork();  
if(child\_d>0)  
{  
sleep(30);  
printf("the parent process is finished\n");  
  
}  
else{  
printf("the child process is finished\n");  
exit(0);  
}  
return 0;  
}**

**场景5（2） waitpid.c**

**#include <stdlib.h>  
#include <stdio.h>  
#include <unistd.h>  
#include <signal.h>  
void handler(int signo){  
if(waitpid(-1,NULL,WNOHANG)<0)  
perror("waitpid() error\n");  
else{  
printf("the child process is finished\n");  
sleep(10);  
  
}  
}  
void main(){  
signal(SIGCHLD,handler);  
pid\_t child\_pid;  
child\_pid=fork();  
if(child\_pid>0)  
{  
sleep(30);  
printf("the parent process is finished\n");  
  
}  
else{  
exit(0);}  
  
}**

**5、总结与讨论；**

通过本次实验，理解了多进程的概念并掌握了其编程应用，理解了多线程的概念并掌握了其编程应用，理解了信号机制的原理并掌握了其编程应用，掌握了间隔定时器的编程应用。