《操作系统》

上机报告

班 级： 1503012

学 号： 15030120044

姓 名： 秦龙

实验 1：



源程序：

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main(){

char father[]={"消息来自父进程.\n"};

char child[]={"消息来自子进程.\n"};

int chan1[2],chan2[2];

char buff[50];

memset(buff,0,sizeof(buff));

pipe(chan1);

pipe(chan2);

if(fork()){

close(chan1[0]);

close(chan2[1]);

write(chan1[1],father,strlen(father));

close(chan1[1]);

read(chan2[0],buff,50);

printf("这是父进程:%s\n",buff);

}else{

close(chan1[1]);

close(chan2[0]);

read(chan1[0],buff,50);

printf("这是子进程:%s\n",buff);

write(chan2[1],child,strlen(child));

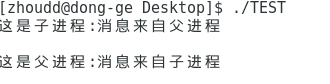
close(chan2[1]);

}

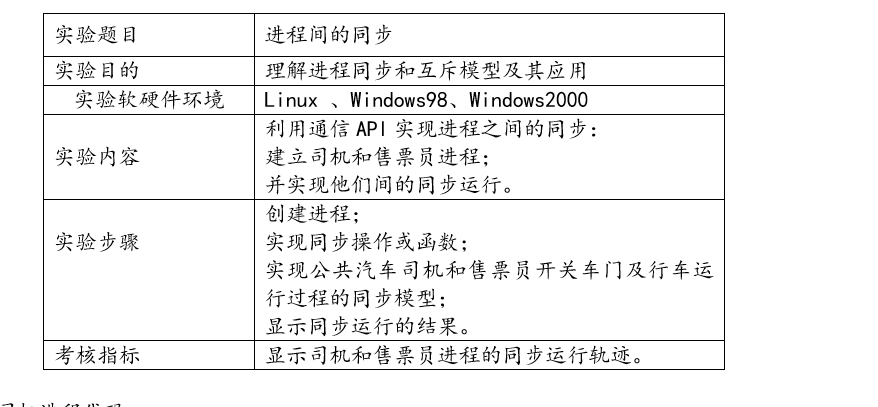
return 0;

}

运行结果



实验 2：



程序代码：

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <signal.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/shm.h>

int set\_semvalue(int sem\_id,int value){

return(semctl(sem\_id,0,SETVAL,value));

}

void del\_semvalue(int sem\_id){

if (semctl(sem\_id,0,IPC\_RMID,0)==-1) {

fprintf(stderr, "删除semaphore失败\n");

}

}

int P(int sem\_id){

struct sembuf sem\_b;

sem\_b.sem\_num=0;

sem\_b.sem\_op=-1;

sem\_b.sem\_flg=SEM\_UNDO;

if(semop(sem\_id,&sem\_b,1)==-1){

fprintf(stderr, "P操作失败\n");

return(1);

}

return(0);

}

int V(int sem\_id){

struct sembuf sem\_b;

sem\_b.sem\_num=0;

sem\_b.sem\_op=1;

sem\_b.sem\_flg=SEM\_UNDO;//zidongshifangchiyouxinhaoliang

if (semop(sem\_id,&sem\_b,1)==-1) {

fprintf(stderr, "V 操作失败\n");

return(1);

}

return(0);

}

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

int s1,s2;

s1=semget(IPC\_PRIVATE,1,IPC\_CREAT);//douke

s2=semget((key\_t)34,1,IPC\_CREAT);

if (set\_semvalue(s1,0)==-1||set\_semvalue(s2,0)==-1) {

fprintf(stderr, "初始化 semvalue失败\n");

exit(EXIT\_FAILURE);

}

pid\_t p;

if((p=fork())<0)

{ printf("creat error");

exit(1);

}

else if (p==0) {

while(1){

sleep(5);

printf("司机正常到达站点，停车\n");

V(s2);

P(s1);

printf("开动车辆离开站点\n");

}

}

else{

while (1) {

sleep(2);

printf("售票员售票\n");

P(s2);

printf("打开车门，上来乘客，关上车门\n");

V(s1);

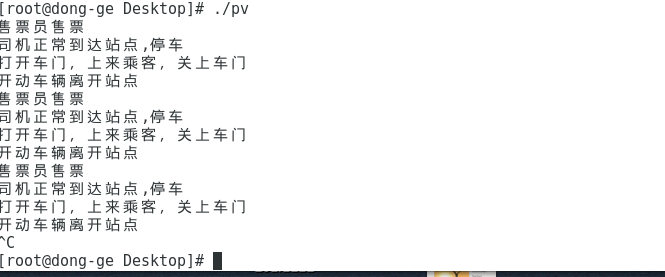
}

}

exit(0);

}

运行结果：



实验 3：

实验题目

线程共享进程数据

实验目的

了解线程与进程之间的数据共享关系。

创建一个线程，在线程中更改进程中的数据。

实验软件环境

VC++6.0 或者Linux 操作系统

实验内容

在进程中定义全局共享数据，在线程中直接引用该数据进行更改并输出

该数据。

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <pthread.h>

void \*thread\_function(void \*arg); //创建线程

int count=10;

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

int res;

pthread\_t a\_thread; //用于声明线程 ID

void \*thread\_result;//

printf("进程正在运行...\n");

printf("进程的count=%d\n",count); //创建线程 pthread\_create, 0 则创建成功 res=pthread\_create(&a\_thread,NULL,thread\_function,(void\*)&count);

if (res!=0) {

perror("Thread creation faild");

exit(EXIT\_FAILURE);

}

res=pthread\_join(a\_thread,&thread\_result);

if (res!=0) {

perror("Thread join failed");

exit(EXIT\_FAILURE);

}

printf("return count=%d\n",(\*(int\*)thread\_result));

printf("线程结束\n");

printf("进程结束\n");

}

void \*thread\_function(void \*arg) {

printf("New thread running\n");

Int sum=0;

pthread\_t pt;

pt=pthread\_self();

printf("Thread %x ran\n",(int)pt);

for(count=1;count<=6;count++){

printf("线程的count=%d\n",count);

sum+=count;

}

printf("sum=%d\n",sum);

printf("线程睡眠 5s\n");

sleep(5);

pthread\_exit(arg);

pthread\_exit

}

实验结果

