**实验二、进程通信（一）  
 ——管道及共享内存**

学号：2017329621125 姓名：田野 班级：计算机科学与技术（全英文）17(1)

1. **实验目的**

(1)加深对管道概念的理解。

(2)掌握利用管道进行进程通信的程序设计。

（3） Linux系统的共享内存机制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支持的共享存储区机制。

1. **实验内容**

**任务一、 管道**

1. 运行源码

阅读以上父子进程利用管道进行通信的例子（例1），写出程序的运行结果并分析。

代码：

#include <unistd.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

int x,fd[2];

char buf[30],s[30];

pipe(fd);

while ((x=fork())==-1);

if (x==0)

{

close(fd[0]);

printf("Child Process!\n");

strcpy(buf, "This is an example\n");

write(fd[1],buf,30);

exit(0);

}

else{

close(fd[1]);

printf("Parent Process!\n");

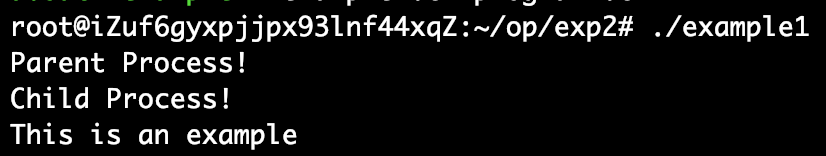
read(fd[0],s,30);

printf("%s\n",s);

}

}

结果：



分析：

首先父进程创建管道，然后父进程生成子进程，子进程与父进程使用同一管道进行通信。子进程关闭了读通道，父进程关闭写通道，子进程向管道写入数据，父进程成功读取到数据并进行输出。

(2)编写程序

父进程利用管道将一字符串交给子进程处理。子进程读字符串，将里面的字符反向后再交给父进程，父进程最后读取并打印反向的字符串。

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/wait.h>

char\* strrev(char \*str){

if(!str || ! \*str)

return str;

int i = strlen(str) -1, j=0;

char ch;

while (i>j)

{

ch = str[i];

str[i] = str[j];

str[j] =ch;

i--;

j++;

}

return str;

}

int main(){

int x,fd[2];

char buf[30],s[30];

pipe(fd);

while((x=fork())==-1);

if (x>0){

strcpy(buf, "pipe");

write(fd[1],buf,30);

wait(0);

read(fd[0],buf,30);

printf("%s\n",buf);

}

else{

read(fd[0],s,30);

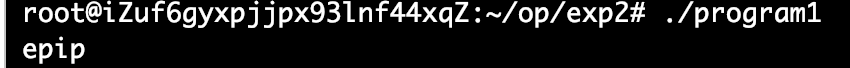
strrev(s);

write(fd[1],s,30);

}

}

结果：



（父进程中的初始字符为pipe）

**任务二、 共享内存**

（1）阅读例2的程序，运行一次该程序，然后用ipcs命令查看系统中共享存储区的情况，再次执行该程序，再用ipcs命令查看系统中共享内存的情况，对两次的结果进行比较，并分析原因。最后用ipcrm命令删除自己建立的共享存储区。 （有关ipcs和ipcrm介绍见后面一页）

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include<stdio.h>

#include<stdlib.h>

int main()

{

key\_t key=125; /\*在实际实验过程中，为了避免每个同学建立的共享存储区关键字一样而相互干扰，关键字请用学号末3位\*/

int shmid\_1,shmid\_2;

if ((shmid\_1=shmget(key,1000,0644|IPC\_CREAT))==-1){

perror("shmget shmid\_1");exit(1);

}

printf("First shared memory identifier is %d\n",shmid\_1);

if ((shmid\_2=shmget(IPC\_PRIVATE,20,0644))==-1){

perror("shmget shmid\_2");exit(2);

}

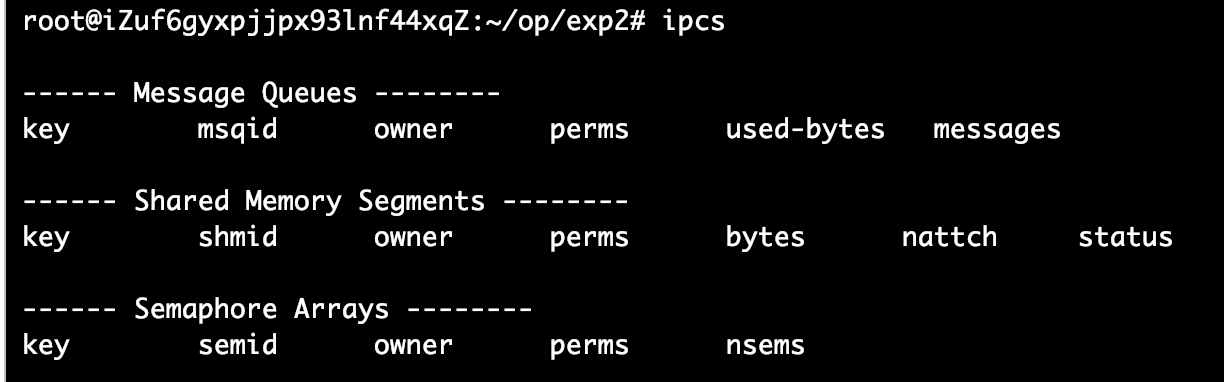
printf("Second shared memory identifier is %d\n",shmid\_2);

exit(0);

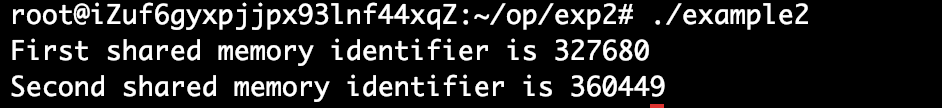
}

结果：

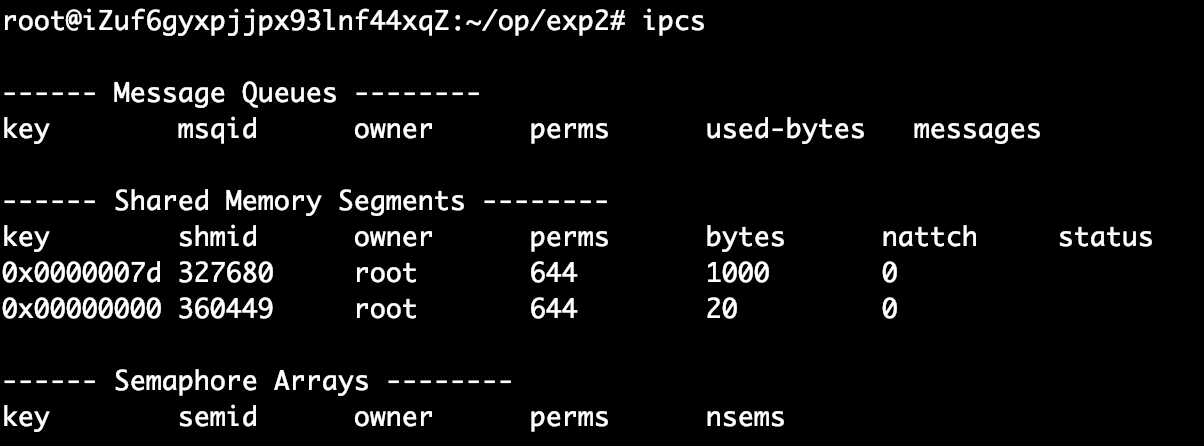
运行第一次程序前，输入ipcs:



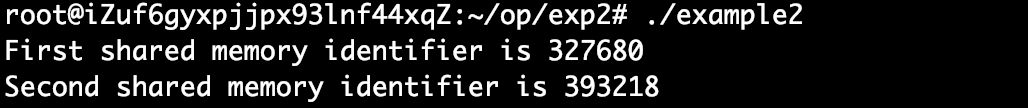
运行第一次程序输出：



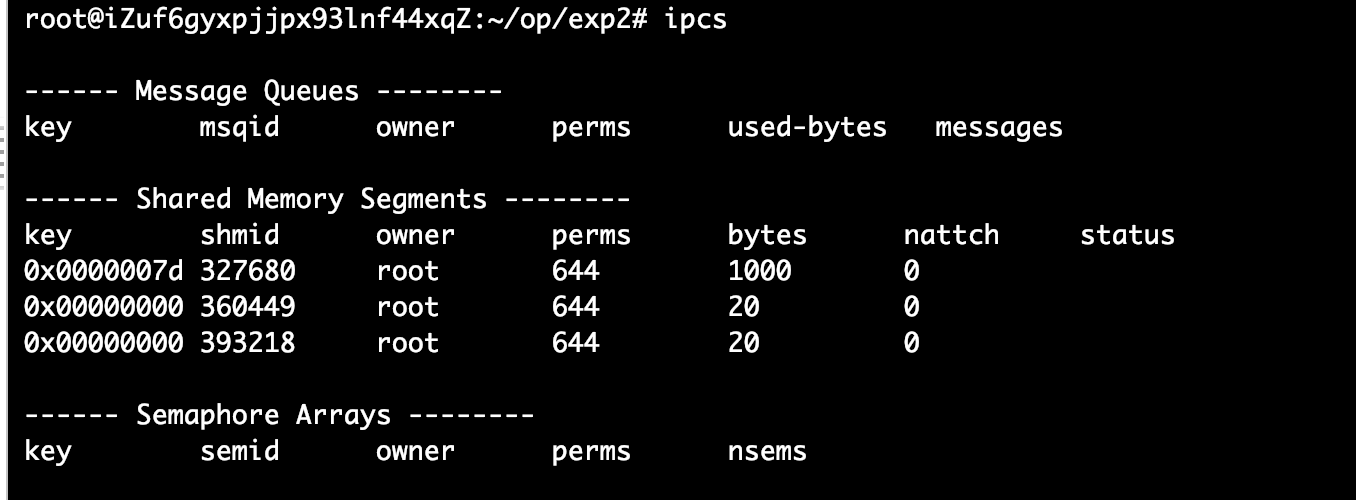
运行第一次程序后，输入ipcs:



运行第二次程序，输出：

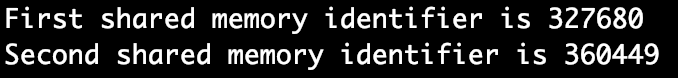


运行第二次程序后，输入ipcs:

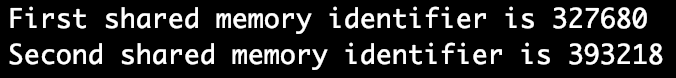


分析两次输出：

第一次输出为：



第二次输出为：



本程序使用shmget()创建了2个共享存储区，shmget()接受三个参数，第一个参数为key,用来创建IPC标识符，shmget返回的标识符与key值一一对应，不同的key值返回不同的标识符；第二个参数为size,决定了共享内存段的大小；第三个参数为shmflag,用于设置访问权限及标识创建条件。

在本程序中，第一处创建共享存储区使用的代码为：

shmid\_1=shmget(key,1000,0644|IPC\_CREAT)

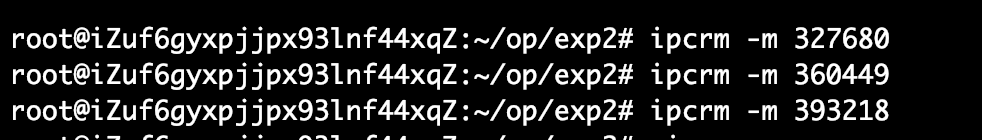
第二个参数为1000，因此该共享内存区的大小为1000bytes,第三个参数使用了0644|IPC\_CREAT，在使用该参数后，如果key值是新的，返回新创建的内存段标识符。若key值是旧的，返回已存在内核中的具有相同关键字值的内存段标识符。因为两次运行程序，key值都一样，所以返回的是相同关键字的内存段标识符。

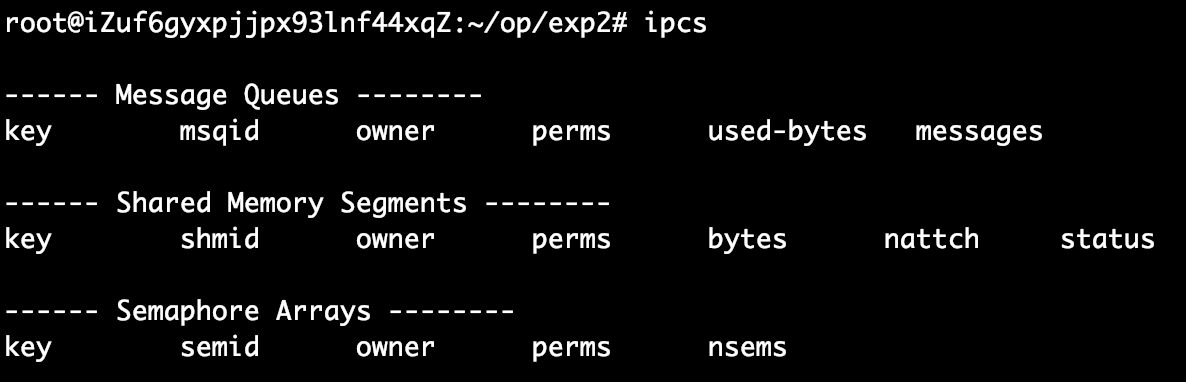
第二处创建共享存储区使用的代码为：

shmid\_2=shmget(IPC\_PRIVATE,20,0644))

第二个参数为20，因此该共享内存区的大小为20bytes，与第一处不同的是，这里每次的共享标识符不一样，会分配不一样的共享内存段。因此两次运行程序，返回的是不同的内存段标识符。

用ipcrm删除





（2）每个同学登陆两个窗口，先在一个窗口中运行例3程序1（或者只登陆一个窗口，先在该窗口中以后台方式运行程序1），然后在另一个窗口中运行例3程序2，观察程序的运行结果并分析。运行结束后可以用ctrl+c结束程序1的运行。

代码：

3.1

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <unistd.h>

#define SHMKEY 125 /\*在实际实验过程中，为了避免每个同学建立的共享存储区关键字一样而相互干扰，关键字请用学号末3位\*/

#define K 1024

int shmid;

void cleanup()

{

shmctl(shmid,IPC\_RMID,0);

exit(0);

}

int main ()

{

int i,\*pint;

char \*addr;

// extern char\* shmat ();

// extern cleanup ();

for(i=0;i<20;i++) signal(i, cleanup);

shmid=shmget(SHMKEY,16\*K,0777|IPC\_CREAT); /\*建立16K共享区SHMKEY \*/

addr=shmat(shmid,0,0);/\*挂接，并得到共享区首地址 \*/

printf ("addr 0x%s \n",addr);

pint=(int \*)addr;

for (i=0;i<256;i++) \*pint++=i;

pause();/\*等待接收进程读 \*/

}

3.2

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define SHMKEY 125 /\*在实际实验过程中，为了避免每个同学建立的共享存储区关键字一样而相互干扰，关键字请用学号末3位\*/

#define K 1024

int shmid;

int main ()

{

int i,\*pint;

char \*addr;

// extern char\* shmat();

shmid=shmget(SHMKEY,8\*K,0777);/\*取共享区SHMKEY的id \*/

addr=shmat(shmid,0,0);/\*连接共享区\*/

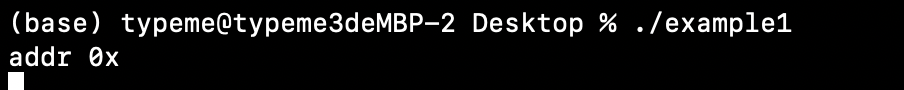
pint=(int \*)addr;

for (i=0;i<256;i++)

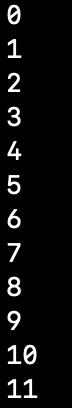
printf("%d\n",\*pint++);/\*打印共享区中的内容\*/

}

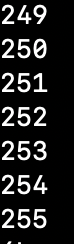
运行例3程序1，输出结果：



运行例3程序2，输出结果：



。。。。



（从1输出到255）

例3程序1，该程序创建了一款大小16K的共享内存空间，并且清空了该空间的内容，在保证空间清空后，从空间头开始，向其中写入由0到255共256个数字，然后进入等待状态，等待程序2读出。

例3程序2，该程序获得了程序1创建的内存空间，并从中读取出程序1写入的数据，并依次输出。

（3）编写程序：使用系统调用shmget()，shmat()，shmdt()，shmctl()，编制程序。要求在父进程中生成一个30字节长的私有共享内存段。接下来，设置一个指向共享内存段的字符指针，将一串大写字母写入到该指针指向的存贮区。调用fork()生成子进程，让子进程显示共享内存段中的内容。接着，将大写字母改成小写，子进程修改共享内存中的内容。之后，子进程将脱接共享内存段并退出。父进程在睡眠5秒后，在此显示共享内存段中的内容（此时已经是小写字母）。

代码：

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <unistd.h>

#include <string.h>

#include <ctype.h>

#define SHMKEY 125 /\*在实际实验过程中，为了避免每个同学建立的共享存储区关键字一样而相互干扰，关键字请用学号末3位\*/

#define B 1

int shmid;

void cleanup()

{

shmctl(shmid,IPC\_RMID,0);

exit(0);

}

void convertToUppercase(char \*sPtr) {

while(\*sPtr) {

if(isupper(\*sPtr))

\*sPtr = tolower(\*sPtr);

sPtr++;

}

}

int main ()

{

int i,x;

char\* pint;

char\* addr;

for(i=0;i<20;i++)

{

signal(i, cleanup);

}

shmid=shmget(SHMKEY,30 \* B,0777|IPC\_CREAT); /\*建立16K共享区SHMKEY \*/

while ((x=fork())==-1);

if (x != 0){

addr=shmat(shmid,0,0);/\*挂接，并得到共享区首地址 \*/

pint=addr;

char\* b = "ABC";

for (i=0; i<= strlen(b);i++) {

\*pint++ = \*b++;

i = 0;

}

sleep(5);

shmid=shmget(SHMKEY,8\*B,0777);/\*取共享区SHMKEY的id \*/

addr=shmat(shmid,0,0);/\*连接共享区\*/

pint=(char\*)addr;

puts(pint);

}

else{

shmid=shmget(SHMKEY,8\*B,0777);/\*取共享区SHMKEY的id \*/

addr=shmat(shmid,0,0);/\*连接共享区\*/

pint=(char\*)addr;

convertToUppercase(pint);

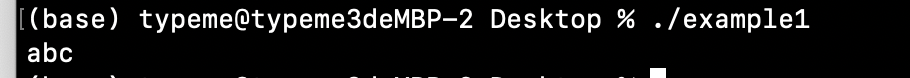
shmdt(pint);

exit(0);

}

}

输出：



（父进程中字符串为:ABC）