

**本科实验报告**

课 程 名 称： 操作系统原理实验 课 程 编 号： 08060157 学 生 姓 名： 阮炜霖 学 号： 2020101603 学 院： 信息科学技术学院

系： 计算机科学系

专 业： 网络工程

指 导 教 师： 郝振明

教 师 单 位： 计算机科学系 开 课 时 间：2021 ~ 2022 学年度第 二 学期

## 暨南大学教务处

2022年 6 月 15 日

操作系统原理实验 **课程实验项目目录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 实验项目 编号 | 实验项目名称 | 实验性质 | 成绩 | 指导教师 |
| 1 | 0806015701 | 实验环境的使用 | 验证 |  | 郝振明 |
| 2 | 0806015702 | 进程的控制 | 设计 |  | 郝振明 |
| 3 | 0806015703 | 进程间的通信 | 设计 |  | 郝振明 |
| 4 | 0806015704 | 生产者消费者问题 | 综合 |  | 郝振明 |
| 5 | 0806015705 | 简单文件系统实现 | 综合 |  | 郝振明 |
| 6 | 0806015706 | 非必须实验。若选择做此项目，则项目名称自定。 |  |  | 郝振明 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

\*此表由学生按顺序填写。

# 暨南大学本科实验报告专用纸

课程名称 操作系统原理实验 成绩评定

实验项目名称 进程间的通信\_ 指导教师 郝振明

实验项目编号 0806015703 实验项目类型 设计型

学生姓名 阮炜霖 学号 2020101603

学院信息科学技术学院 系 计算机科学系 专业 网络工程

实验时间 2022 年 3 月 17 日 上 午 3、4节 实验地点 N116

**一、 实验目的**

1. 实验目的：学习如何利用管道机制、消息缓冲队列、 共享存储区机制进行进程间的通信，并加深对上述通信机制的理解。

2. 实验要求：任选 C 高级程序语言编写源程序，在 Linux 操作系统下调试通过，测试正确。

**二、 实验内容**

1.实验内容

（1）了解系统调用pipe()、msgget()、msgsnd()、msgrcv()、msgctl()、shmget()、shmat()、shmdt()、shmctl()的功能和实现过程。

（2）编写一段程序，使其用管道来实现父子进程之间的进程通信。子进程向父进程发送自己的进程标识符，以及字符串“is sending a message to parent!”。父进程则通过管道读出子进程发来的消息，将消息显示在屏幕上，然后终止。

（3）编写一段程序，使其用消息缓冲队列来实现client进程和server进程之间的通信。Server进程先建立一个关键字为SVKEY(如75)的消息队列，然后等待接收类型为REQ(如1)的消息；在收到请求消息后，它便显示字符串“serving for client”和接收到的client进程的进程标识数，表示正在为client进程服务；然后再向client进程发送一应答消息，该消息的类型是client进程的进程标识数，而正文则是server进程自己的标识数。Client进程向消息队列发送类型为REQ的消息（消息的正文为自己的进程标识数）以取得server进程的服务，并等待server进程发来的应答；然后显示字符串“receive reply from”和接收到的server进程的标识数。

（4）编写一段程序，使其用共享存储区来实现两个进程之间的进程通信。进程A创建一个长度为512字节的共享内容，并显示写入该共享内存的数据；进程B将共享内存附加到自己的地址空间，并向共享内存中写入数据。

2.实验原理

使用的系统调用命令说明或算法及程序详细设计：

（1） 用 pipe 创建管道，再调用 fork 产生两个子进程，子进程分别向管道写入信息，父进程读取管道的信息并输出到标准输出

（2） 通过 ftok 在 /tmp/msg.txt 上创建一个 key，然后通过这个key 创建消息队列，子进程通过 ftok 返回的 key 进行读写。

（3） 用 shmget 创建共享内存，然后通过返回的指针进行读写。

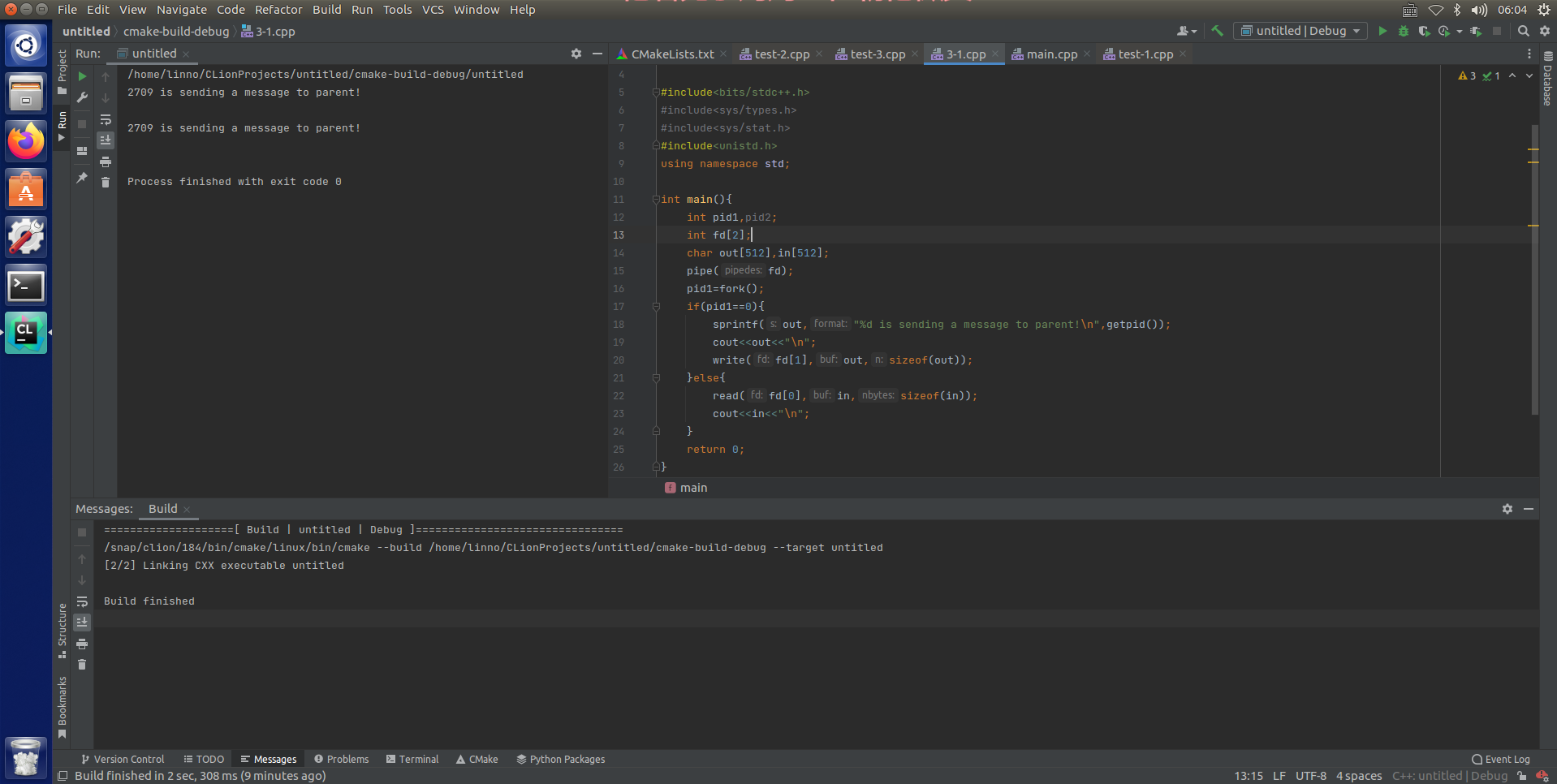
## 三、 实验步骤

（一）使用准备

在虚拟机上用Clion新建一个C/C++程序项目，接着开始编写程序实现进程间的通信。

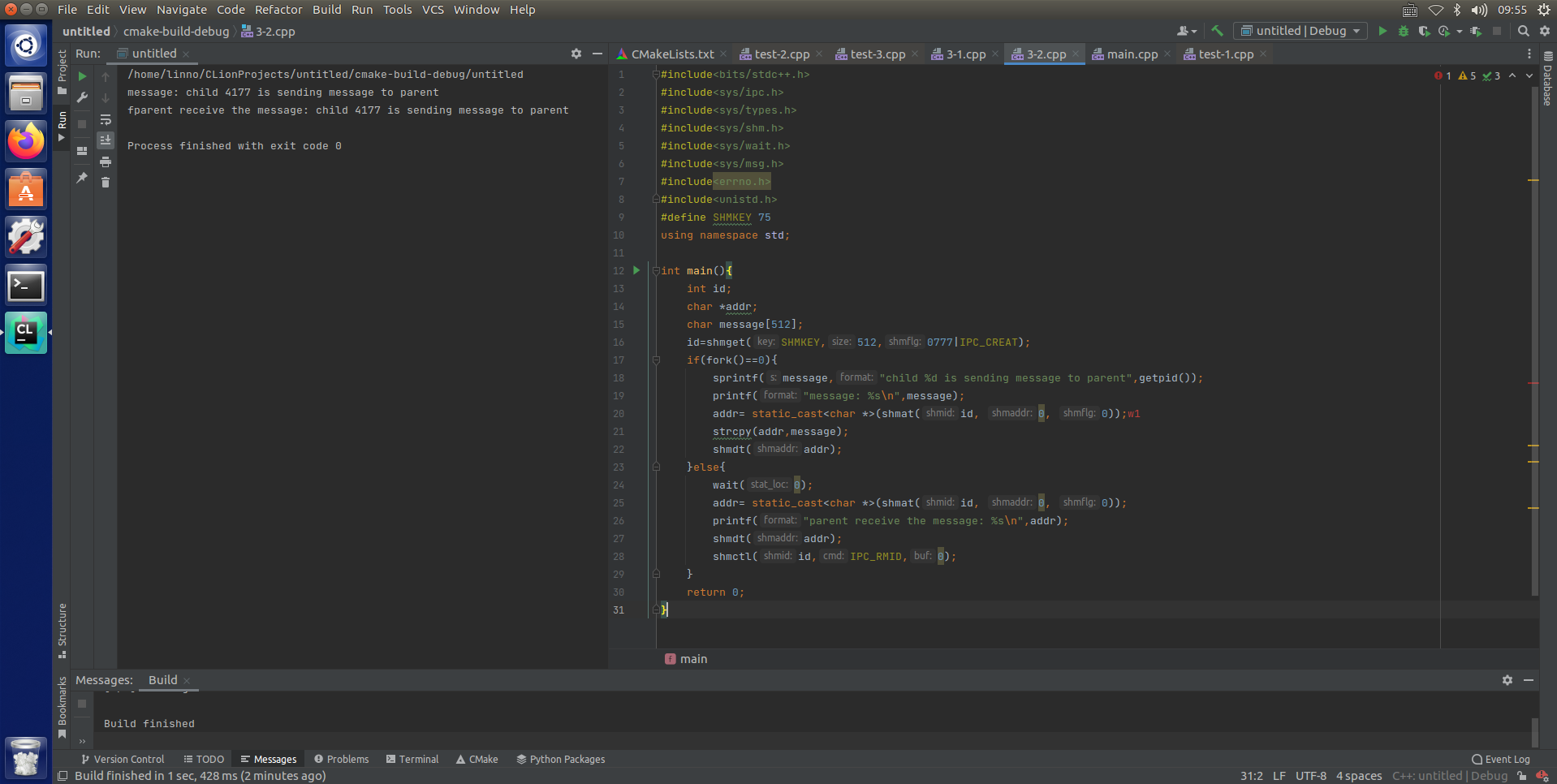
（二）编写程序

1.在CLion中， 编写以下代码。(程序3-1)



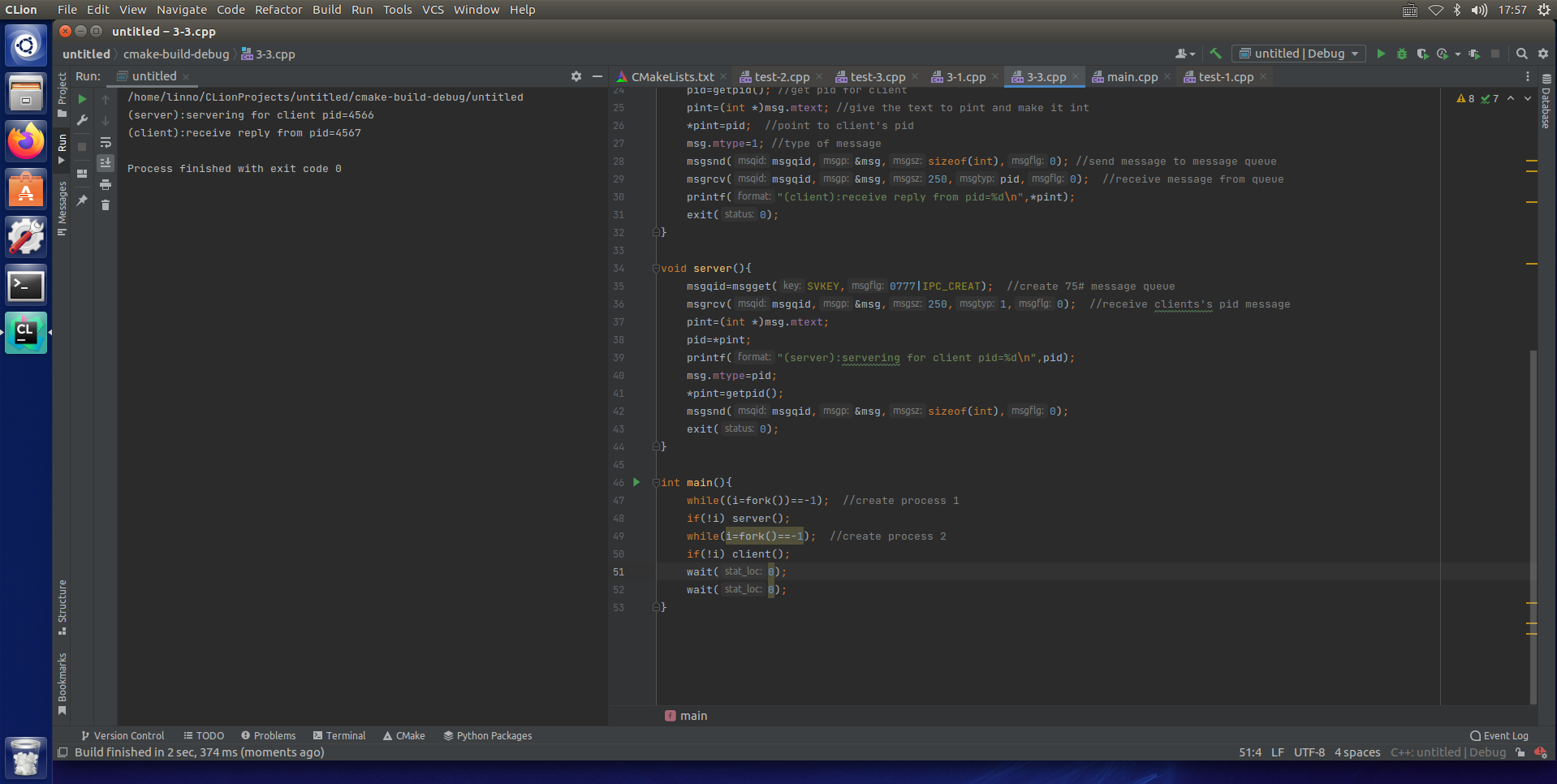
观察左边的结果，可以发现子进程发送的信息被父进程收到，说明上面程序使用管道实现了父子进程之间的通信。

2.使用共享存储区来实现父子进程之间的进程通信。（程序3-2）



观察左边输出结果，发现在子进程中的信息被成功传回父进程中输出。实验结果表明已成功使用共享存储区实现父子进程之间的进程通信。

3.使用消息缓冲队列来实现父子进程之间的进程通信。（程序3-2）



观察左边输出结果，发现server成功发送了一条”servering for client”+pid的消息，并且在client回应了一条”receive reply from”+pid的消息。实验结果表明已成功使用消息队列实现server和client之间的通信。

## 四、 思考与练习

(1)上述哪些通信机制提供了发送进程和接收进程之间的同步功能？这些同步是如何进行的？

消息缓冲机制提供了同步功能。当写进程把一定数量的数据写入管道中，然后睡眠等待，直到读进程读到空的管道时，也开始睡眠等待，直到写进程将数据写入管道中，才将其唤醒，从而实现进程的同步。

(2)上述通信机制各有什么特点，它们分别适合于何种场合？

**管道通信（PIPE）**

两个进程利用管道进行通信时，发送消息的进程称为写进程，接受信息的进程称为读进程。管道通信方式的中间介质是管道文件，它将一个写进程和一个都进程连接在一起，实现两个进程之间的通信。写进程通过写入端（发送端）往管道文件中写入信息；读进程通过读出端（接收端）从管道文件中读取信息。两个进程协调不断地进行写和读，便会构成双方通过管道传递信息的流水线。

适用场合：数据单向流动，只能在具有亲缘关系的进程间使用。

**共享存储区机制**

针对消息缓冲需要占用CPU进行消息复制的缺点。OS提供了一个进程间直接进行数据交换的通信方式——共享内存。这种通信方式允许多个进程在外部通信协议或同步，互斥机制的支持下使用同一个内存段进行通信。它是一种最有效的数据通信方式，其特点是没有中间环节，直接将共享的内存页面通过附接，映射到相互通信的进程各自的虚拟地址空间中，从而使多个进程可以直接访问到同一个物理内存页面。因此这种进程间通信方式是在同一个计算机系统中的诸进程间实现通信的最快捷的方法。而它的局限性也在于此，即共享内存的诸进程必须处于同一个计算机系统，有物理内存可以共享才行。

适用场合：内存需要能被其他进程所访问且高效率通讯。

## 五、附录

程序3-1

*// Program 3.1*

*// Created by linno on 2022/4/12.*

*//*

#include<bits/stdc++.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

using namespace std;

int main(){

    char out[512],in[512];

    int pid1,pid2;

    int fd[2];

    pipe(fd);

    pid1=fork();

    if(pid1==0){

        sprintf(out,"%d is sending a message to parent!\n",getpid());

        cout<<out<<"\n";

        write(fd[1],out,sizeof(out));

    }else{

        read(fd[0],in,sizeof(in));

        cout<<in<<"\n";

    }

    return 0;

}

程序3-2

1. *// Program 3.2*
2. *// Created by linno on 2022/4/12.*
3. *//*
4. #include<bits/stdc++.h>
5. #include<sys/ipc.h>
6. #include<sys/types.h>
7. #include<sys/shm.h>
8. #include<sys/wait.h>
9. #include<sys/msg.h>
10. #include<errno.h>
11. #include<unistd.h>
12. #define SHMKEY 75
13. using namespace std;
14. int main(){
15. int id;
16. char \*addr;
17. char message[512];
18. id=shmget(SHMKEY,512,0777|IPC\_CREAT);
19. if(fork()==0){
20. sprintf(message,"child %d is sending message to parent",getpid());
21. printf("message: %s\n",message);
22. addr= static\_cast<char \*>(shmat(id, 0, 0));w1
23. strcpy(addr,message);
24. shmdt(addr);
25. }else{
26. wait(0);
27. addr= static\_cast<char \*>(shmat(id, 0, 0));
28. printf("parent receive the message: %s\n",addr);
29. shmdt(addr);
30. shmctl(id,IPC\_RMID,0);
31. }
32. return 0;
33. }

程序3-3

1. *// Program 3.3*
2. *// Created by linno on 2022/4/13.*
3. *//*
4. #include<bits/stdc++.h>
5. #include<sys/ipc.h>
6. #include<sys/types.h>
7. #include<sys/shm.h>
8. #include<sys/wait.h>
9. #include<sys/msg.h>
10. #include<errno.h>
11. #include<unistd.h>
12. #define SVKEY 75 *//KEY*
13. using namespace std;
14. struct MSG{  *//node of message*
15. long mtype;
16. char mtext[250];  *//length of text*
17. }msg;
18. int msgqid,pid,\*pint,i;
19. void client(){
20. msgqid=msgget(SVKEY,0777);  *//open 75#message queue*
21. pid=getpid(); *//get pid for client*
22. *//printf("client's pid:%d\n",pid);*
23. pint=(int \*)msg.mtext; *//give the text to pint and make it int*
24. \*pint=pid;  *//point to client's pid*
25. msg.mtype=1; *//type of message*
26. msgsnd(msgqid,&msg,sizeof(int),0); *//send message to message queue*
27. msgrcv(msgqid,&msg,250,pid,0);  *//receive message from queue*
28. printf("(client):receive reply from pid=%d\n",\*pint);
29. exit(0);
30. }
31. void server(){
32. msgqid=msgget(SVKEY,0777|IPC\_CREAT);  *//create 75# message queue*
33. msgrcv(msgqid,&msg,250,1,0);  *//receive clients's pid message*
34. pint=(int \*)msg.mtext;
35. pid=\*pint;
36. printf("(server):servering for client pid=%d\n",pid);
37. msg.mtype=pid;
38. \*pint=getpid();
39. *//printf("server's pid:%d\n",getpid());*
40. msgsnd(msgqid,&msg,sizeof(int),0);
41. exit(0);
42. }
43. int main(){
44. while((i=fork())==-1);  *//create process 1*
45. if(!i) server();
46. while((i=fork())==-1);  *//create process 2*
47. if(!i) client();
48. wait(0);
49. wait(0);
50. }