实验5 进程通信（1）无名管道和共享内存通信

实验目的

1.了解无名管道通信的基本原理

2.掌握父子进程使用管道通信的方法

3.掌握共享内存通信方式的原理

实验内容

1. 父子进程基于无名管道的简单通信。

/\*pipe1.c\*/

#include <sys/types.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <stdio.h>

#include <string.h>

main()

{

int fd[2],pid,n;

char outpipe[50],inpipe[50];

pipe(fd); /\*创建无名管道\*/

pid=fork();

if(pid==0)

{

sprintf(outpipe,"I am child process");

write(fd[1],outpipe,strlen(outpipe)); /\*向无名管道写入信息\*/

printf("child process writes%d bytes: %s\n",strlen(outpipe),outpipe);

}

else

{

wait(0);

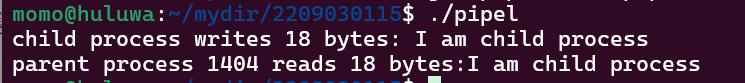
n=read(fd[0],inpipe,25); /\*从无名管道读出信息\*/

printf("parent process %d reads %d bytes:%s\n",getpid(),n,inpipe);

}

}

要求：写出程序的运行结果截图及分析



使用pipe函数创建一个无名管道，含有两个文件描述符fd[0]（读端）和fd[1]（写端），通过fork函数创建一个子进程。

**子进程操作**：

子进程将字符串"I am child process"写入outpipe数组。

子进程使用write函数将outpipe数组中的字符串写入管道的写端fd[1]。

子进程打印写入管道的字节数和内容。

**父进程操作**：

父进程使用wait函数等待子进程结束。

父进程从管道的读端fd[0]读取最多25个字节到inpipe数组。

父进程打印读取的字节数和内容。

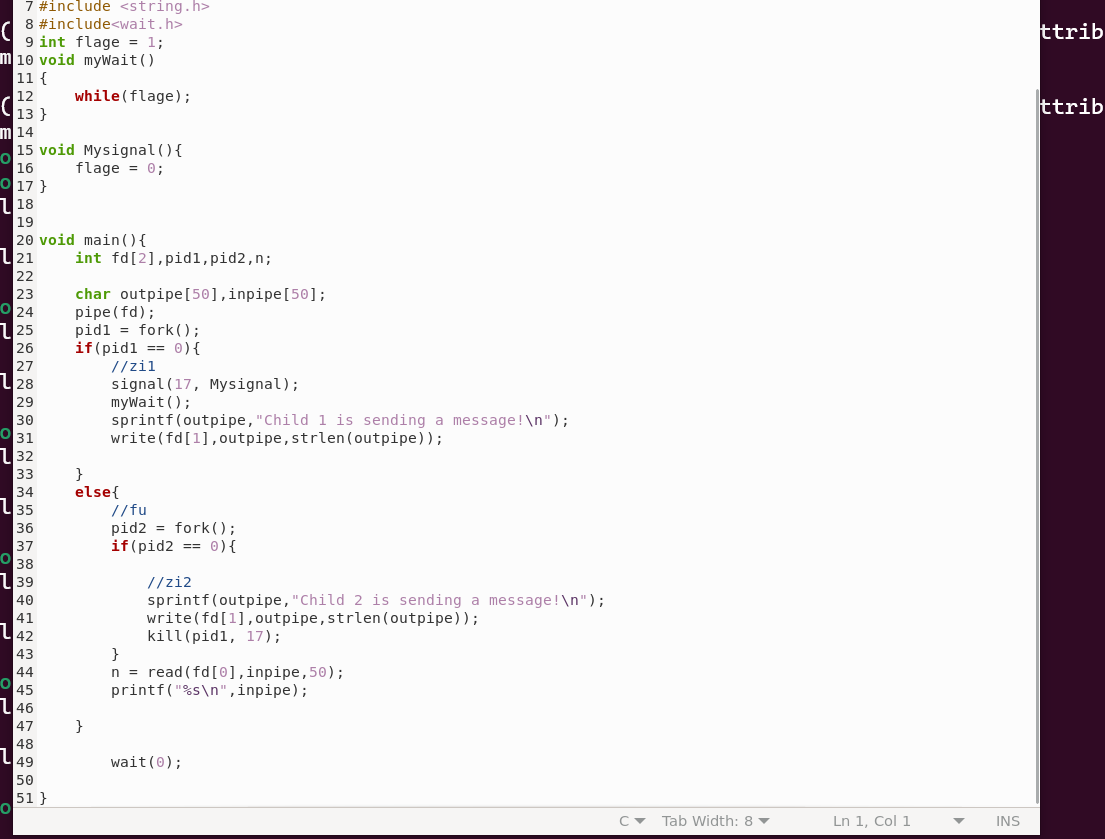
2. 编写程序实现多个进程基于无名管道进行通信。用系统调用pipe()建立一无名管道，两个子进程P1和P2分别向管道写入一句话：

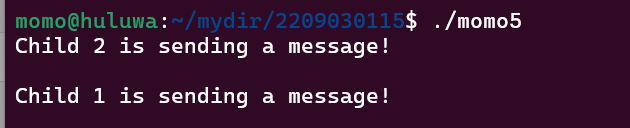
Child 1 is sending a message!

Child 2 is sending a message!

父进程从无名管道中读出两个来自子进程的信息并显示，子进程发送先后顺序没有要求。

要求：（1）写出主要程序代码及运行结果截图





（2）总结pipe管道通信的主要原理和特点

管道通过pipe系统调用创建，它返回两个文件描述符，通常称为读端（fd[0]）和写端（fd[1]）。

管道是半双工的，数据只能在一个方向上流动。数据从写端流入管道，从读端流出。

管道有一个内部缓冲区，用于暂存写入的数据，直到它们被读取。这个缓冲区的大小通常是有限的。

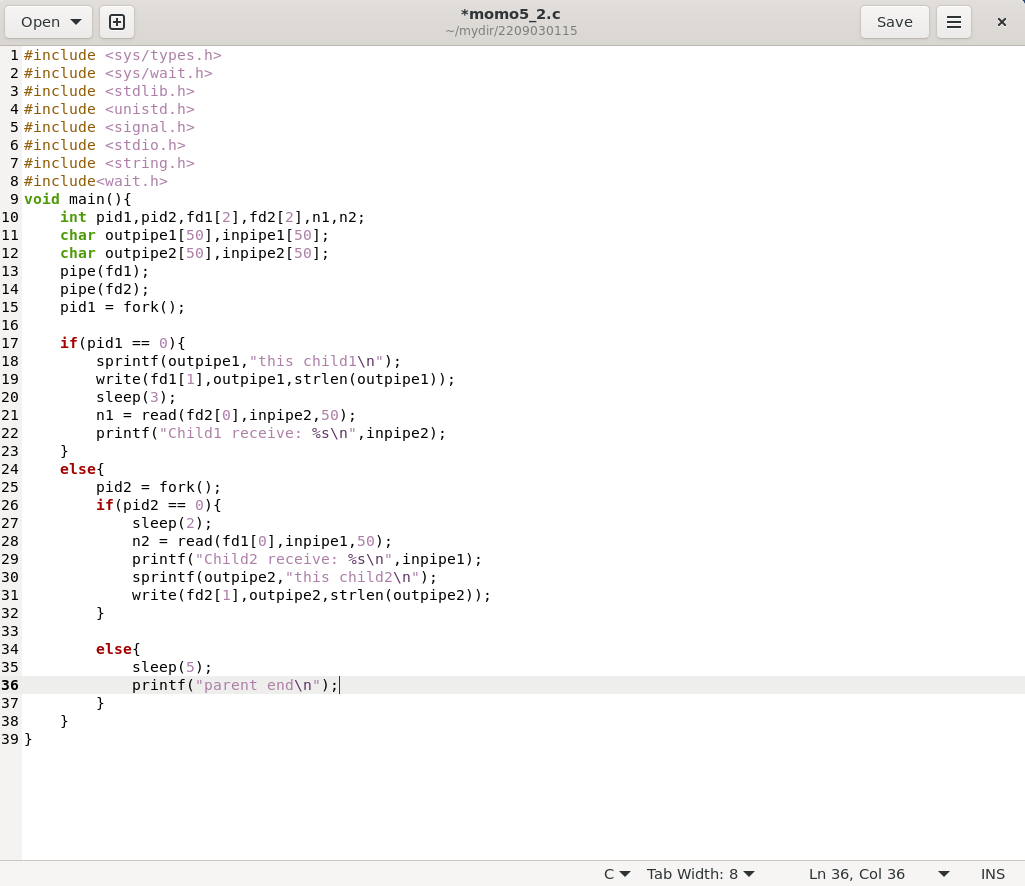
无名管道，只能在具有亲缘关系的进程间使用，且没有名字，因此只能在创建它的进程及其子进程中访问。

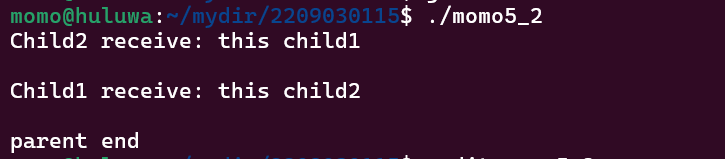
管道的写操作通常是原子的，这意味着系统调用要么写入全部数据，要么不写入。

（3）如果从管道中读出乱码如何解决？（memset函数）

在读取数据之前，使用memset函数将缓冲区初始化为零或特定的字符。

3.（选做）编程实现父进程建立的两个子进程p1，p2利用pipe实现双向通信（p1发一条消息收一条消息，p2收一条发一条消息，或者相反；可用sleep控制同步），父进程等待子进程结束后输出通信结束信息终止。





4. 运行以下程序，分析程序的运行结果。

接收进程receive.c:

#include<stdio.h>

#include<stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

int main()

{

int shmid; char \*viraddr;

shmid=shmget(1234,BUFSIZ,0666|IPC\_CREAT); viraddr=shmat(shmid, 0,0);

printf("Your message is :\n%s",viraddr);   
 shmdt(viraddr);

shmctl(shmid,IPC\_RMID,0);

exit(0);

return 0;

}

发送进程send.c：

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

int main()

{ int shmid; /\*定义共享内存内部标识shmid \*/

char \*viraddr; /\*定义附接共享内存的虚拟地址\*/

char buffer[BUFSIZ]; shmid=shmget(1234,BUFSIZ,0666|IPC\_CREAT);

viraddr=(char\*)shmat(shmid, 0,0);

while(1)

{ puts("Enter some text:");

fgets(buffer,BUFSIZ,stdin);

strcat(viraddr,buffer);

if(strncmp(buffer,"end",3)==0) break; }

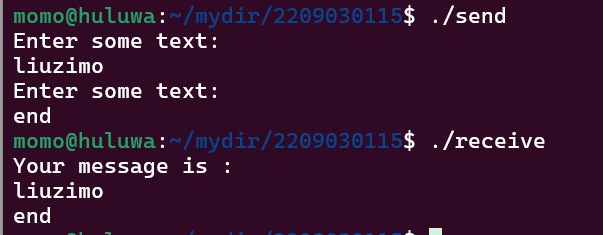
shmdt(viraddr); exit(0);

return 0;

}

要求：

1. 运行截图。



1. 根据程序分析共享内存的实现原理

在send.c和receive.c中，共享内存段通过shmget函数创建。然后通过shmat函数，进程可以将共享内存段映射到自己的地址空间。在send.c中，使用fgets从标准输入读取字符串，并使用strcat将其追加到共享内存中。在receive.c中，程序直接打印共享内存中的内容。当进程完成对共享内存的操作后，使用了shmdt函数将其从自己的地址空间分离。

共享内存通常与信号量（semaphores）结合使用，以提供同步机制，确保数据的一致性和完整性。