|  |
| --- |
| 北 京 邮 电 大 学  实 验 报 告  课程名称： Linux 环境及开发工具应用实践  院系：计算机学院（国家示范性软件工程学院）  班级： 2021211318  姓名： 万志恒  学号： 2021212413  教师： 赵方 金昕  成绩：  2023年 春季学期 |
| 实验目的 （1）消息缓冲队列、共享存储区机制进行进程间的通信；  （2）理解通信机制。 实验环境 硬件：PC微型计算机、8核、16.0G内存、467G硬盘  软件：Vmware、Ubuntu操作系统、Vi、GCC、GDB   1. **实验任务及内容**   1.使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 （例如 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 ID。client 进程则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 ID） 以取得 sever 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 ID。  源程序：  #include <unistd.h>  #include <stdio.h>  #include <stdlib.h>  #include <sys/wait.h>  #include <sys/types.h>  #include <sys/msg.h>  #include <sys/ipc.h>  #define SVKEY 75  struct msgform {//消息结构  long mtype;  char mtext[250];  }msg;  int msgqid,pid,\*pint,i;  void client() {  msgqid=msgget(SVKEY,0777);//打开75#消息队列  pid=getpid();  pint=(int \*)msg.mtext;//把正文的内容传给 pint，并强制转换类型  \*pint=pid;//pint指针指向client进程标识符  msg.mtype=1;//消息类型为1  msgsnd(msgqid,&msg,sizeof(int),0);//发送消息msg入msgqid消息队列  msgrcv(msgqid,&msg,250,pid,0);//从队列msgqid接收消息msg  printf("(client):receive reply from pid=%d\n",\*pint);//显示 server进程标识数  exit(0);  }  void server( )  {  msgqid=msgget(SVKEY,0777|IPC\_CREAT);//创建75#消息队列  msgrcv(msgqid,&msg,250,1,0);//接收client进程标识数消息  pint=(int \*)msg.mtext;//把正文的内容传给 pint，并强制转换类型  pid=\*pint;//获得 cilent 进程标识数  printf("(server):serving for client pid=%d\n",pid);  msg.mtype=pid;//消息类型为 client 进程标识数  \*pint=getpid();//获取 server 进程标识数  msgsnd(msgqid,&msg,sizeof(int),0);//发送消息  exit(0);  }  int main()  {  while((i=fork())==-1);//创建进程 1  if(!i)server();  while((i=fork())==-1);//创建进程 2  if(!i) client();  wait(0);  wait(0);  return 0;  }  显示结果：    2. 使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 字节的共享内存，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。  源程序：  #include<stdio.h>  #include<stdlib.h>  #include<string.h>  #include<unistd.h>  #include<sys/wait.h>  #include<sys/shm.h>  #define SHMKEY 318  int main(){  int id;//共享内存标识符  char \*addr;//指向共享内存  char message[512];  id=shmget(SHMKEY,512,0666|IPC\_CREAT);//创建共享内存  if(fork()==0){//子进程  sprintf(message,"%d wrote to parent",getpid());  printf("%s\n",message);  addr=shmat(id,0,0);//将共享内存连接到当前进程的地址空间  strcpy(addr,message);//将内容放入共享内存  shmdt(addr);//将共享内存从当前进程中分离  }else{  wait(NULL);//等待子进程执行完成  addr=shmat(id,0,0);//将共享内存连接到当前进程的地址空间  printf("%s\n",addr);//将共享内存中的内容输出  printf("parent copy.\n");  shmdt(addr);//将共享内存从当前进程中分离  shmctl(id,IPC\_RMID,0);//删除共享内存段  }  return 0;  }  显示结果：     实验心得及体会 这两个实验加深了我对进程间通讯的了解，使我更加深入地认识了消息缓冲队列和共享存储区这两种进程通讯的实现方式，了解了他们各自的特点，有助于理解操作系统中进程通讯的概念。 |