

**实 验 报 告**

**（ 2024/2025 学年 第 一 学期 ）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 计算机操作系统 | | | | | |
| 实验名称 | 进程管理 | | | | | |
| 实验时间 | 2024 | 年 | 10 | 月 | 23 | 日 |
| 指导单位 | 物联网学院 | | | | | |
| 指导教师 |  | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 |  | 班级学号 |  |
| 学院(系) | 物联网学院 | 专 业 | 网络工程 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **实验名称** | 进程管理 | | | | | | | | | **指导教师** | |  |
| **实验类型** | | 设计性实验 | | **实验学时** | | 4 | | | **实验时间** | | | 2024.10.23 |
| 1. **实验目的和要求** | | | | | | | | | | | | |
| **实验目的：**了解有关 Linux 系统调用； 加深对进程概念的理解，明确进程和程序的区别；进一步认识并发执行的实质；学习进程控制和进程通信机制。  **实验要求：**  1.在 Linux 环境下，采用 C 语言编程，使用系统调用 fork 创建进程，观察程序运行的结果；  2.Linux系统的进程通信机构 (IPC) 允许在任意进程间大批量地交换数据，要求熟悉Linux支持的消息通信机制，多次反复运行改进后的程序，观察并记录运行结果，比较设计内容第4项和第5项两种通信机制的数据传输的时间，并分析原因。 | | | | | | | | | | | | |
| **二、实验环境(实验设备)** | | | | | | | | | | | | |
| Ubuntu18.04 gcc7.5.0 make4.1 vim | | | | | | | | | | | | |
| 1. **实验原理及内容**   1.基本操作  实验过程：使用vim编辑文本，gcc编译源程序，写源代码和makefile脚本，在电脑上编下给出代码，查看执行结果。  实验内容：  源代码：  #include <stdio.h>  #include <unistd.h>  void main()  {      int pid;      pid = fork();      if (pid == 0)      {          printf("i am a son process!\n");          printf("my process id is %d\n", pid);      }      else      {          printf("i am the father process!\n");          printf("my process id is %d\n", pid);      }  }  实验结果：    2.管道  实验过程：使用系统调用pipe()建立一条管道，创建2个子进程P1和P2分别向管道各写一句话，父进程从管道中读出来自于两个子进程的信息，并显示在屏幕上  实验内容：  源代码：  #include <unistd.h>  #include <signal.h>  #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/types.h>  #include <sys/wait.h>  int pid1, pid2;  int main()  {      int fd[2];      char outpipe[100], inpipe[100];      pipe(fd); /\*创建一个管道\*/      while ((pid1 = fork()) == -1);      if (pid1 == 0)      {          lockf(fd[1], 1, 0);          sprintf(outpipe, "child 1 process is sending message!");          /\*把串放入数组outpipe中\*/          write(fd[1], outpipe, 50); /\*向管道写长为50字节的串\*/          sleep(5);                  /\*自我阻塞5秒\*/          lockf(fd[1], 0, 0);          exit(0);      }      else      {          while ((pid2 = fork()) == -1);          if (pid2 == 0)          {              lockf(fd[1], 1, 0); /\*互斥\*/              sprintf(outpipe, "child 2 process is sending message!");              write(fd[1], outpipe, 50);              sleep(5);              lockf(fd[1], 0, 0);              exit(0);          }          else          {              wait(0);                 /\*同步\*/              read(fd[0], inpipe, 50); /\*从管道中读长为50字节的串\*/              printf("%s\n", inpipe);              wait(0);              read(fd[0], inpipe, 50);              printf("%s\n", inpipe);              exit(0);          }      }      return 0;  }  实验结果：    3.消息缓冲队列：  实验过程：编写一段程序，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（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 form”和接收到的server进程的标识符。  实验内容：  源代码：  #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <sys/types.h>  #include <sys/ipc.h>  #include <sys/msg.h>  int msgqid, pid, \*pint, i;  struct msgform  {      long mtype;      char mtext[50];  } msg;  void client()  {      msgqid = msgget(66, 8888);      pid = getpid();      pint = (int \*)msg.mtext;      \*pint = pid;      msg.mtype = 1;      msgsnd(msgqid, &msg, sizeof(int), 0);      msgrcv(msgqid, &msg, 50, pid, 0);      printf("(client):receive reply from pid=%d\n", \*pint);      exit(0);  }  void server()  {      msgqid = msgget(66, 8888 | IPC\_CREAT);      msgrcv(msgqid, &msg, 50, 1, 0);      pint = (int \*)msg.mtext;      pid = \*pint;      printf("(server):serving for client pid=%d\n", pid);      msg.mtype = pid;      \*pint = getpid();      msgsnd(msgqid, &msg, sizeof(int), 0);      exit(0);  }  int main()  {      while ((i = fork()) == -1);      if (!i)          server();      while ((i = fork()) == -1);      if (!i)          client();      wait(0);      wait(0);      return (0);  }  实验结果：    4.共享存储区：  实验过程：用fork( )创建子进程，利用共享存储区实现SERVER和 CLIENT的通信。SERVER建立一个KEY为75的共享区：若共享区的第一个字节为-1 ，则表示空闲。若该字节值发生变化，则表示收到了消息，读取后再次把值设为-1。SERVER每接收到一次数据后显示“(server)received”。若遇到的值为0，则视为结束信号，取消该队列，并退出。CLIENT端建立一个为75的共享区：若共享区的第一个字节为-1， 则表示该单元空闲，可发送请求，CLIENT 随即填入9到0，期间等待Server端再次空闲。CLIENT每发送一次数据后显示“ (client)sent”。进行完这些操作后， CLIENT退出。  实验内容：  源代码：  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <sys/msg.h>  #include <sys/shm.h>  #include <sys/ipc.h>  #include <sys/sem.h>  #include <unistd.h>  #define SHMKEY 75 /\*定义共享区关键词\*/  void CLIENT()  {      int shmid, i;      int \*addr;      shmid = shmget(SHMKEY, 1024, 0777 | IPC\_CREAT);      /\*获取共享区，长度1024，关键词SHMKEY\*/      addr = shmat(shmid, 0, 0); /\*共享区起始地址为addr\*/      for (i = 9; i >= 0; i--)      {          while (\*addr != -1);          printf("(client)sent\n");          \*addr = i; /\*把i赋给addr\*/      }      exit(0);  }  void SERVER()  {      int shmid;      int \*addr;      shmid = shmget(SHMKEY, 1024, 0777 | IPC\_CREAT);      /\*创建共享区\*/      addr = shmat(shmid, 0, 0); /\*共享区起始地址为addr\*/      \*addr = -1;      do      {          while (\*addr == -1);          printf("(server)received\n%d", \*addr);          if (\*addr != 0)              \*addr = -1;      } while (\*addr != 0);      wait(0);      shmctl(shmid, IPC\_RMID, 0);  }  void main()  {      if (fork())      {          SERVER();      }      else      {          CLIENT();      }  }  实验结果： | | | | | | | | | | | | |
| **四、实验小结（包括问题和解决方法、心得体会、意见与建议等）** | | | | | | | | | | | | |
| 在本次实验中，我深入了解了Linux的进程管理及进程间通信机制，特别是管道、消息队列和共享存储区的应用。过程中遇到的问题主要集中在进程间同步与互斥上，例如在使用管道时，数据竞争导致读取内容不完整。通过引入锁机制解决了这一问题。  通过这些实验，我深刻感受到操作系统在管理资源、调度任务和进程间通信方面的复杂性与精妙之处。每种通信机制都有其独特的优缺点，适用于不同的应用场景。这种理解不仅让我在编程上受益，也让我在思考系统设计时更加全面。此次实验让我更加热爱操作系统的学习，并激发了我对深入研究计算机科学的热情。 | | | | | | | | | | | | |
| **五、指导教师评语** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **成绩** | | |  | | **批阅人** | |  | **日期** | | |  | |