**《操作系统》实验报告**

徽标, 公司名称

描述已自动生成

**题目: 实验三 进程通信**

**学 院 计算机学院**

**班 级 2021211304**

**学 号 2021212171、2021212484**

**姓 名 杨晨、张梓良**

**2023年 11月**

##### 1 实验内容

查阅资料，学习掌握Linux系统提供的用于四种进程间通信的系统调用、库函数的使用方法和参数，参照样例程序，设计完成以下两组实验。

**基于消息队列的进程通信**

Linux 消息队列包括POSIX消息队列、System V消息队列。发送者进程向消息队列中写入数据，接收者进程从队列中接收数据，实现相互通信。消息队列克服了信号signal承载信息量少，管道pipe只能承载无格式字节流以及缓冲区大小受限等缺点。

要求：编程实现发送者和接收者两个并发进程：

（1）发送者和接收者使用msgget(key\_t, key, int msgflg)、msgctl(int msgid, int command, struct msgid\_ds \*buf)创建、管理消息队列。只有接收者在接收完最后一个消息之后，才删除消息。

（2）发送者使用msgsend(int msgid, const void \*msg\_ptr, size\_t msg\_sz, int msgflg)向消息队列不断写入数据，并打印提示信息；

（3）接收者使用msgrcv(int msgid, void \*msg\_ptr, size\_t msg\_st, long int msgtype, int msgflg)从消息队列中接收消息，并打印提示信息。

**共享内存通信**

Linux 内核支持多种共享内存方式，如mmap()系统调用，POSIX共享内存，以及System V共享内存。本实验采用System V共享内存实现方法。

要求：编程实现写者进程Writer和读者进程Reader，

（1）写者进程和读者进程使用shmget(key\_t key, int size, int shmflg)创建在内存中创建用于两者间通信的共享内存，使用shmat(int shmid, char\*shmaddr, int flag)将共享内存映射到进程地址空间中，以便访问共享内存内容；

（2）写者进程向共享内存写入多组数据，读者进程从共享内存d读出数据；

（3）进程间通信结束后，写者进程和读者进程使用shmdt(char\*shmaddr)解除共享内存映射，使用shmctl(int shmid, int cmd, struct shmid\_ds\*buf)删除共享内存。

##### 2 基于消息队列的进程通信

**系统调用**



**代码实现**

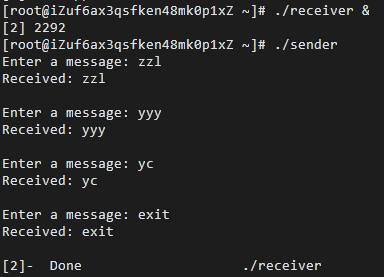
**sender.c**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <sys/types.h>
6. #include <sys/ipc.h>
7. #include <sys/msg.h>
8. #define MSG\_SIZE 256
9. struct message
10. {
11. long mtype;
12. char mtext[MSG\_SIZE];
13. };
14. int main()
15. {
16. key\_t key = ftok(".", 'a'); *// Generate a unique key*
17. int msgid = msgget(key, IPC\_CREAT | 0666); *// Create a message queue*
18. if (msgid == -1)
19. {
20. perror("msgget");
21. exit(1);
22. }
23. struct message msg;
24. msg.mtype = 1;
25. char buffer[MSG\_SIZE];
26. while (1)
27. {
28. printf("Enter a message: ");
29. fgets(buffer, MSG\_SIZE, stdin);
30. sprintf(msg.mtext, "%s", buffer);
31. if (msgsnd(msgid, &msg, MSG\_SIZE, 0) == -1)
32. {
33. perror("msgsnd");
34. exit(1);
35. }
37. *// Exit if the user enters "exit"*
38. if (strncmp(buffer, "exit", 4) == 0)
39. {
40. break;
41. }
42. sleep(1);
43. }
44. return 0;
45. }

**receiver.c**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <sys/types.h>
6. #include <sys/ipc.h>
7. #include <sys/msg.h>
8. #define MSG\_SIZE 256
9. struct message
10. {
11. long mtype;
12. char mtext[MSG\_SIZE];
13. };
14. int main()
15. {
16. key\_t key = ftok(".", 'a');
17. int msgid = msgget(key, IPC\_CREAT | 0666); *// get the message queue id*
18. if (msgid == -1)
19. {
20. perror("msgget");
21. exit(1);
22. }
23. struct message msg;
24. while (1)
25. {
26. if (msgrcv(msgid, &msg, MSG\_SIZE, 0, 0) == -1)
27. {
28. perror("msgrcv");
29. exit(1);
30. }
31. printf("Received: %s\n", msg.mtext);
32. *// Exit if the user enters "exit"*
33. if (strncmp(msg.mtext, "exit", 4) == 0)
34. {
35. break;
36. }
37. }
38. *// Remove the message queue*
39. if (msgctl(msgid, IPC\_RMID, NULL) == -1)
40. {
41. perror("msgctl");
42. exit(1);
43. }
44. return 0;
45. }

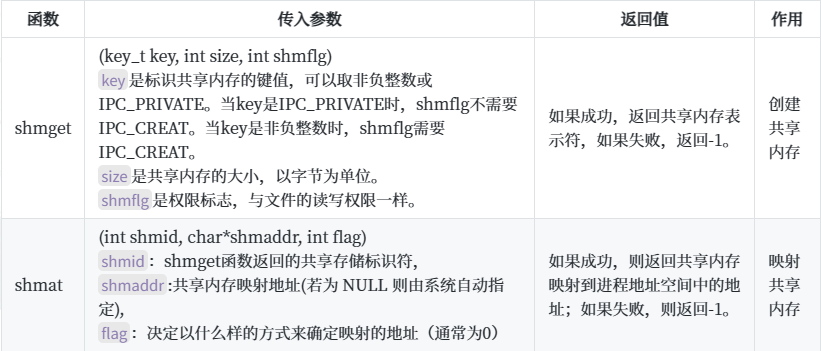
**运行结果**

****

进程间可以正常通信，运行结果符合预期。

##### 3 共享内存通信

**系统调用**

****

**代码实现**

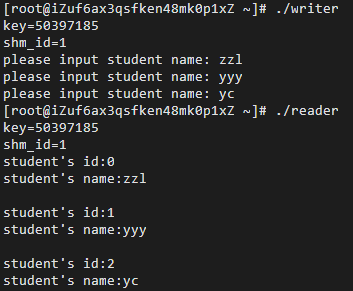
**writer.c**

1. #include <stdio.h>
2. #include <sys/ipc.h>
3. #include <sys/shm.h>
4. #include <sys/types.h>
5. #include <unistd.h>
6. typedef struct student
7. {
8. int id;
9. char name[8];
10. } student;
11. int main(int argc, char \*\*argv)
12. {
13. int shm\_id;
14. int i = 0;
15. key\_t key;
16. char buffer[BUFSIZ];
17. student \*school;
18. char pathname[30];
19. *// 根据路径名和项目id生成key*
20. strcpy(pathname, "/tmp");
21. key = ftok(pathname, 0x03);
22. if (key == -1)
23. {
24. perror("ftok error");
25. return -1;
26. }
27. printf("key=%d\n", key);
28. *// 创建共享内存*
29. shm\_id = shmget(key, 4096, IPC\_CREAT | IPC\_EXCL | 0600); *// IPC\_CREAT | IPC\_EXCL时, 如果没有该块共享内存，则创建，并返回共享内存ID*
30. if (shm\_id == -1)
31. {
32. perror("shmgct error");
33. return -1;
34. }
35. printf("shm\_id=%d\n", shm\_id);
36. *// 映射共享内存*
37. school = (student \*)shmat(shm\_id, NULL, 0);
38. for (i = 0; i < 3; i++)
39. {
40. printf("please input student name: ");
41. fgets(buffer, BUFSIZ, stdin);
42. (school + i)->id = 0 + i;
43. strncpy((school + i)->name, buffer, 5);
44. }
45. shmdt(school); *// 解除映射*
46. return 0;
47. }

**reader.c**

1. #include <stdio.h>
2. #include <sys/ipc.h>
3. #include <sys/shm.h>
4. #include <sys/types.h>
5. #include <unistd.h>
6. typedef struct student
7. {
8. int id;
9. char name[8];
10. } student;
11. int main(int argc, char \*\*argv)
12. {
13. int shm\_id, i;
14. key\_t key;
15. student \*school;
16. char pathname[30];
17. *// 根据路径名和项目id生成key*
18. strcpy(pathname, "/tmp");
19. key = ftok(pathname, 0x03);
20. if (key == -1)
21. {
22. perror("ftok error");
23. return -1;
24. }
25. printf("key=%d\n", key);
26. *// 获取共享内存ID*
27. shm\_id = shmget(key, 0, 0);
28. if (shm\_id == -1)
29. {
30. perror("shmget error");
31. return -1;
32. }
33. printf("shm\_id=%d\n", shm\_id);
34. *// 映射共享内存*
35. school = (student \*)shmat(shm\_id, NULL, 0);
36. for (i = 0; i < 3; i++)
37. {
38. printf("student's id:%d\n", (\*(school + i)).id);
39. printf("student's name:%s\n", (\*(school + i)).name);
40. }
41. *// 解除映射*
42. if (shmdt(school) == -1)
43. {
44. perror("detach error");
45. return -1;
46. }
47. shmctl(shm\_id, IPC\_RMID, 0); *// 删除共享内存*
48. return 0;
49. }

**运行结果**

****

进程间可以正常通信，运行结果符合预期。