201311211914_赵帅帅_实验 4

1、 实验目的

了解和熟悉 linux 支持的消息通信机制。

2、 实验任务

使用 linux 系统提供的系统调用 msgget(),msgrev(),msgctl()编制一个长度为 1K 的消息发送和接受的程序。

3、 实验要求

（1）用一个程序作为“引子”，先后 fork()两个进程，SERVER 和 CLIENT，进行通信。

（2）SERVER 端建立一个 Key 为 75 的消息队列，等待其他进程发来的消息。当遇到类型为 1 的消息，则作为结束信号，取消该队列，并退出 SERVER。SERVER 每接受到一个消息后显示一句“(Server)received”。

（3）CLIENT 端使用 key 为 75 的消息队列，先后发送类型从 10 到 1 的消息，然后退出。最后一个消息，即是 SERVER 端需要的结束信号。CLIENT 每发送一条消息后显示一句“(Client)sent”。

（4）父进程在 SERVER 和 CLIENT 都退出后结束。

4、 代码

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<stdlib.h>

const int MSGKEY = 75;
struct msgform
{
 long mtype;
 char mtext[1024];
}msg;
int msgqid;

cleanup()
{
 msgctl(msgqid,IPC_RMID,0);
 exit(0);
```

```
}

void client()
{
 int i = 10;
 msgqid = msgget(MSGKEY, 0777);

 for(i =10; i >= 1; i --)
 {
     msg.mtype = i;
     msgsnd(msgqid, &msg, 1032, 0);
     printf("(Client)sent:%d\n", i);

     sleep(1);

     msgrcv(msgqid, &msg ,1032, 100, 0);
     printf("(Client)sure\n");

 }
 //exit(0);
}

void server()
{
 msgqid = msgget(MSGKEY, 0777|IPC_CREAT);

 int i = 10;
 for(;;)
 {
     msgrcv(msgqid, &msg, 1032, i, 0);
     printf("(Server)received:%d\n", msg.mtype);
     long temp = msg.mtype;

     sleep(1);

     msg.mtype = 100;
     msgsnd(msgqid, &msg ,1032,0);
     printf("(Server)return\n");

     if(temp == 1)
         break;
     i --;
 }
 //exit(0);
```

```
}

int main(void)
{
 int  Server, Client;
 Client = fork();
 if(!Client)
 {
     client();
 }
 else
 {
     Server = fork();
     if(!Server)
     {
         server();
     }
 }
 msgctl(msgqid,IPC_RMID,0);
 wait(0);
 wait(0);
 return 0;
}
```

5、 运行结果和解释

```
(Client)sent:10
(Server)received:10
(Server)return
(Client)sure
(Client)sent:9
(Server)received:9
(Server)return
(Client)sure
(Client)sent:8
(Server)received:8
(Server)return
(Client)sure
(Client)sent:7
(Server)received:7
(Server)return
(Client)sure
(Client)sent:6
(Server)received:6
(Server)return
(Client)sure
```

(Client)sent:5
(Server)received:5
(Server)return
(Client)sure
(Client)sent:4
(Server)received:4
(Server)return
(Client)sure
(Client)sent:3
(Server)received:3
(Server)return
(Client)sure
(Client)sent:2
(Server)received:2
(Server)return
(Client)sure
(Client)sent:1
(Server)received:1
(Server)return
(Client)sure

结果解释：

依照实验要求，客户端先发了一条类型为 10 的消息，服务端接收后要说明自己收到了，再回信给客户端表明自己收到了，客户端表明自己收到服务端的回信从而确认自己发给服务端的信件已经被服务端受到了，本次通信结束，可以进行下一次通信。

例：

(Client)sent:10　　//客户端向服务端发消息

(Server)received:10　　//服务端收到信

(Server)return　　//服务端回信给客户端

(Client)sure　　//客户端确定服务端收到消息

6、 实验感想

实验算法设计时容易把人搞糊涂，主要是对消息传送的函数不熟悉导致的，只要认真理解函数中的参数并且弄清楚函数中所对应的消息中的消息类型和消息正文存的是什么，那么这次的实验就没有什么难度了。

此次实验还有许多细节需要注意，如 exit()函数和 wait()函数要弄明白是干什么的，有什么区别。