## 实验九 进程通信——消息队列

课程名称 操作系统原理实验

实验项目名称 实验九 消息队列

学生姓名 司晨旭 专业班级 计算2002 学号 32001019

1. **两个进程并发执行，通过消息队列，分别进行消息的发送和接受**

|  |
| --- |
| **课本代码略** |

|  |
| --- |
| **运行结果如下** |

**1．熟悉和消息队列相关的系统调用**

|  |
| --- |
| **int msgget(key\_t key,int msgflg)**  **创建消息队列**  **int msgsnd(int msqid,const void \*msgp,size\_t msgsz, int msgflg)**  **向消息队列中发送消息**  **ssize\_t msgrcv(int msqid,void \* msgp,size\_t msgsz, long msgtyp,int msgflg)**  **从消息队列中接受消息**  **int msgctl(int msgid,int cmd,struct msqid\_ds \*buf)**  **在消息队列上执行指定的操作** |

**2.尝试运行多个发送进程和多个接受进程，观察进程的并发执行情况，并解释原因**

|  |
| --- |
|  |
| **消息队列本身提供同步和互斥机制** |

**二.编译链接通过后，运行程序，通过程序提示，观察多线程得并发执行，并思考下述问题**

|  |
| --- |
| **课本代码略过** |

**1．熟悉和消息队列相关得系统调用**

|  |
| --- |
| **int msgget(key\_t key,int msgflg)**  **创建消息队列**  **int msgsnd(int msqid,const void \*msgp,size\_t msgsz, int msgflg)**  **向消息队列中发送消息**  **ssize\_t msgrcv(int msqid,void \* msgp,size\_t msgsz, long msgtyp,int msgflg)**  **从消息队列中接受消息**  **int msgctl(int msgid,int cmd,struct msqid\_ds \*buf)**  **在消息队列上执行指定的操作** |

**2.回顾与POSIX线程控制和信号量相关的函数**

|  |
| --- |
| **int sem\_init(sem\_t \*sem,int pshared unsigned int value)**  **信号量初始化**  **int sem\_wait(sem\_t \*sem)**  **P 信号量减一的操作**  **int sem\_post(sem\_t \*sem)**  **V 信号量加一的操作** |

**3.尝试删除信号量的同步控制，观察并发线程运行混乱的情况**

|  |
| --- |
| **正常运行结果如上图**  **下面给出删除P V操作的图**    **删除了之后 三个线程之间的运行十分混乱** |

**4.理清例程中并发线程同步和互斥的关系**

|  |
| --- |
| **一共有三个同步关系 一个互斥关系**  **三个同步关系分别为**  **snd\_dp rcv\_dp 为一个同步关系**  **empty 和full 为一个同步关系**  **over 为一个同步关系**  **一个互斥为**  **w\_mutex** |

**编程题**

|  |
| --- |
| **运行结果如上图**  **代码如下**  **#include <stdlib.h>**  **#include <stdio.h>**  **#include <string.h>**  **#include <errno.h>**  **#include <unistd.h>**  **#include <sys/types.h>**  **#include <sys/ipc.h>**  **#include <sys/msg.h>**  **#define MAX\_TEXT 512**  **struct my\_msg\_st**  **{**  **long int my\_msg\_type;**  **char some\_text[BUFSIZ];**  **};**  **int main()**  **{**  **int running = 1;**  **int msgid;**  **struct my\_msg\_st some\_data;**  **long int msg\_to\_receive = 0;**  **char buffer[BUFSIZ];**  **int flag;**  **/\* First, we set up the message queue. \*/**  **msgid = msgget((key\_t)1234, 0666 | IPC\_CREAT);**  **if (msgid == -1)**  **{**  **fprintf(stderr, "msgget failed with error: %d\n", errno);**  **exit(EXIT\_FAILURE);**  **}**  **/\* Then the messages are retrieved from the queue, until an end message is encountered.**  **Lastly, the message queue is deleted. \*/**  **while (running)**  **{**  **scanf("%d", &flag);**  **some\_data.my\_msg\_type = 0;**  **if (flag == 1)**  **{**  **if (msgrcv(msgid, (void \*)&some\_data, BUFSIZ,**  **msg\_to\_receive, 0) == -1)**  **{**  **fprintf(stderr, "msgrcv failed with error: %d\n", errno);**  **exit(EXIT\_FAILURE);**  **}**  **else**  **{**  **printf("You wrote: %s", some\_data.some\_text);**  **}**  **}**  **else**  **{**  **printf("Enter some text: ");**  **fgets(buffer, BUFSIZ, stdin);**  **some\_data.my\_msg\_type = 1;**  **strcpy(some\_data.some\_text, buffer);**  **if (msgsnd(msgid, (void \*)&some\_data, MAX\_TEXT, 0) == -1)**  **{**  **fprintf(stderr, "msgsnd failed\n");**  **exit(EXIT\_FAILURE);**  **}**  **}**  **if (strncmp(some\_data.some\_text, "end", 3) == 0)**  **{**  **running = 0;**  **}**  **if (strncmp(some\_data.some\_text, "1", 1) == 0)**  **{**  **flag=1;**  **}**  **}**  **if (msgctl(msgid, IPC\_RMID, 0) == -1)**  **{**  **fprintf(stderr, "msgctl(IPC\_RMID) failed\n");**  **exit(EXIT\_FAILURE);**  **}**  **exit(EXIT\_SUCCESS);**  **}** |