/\*\*－－ 生产者/消费者问题

\*/

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#define BUFFER "./buffer" //生产者 消费者的一个非共享式资源。本地文件

//如果你收到一个多semun的双重定义错误，则注释掉union定义

union semun //是一个sem联合结构

{

int val;

struct semid\_ds \*buf;

ushort \*array;

};

main (int argc,char \*argv[])

{

FILE \*fptr;

static struct sembuf acquire = { 0,-1, SEM\_UNDO }, //两个信号量模板

release = { 0, 1, SEM\_UNDO }; //SEM\_UNDO表示可以撤销

pid\_t c\_pid;

key\_t ipc\_key;

static ushort start\_val[2] = {1,0};

int semid,producer = 0,i,n,p\_sleep,c\_sleep;

union semun arg;

enum {READ,MADE};

if (argc != 2) //检查是否有一个参数：进程在生命周期中的slepp秒数

{

fprintf (stderr,"%s sleep\_time\n",argv[0]);

exit (-1);

}

ipc\_key = ftok (".",'S'); //获取信号量标识

if ((semid = semget(ipc\_key,2,IPC\_CREAT | IPC\_EXCL | 0666)) != -1) //创建一个信号量集合或者获得

{

producer = 1; //开始是1，表示最开始为生产者生产

arg.array = start\_val;

if(semctl(semid,0,SETALL,arg) == -1) //初始化信号量集合

{

perror("semctl -- producer -- initialization");

exit (1);

}

}

else if ((semid = semget(ipc\_key,2,0)) == -1)

{

exit(2);

}

/\*

\*生产者/消费者问题的第二部分

\*/

switch (producer)

{

case 1: //生产者

p\_sleep = atoi(argv[1]); //睡眠时间

srand ((unsigned) getpid());

for (i = 0; i < 10; i++)

{

sleep (p\_sleep);

n = rand()%99 + 1;

printf("A. The number [%2d] enerated by producer\n",n);

acquire.sem\_num = READ; //

if (semop(semid,&acquire,1) == -1)

{

perror ("semop -- producer -- waiting for cnsumer to read number");

exit(3);

}

if ((fptr = fopen(BUFFER,"w")) == NULL)

{

perror(BUFFER);

exit(4);

}

fprintf(fptr,"%d\n",n); //写入

fclose(fptr);

release.sem\_num = MADE; //类似于full

printf("B.The number [%2d] deposited by producer\n",n);

if (semop(semid,&release,1) == -1)

{

perror ("semop -- producer -- indicating new number has been made");

exit(5);

}

}

sleep(5);

if (semctl(semid,0,IPC\_RMID,0) == -1)

{

perror("semctl -- producer");

exit(6);

}

printf("Semaphore removed\n");

break;

case 0: //消费者

c\_sleep = atoi(argv[1]);

c\_pid = getpid();

while(1)

{

sleep(c\_sleep);

acquire.sem\_num = MADE;

if (semop(semid,&acquire,1) == -1)

{

perror ("semop -- consumer -- waiting for new number to be made");

exit(7);

}

if ((fptr = fopen(BUFFER,"r")) == NULL)

{

perror(BUFFER);

exit(8);

}

fptr = fopen(BUFFER,"r");

fscanf(fptr,"%d",&n);

fclose(fptr);

release.sem\_num = READ;

if(semop(semid,&release,1) == -1)

{

perror ("semop -- consumer -- indicatingnumberhas been read");

exit(9);

}

printf("C.The number [%2d] obtained by consumer %6d\n",n,c\_pid);

}

exit(0);

}

}