**内蒙古农业大学**

**计算机与信息工程学院**

实  验  报  告

实验三

课程名称：     嵌入式系统开发实践

班    级：        21计科四班

姓    名：        李振浩

学    号：         2021122143170

授课教师：      张永安

**实验三 线程编程**

**一、      实验目的**

1.     掌握常用函数的使用；

2.     掌握嵌入式Linux多线程编程；

3.     掌握线程的同步与互斥机制。

**二、      实验工具和环境**

PC机、Linux Ubuntu操作系统。

**三、      实验内容与结果**

1、（1）创建线程文件pth.c，在主线程中利用pthread\_create()函数创建一个线程，线程号为id；线程函数名称为thread1()。在主线程中输出5条“This is main thread!”,在线程函数中输出5条“This is pthread1!”。主线程中利用线程阻塞函数pthread\_join()，使线程函数得以执行，并利用pthread\_self()分别输出显示主线程和线程thread1的线程号。该题完成后，将所有文件保存在文件夹expr1中，并保留linux输出结果的截屏。

（2）在上题的pth.c文件的基础上复制并修改程序，练习使用pthread\_cancel()函数取消线程的执行。即在线程程序中增加while(1)，让线程不断输出内容，在主程序中增加pthread\_cancel()，来终止线程的输出。该题完成后，将所有文件保存在文件夹expr2中，并保留linux输出结果的截屏。

（3）在上题的pth.c文件的基础上复制并修改程序，练习使用pthread\_create()函数的参数传递（传递字符型还是数值型自定），利用pthread\_create()函数传递参数到线程程序，并输出所传递参数的内容。该题完成后，将所有文件保存在文件夹expr3中，并保留linux输出结果的截屏。

（4）在上题的pth.c文件的基础上复制并修改程序，实现多线程操作。增加线程函数thread2()，每个线程函数输出5条信息，注意输出时增加sleep(1)的延时。查看输出结果。该题完成后，将所有文件保存在文件夹expr4中，并保留linux输出结果的截屏。

（5）在上题的pth.c文件的基础上复制并修改程序，实现多线程互斥操作。增加互斥锁，让每个得到互斥锁的线程能够锁定临界资源，完成该线程的所有操作。查看输出结果。该题完成后，将所有文件保存在文件夹expr5中，并保留linux输出结果的截屏。

（6）在上题的pth.c文件的基础上复制并修改程序，练习信号量互斥操作。增加两个信号量，完成类似互斥锁的线程互斥，让每个线程能够锁定临界资源，完成该线程的所有操作。查看输出结果。该题完成后，将所有文件保存在文件夹expr6中，并保留linux输出结果的截屏。

（7）在上题的pth.c文件的基础上复制并修改程序，练习信号量同步操作。让线程thread1()和thread2()能够交替执行。查看输出结果。该题完成后，将所有文件保存在文件夹expr7中，并保留linux输出结果的截屏。

（8）在上题的pth.c文件的基础上复制并修改程序，练习单个信号量同时并行多个任务的操作。创建一个信号量和一个线程函数，但该信号量可同时并行运行多个任务（具体任务个数自己决定，一般不小于3个）。在线程函数中输出线程申请资源的信息、申请成功和释放资源的信息，并随机延时一段时间（随机延时2-5秒）。查看输出结果。该题完成后，将所有文件保存在文件夹expr8中，并保留linux输出结果的截屏。

2、利用线程的同步与互斥实现“生产者消费者问题”：

      在主线程中创建生产者与消费者线程，生产者每次生产时锁定临界资源，向资源池添加一个资源（资源可以用一个公有变量代替）；消费者消费每次消费时锁定临界资源，并使资源池的数量减一。注意生产和消费时都需要一定的时间，可以用随机时间，或自定义生产消费时间。

**四、      实验结果**

将以上综合案例的代码复制到下面，并将执行的输出结果截图放到代码后面。

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 void \*thread1(){

5 int i;

6 for(i=0;i<5;i++){

7 printf("This is pthread1!\n");

8 usleep(1000000);

9 }

10 return (void \*)i;

11 }

12

13 int main()

14 {

15 pthread\_t tid;

16 pthread\_create(&tid,NULL,thread1,NULL);

17 int i;

18

19 for(i=0;i<5;i++){

20 printf("This is main thread!\n");

21 usleep(1000000);

22 }

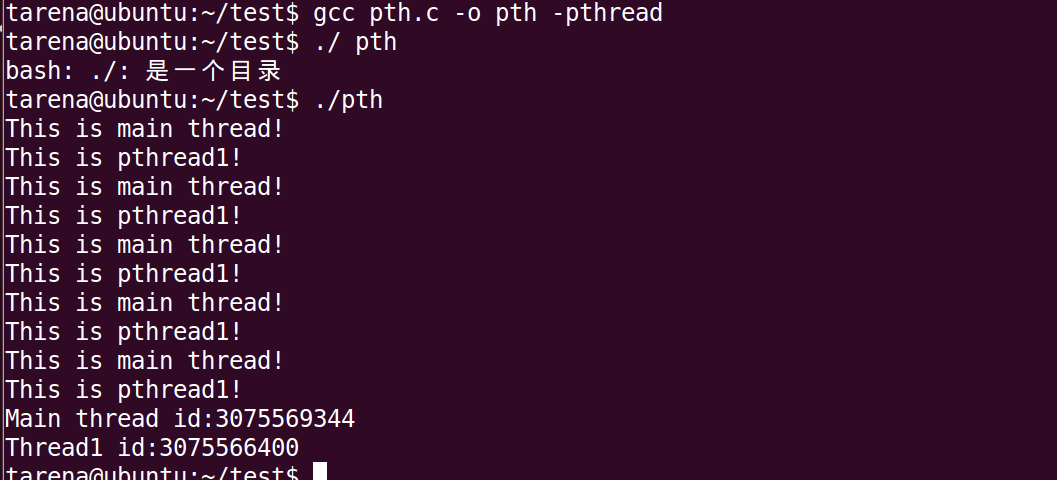
23 pthread\_join(tid,NULL);

24 printf("Main thread id:%u\n",(unsigned int)pthread\_self());

25 printf("Thread1 id:%lu\n",tid);

26 return 0;

27 }



二

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 void \*thread1(){

5 int i;

6 while(1){

7 printf("This is pthread1!\n");

8 sleep(1);

9 }

10 return (void \*)i;

11 }

12

13 int main()

14 {

15 pthread\_t tid;

16 pthread\_create(&tid,NULL,thread1,NULL);

17 int i;

18

19 for(i=0;i<5;i++){

20 printf("This is main thread!\n");

21 usleep(1000000);

22 }

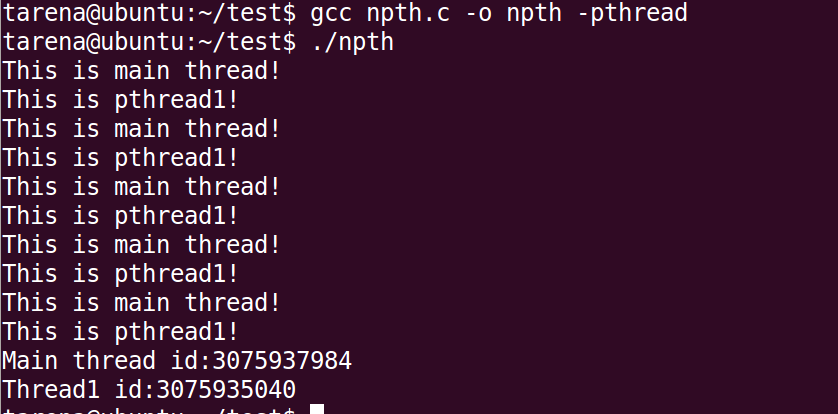
23 pthread\_cancel(tid);

24 printf("Main thread id:%u\n",(unsigned int)pthread\_self());

25 printf("Thread1 id:%lu\n",tid);

26 return 0;

27 }



三

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 void \*thread1(void \*buf){

5 char \*str=(char \*)buf;

6 while(1){

7 printf("Thread1 :%s\n",str);

8 usleep(1000000);

9 }

10 }

11

12 int main()

13 {

14 pthread\_t tid;

15 char buf[256]="abcdef";

16 pthread\_create(&tid,NULL,thread1,buf);

17 int i;

18

19 for(i=0;i<5;i++){

20 printf("This is main thread!\n");

21 usleep(1000000);

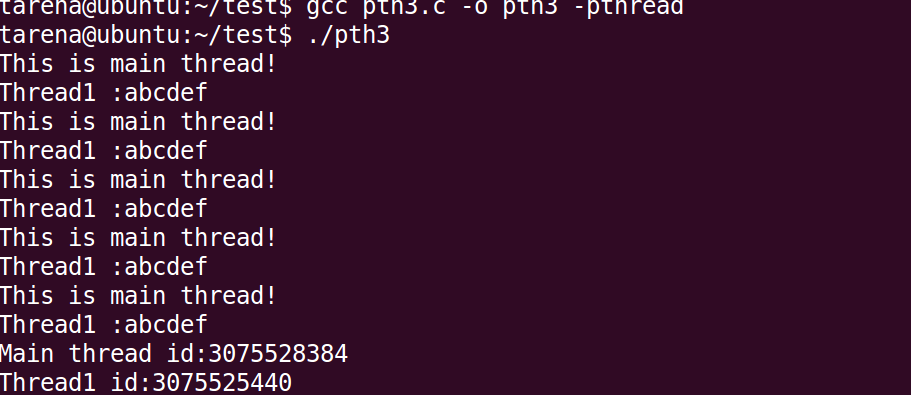
22 }

23 pthread\_cancel(tid);

24 printf("Main thread id:%u\n",(unsigned int)pthread\_self());

25 printf("Thread1 id:%lu\n",tid);

26 return 0;

27 }

四

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 void \*thread1(){

5 int i;

6 for(i=0;i<5;i++){

7 printf("This is pthread1!\n");

8 sleep(1);

9 }

10 return (void \*)i;

11 }

12

13 void \*thread2(){

14 int i;

15 for(i=0;i<5;i++){

16 printf("This is pthread2!\n");

17 sleep(1);

18 }

19 return (void \*)i;

20 }

21 int main()

22 {

23 pthread\_t tid1,tid2;

24 pthread\_create(&tid1,NULL,thread1,NULL);

25 pthread\_create(&tid2,NULL,thread2,NULL);

26

27 int i;

28

29 for(i=0;i<5;i++){

30 printf("This is main thread!\n");

31 usleep(1000000);

32 }

33 pthread\_join(tid1,NULL);

34 pthread\_join(tid2,NULL);

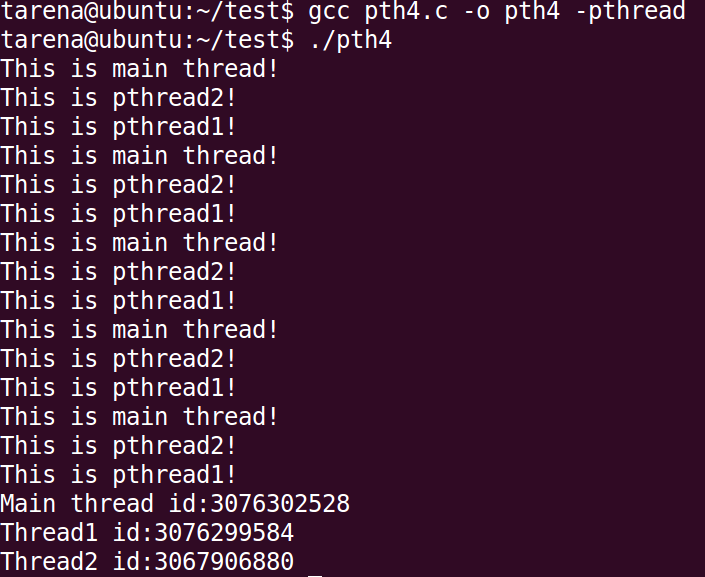
35 printf("Main thread id:%u\n",(unsigned int)pthread\_self());

36 printf("Thread1 id:%lu\n",tid1);

37 printf("Thread2 id:%lu\n",tid2);

38 return 0;

39 }



五

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 char str[1024];

5 pthread\_mutex\_t mutex;

6 void \*thread1(void \*buf){

7 while(1){

8 pthread\_mutex\_lock(&mutex);

9 sprintf(str,"This is pyhread1!\n");

10 printf("%s\n",str);

11 sleep(5);

12 pthread\_mutex\_unlock(&mutex);

13 usleep(1);

14 }

15 }

16

17 void \*thread2(void \*buf){

18 while(1){

19 pthread\_mutex\_lock(&mutex);

20 sprintf(str,"This is pyhread2!\n");

21 printf("%s\n",str);

22 sleep(2);

23 pthread\_mutex\_unlock(&mutex);

24 usleep(1);

25 }

26 }

27 int main()

28 {

29 pthread\_mutex\_init(&mutex,NULL);

30

31 pthread\_t tid1,tid2;

32 pthread\_create(&tid1,NULL,thread1,NULL);

33 pthread\_create(&tid2,NULL,thread2,NULL);

34

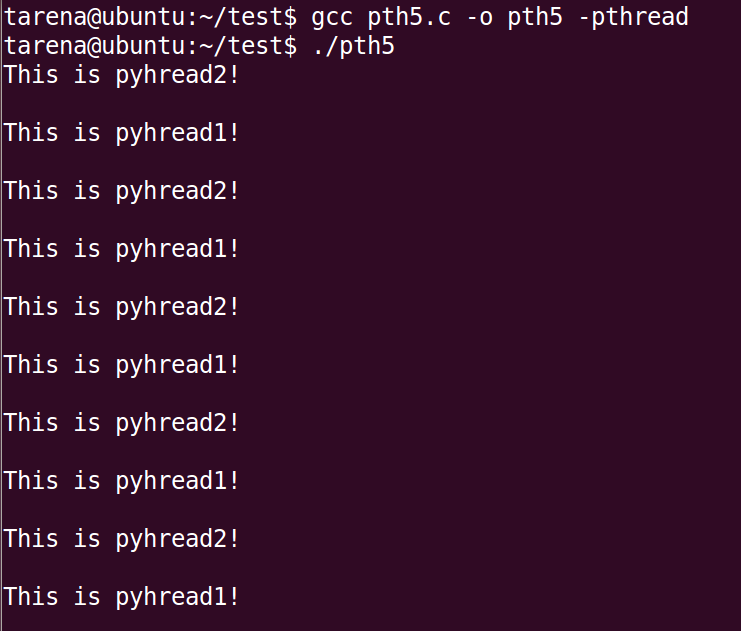
35 pthread\_join(tid1,NULL);

36 pthread\_join(tid2,NULL);

37 pthread\_mutex\_destroy(&mutex);

38 return 0;

39 }



六

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 #include<string.h>

5 #include<pthread.h>

6 #include<semaphore.h>

7 sem\_t sem;

8 void \*thread1(void \*buf){

9 int i=0;

10 while(1){

11 sem\_wait(&sem);

12 printf("This is pthread1!,i=%d\n",i++);

13 sem\_post(&sem);

14 usleep(1);

15 }

16 }

17

18 void \*thread2(void \*buf){

19 int i=0;

20 while(1){

21 sem\_wait(&sem);

22 printf("This is pthread2!,i=%d\n",i++);

23 sleep(2);

24 sem\_post(&sem);

25 usleep(1);

26 }

27 }

28 int main()

29 {

30 sem\_init(&sem,0,1);

31 pthread\_t tid1,tid2;

32 pthread\_create(&tid1,NULL,thread1,NULL);

33 pthread\_create(&tid2,NULL,thread2,NULL);

34

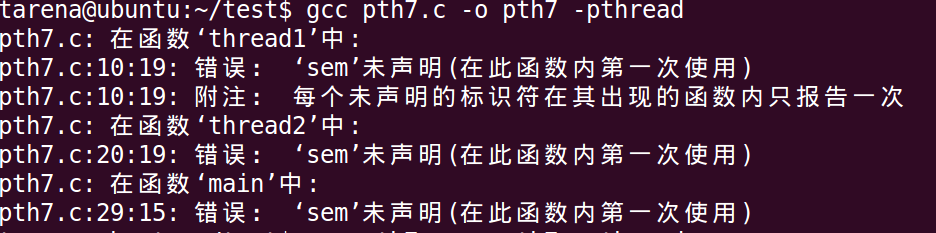
35 pthread\_join(tid1,NULL);

36 pthread\_join(tid2,NULL);

37 sem\_destroy(&sem);

38 return 0;

39 }



七

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 char str[1024];

5 pthread\_mutex\_t mutex;

6 pthread\_cond\_t cond;

7 int count=0;

8 void \*thread1(void \*buf){

9 while(1){

10 pthread\_cond\_wait(&cond,&mutex);

11 count-=2;

12 sprintf(str,"This is pyhread1!");

13 printf("%s,count=%d\n",str,count);

14 sleep(5);

15 usleep(1);

16 }

17 }

18

19 void \*thread2(void \*buf){

20 while(1){

21 sprintf(str,"This is pyhread2!");

22 printf("%s,count=%d\n",str,count);

23 sleep(2);

24 if(count%10==0)

25 pthread\_cond\_signal(&cond);

26 count++;

27 usleep(1);

28 }

29 }

30 int main()

31 {

32 pthread\_mutex\_init(&mutex,NULL);

33 pthread\_cond\_init(&cond,NULL);

34 pthread\_t tid1,tid2;

35 pthread\_create(&tid1,NULL,thread1,NULL);

36 pthread\_create(&tid2,NULL,thread2,NULL);

37

38 pthread\_join(tid1,NULL);

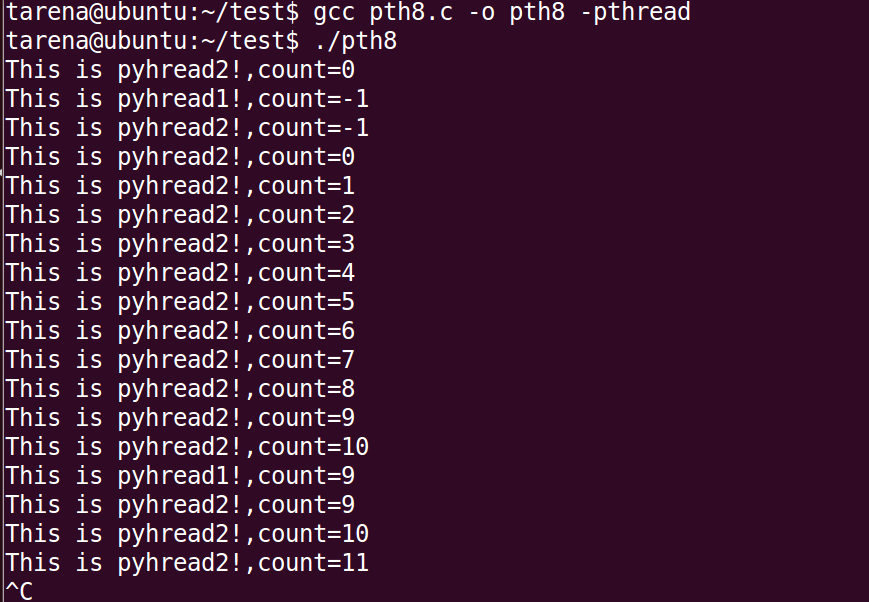
39 pthread\_join(tid2,NULL);

40 pthread\_mutex\_destroy(&mutex);

41 pthread\_cond\_destroy(&cond);

42 return 0;

43 }



八

1 #include<stdio.h>

2 #include<pthread.h>

3 #include<stdlib.h>

4 #include<unistd.h>

5 #include<semaphore.h>

6

7 sem\_t sem;

8 void \*task(void \*buf){

9 int task\_id=\*((int \*)buf);

10 printf("Task %d is waiting for resource\n",task\_id);

11 sem\_wait(&sem);

12 printf("Task %d acquired resource\n",task\_id);

13 sleep(2);

14 printf("Task %d released resource\n",task\_id);

15 sem\_post(&sem);

16 pthread\_exit(NULL);

17 }

18

19 int main()

20 {

21 pthread\_t threads[3];

22 int task\_ids[3];

23 sem\_init(&sem,0,2);

24 int i;

25 for(i=0;i<3;i++){

26 task\_ids[i]=i+1;

27 pthread\_create(&threads[i],NULL,task,&task\_ids[i]);

28 }

29 for(i=0;i<3;i++){

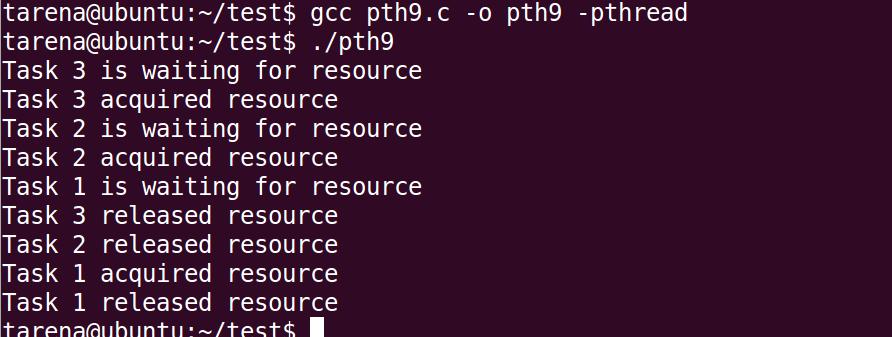
30 pthread\_join(threads[i],NULL);

31 }

32 sem\_destroy(&sem);

33 return 0;

34 }



八

1 #include<stdio.h>

2 #include<stdlib.h>

3 #include<pthread.h>

4 #include<unistd.h>

5 #include<semaphore.h>

6 int buf[5];

7 int in=0;

8 int out=0;

9 pthread\_mutex\_t mutex;

10 sem\_t full,empty;

11 void \*producer(void \*arg){

12 int item;

13 while(1){

14 item=rand()%100+1;

15 sem\_wait(&empty);

16 pthread\_mutex\_lock(&mutex);

17 printf("producer produced item%d\n",item);

18 buf[in]=item;

19 in=(in+1)%5;

20 pthread\_mutex\_unlock(&mutex);

21 sem\_post(&full);

22 sleep(3);

23 }

24 }

25 void \*consumer(void \*arg){

26 int item;

27

28 while(1){

29

30 sem\_wait(&full);

31 pthread\_mutex\_lock(&mutex);

32

33 item=buf[out];

34 printf("consumer consumed item%d\n",item);

35 out=(out+1)%5;

36 pthread\_mutex\_unlock(&mutex);

37 sem\_post(&empty);

38 sleep(3);

39 }

40 }

41 int main(){

42 pthread\_t pro,con;

43 pthread\_mutex\_init(&mutex,NULL);

44 sem\_init(&full,0,0);

45 sem\_init(&empty,0,5);

46 pthread\_create(&pro,NULL,producer,NULL);

47 pthread\_create(&con,NULL,consumer,NULL);

48

49 pthread\_join(pro,NULL);

50 pthread\_join(con,NULL);

51

52 pthread\_mutex\_destroy(&mutex);

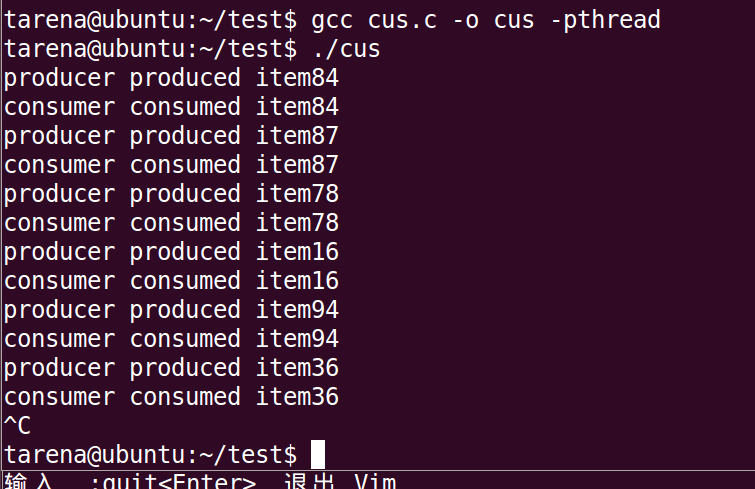
53 sem\_destroy(&full);

54 sem\_destroy(&empty);

55

56 return 0;

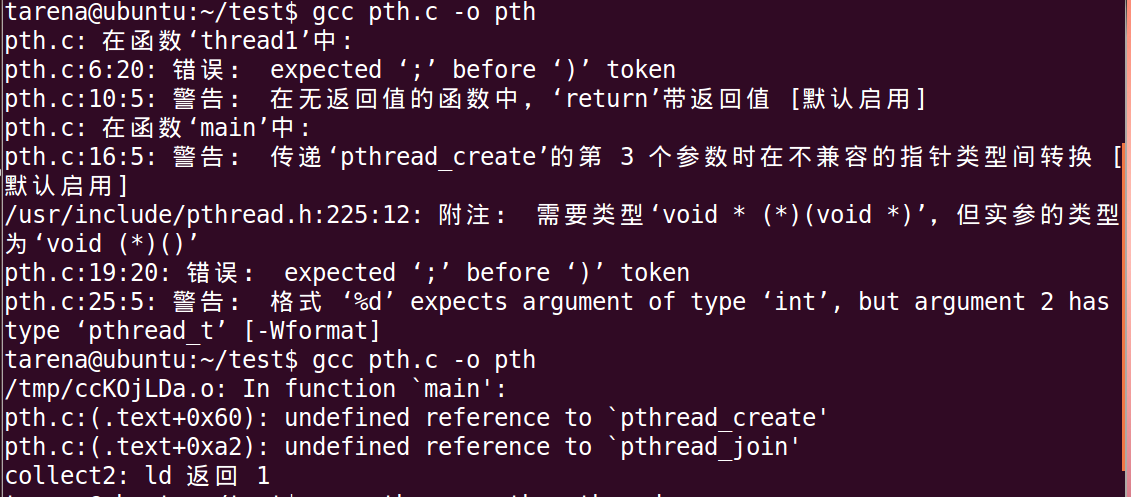
57 }



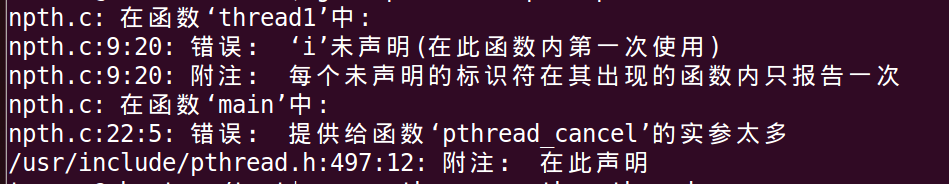
**五、      实验总结**

简述案例设计思路，并给出程序设计和调试过程中遇到问题的解决思路和方法。

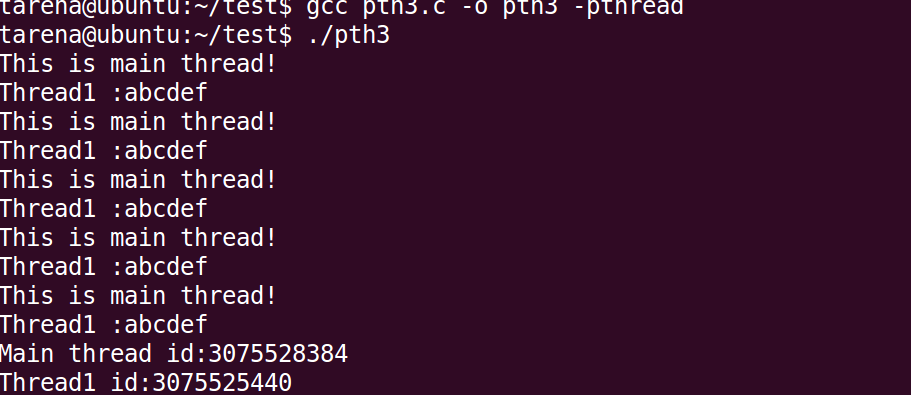
一



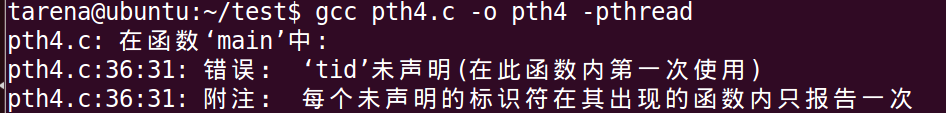
二



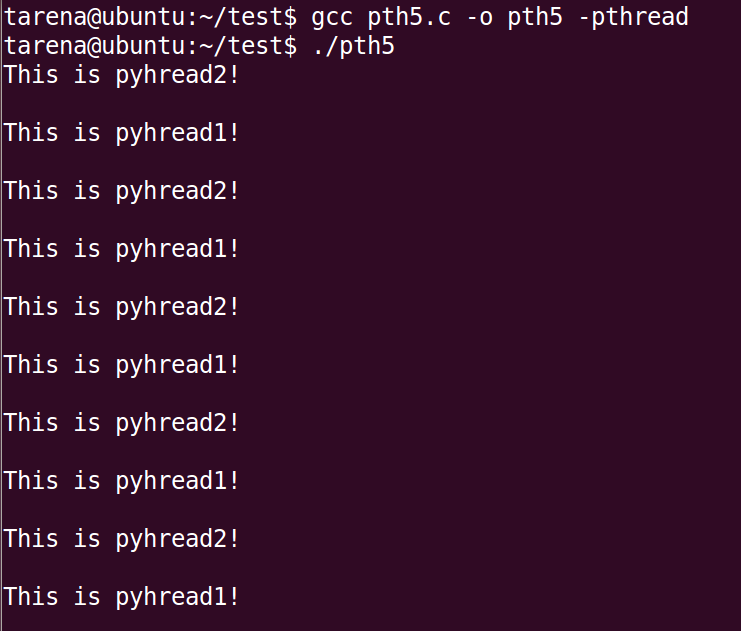
三



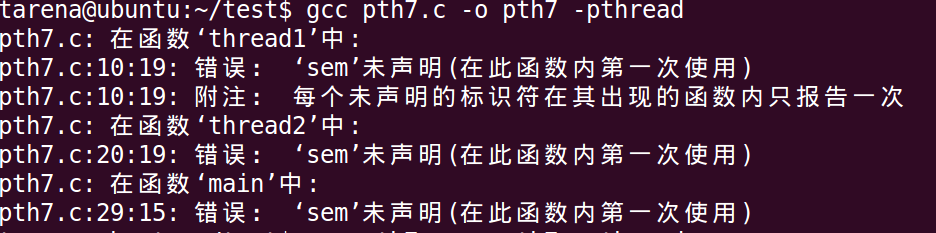
四



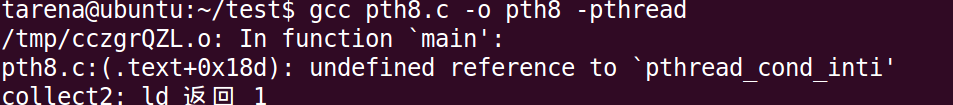
五



六



七



八

