《Linux应用程序开发实验报告》

题目: 多线程编程

班级:\_\_ \_\_ \_\_17计科3班 \_

学号: 10417307 \_\_\_

姓名: \_\_薛冬冬\_ \_\_\_

东南大学成贤学院计算机系

2020年4月30日

**多线程编程**

1. **实验目的**

1. 掌握 Linux 中线程的基本概念；

2. 掌握 Linux 中线程的创建及使用；

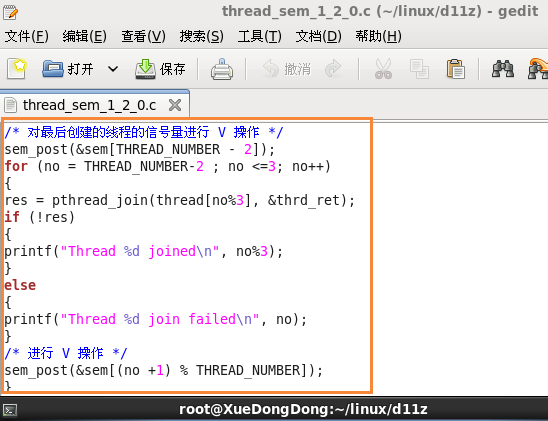
3. 掌握 Linux 中线程属性的设置；

4. 能够独立编写多线程程序。

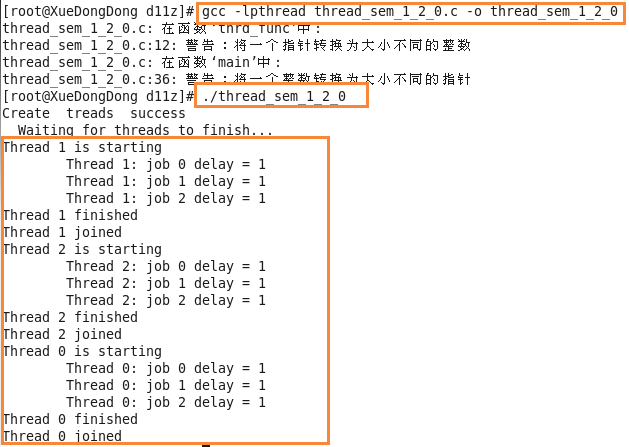
1. **实验内容**

**1. 用信号量同步机制实现 3 个线程之间的有序执行。改写课本程序，使线程执行顺序为1，2，0。**

代码改写位置：

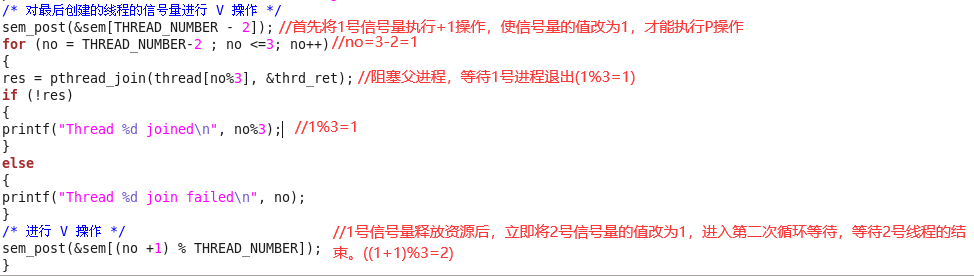


终端运行效果：

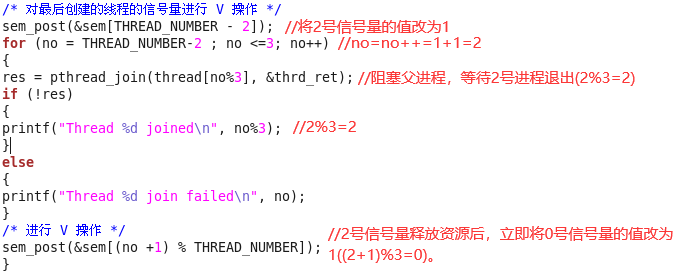


具体分析：

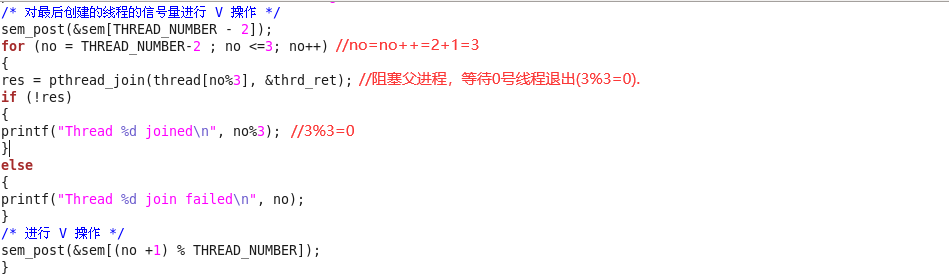
●1号线程：



●2号线程：



●0号线程：



具体完整代码见四、附录。

1. **实验体会**

本次实验相较于之前的实验简单了很多，题量也减少了很多，只有一道题目。这次实验，只要将老师的讲解视频耐心认真的看和分析，即可得到具体的解决方案。老师在视频中对该章内容的讲解认真且详细，不仅讲解了本节课的知识点，还对代码进行了具体的讲解，细化到一条一条代码的讲解分析，听了老师的清晰讲解，实验那条题目很快就找到了突破口和解决方案。

通过本次实验，使我深入掌握了线程的基本概念和线程的创建及使用，再加上老师将对函数的讲解都放到具体实例中讲解，使我对知识点中涉及到的函数作用有了更好的了解。本节课中，老师还讲解了生产者消费者实验，使用了知识点有名管道和信号量，再结合第六章的实验，使我意识到了生产者消费者实验的重要性。

1. **附录**

具体代码：

/\*thread\_sem\_1\_2\_0.c\*/

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#define THREAD\_NUMBER 3 /\* 线程数 \*/

#define REPEAT\_NUMBER 3 /\* 每个线程中的小任务数 \*/

#define DELAY\_TIME\_LEVELS 5 /\*小任务之间的最大时间间隔\*/

sem\_t sem[THREAD\_NUMBER];

void \*thrd\_func(void \*arg)

{

int thrd\_num = (int)arg;

int delay\_time = 0;

int count = 0;

/\* 进行 P 操作 \*/

sem\_wait(&sem[thrd\_num]);

printf("Thread %d is starting\n", thrd\_num);

for (count = 0; count < REPEAT\_NUMBER; count++)

{

delay\_time = (int)(rand() \* DELAY\_TIME\_LEVELS/(RAND\_MAX)) + 1;

sleep(delay\_time);

printf("\tThread %d: job %d delay = %d\n",thrd\_num, count, delay\_time);

}

printf("Thread %d finished\n", thrd\_num);

pthread\_exit(NULL);

}

int main(void)

{

pthread\_t thread[THREAD\_NUMBER];

int no = 0, res;

void \* thrd\_ret;

srand(time(NULL));

for (no = 0; no < THREAD\_NUMBER; no++)

{

sem\_init(&sem[no], 0, 0);

res = pthread\_create(&thread[no], NULL, thrd\_func, (void\*)no);

if (res != 0)

{

printf("Create thread %d failed\n", no);

exit(res);

}

}

printf("Create treads success\n Waiting for threads to finish...\n");

/\* 对最后创建的线程的信号量进行 V 操作 \*/

sem\_post(&sem[THREAD\_NUMBER - 2]);

for (no = THREAD\_NUMBER-2 ; no <=3; no++)

{

res = pthread\_join(thread[no%3], &thrd\_ret);

if (!res)

{

printf("Thread %d joined\n", no%3);

}

else

{

printf("Thread %d join failed\n", no);

}

/\* 进行 V 操作 \*/

sem\_post(&sem[(no +1) % THREAD\_NUMBER]);

}

for (no = 0; no < THREAD\_NUMBER; no++)

{

/\* 删除信号量 \*/

sem\_destroy(&sem[no]);

}

return 0;

}