

**课 程 实 验 报 告**

**课程名称： 操 作 系 统 原 理**

**专业班级： 计算机科学与技术1609**

**学 号： U201614758**

**姓 名： 梅朝瑞**

**指导教师： 谢 夏**

**报告日期： 2019/1/1**

**计算机科学与技术学院**

# 实验二：进程控制

## 实验目的

1. 掌握Linux下线程的概念；
2. 了解Linux线程同步与通信的主要机制；
3. 通过信号灯操作实现线程间的同步与互斥。

## 实验内容

1. **程序要求**

通过Linux多线程与信号灯机制，设计并实现计算机线程与I/O线程共享缓冲区的同步与通信。

程序要求:两个线程,共享公共变量a

线程1负责计算(1到100的累加，每次加一个数)

线程2负责打印（输出累加的中间结果)

1. **运行环境**

**软件配置：**

主机：Windows 10 专业版 Build 17763.195

虚拟机：VMware Workstation 14 Pro，Deepin 15.8 Linux/GNU

开发环境：Eclipse IDE for C/C++ Developers Oxygen Release (4.7.0)

**硬件：**

AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz

Kinston HyperX Fury DDR4 2400 8G

NVIDIA GeForce GTX 1060 6GB

1. **源程序**

//头文件，包含了各种系统调用的库

#include <pthread.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <stdio.h>

#include <stdlib.h>

#define LOOPS 100

//由于标准库中已删去了关于union semun的定义，此处给出规范的自定义

union semun {

int val; /\* value for SETVAL \*/

struct semid\_ds \*buf; /\* buffer for IPC\_STAT, IPC\_SET \*/

unsigned short \*array; /\* array for GETALL, SETALL \*/

struct seminfo \*\_\_buf; /\* buffer for IPC\_INFO \*/

};

int semid;

pthread\_t p1,p2;

int a;

//P、V操作的定义

void P(int semid,int index){

struct sembuf sem;

sem.sem\_num = index;

sem.sem\_op = -1;

sem.sem\_flg = 0; //操作标记：0或IPC\_NOWAIT等

semop(semid,&sem,1); //1:表示执行命令的个数

return;

}

void V(int semid,int index){

struct sembuf sem;

sem.sem\_num = index;

sem.sem\_op = 1;

sem.sem\_flg = 0;

semop(semid,&sem,1);

return;

}

//两个线程的行为定义

void \*subp2(void\*){

a = 0;

for(int n = 0;n != (LOOPS+1);n++){ //循环累加计算

P(semid,1);

a = a+n;

V(semid,0);

}

return 0;

}

void \*subp1(void\*){

for(int n = 0;n != (LOOPS+1);n++){

P(semid,0);

printf("subp1:a=%d\n",a); //打印结果

V(semid,1);

}

return 0;

}

int main(void){

union semun semopts;

int res;

//信号灯集定义

semid = semget(0,2,IPC\_CREAT|0666);

semopts.val = 0;

res = semctl(semid,0,SETVAL,semopts);

semopts.val = 1;

res = semctl(semid,1,SETVAL,semopts);

//线程创建

pthread\_create(&p1,NULL,subp1,NULL);

pthread\_create(&p2,NULL,subp2,NULL);

pthread\_join(p1,NULL);

pthread\_join(p2,NULL);

semctl(semid,0,IPC\_RMID,0);

return 0;

}

1. **实验结果**

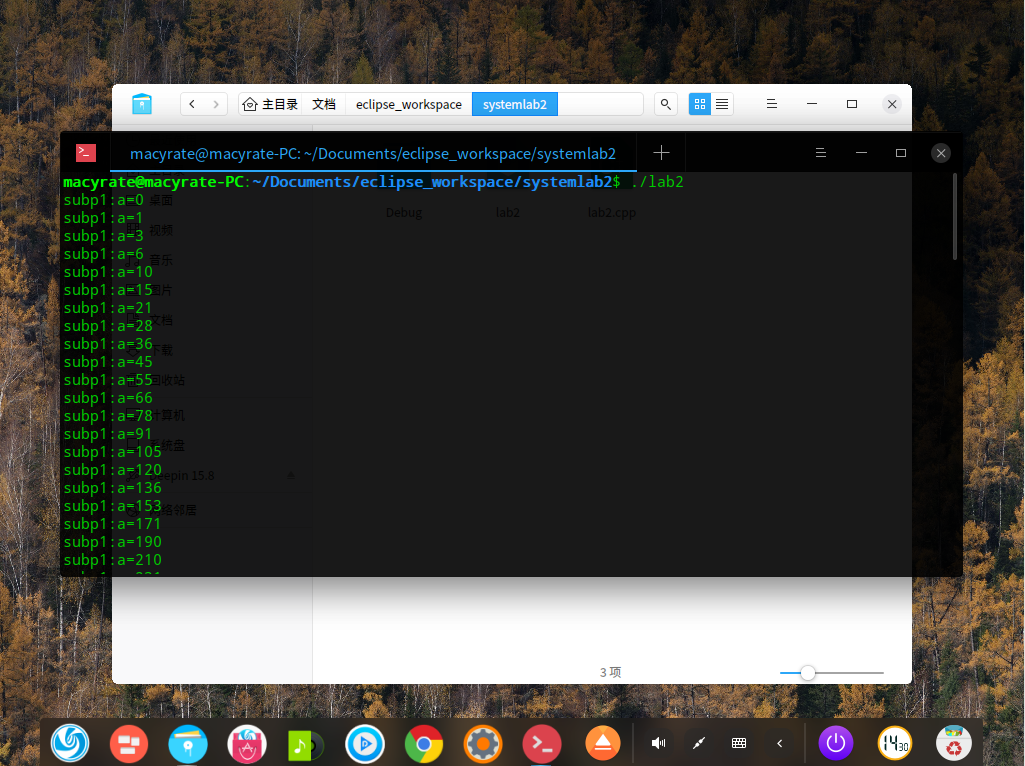


图2-1 程序执行中

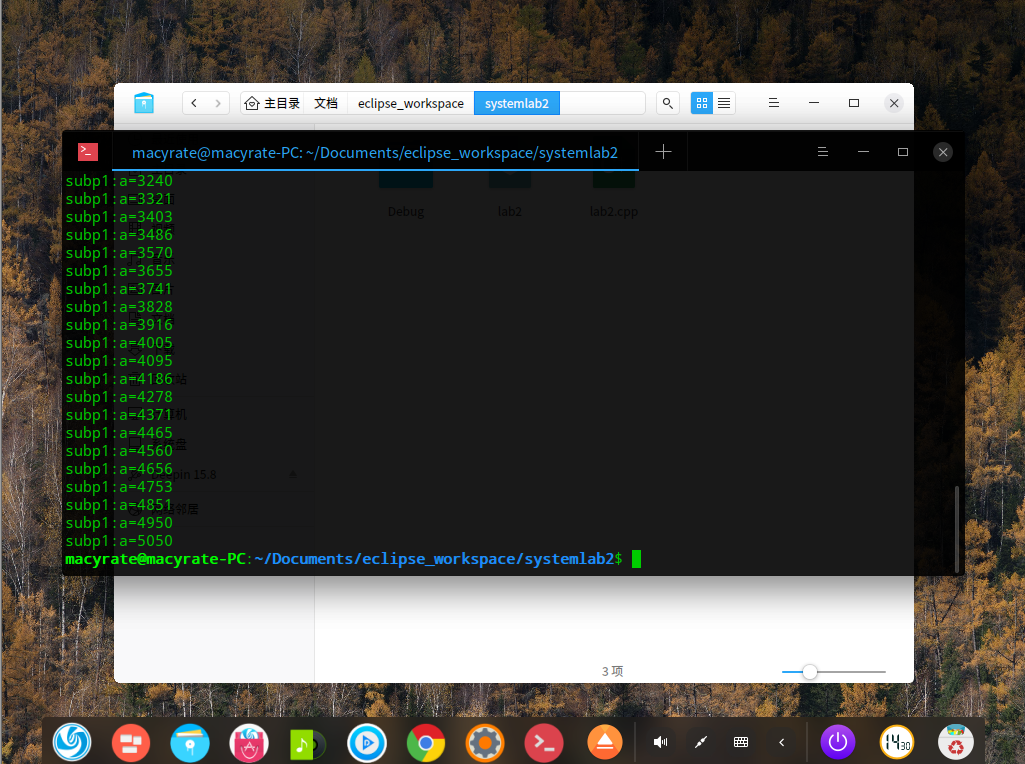


图2-2 程序执行完成

1. **实验心得**

这是操作系统的第二次实验，有了第一次实验的基础，这次所花的时间更少一些。

最大的难点，显然是在信号灯上。在教材中，关于信号灯定义的伪代码相当简洁，但到了Linux环境下实际应用时，才发现完全不是一回事，几个函数都是看得云里雾里，信号量集这个概念也是让我一阵茫然。

不过，其实只要一把梭，把每个函数都写上去就行了。只是，这个理解的过程依旧十分痛苦。

让我意外的是，P、V操作这种“原语”竟然不是库里自带的，也没有系统调用，竟然是需要自己写的……

另外，需要自己定义的union semun也是个坑，似乎是某种历史渊源所造成的，不过，姑且放在一边吧，总算是把这个看似简单的实验弄出来了。

最后，这次实验更让我稍稍体会到了“进程”与“线程”之间的不同之处。