**云南大学软件学院**

**实 验 报 告**

课程： 操作系统实验 任课教师： 储　星 实验指导教师： 谢 诚

姓名： 陈俊宏 学号： 20211060245 专业： 人工智能 日期： 2023-05-12 成绩：

**实验四 进程的同步与互斥**

1. **实验目的**

1、学习信号量的使用；

2、多线程程序的编程方法来完成进程的同步与互斥。

**二、实验内容**

1、 学习多线程程序如何编译；

2、 学习在 Linux 中，如何使用信号量；

3、 实现哲学家就餐问题，里面存在死锁现象，请用资源顺序分配法改写该程序，避免死锁。

**三、 实验要求**

1、本次实验不分组。

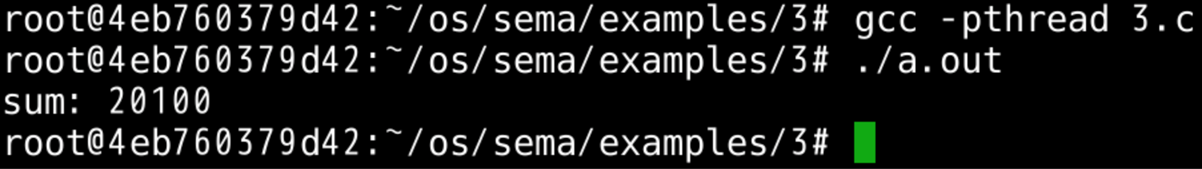
2、5月30日前提交纸质报告。

3、描述清楚实验要点，特别是多线程编程、信号量的应用以及资源顺序分配法。撰写实验总结。

**四、 实验过程**

1、多线程程序如何编译：

用 pthread 库实现的多线程程序，可以使用 gcc 来编译，且需加上 -pthread 参数：



#### pthread\_create 函数的各个参数的含义是什么

该函数的原型和各参数的含义如下：

1. int pthread\_create(pthread\_t \*tidp, const pthread\_attr\_t \*attr,
2. (void\*)(\*start\_rtn)(void\*), void \*arg);

第一个参数为指向线程标识符的指针； 第二个参数用来设置线程属性； 第三个参数是线程运行函数的起始地址； 最后一个参数是运行函数的参数。

#### 主线程结束时，子线程会被杀死吗

是的，如果主线程结束时，子线程还没有结束，那么子线程会被操作系统杀死。可以在主线程中使用 pthread\_join 来等待子线程结束。

2、在 Linux 中，如何使用信号量：

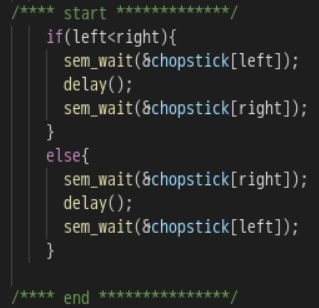
信号量的初始化、P 操作、V 操作分别对应于函数 sem\_init、sem\_wait 和 sem\_post 。一个例子程序如下：

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <pthread.h>
5. #include <semaphore.h>
6. double balance = 100;
7. sem\_t s;
8. void \*s1(void \*amount){
9. sem\_wait(&s);
10. double t = balance + \*(double\*)amount;
11. balance = t;
12. sem\_post(&s);
13. }
14. void s2(double amount){
15. sem\_wait(&s);
16. double t = balance - amount;
17. balance = t;
18. sem\_post(&s);
19. }
20. void s3(){
21. printf("new balance: %f\n", balance);
22. }
23. int main(){
24. int res;
25. pthread\_t t1;
26. void \*thread\_result;
27. double b = 100;
28. sem\_init(&s, 0, 1);
29. res = pthread\_create(&t1, NULL, s1, (void\*) &b);
30. if (res != 0){
31. perror("failed to create thread");
32. exit(1);
33. }
34. s2(50);
35. res = pthread\_join(t1, &thread\_result);
36. if (res != 0){
37. perror("failed to join thread");
38. exit(2);
39. }
40. s3();
41. return 0;
42. }

3、资源顺序分配法

#### 什么是资源顺序分配法

资源顺序分配法的基本思想是：给每类资源编号，进程只能按序号由小到大的顺序申请资源，若不满足则拒绝分配。



**五、实验总结**

答：

本次实验主要学习了多线程程序的编译方法以及在 Linux 中如何使用信号量。通过实现哲学家就餐问题，深入了解了进程的同步和互斥操作，以及如何使用信号量来实现它们。同时，还学习了资源顺序分配法，通过改变资源的获取顺序，来避免死锁的发生。

通过本次实验，深入了解了多线程编程和信号量的应用，以及如何避免死锁的发生。同时，我们也学习了如何编写高效、正确的多线程程序，以提高程序的性能和可靠性。

总的来说，本次实验让我受益匪浅，不仅提高了编程技能，还让我更好地理解了进程同步和互斥的概念。