电子科技大学

University of Electronic Science and Technology of China

Linux操作系统编程

实验六 | 线程同步实验

主讲老师：杨珊

实验六 | 线程同步实验目的

目的一：了解临界资源、哲学家进餐问题、死锁等经典的**操作系统同步互斥问题**的基本概念

目的二：掌握利用**互斥量机制**实现多线程对临界资源**互斥访问**的**方法**

目的三：掌握互斥量**非阻塞加锁**的操作方法

目的三：掌握在保证对临界资源互斥访问的基础上通过**让权等待**的方式**预防死锁**的编程思想

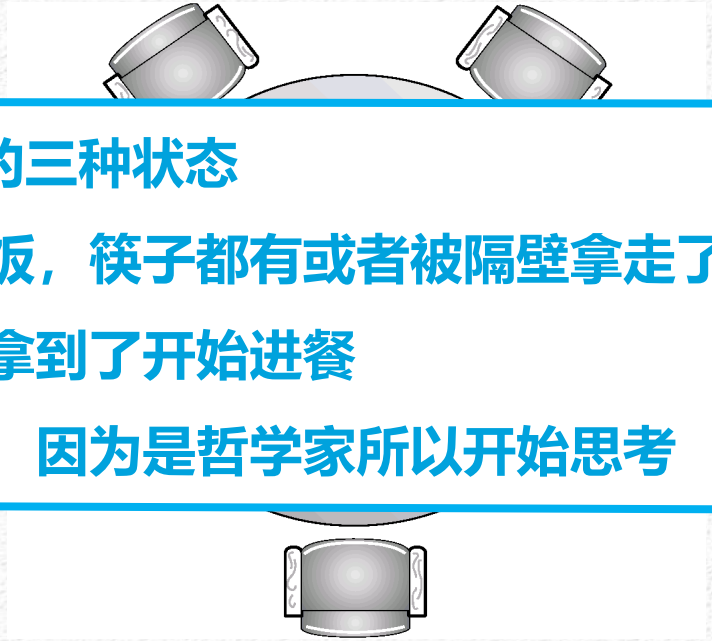
实验六 | 线程同步实验目的

问题描述：五个哲学家围坐在一张圆桌，分别坐在周围的五把椅子上，在圆桌上有五个碗和五只筷子，他们的生活方式是交替地进行思考和进餐。平时，一个哲学家左右最靠近他的两根筷子，只有在他拿起两根筷子时才能进餐。进餐完毕，放下筷子继续思考。

哲学家不会谦让，吃饭和思考都没有固定模式

只有五个碗和五只筷子，能同时吃上饭的只有两位哲学家

考虑将哲学家模拟为同一例程的五个线程，筷子是共享的资源



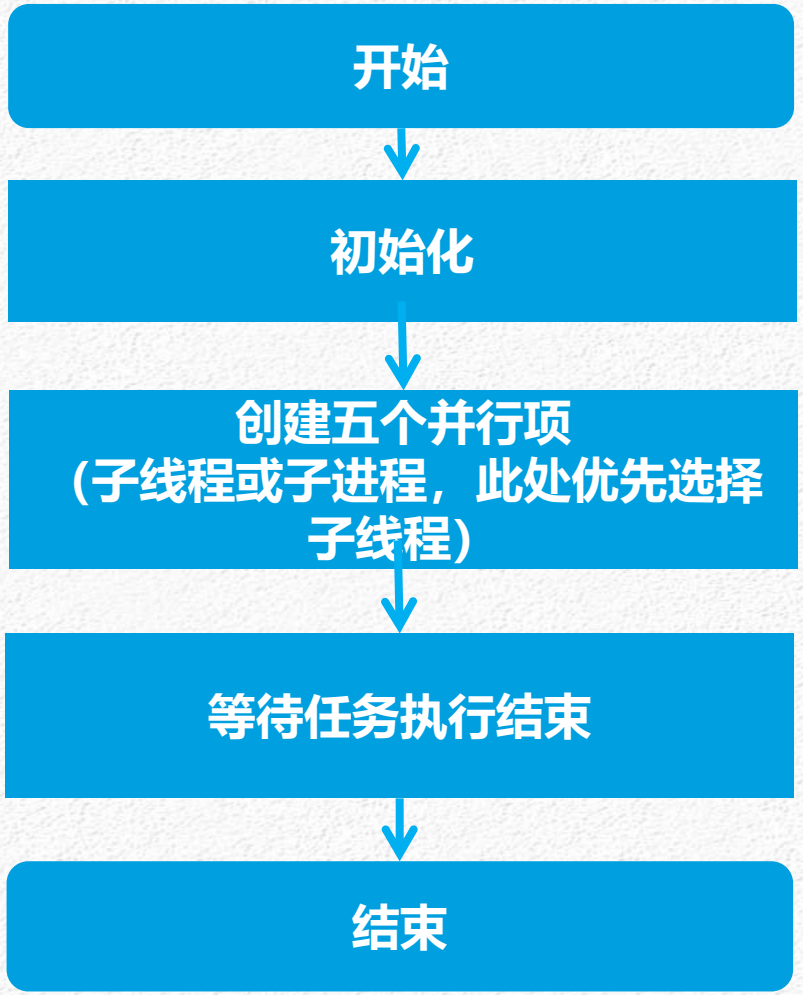
哲学家的三种状态

1. 想吃饭，筷子都有或者被隔壁拿走了
2. 筷子拿到了开始进餐
3. 饱了，因为是哲学家所以开始思考

这段描述中有哪些实现的限制

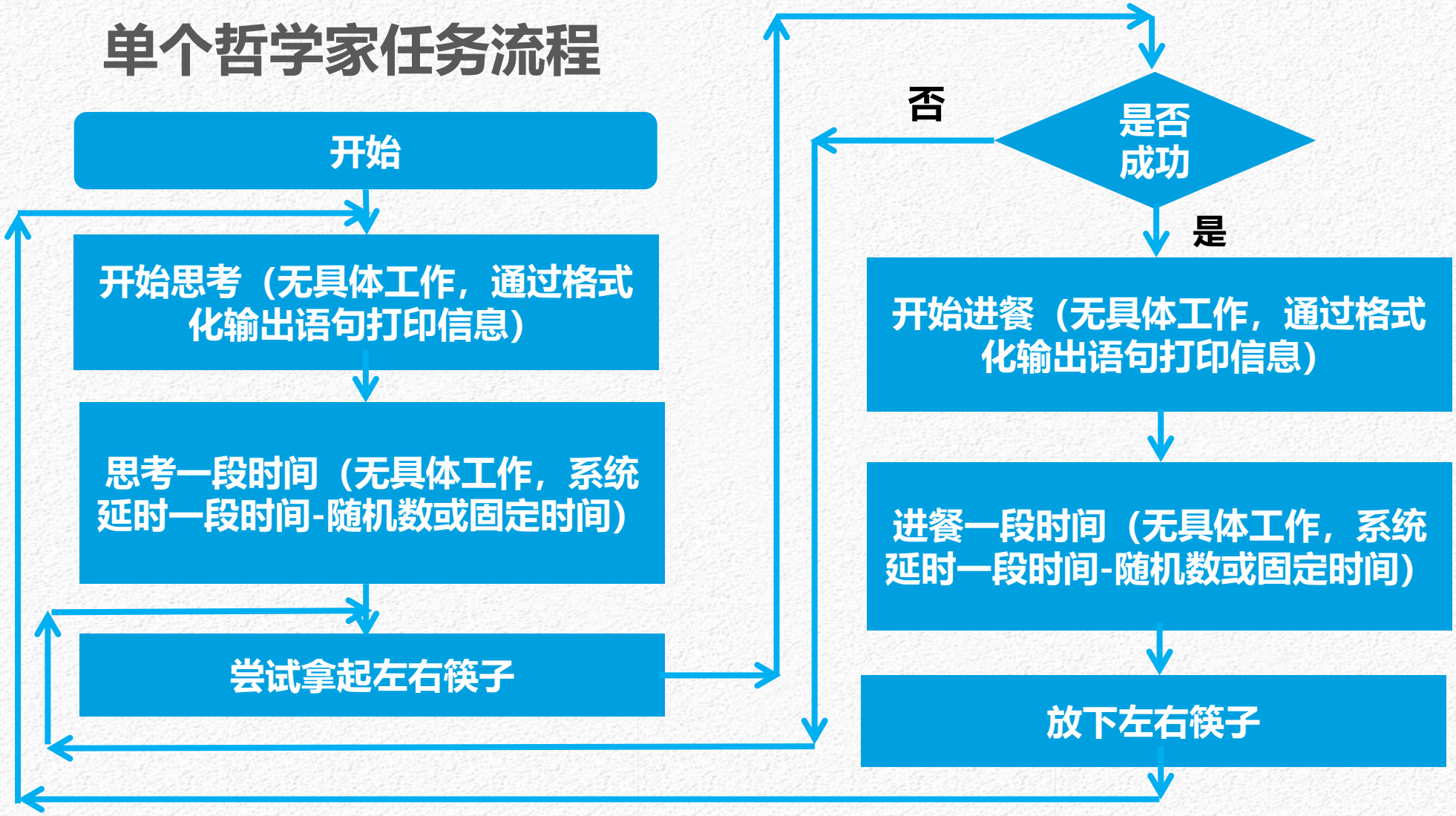
实验六 | 进餐问题主函数流程

main函数



实验六 | 进餐问题主函数流程

单个哲学家任务流程



实验六 | 程序实现示例

哲学家任务用线程实现，并对哲学家和筷子进行编号，并通过参数传递各种编号

```
void philosopher(int number) {//线程例程
    while(1){
     printf(“philosopher %d is thinking\n”,number);
     sleep(2);
     takechopstick(number);
     takechopstick((number+1)%5);
     printf(“philosopher %d is eating\n”,number);
     sleep(2);
     putchopstick(number);
     putchopstick((number+1)%5);}}
```

实验六 | 互斥访问的实现机制

筷子（临界资源），各个哲学家（不同任务）对其进行**互斥访问**

临界资源的访问与释放，Linux中有多种实现机制：

- ✓ 互斥量（加锁，解锁）
- ✓ XSI信号量集（P，V操作）
- ✓ POSIX信号量（P，V操作）

实验六 | 互斥量使用步骤

互斥量（加锁，解锁）的使用步骤

1. 定义一个互斥量变量（ pthread_mutex_t 类型)
2. 初始化互斥量
3. 访问临界资源前对互斥量加锁
4. 访问临界资源后对互斥量解锁
5. 销毁互斥量

实验六 | 互斥量初始化

头文件：pthread.h

定义和静态初始化互斥量变量：

pthread_mutex_t mlock = PTHREAD_MUTEX_INITIALIZER

动态初始化：

函数原型：int pthread_mutex_init(pthread_mutex_t *mutex,
const pthread_mutexattr_t *attr);

参数与返回值：mutex：互斥变量指针

attr：设置互斥量的属性，通常采用默认属性，将attr设为NULL。成功返回0，出错返回错误码

实验六 | 互斥量销毁

头文件：pthread.h

互斥量销毁：int pthread_mutex_destroy(pthread_mutex_t *mutex);

作用：互斥量在使用完毕后，必须要对互斥量进行销毁，以释放资源

参数与返回值

- ✓ mutex：互斥量变量指针
- ✓ 成功返回0，出错返回错误码

实验六 | 互斥量的加锁和解锁

头文件：pthread.h

函数：int pthread_mutex_lock(pthread_mutex_t *mutex);

作用：对共享资源访问之前需要加锁互斥量

当调用pthread_mutex_lock时，若互斥量已被加锁，则调用线程将被阻塞（哲学家线程进入阻塞状态，等待操作系统唤醒）

函数：int pthread_mutex_unlock(pthread_mutex_t *mutex);

作用：对共享资源访问之前需要加锁互斥量

两个函数的返回值：成功返回0，出错返回错误码

拿筷子操作导致死锁的原因

- ✓ 筷子需要互斥访问（临界资源）
- ✓ 任务无法抢占其他任务已经拿起的筷子
- ✓ 任务拿起筷子直到进餐完毕后才放下（长时间占有资源）
- ✓ 圆桌导致可能出现循环等待

实验六 | 预防死锁的实现方法一

方法一：破坏请求和保持条件

- ✓ 任务如果无法同时拿起两支筷子，则放下已经拿起的筷子，等待一段时间再尝试
- ✓ 利用POSIX API中的非阻塞操作实现能否拿起筷子的判断
  - pthread_mutex_trylock操作
  - sem_trywait操作
- ✓ XSI信号量集P操作设置IPC_NOWAIT参数

实验六 | 互斥量初始化

头文件：pthread.h

函数：int pthread_mutex_trylock(pthread_mutex_t *mutex);

作用：

- ✓ 调用该函数时，若互斥量未加锁，则锁住该互斥量，返回0；
- ✓ 若互斥量已加锁，则函数直接返回失败（哲学家线程进入代码安排的下一步活动，思考或者饥饿）

实验六 | 预防死锁的实现方法二

方法二：破坏“循环等待条件”

- ✓ 对哲学家编号，奇偶号哲学家拿起筷子的顺序不同
- ✓ 创建一个服务生任务，哲学家拿起筷子时向该任务发起申请，由该任务根据当前筷子的分配情况判断系统是否由安全状态向不安全状态转换，从而允许或拒绝该次申请。

实验六 | 实验内容

利用POSIX API在Linux系统上编写应用程序，通过**多线程**和**互斥量机制**实现哲学家进餐问题（哲学家的数量可以通过简单的配置进行修改）

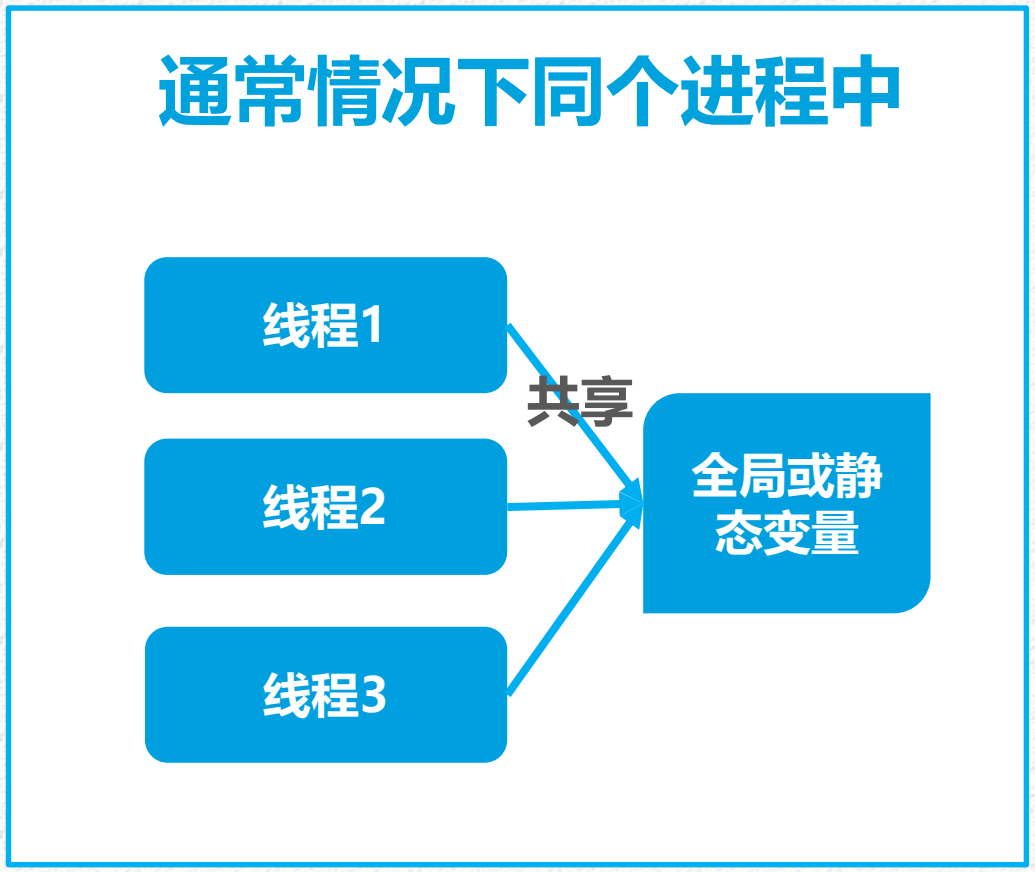
1. 首先通过**阻塞加锁**的操作方式实现哲学家之间（线程）对筷子（临界资源）的互斥访问，观察程序运行一段时间以后会出现的**死锁状态**。如果未出现死锁状态的可以通过**修改哲学家数量以及修改延时设置**来增大出现死锁的机率。
2. 将互斥量的加锁操作从阻塞方式修改为**非阻塞方式**，通过让权等待的思想预防死锁状态的出现。

实验六 | 实验内容扩展

基本功能实现的基础上鼓励尝试多种思路预防死锁，包括并不限于以下思路：

- ✓ 对哲学家编号，**奇偶号哲学家**拿起筷子**顺序不同**
- ✓ 再创建一个任务，哲学家拿起筷子时向该任务发起申请，由**该任务对当前筷子的分配情况进行判断**，判定系统是否由安全状态向不安全状态转换，从而允许或拒绝该次申请

实验六 |编程技巧| 线程局部存储



线程局部存储:

对于某个全局变量想做出

本线程范围内的修改和读取

例： errorno

实验六 |编程技巧| 线程局部存储

```
static __thread int buf[100];
```

- ✓ __thread说明符让线程拥有自己对该全局变量的拷贝
- ✓终止时自动释放该存储
- ✓例：每个线程有自己的errorno拷贝，防止被其他线程干扰

电子科技大学

University of Electronic Science and Technology of China

Linux操作系统编程

感谢观看

主讲老师：杨珊