2020春

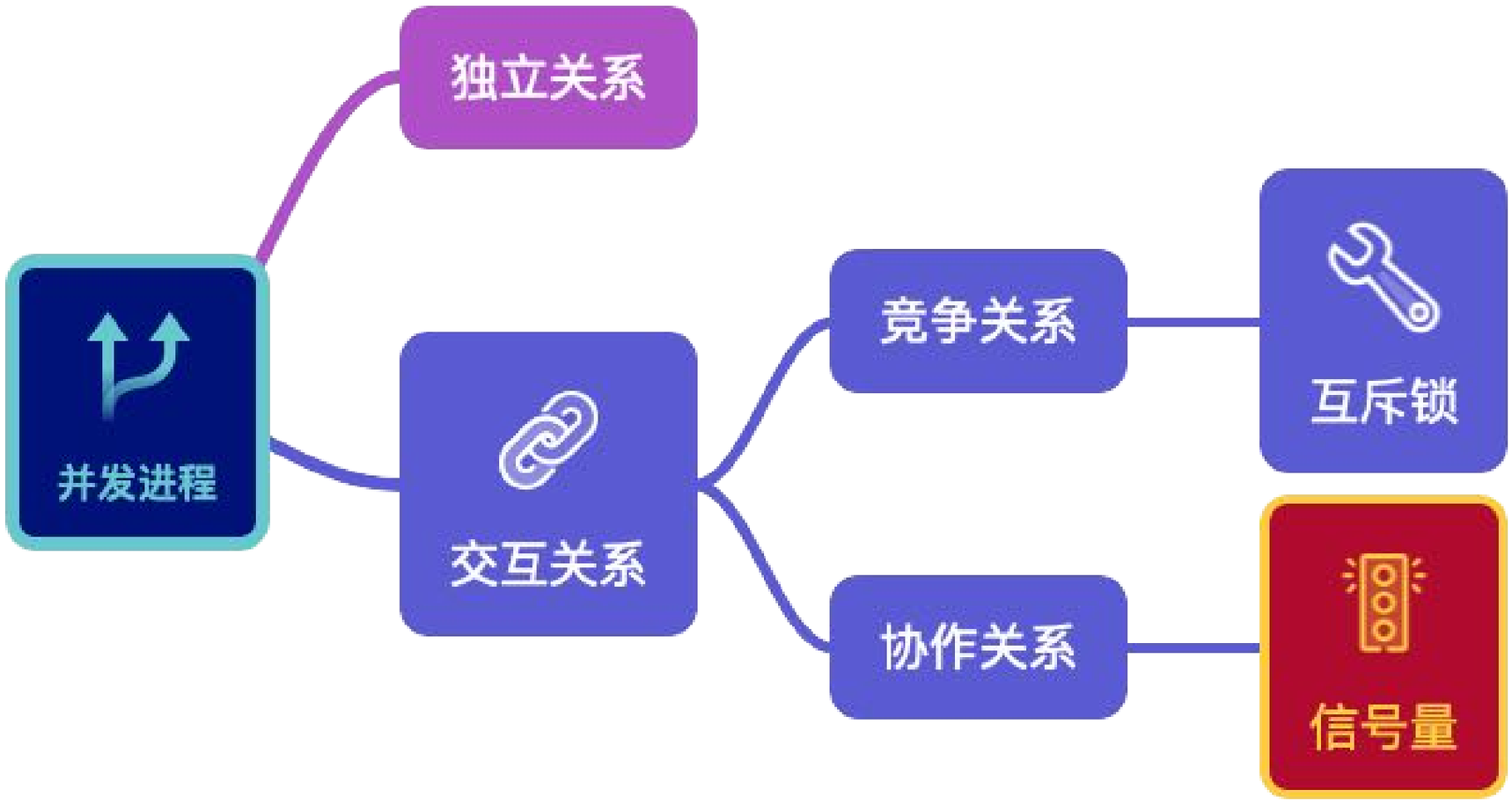
LINUX OPERATING SYSTEM

YANG

LINUX操作系统（双语）

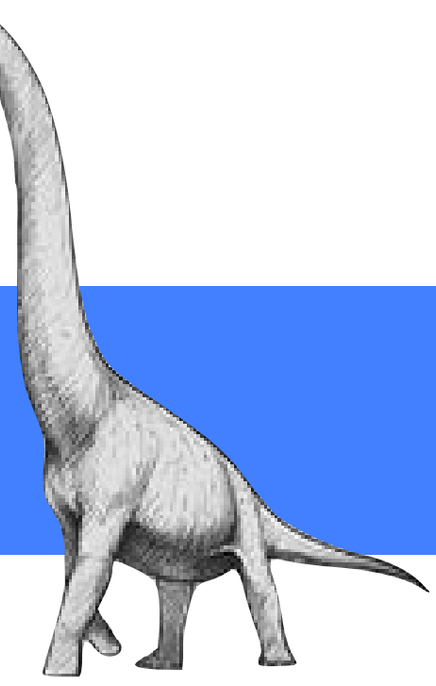
双语课→课件内容中英混排

REVIEW



Lecture 9

Semaphores I



本讲内容

- 信号量与PV操作
- 信号量的使用

信号量与PV操作

信号量

- 信号量（Semaphore）是一种比互斥锁更强大的同步工具，它可以提供更高级的方法来同步并发进程。
  - 1965年由荷兰学者Dijkstra提出

- A semaphore S is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: P (*proberen in Dutch*) and V(*verhogen in Dutch*).
  - P： wait() operation
  - V： signal() operation

信号量的实现

```
P(s){
   while(s<=0)
      do nothing;

   s--;
}

V(s){
   s++;
}
```

s value->	s<=0	s=1	s>1
P(s)			
V(s)			

信号量的实现

```
P(s){
   while(s<=0)
      do nothing;

   s--;
}

V(s){
   s++;
}
```

no.	operations	s value
1	initialize	
2	A:P(s)	
3	A:V(s)	
4	B:P(s)	
5	C:P(s)	
6	D:P(s)	
7	B:V(s)	
8	C:V(s)	
9	D:V(s)	

s = //init

信号量的使用

BINARY SEMAPHORE

- 顾名思义，二值信号量的值只能是0或1，通常将其初始化为1，用于实现互斥锁的功能。

```
semaphore mutex = 1;
process pi{
    P(mutex);

    critical section

    V(mutex);

}
```

BINARY SEMAPHORE

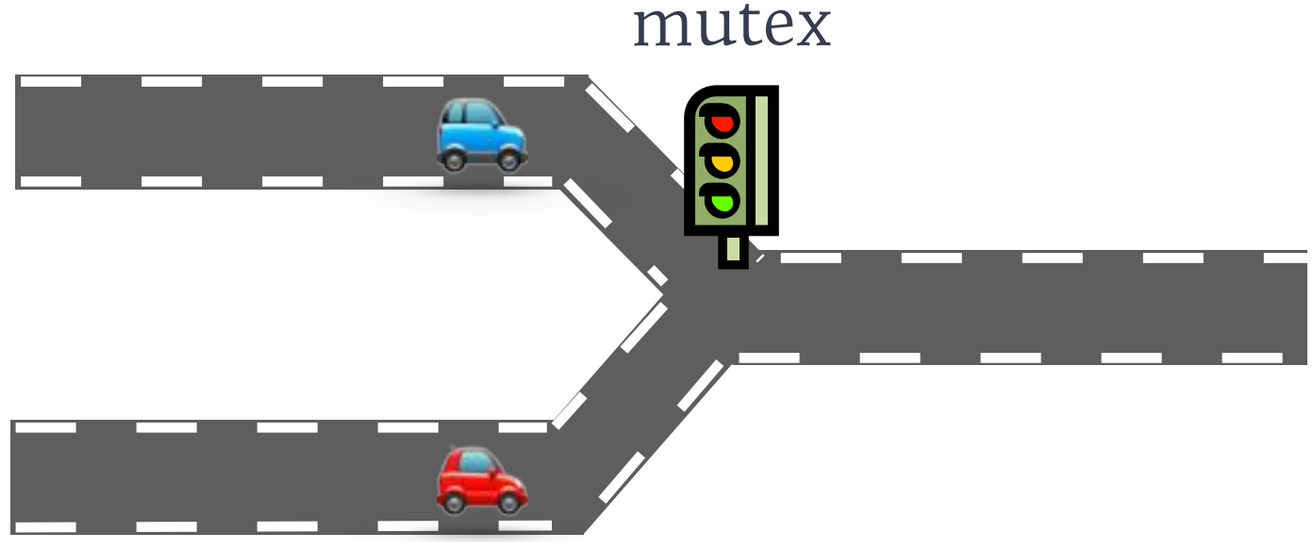
- 顾名思义，二值信号量的值只能是0或1，通常将其初始化为1，用于实现互斥锁的功能。

```
semaphore mutex = 1;
process pi{
    P(mutex);

    critical section

    V(mutex);

}
```



COUNTING SEMAPHORE

- 一般信号量的取值可以是任意数值，用于控制并发进程对共享资源的访问。

```
semaphore road = 2;
process Car_i{
    P(road);

    pass the fork
    in the road.

    V(road);

}
```

COUNTING SEMAPHORE

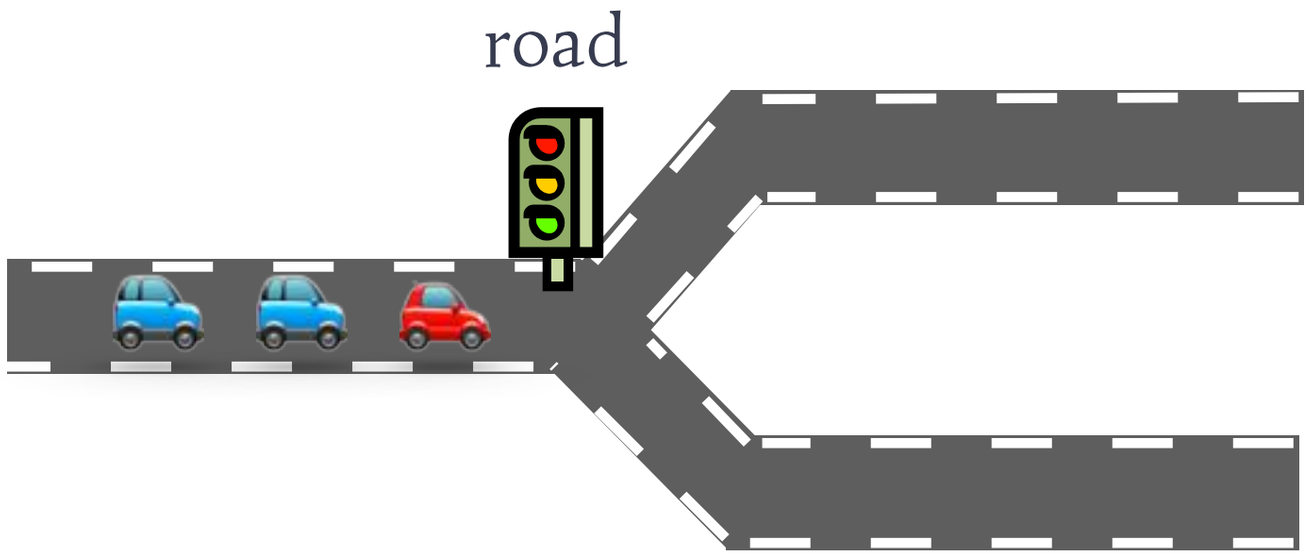
- 一般信号量的取值可以是任意数值，用于控制并发进程对共享资源的访问。

```
semaphore road = 2;
process Car_i{
    P(road);

    pass the fork
    in the road.

    V(road);

}
```



TAKE A BREAK

下期预告

- 本期视频会分成理论和实践2个部分
- 最近这段时间B站审核时间超过6小时，请大家耐心等待
- 下次直播时间：3月6日 上午9:30
- 课程内容
  - Lecture 9 Semaphores II

Lecture 9

The End

实践4 信号量

解决订票终端的临界区管理

```c
int ticketAmount = 2; //Global Variable

void* ticketAgent(void* arg){
    int t = ticketAmount;
    if (t > 0)
    {
        printf("One ticket sold!\n");
        t--;
    }else{
        printf("Ticket sold out!!\n");
    }
    ticketAmount = t;
    pthread_exit(0);
}
```

- 要使用信号量，请先包含头文件<semaphore.h>
- sem_t：信号量的数据类型
- int sem_init(sem_t *sem, int pshared, unsigned int val);
  - 该函数第一个参数为信号量指针，第二个参数为信号量类型（一般设置为0），第三个为信号量初始值。第二个参数pshared为0时，该进程内所有线程可用，不为0时不同进程间可用。
- int sem_wait(sem_t *sem);
  - 该函数申请一个信号量，当前无可用信号量则等待，有可用信号量时占用一个信号量，对信号量的值减1。
- int sem_post(sem_t *sem);
  - 该函数释放一个信号量，信号量的值加1。
- int sem_destory(sem_t *sem);
  - 该函数销毁信号量。