Linux自旋锁（ARM版本）

锁本身是个32位整数，高16位表示叫号，低16位表示当前服务号

上锁：

锁内存总线执行：

锁变量读入临时变量lockval

lockval高16位加1后写回锁变量（lockval不变）

如果失败再回去试一次

while(lockval高16位 != lockval低16位)

锁变量的低16位读入lockval低16位

解锁：

锁变量低16位加1

使用信号量的基本程序模型

系统调用，触发内核中的wait

系统调用，触发内核中的signal

进程A：

id=创建信号量(预定义Key)

…

请求资源(id)

临界区程序（可能耗时长）

释放资源(id)

…

进程B：

id=获取信号量(预定义Key)

…

请求资源(id)

临界区程序（可能耗时长）

释放资源(id)

…

进程C：

id=获取信号量(预定义Key)

…

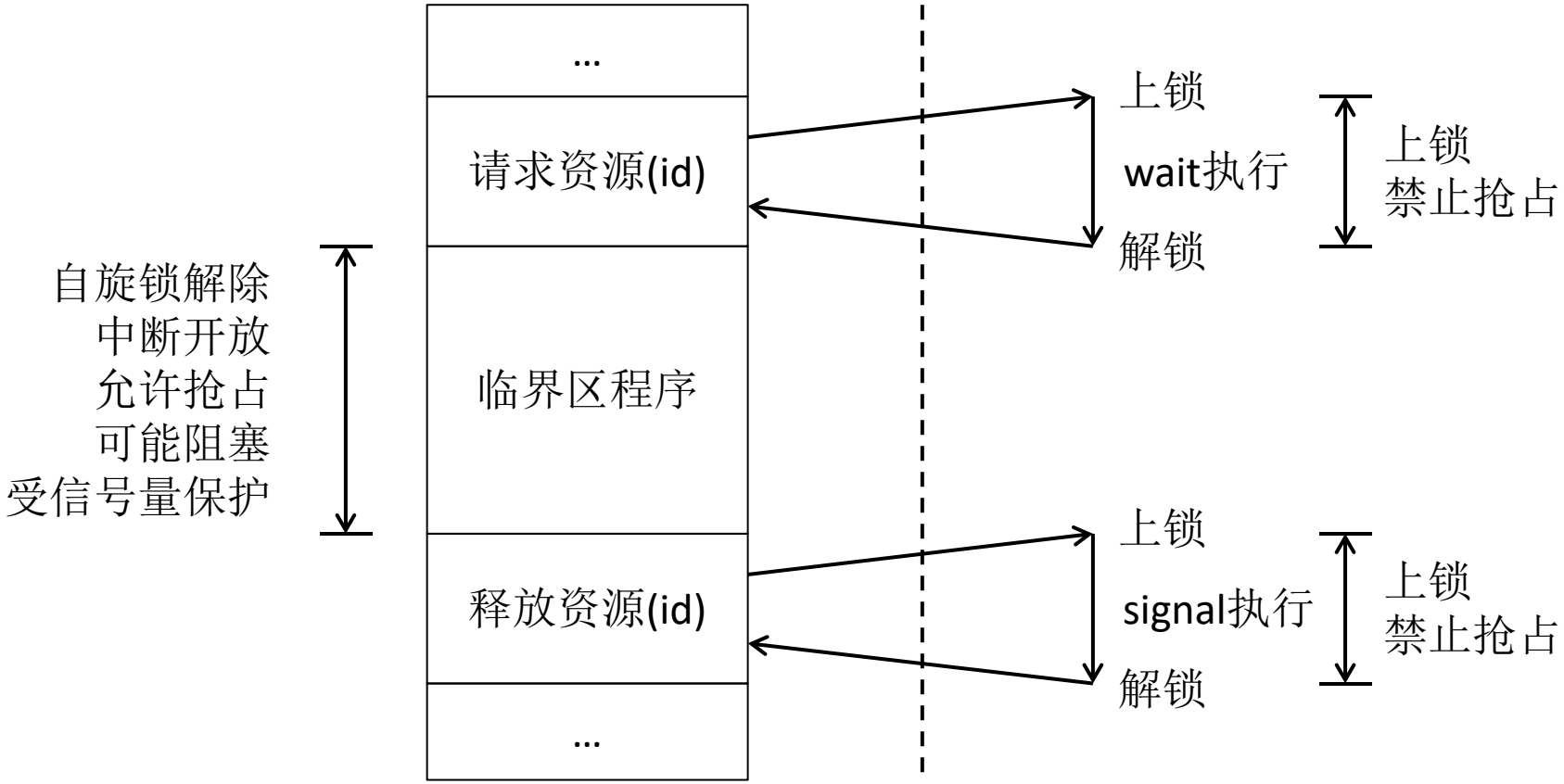
请求资源(id)

临界区程序（可能耗时长）

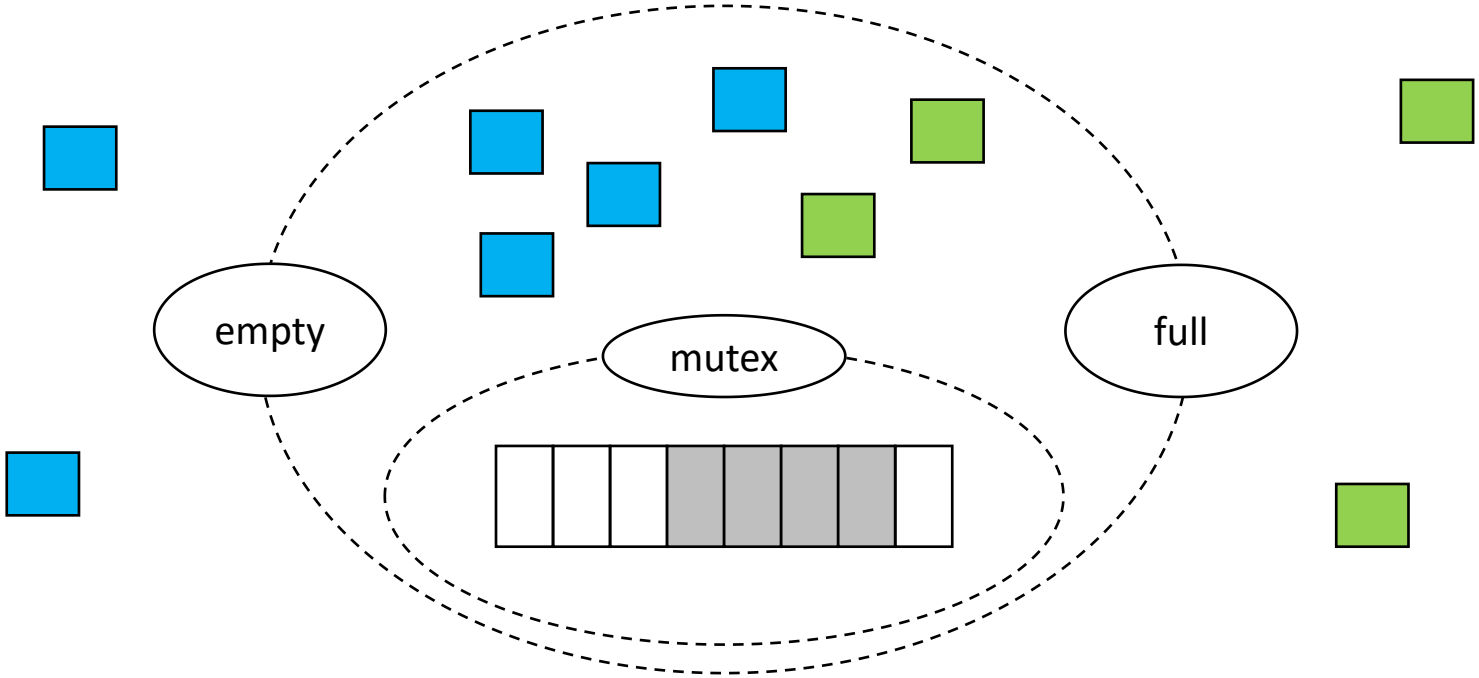
释放资源(id)

…

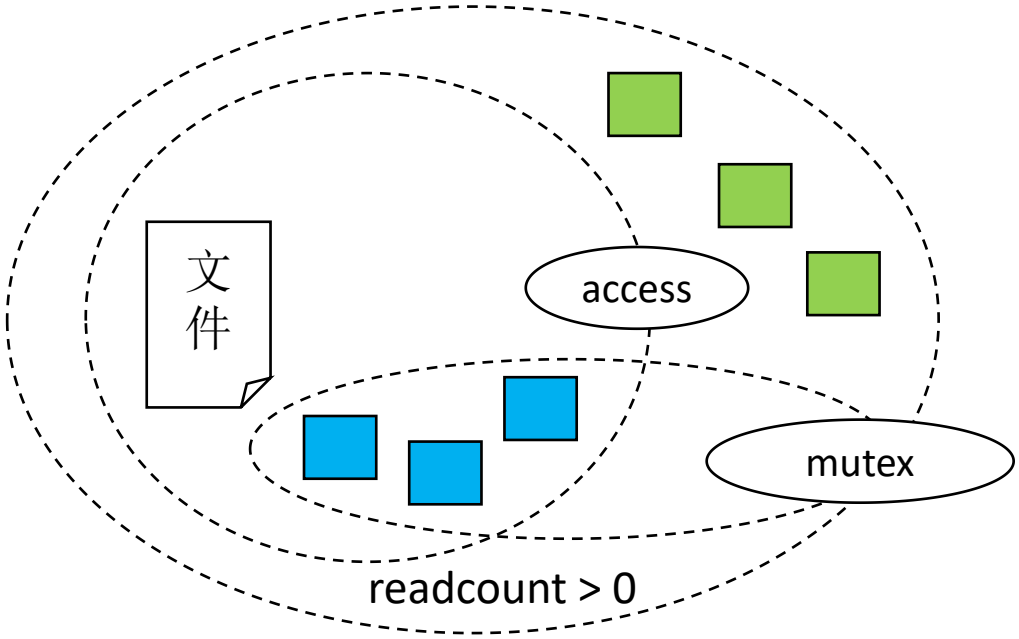
使用信号量的基本程序模型



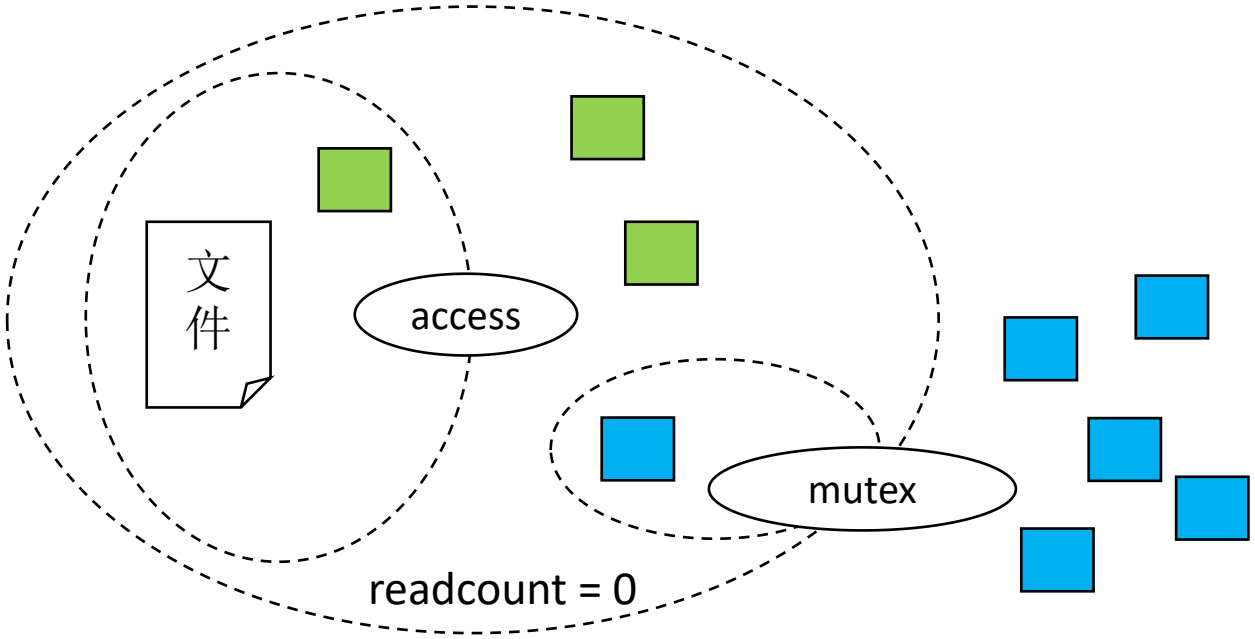
生产者-消费者问题



读者-写者问题



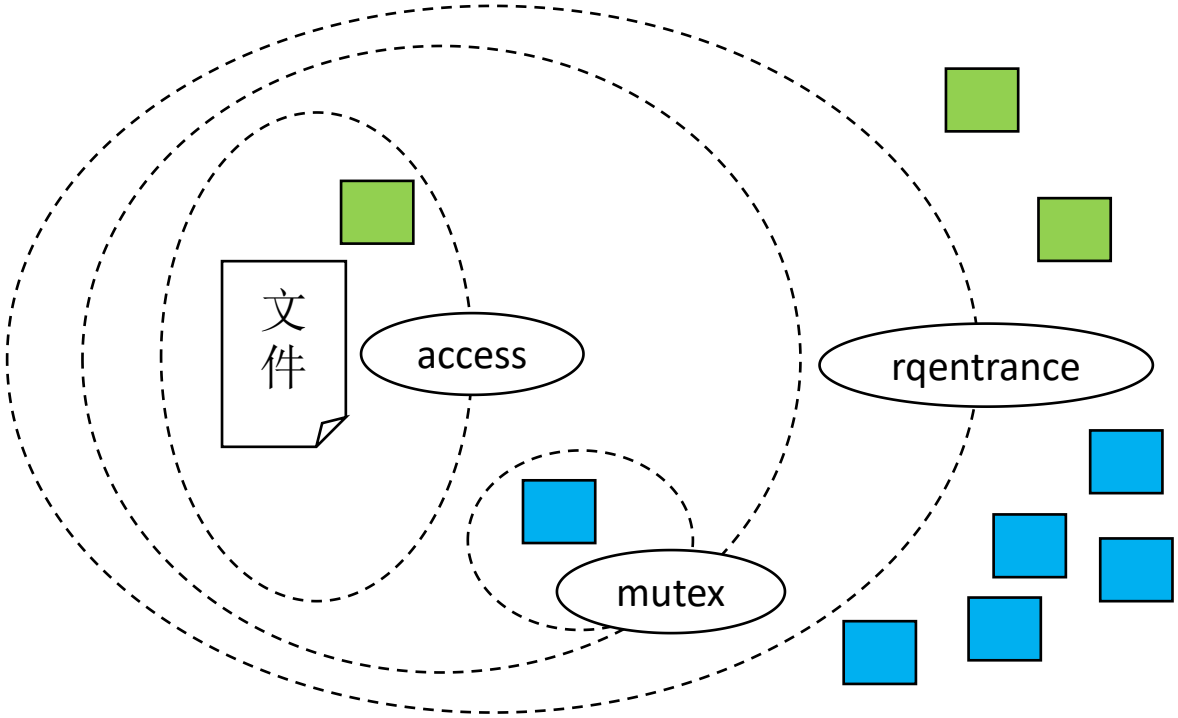
读者-写者问题



读者-写者问题

```
wait(rqentrance);
wait(access);
// writing
signal(access);
signal(rqentrance);
```

```
wait(rqentrance);
wait(mutex);
readcount++;
if (readcount==1)
    wait(access);
signal(mutex);
signal(rqentrance);
// reading
wait(mutex);
readcount--;
if (readcount==0)
    signal(access);
signal(mutex);
```



管程：生产者-消费者

```
Monitor PC;
int nextin,nextout,count;
char buffer[n];
condition notfull，notempty;
append(char x) {
if (count==n) Cwait(notfull);
    buffer[nextin]=x;
    nextin=(nextin+1) % n;
    count++;
    Csignal(notempty);
}
take(char x) {
    if (count==0) Cwait(notempty);
    x=buffer[nextout];
    nextout=(nextout+1) % n;
    count--;
    Csignal(notfull);
}
{
    count=0; nextin=0; nextout=0;
}
```

```
main() {
   cobegin
     producer();
     consumer();
   coend
}
producer() {
    char x;
    while(true) {
        produce(x);
        append(x);
    }
}
void consumer() {
    char x;
    while(true) {
        take(x);
        consume(x);
    }
}
```

管程：哲学家进餐

```
Monitor dp;
enum {thinking, hungry, eating} state[5];
condition self[5];
int i;
{
  for(i=0;i<=4;i++) state[i]=thinking;
}
void pickup(int i) {
  state[i]=hungry;
  test(i);
  /*左右有人进餐则阻塞*/
  if (state[i]!=eating)
    self[i].wait;
}
void putdown(int i) {
  state[i]=thinking;
  /*唤醒左右*/
  test((i+4)% 5));
  test((i+1)% 5);
}
```

```
void test(int k) {
  if (state[(k+4)%5]!=eating) &&
     (state[k]==hungry) &&
     (state[(k+1)%5]!=eating) {
    state[k]=eating;
    self[k].signal);
  }
}
```

```
philosopher(int i):
while (true) {
     Thinking;
     dp.pickup(i);
     Eating;
     dp.putdown(i);
}
```