失败的

DI

while (LOCK=T);

LOCK=T;

EI

CS

LOCK=F;

要注意的一点就是会不会无限while

总而言之我们要基本做到

1.get lock();CS,release lock(); 基本结构

2.在while无限循环的时候，我们要EI,不然会卡死

getLock(){

DI.

While (lock=T){

E.I.

//这一段创造了一个跳出当前循环并接受当前Lock的机会

D.I.

}

LOCK=T

EI

}

RealeaseLock(){

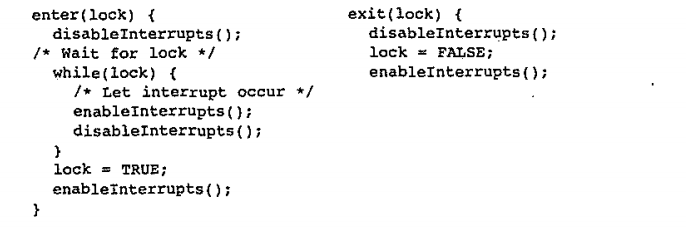
DI

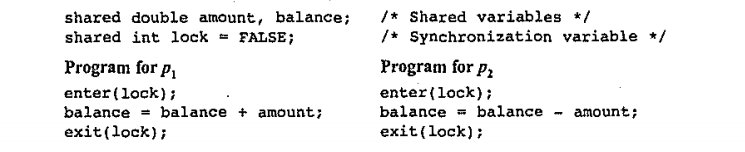
lock=F

EI

}

release有没有DIEI，区别在于依然能够运行，但速度变慢，有可能在LOCK=F之前interrupt,白跑了一次。





PS1

lock

c,s,1

c,s,2

c,s,3

unlock

PS2同样

当PS1在CS1的时候，PS2应该有cs2 cs3的权利

PS1

lock1

add node

lock2

size++

unlock2

unlock1

PS2

lock2

size—

lock1

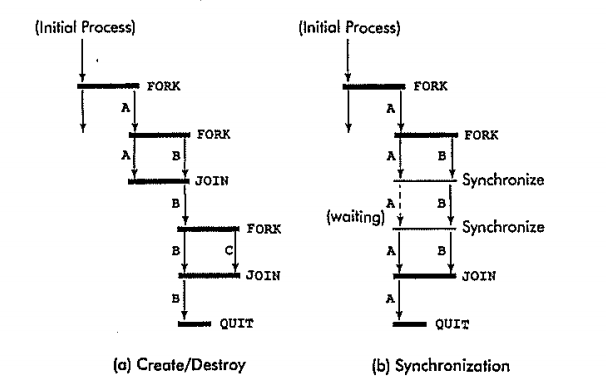
delete node

unlock1

unlock2

这一套的逻辑在于，假设有个length为4的node list，我们Lock1 add node，这时实际有5个node，还没size++，然后Interrupt,size--，这样就是length为3，实际5个node，而且永远无法继续，dead lock,这的size—与对node的操作就是我们的CS

解决方法就是synchronization



左边的creat/destroy，当A与B同时CS的时候，thread A die

右边的是synchronization，A与B有冲突的时候，A会waiting，先进性B，再进行A，

synchronization是一种思想，对于同一个Process,只有一个thread可以在同一时间内改变他，其他的thread要wait

共用一个Lock

初始为lock

PS1

write(x)

unlock

lock

read(y)

PS2

lock

read(x)

write(y)

unlock

我们们想要的顺序是writex read x write y read y

但是实际上我们可以writex ready，卡死

有时候我们需要多个lock来给一个顺序

给你两个lock,lock1,lock2，初始都是lock

PS1

write(x)

unlock1

lock2

read(y)

PS2

lock1

read(X)

write(Y)

unlock2

semaphore



P就是当==0，wait，或者等于1，-1

Vs就是加1,

最简单的例子

sim s=1

ps1

P(S)

cs

V(S)

PS2同理

ps1 start

s=0, interrupt ps2,ps2 wait till s≠0

vs

ps2 p(s)继续

sim s=2就不行

sim s1=0,sim s2=0

PS1

write(X)

V(S1)

p(S2)

read(Y)

ps2

p(s1)

read(X)

write(Y)

V(S2)

如果一开始=0.就VP, 如果一开始=1，就PV，可以看出PV会制定一个顺序

V操作默认是无法interrupt的，直到我们成功+1

是不是synchronization这个问题只有一个判断条件，要wait，就synchronization，不wait，就不synchronization，前面的semaphore就是要wait的

以上都是binary semaphore，

s=4这种叫做 general semaphore

GENERAL SEMAPHORE

bound-buffer

左边有个process叫producer， 右边有个consumer process

中间是buffer(一个长度有限的array，缓存区)

总体思路，，对于producer来说，创造的时候必须在buffer有empty,对于consumer来说，必须有东西才能consume

sim empty=size//5

sim full=0

Produce

P(empty)

<produce>

V(full)

P(full)

<consume>

V(empty)

注意这个解是有问题的

例如我们用buffer传递strinng

想Produce I love here， 但只写出I love her，这时interrupt， wife pick，离婚

对buffer的操作实际上还是critical section,

我们要避免同时对CS操作

完美解，用另外一个semaphore mutex来保证对critical section的控制

sim empty=5,full=0,mutex=1

Prod

P(Empty)

P(mutex)

<produce>

V(mutex)

V(Full)

Cons

P(full)

P(mutex)

<consume>

V(mutex)

v(empty)

empty=0.full=5,在consume这里interrupt P(EMPTY),因为-0，会wait，所以不会打破0的界限

buffer问题，用两个semaphore，初始的空为size，另外一个0，然后用Mutex控制buffer操作

制造就p empty V full，消耗empty，增加

但还有一个问题

买卖车问题

假设produce这个过程要3个month

而thread是随机的，有可能连造两辆，那么cons就要等六个月，实际上他只需要等三个月

解决方法，把buffer看作linked list

produce process

P(EMPTY)

P(mutex)

get node

V(mutex)

produce

P(mutex)

put node//塞回写好的node

v(mutex)

V(full)

consume process

P(full)

P(mutex)

get node

v(mutex)

<consume>

P(mutex)

put node//塞回消耗的node

V(mutex)

V(empty)

‘