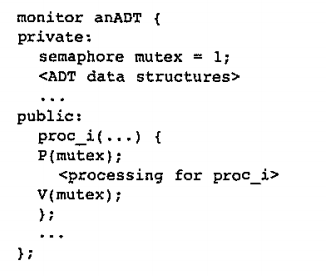
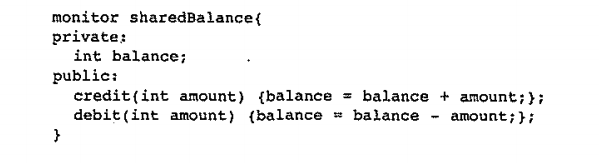
high level synchronization之Monitor



monitor会内置一个Mutex，然后里面有很多个process，同一时间只会进行一个process，用mutex来synrchronize



balance例子，同一时间只能进行credit或者debit

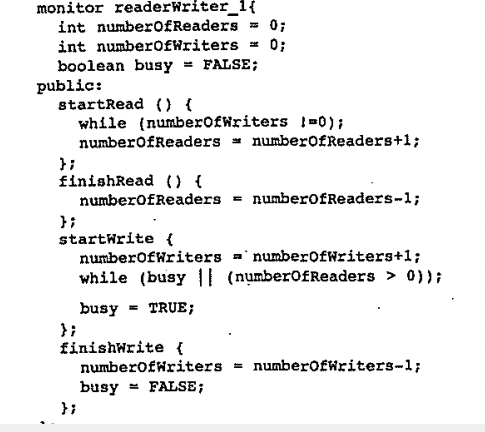
用Monitor来解决reader writer的问题

一般结构是

BeforeRead()

<Read>

AfterRead()

.. 

缺陷：

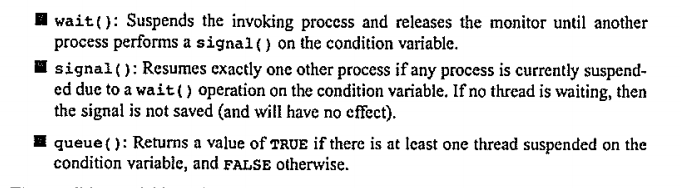
startWrite完美进行，这时finishWrite还没开始

我们进行startRead()

会停在while(numberOfWriters!=0)上

这是因为hidden semaphore的关系，就只能startRead没法finishWrite,没人继续

所以我们要抛弃semaphore，要引入condition来构建monitor



Condition变量的三个操作，

wait（）:延缓call wait的process，释放monitor，直到另一个process 用这个condition variable call signal()，

signal()： 明确指定一个process继续，如果这个process被wait()操作延缓了。如果没有thread在wait，那么这个signal不生效，跳过

queue()：如果这个condition variabe有至少一个延缓的process，return true，不然就是fase

延缓：suspend

用了condition的READERWRITER问题

Boolean busy=F

Cond OKTOREAD,OKTOWRITE

int RC=0;

startRead(){

if (busy||OKTOWRITE.queue==true)//有人正在写，或者想写,wait

OKtoREAD.wait;

RC++; //如果没有人在排队，才能rc

OKTOREAD.signal;­­­­//给READ发信号，这个信号是发给其他reader的，告诉大家一起上

}

startWrite{

if (busy||RC!=0) //有read或者额有其他writer，等

OkToWrite.wait();

busy=true; //成功write， busy=true

}

FinishRead()

RC--; rc减少

if (RC==0) OKtoWrite.signal;

}//没有reader的时候开始write

FinishWrite()

{Busy=F;

if(OKTOWRITE.queue==true)

OkToWrite.signal;//还有write的时候

else

OKTOread.signal;

}

上面的也是有小问题的

如果有R1R2R3R4，W1正在book内

W1 leave, finish write，给OKTOREAD发信号

会随机激活一个，激活R1，这时­W4进入，我们应该在R1结束后进入W4，但是R1会发射OKTOREAD.SIGNAL，这样就让R3在W4之前

改变

if (busy||OKTOWRITE.queue==true)//有人正在写，或者想写,wait

OKtoREAD.wait;

RC++; //如果没有人在排队，才能rc

if(OkToWrite.queue==false)

OKTOREAD.signal;­­­­//给READ发信号，这个信号是发给其他reader的，告诉大家一起上

为啥改变里不加上else,,OKTOWRITE.signal

假设W4wait. R2进行，我们要等R2进行完，才能W4