经典同步问题

读者-写者问题

> *共享数据的访问*
>
> *读者：不需要修改数据*
>
> *写者：读取和修改数据*

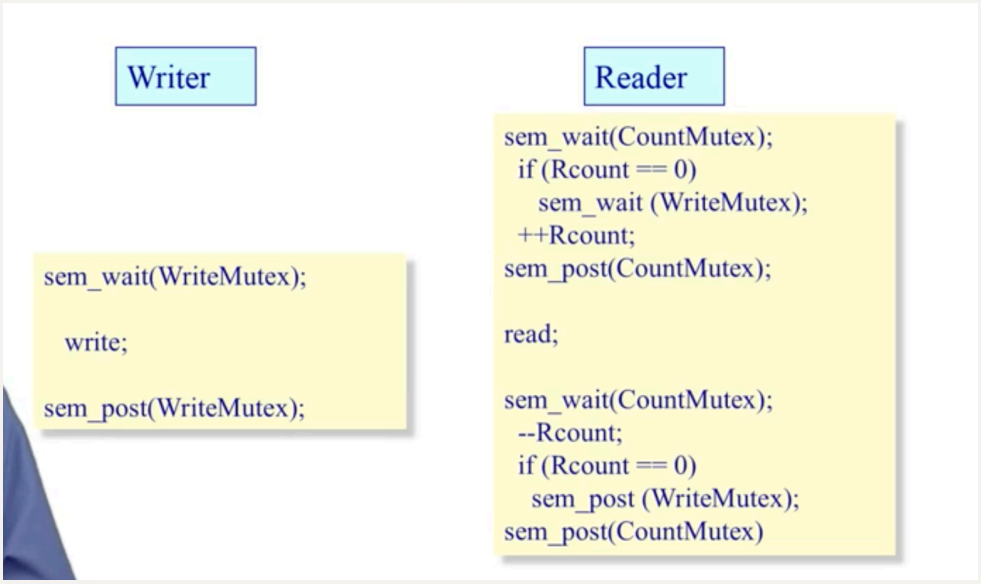
问题的约束

- 允许同一时间有多个读者,但在任何时候只有一个写者
- 当没有写者时，读者才能访问数据
- 当没有读者和写者时，写者才能访问数据
- 在任何时候只有一个线程可以操作共享变量

共享数据

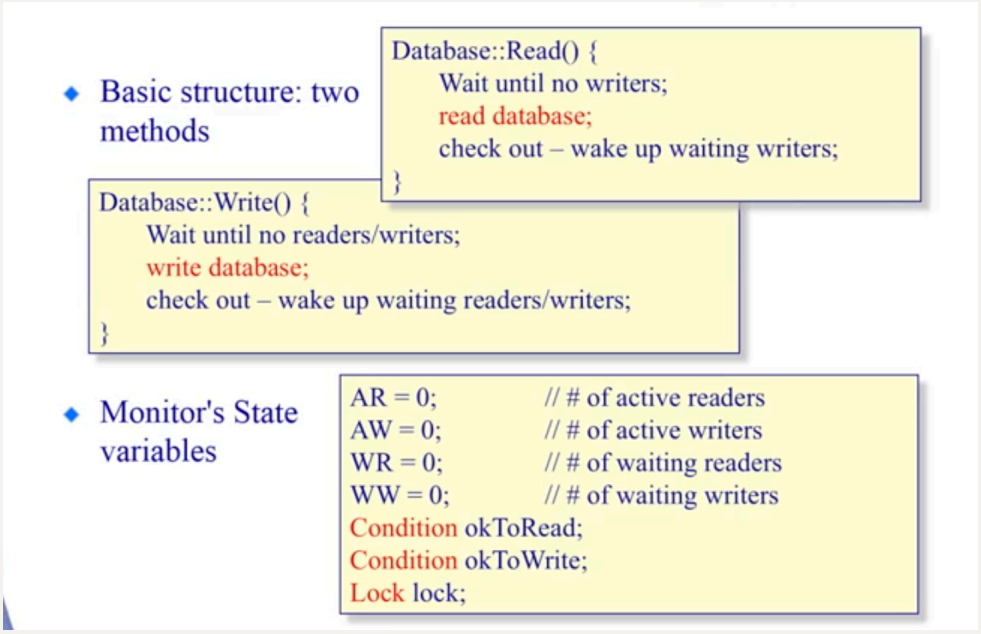
- 数据集data
- 信号量CountMutex初始化为1
  修改Rcount变量时候需要互斥
- 信号量WriteMutex初始化为1
  写操作的时候需要互斥
- 整数Rcount 初始化为0
  当前有多少个读者

读者优先

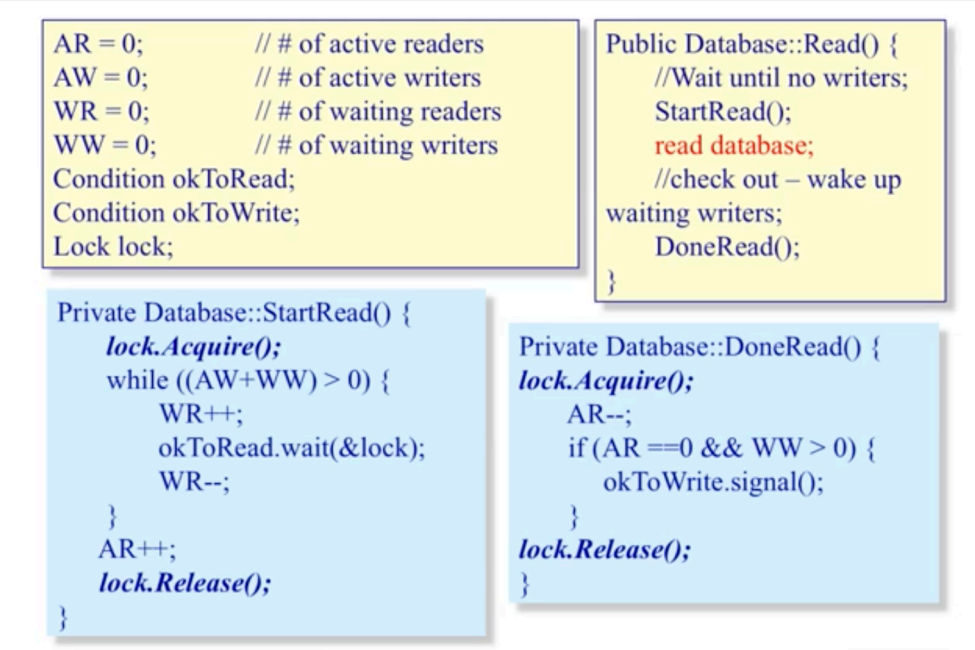


写者优先

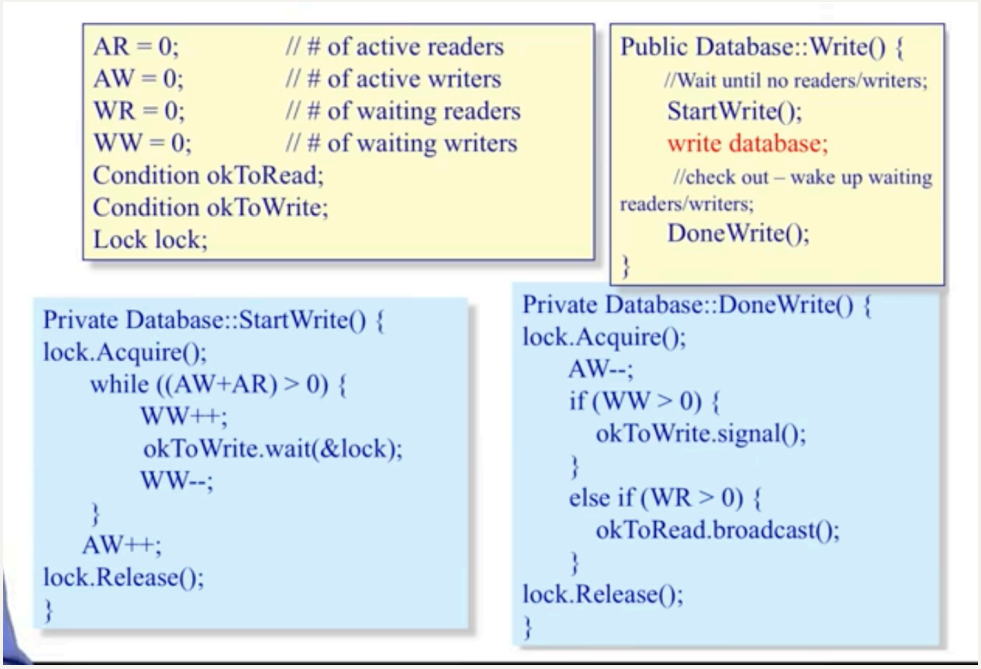
使用管程来实现写者优先



读


AR = 0; // # of active readers
AW = 0; // # of active writers
WR = 0; // # of waiting readers
WW = 0; // # of waiting writers
Condition okToRead;
Condition okToWrite;
Lock lock;
Public Database::Read() {
//Wait until no writers;
StartRead();
read database;
//check out – wake up waiting writers;
DoneRead();
}
Private Database::StartRead() {
lock.Acquire();
while ((AW+WW) > 0) {
WR++;
okToRead.wait(&lock);
WR--;
}
AR++;
lock.Release();
}
Private Database::DoneRead() {
lock.Acquire();
AR--;
if (AR ==0 && WW > 0) {
okToWrite.signal();
}
lock.Release();
}

写


AR = 0; // # of active readers
AW = 0; // # of active writers
WR = 0; // # of waiting readers
WW = 0; // # of waiting writers
Condition okToRead;
Condition okToWrite;
Lock lock;
Public Database::Write() {
//Wait until no readers/writers;
StartWrite();
write database;
//check out – wake up waiting readers/writers;
DoneWrite();
}
Private Database::StartWrite() {
lock.Acquire();
while ((AW+AR) > 0) {
WW++;
okToWrite.wait(&lock);
WW--;
}
AW++;
lock.Release();
}
Private Database::DoneWrite() {
lock.Acquire();
AW--;
if (WW > 0) {
okToWrite.signal();
}
else if (WR > 0) {
okToRead.broadcast();
}
lock.Release();
}

哲学家就餐问题