10/11/2010

Midterm	1A	Average	66
Midterm	1B	Average	66.7

MT 1A

1.

MT 1B 1.

b.

```
while (true){
  lock1->Acquire();
  - //do s - hit the ball
  lock1->Release();
}
```

d.

e.

1A Problem 4 - Semaphore

20 Cust's in Line only

- if there are already 20 people, the Cust must leave

Problem: Semaphores are hidden. you cannot use a Semaphore for the 20-Cust limit

Data: ① Need an int for Line count

int LineCount = 0;

② Binary semaphore as a lock to control access to LineCount

Semaphore mutex(1);

③ Semaphore for Customers who are waiting in Line

Semaphore waitingInLine(0);

A

10 Prob4 Customer

```
mutex.Down();
if ( LineCount >= 20) {
    // I ~~can get in line~~ must leave
    mutex.Up();
    return;
}
// I can get in line
~~mutex.Up();~~
LineCount ++;
mutex.Up();
waitingInLine.Down();
```

1B Prob 4

Play tennis

- Two threads only. They take turns hitting the ball

Data: No counter

2 semaphores

Semaphore ~~Player~~ Federer(1)

Semaphore Roddick(0)

~~Thr~~ Roger Andy

```
Federer.Down();          Roddick.Down();
{//hits ball            //hits ball
Roddick.Up();            Federer.Up();
```

~~Hw~~ Prob3

Key: Priority is determined simply by what a thread checks first - when it is exiting a C.R.

On entry, you check the appropriate monitor variables to see if you can enter the C.R.

Data: I need 4 monitor variables

M.V.
- * waiting Writers / waiting Readers — waiting Producers / waiting Consumers
- * (1) active Writers / active Readers — active Producers / active Consumers

When waiting in Line for M.E.
- waitingWriterCV / waitingReaderCV — waitingProducerCV / waitingConsumerCV
- ReaderWriterLock — PCLock

1B Prob 2 — Table Cleaners

Cust's not seated @ dirty tables
- When a Cust leaves a table they change its state to dirty
- After a TC cleans a table, the table state is clean

Table state array

Mgr provides a table number to a Cust.

4 Parts
- Table Cleaner
- Mgr getting table # for Cust
- Cust receiving table #
- Cust making table dirty

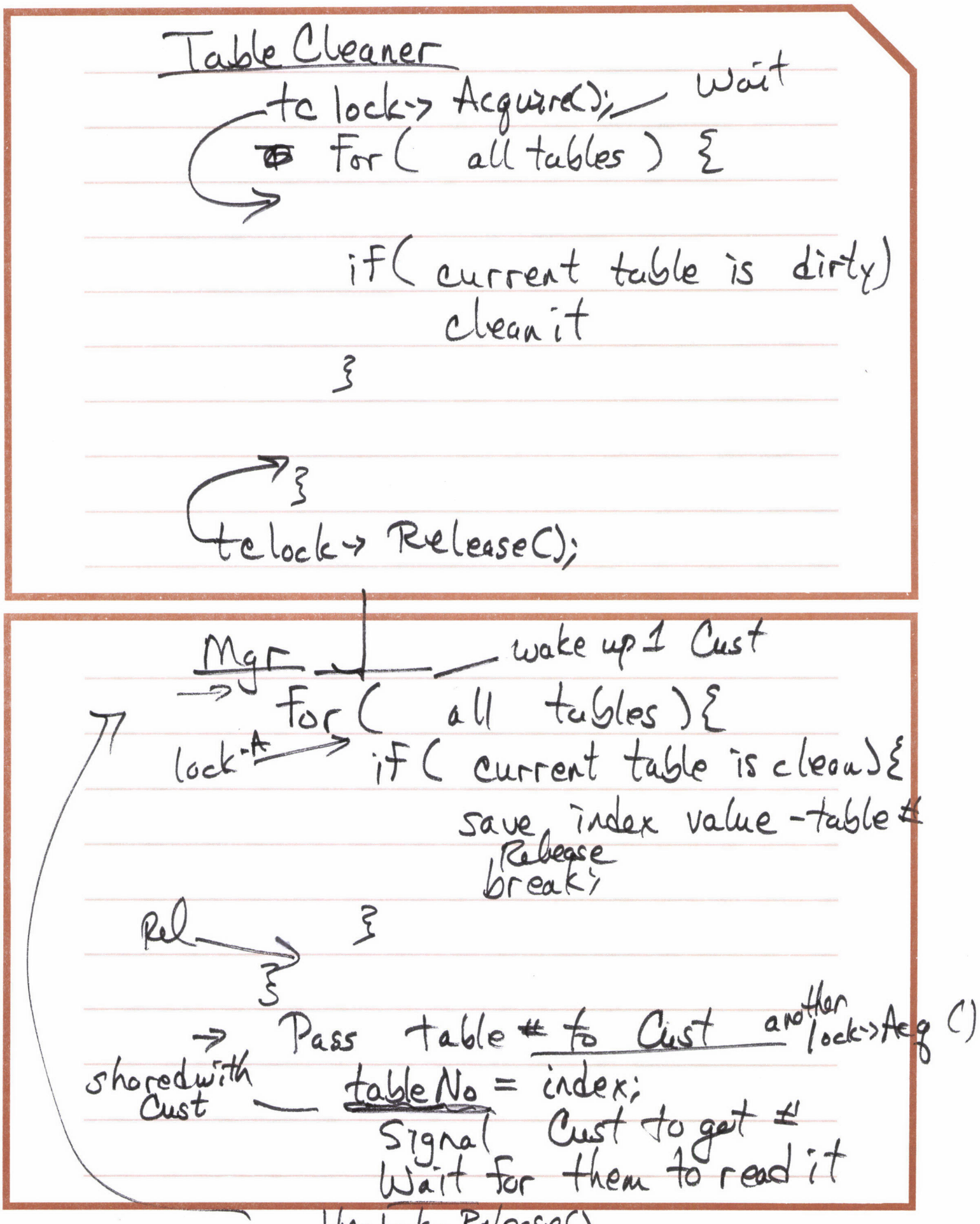
Table Cleaner
tc lock-> Acquire(); wait
For (all tables) {
if(current table is dirty)
clean it
}
}
tclock-> Release();
Mgr
wake up 1 Cust
For (all tables){
lock-A
if (current table is clean){
save index value - table #
Release
break;
}
Rel
}
Pass table # to Cust
another lock->Acq ()
shared with Cust
tableNo = index;
Signal Cust to get #
Wait for them to read it
another lock-Release()

Cust's waiting for table #

Cust

- Get table no From Mgr
  anotherlock -> Acquire()
  myTableNo = tableNo;
  Signal the Mgr
  anotherlock -> Release()

local to my thread

- Make table dirty
  tableLock[myTableNo] -> Acquire()
  table[myTableNo].status = DIRTY;
  tableLock[myTableNo] -> Release();

#1A Prob 2

- Each OT has their own line
- Customers choose shortest line
- Not all OTs will be taking orders
- There is always 1 OT taking orders

Data: An array of line lengths
- 1 for each OT

An array of OT status as to whether they are taking orders, or not

Algorithm: Single for loop to find smallest value of OT taking orders

int smallest, index = -1;

Cust

for (all OTs) { OTLock[ ] -> Acq

if(this order taker is taking orders

1st OT taking orders

&& index == -1)

set smallest to this value

index = i;

} else if(this OT taking orders){

// Check this line length is shorter than "smallest"

• if so, save this position

} } → Release

Cust gets in line for their OT

OT

Has their own line

• LineLock [myIndex] * or 1 Lock

* LineCV [myIndex]

* LineLength [myIndex]

Project 3- 3 parts

Parts 1 & 2 - Demand Paged Virtual Memory

To test: matmult - 7220
sort - 1023

Part 3- Nachos networking

Implement a ~~Lock~~ Monitor server in Nachos