

The Midterm

ICS412 - Fall 2009 Operating Systems

Henri Casanova (henric@hawaii.edu)

What to Study?

- All lecture notes and reading assignments up to and including “CPU Scheduling”
- Past homework and solutions
 - Topics not in homework will be on the exam
- Past programming assignments and solutions
- There should be very few surprises
- Type of Questions:
 - quiz-like questions
 - general questions
 - problems

Sample Quiz-like questions

- Which of the following scheduling algorithms can lead to starvation?
 - FCFS, Priority, SJF
- A program containing a race condition will result in data corruption
 - Always, sometimes, never
- A system that meets the four necessary deadlock conditions will deadlock
 - Always, sometimes, never
- Can the system call instruction be privileged?
 - Yes, No
- A condition variable is a kind of semaphore?
 - Yes, No
- Does exec() create a new process?
 - Yes, No

Sample Quiz-like questions

- Which of the following scheduling algorithms can lead to starvation?
 - FCFS, **Priority**, SJF
- A program containing a race condition will result in data corruption
 - Always, **sometimes**, never
- A system that meets the four necessary deadlock conditions will deadlock
 - Always, **sometimes**, never
- Can the system call instruction be privileged?
 - Yes, **No**
- A condition variable is a kind of semaphore?
 - Yes, **No**
- Does exec() create a new process?
 - Yes, **No**

Sample General Questions

- What is the difference between an Interrupt and an Exception/Trap? Give two examples of each

Sample General Questions

- What is the difference between an Interrupt and an Exception/Trap? Give two examples of each
 - **Both are events that the OS must react to**
 - **An Interrupt is due to some external event**
 - e.g., keyboard input
 - e.g., disk operation completion
 - **A trap/exception is caused by an instruction**
 - e.g., illegal memory access
 - e.g., system call instruction

Sample General Questions

- The value of a semaphore cannot be directly read by a process. Why is this the case? Why is it not useful to read the value of a semaphore?

Sample General Questions

- The value of a semaphore cannot be directly read by a process. Why is this the case? Why is it not useful to read the value of a semaphore?
 - Reading the value doesn't mean anything because by the time one tries to use this value for any purpose, any other process may have decreased/increase the value
 - The only conceivable use would be to do a "tryP()"

Sample General Questions

- What is the difference between a race condition and a deadlock?

Sample General Questions

- What is the difference between a race condition and a deadlock?
 - In a deadlock no thread can make progress
 - In a race condition all threads make progress but there may be data corruption

Sample General Questions

- What are the advantages/disadvantages of User vs. Kernel Threads?

Sample General Questions

- What are the advantages/disadvantages of User vs. Kernel Threads?
 - User-level
 - good: very low overhead for creation/maintenance because the kernel's not involved
 - bad: cannot take advantage of multi-core architectures
 - bad: if on thread blocks, they all block
 - Kernel-level
 - bad: high overhead because the kernel is involved
 - removes the two "bad" of user threads

Problems

- There will be one concurrency problem where you have to write pseudo-code using locks and condition variables or semaphores
- Let's do a few such problems
 - Ambitious Co-Worker
 - Meeting Room
 - Busy Bridge

Ambitious Co-Worker

- Your co-worker says she's solved it all with *tagged critical regions!*
- Example:


```
critical x, z
S1; // mutual exclusion with anybody having x or z
end critical

critical x, y;
S2; // mutual exclusion with anybody having x or y
end critical

critical y, u, v;
S3; // mutual exclusion with anybody having y, u, or v
end critical
```
- Question #1: How can this be implemented with semaphore. the solution must be safe and live
- Question #2: is it true that one can do anything with this? Is it as powerful as semaphores?

Ambitious Co-Worker

- Your co-worker says she's solved it all with tagged *critical regions!*
- Example:


```
critical x, z // x.P(); z.P();
S1;
end critical // x.V(); z.V();

critical x, y; // x.P(); y.P();
S2;
end critical // x.V(); y.V();

critical y, u, v; // u.P(); v.P(); y.P();
S3;
end critical // u.V(); v.V(); y.V();
```
- Question #1:
 - Just have one semaphore per tag
 - Sort them all by lexicographical order
 - to avoid circular waiting, and therefore deadlocks!
 - "critical x,y" and "critical y,x" in the same program
 - Call P() on them in that order when entering a tagged critical region
 - and V() when exiting

Ambitious Co-Worker

- Your co-worker says she's solved it all with tagged *critical regions!*
- Example:


```
critical x, z // x.P(); z.P();
S1;
end critical // x.V(); z.V();

critical x, y; // x.P(); y.P();
S2;
end critical // x.V(); y.V();

critical y, u, v; // u.P(); v.P(); y.P();
S3;
end critical // u.V(); v.V(); y.V();
```
- Question #2:
 - Not as powerful as semaphore as semaphores provide signaling
 - which is why we added condition variables to locks to equate the power of semaphores

Meeting Rooms

- ICS412 students and ICS faculty are having a party in a POST 1217
 - no more than 40 people in a room
 - an equal number of students and faculty at all times
- Using cond/locks, write procedures for
 - FacultyEnter(), FacultyLeave()
 - called in sequence by faculty threads
 - StudentEnter(), StudentLeave()
 - called in sequence by faculty threads

Meeting Rooms

```

new_student = false; totalf, totals = 0;
new_faculty = false; lock mutex;
cond NoNewFaculty, NewFaculty;
cond NoNewStudent, NewStudent;
cond NotFull;

FacultyEnter() {
lock(mutex);
while (new_faculty)
wait(NoNewFaculty, mutex);
new_faculty = true;
signal(NewFaculty);
while (!new_student) {
wait(NewStudent, mutex);
}
while (totals >= 19) {
wait(NotFull, mutex);
}
totalf++;
new_faculty = false;
signal(NoNewFaculty);
unlock(mutex);
}

StudentEnter() {
lock(mutex);
while (new_student)
wait(NoNewStudent, mutex);
new_student = true;
signal(NewStudent);
while (!new_faculty) {
wait(NewFaculty, mutex);
}
while (totals >= 19) {
wait(NotFull, mutex);
}
totals++;
new_student = false;
signal(NoNewStudent);
unlock(mutex);
}

Leave() {
// just like the enter
}
    
```

Busy Bridge

- A bridge is undergoing repair and only one lane is open for traffic. To prevent accidents traffic lights have been installed to synchronize traffic going in different directions
- A car can only cross the bridge if there are no cars going the opposite direction. Sensors at either end of the bridge detect when cars arrive and depart from the bridge, and these sensors control the traffic lights
- Cars are implemented as
 - Arrive(dir my_dir)
 - Depart()
- Let's look at pseudo-code

Busy Bridge

```
int num cars = 0;      enum dir={open, north, south};
dir cur_dir = open;
```

```
Arrive (dir my_dir) {                Depart () {
    while (cur_dir != my_dir ||      num_car--;
           cur_dir != open) {        if (num_cars == 0) {
    }                                  cur_dir = open;
    num_car++;                        }
    cur_dir = my_dir;                }
}
```

- Question #1: The code above has no synchronization. Outline an execution sequence where two threads can cause two cars to be on the bridge traveling in opposite direction

Busy Bridge

```
int num cars = 0; enum dir={open, north, south};
dir cur_dir = open;
```

```
Arrive (dir my_dir) {                Depart () {
    while (cur_dir != my_dir ||      num_car--;
           cur_dir != open) {        if (num_cars == 0) {
    }                                  cur_dir = open;
    num_car++;                        }
    cur_dir = my_dir;                }
}
```

- Car #1: arrives, sees the bridge as open, increases num_car++, and gets preempted
- Car #2: arrives, sees the bridge as open, increases num_carr+, and sets cur_dir to North
- Car #1: resumes, sets cur_dir to South
- At this point, both areas have exited from Arrive() and are traveling on the bridge, towards a head-on collision

Busy Bridge

```
int num cars = 0; enum dir={open, north, south};
dir cur_dir = open;
```

```
Arrive (dir my_dir) {                Depart () {
    while (cur_dir != my_dir ||      num_car--;
           cur_dir != open) {        if (num_cars == 0) {
    }                                  cur_dir = open;
    num_car++;                        }
    cur_dir = my_dir;                }
}
```

- Question #2: Fix the above code with Locks and Condition Variables

Busy Bridge

```
int num cars = 0;      enum dir={open, north, south};
dir cur_dir = open;   lock mutex;      cond light;
```

```
Arrive (dir my_dir) {                Depart () {
    lock(mutex);                      lock(mutex);
    while (cur_dir != my_dir ||      num_car--;
           cur_dir != open) {        if (num_cars == 0) {
        wait(light, mutex);           cur_dir = open;
    }                                  }
    num_car++;                        unlock(mutex);
    cur_dir = my_dir;                signal(light);
    unlock(mutex);                    }
    signal(light);                    }
}
```

Busy Bridge

```
int num cars = 0;      enum dir={open, north, south};
dir cur_dir = open;   lock mutex;      cond light;
```

```
Arrive (dir my_dir) {                Depart () {
    lock(mutex);                      lock(mutex);
    while (cur_dir != my_dir ||      num_car--;
           cur_dir != open) {        if (num_cars == 0) {
        wait(light, mutex);           cur_dir = open;
    }                                  }
    num_car++;                        unlock(mutex);
    cur_dir = my_dir;                signal(light);
    unlock(mutex);                    }
    signal(light);                    }
}
```

- Question #3: Can there be starvation?

Busy Bridge

```
int num_cars = 0;          enum dir={open, north, south};
dir cur_dir = open;       lock mutex;          cond light;

Arrive (dir my_dir) {
    lock(mutex);
    while (cur_dir != my_dir ||
           cur_dir != open) {
        wait(light, mutex)
    }
    num_car++;
    cur_dir = my_dir;
    unlock(mutex);
    signal(light);
}

Depart () {
    lock(mutex);
    num_car--;
    if (num_cars == 0) {
        cur_dir = open;
    }
    unlock(mutex);
    signal(light);
}
```

- Question #3: Can there be starvation?
 - Yes
 - Just like in the unbounded reader-preferred solution for reader-writer