The Importance of Safety and Synchronization Review

CS439: Principles of Computer Systems
October 1, 2018
Bringing It All Together

- **Processes**
  - Abstraction for protection
  - Define address space
- **Threads**
  - Share (and communicate) through global and static data, share the heap, each has its own stack and full use of the registers
  - Race conditions may be a problem!
- **CPU Schedulers**
  - May pre-empt a process or thread at any time
- **Ensuring correctness (OR eliminating race conditions and deadlock)**
  - Safety and liveness
  - Atomic instructions
  - Synchronization: mutual exclusion, counted resources...
  - Locks, semaphores, monitors, transactions, conservative two-phase locking
  - The Six Commandments of multi-threaded programming
  - Common patterns: Bounded Buffer, Dining Philosophers, Readers/Writers
Today’s Additions

• The Importance of Safety (Therac-25)
• Synchronization Review
• Exam Logistics
Therac-25
or The Importance of Safety
What is the Therac-25?

- Linear accelerator
- Used to treat patients...
## Modes of Operation

<table>
<thead>
<tr>
<th></th>
<th>Beam Energy</th>
<th>Beam Current</th>
<th>Beam Modifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>For electron therapy</td>
<td>5-25 MeV</td>
<td>low</td>
<td>magnets</td>
</tr>
<tr>
<td>For X-ray therapy, photo mode</td>
<td>25 MeV</td>
<td>high (100x)</td>
<td>flattener</td>
</tr>
<tr>
<td>For field light mode</td>
<td>0</td>
<td>0</td>
<td>none</td>
</tr>
</tbody>
</table>
What Went Wrong?

• Two (major) software problems
• Tons of bad software design/human failures that might have prevented this:
  – False alarms
  – Errors reported by number only *and there was no documentation!*
  – No clearinghouse for mistakes *and company hid failures from other users*
  – No end-to-end consistency checks
  – No quality control
  – Don’t trust software---hardware should have prevented this, too
What about more recent disasters?

• We don’t know for sure

• Possibly software lost treatment plan and defaulted to “all leaves open”

• Software should have sensible defaults!
Lessons

• Complex systems fail for complex reasons
• Be tolerant of inputs
• Be strict on outputs
• Assume buggy software and protect against it!
Synchronization Review
Concurrency is great...

```c
int a=1, b=2;
main() {
    createThread(fn1, 4);
    createThread(fn2, 5);
    thread_join(all);
}

fn1(int arg1){
    if(a) b++;
}

fn2(int arg1){
    a=arg1;
}
```

What are the values of a and b after execution?

A. a=1, b=2  
B. a=1, b=3  
C. a=5, b=2  
D. a=5, b=3
... but can be problematic

```c
int a=1, b=2;
main() {
    createThread(fn1, 4);
    createThread(fn2, 5);
    thread_join(all);
}

fn1(int arg1){
    if(a) b++;
}

fn2(int arg1){
    a=0;
}
```

What are the values of a and b after execution?

A. a=0, b=2  
B. a=0, b=3
C. a=1, b=2  
D. a=1, b=3
int flag1=0, flag2=0;

int main(){
    tid id=thread_create(p1, NULL);
    p2(); thread_join(id);
}

void p1 (void *ignored){
    flag1=1;
    if(!flag2){
        critical_section_1();
    }
}

void p2(void * ignored){
    flag2=1;
    if(!flag1){
        critical_section_2();
    }
}

Can both critical sections execute during a single execution of the code?

A. Yes
B. No
Atomicity

• Required to reason about multi-threaded code without considering all interleavings
• Requires mutual exclusion
• Locks provide that solution
• Looked at lock implementation
  – Requires waiting
  – Requires hardware support
• Use software abstractions
  – Semaphores
  – Monitors (lock+condition variables)
Tradeoff and Problems: Difficult to Get Right

• Ensure safety
• Ensure liveness
• No race conditions
• No starvation
• No priority inversion
• No deadlock
In Addition... the Cost of Parallelization

for($k = 0; k < n; k++)
    a[k] = b[k]*c[k] + d[k]*e[k];

How would you parallelize this?
How many threads?
The Six Commandments

- Thou shalt always do things the same way
- Thou shalt always synchronize with locks and condition variables
- Thou shalt always acquire the lock at the beginning of a function and release it at the end
- Thou shalt always hold lock when operating on a condition variable
- Thou shalt always wait in a while loop
- (Almost) Never sleep()
Why Thread Coding Standards?

• History has tested this approach
• If you follow these commandments, you will find it easier to write correct code.
• In this class, you must use them or lose points.
• We highly recommend that you continue to do so after this class
But...

• After this class, if you can come up with something better, please use it!

• BUT...
  – Lots of really smart people have thought really hard about this already, so a day or two of thought is unlikely to change the best practice
  – The consequences of getting code wrong can be atrocious
  – People who are confident about their abilities tend to perform *worse*. If you think you are a Threading and Concurrency Ninja and truly understand, then you may wish to re-evaluate...
    • Dunning-Kruger effect
In this class...

- Six commandments
- Coarse-grained locking
- Order your resources
Exam Logistics
Exam Review

He who asks is a fool for five minutes; he who does not ask remains a fool forever.

- Anonymous Chinese Proverb
iClicker Question

What might be on the exam?

A. Information from lectures and reading
B. Coding questions
C. Concept questions (general understanding/thought)
D. All of the above (and more!)
Exam Procedures

• Arrive on time
  – No one may start the exam after the first person leaves
• Bring your UT ID
• Find your EID and assigned seat on the chart outside the classroom
• Do not enter the room until told to do so
• When you enter, proceed to your seat
Exam Procedures

• Leave all extra paper, electronics, hats, etc. in your bag.
• Do not begin the exam until told to do so
• No questions may be asked during the exam
  – Write any assumptions
• When finished
  – turn in exam and all scratch paper to myself or the proctor
  – present your ID
iClicker Question

What should you bring to the exam?

A. A writing utensil and your ID
B. Nothing
My Best Advice

Do NOT panic!

You have been taught how to do each question, and you can do it.
Summary

• Please Think!

• Safety first!
  – Coarse-grained locking is the easiest to get right, so do that
  – Don’t worry about performance at first
  – In fact, don’t even worry about liveness at first

• Follow the thread coding standards
  – If you don’t, it is wrong!
Announcements

• Exam 1 is next Wednesday at 7p UTC 2.112A
  – If you have a conflict, you should have already told me (if you
    don’t receive instructions by noon Tuesday, contact me again)
  – Show up ON TIME
• Class on Wednesday is shortened and optional
  – 10:15a-11:45a in GDC 6.302 (probably)
  – Review sessions (driven by your questions!)
  – Any student may attend either section
• My Wednesday office hours are canceled
• Solutions to the sample exam will be posted later today
• Project 1 due Friday at 5:59p/11:59p
• No discussion sections this week