

15-410
“...misbehave(7)...”

Project 2
Sep. 20, 2006

Dave Eckhardt

Bruce Maggs

Synchronization

P2 (et seq.) partners

- “Partner Registration Page” on web site
- Everybody registered - Thanks!

Please make sure you've discussed

- How many late days?
- Project *schedule* in other classes
 - *Write down* a joint project schedule
- Auditing or pass/fail? Target 410 grade?
- Prior experience
- Interviews

Outline

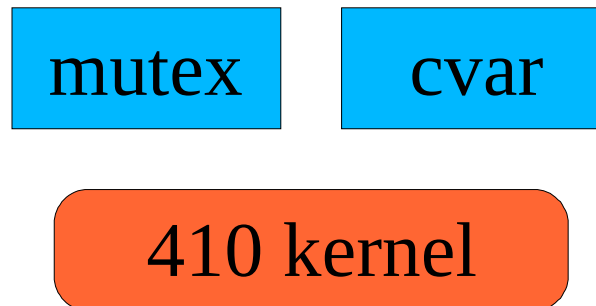
What you'll build

- Mutex, condition variable
- Thread library
- Supplemental library routines
- Tests

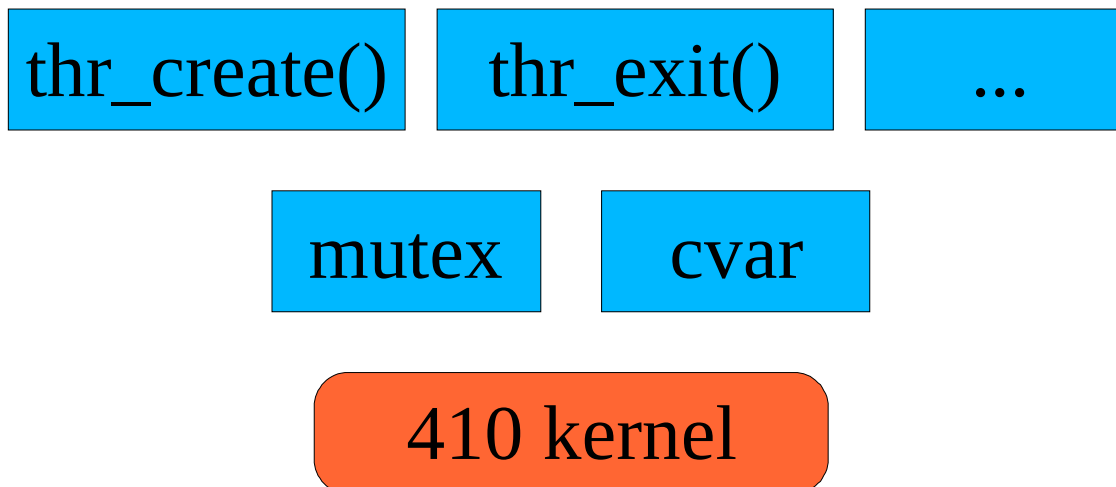
How the pieces fit together

- A picture is worth 1000 words
- You'll need to read the handouts too
 - (two, each >1000 words)
 - `kspec` –specifies our kernel for P2, your kernel for P3
 - `thr_lib` –specifies thread library

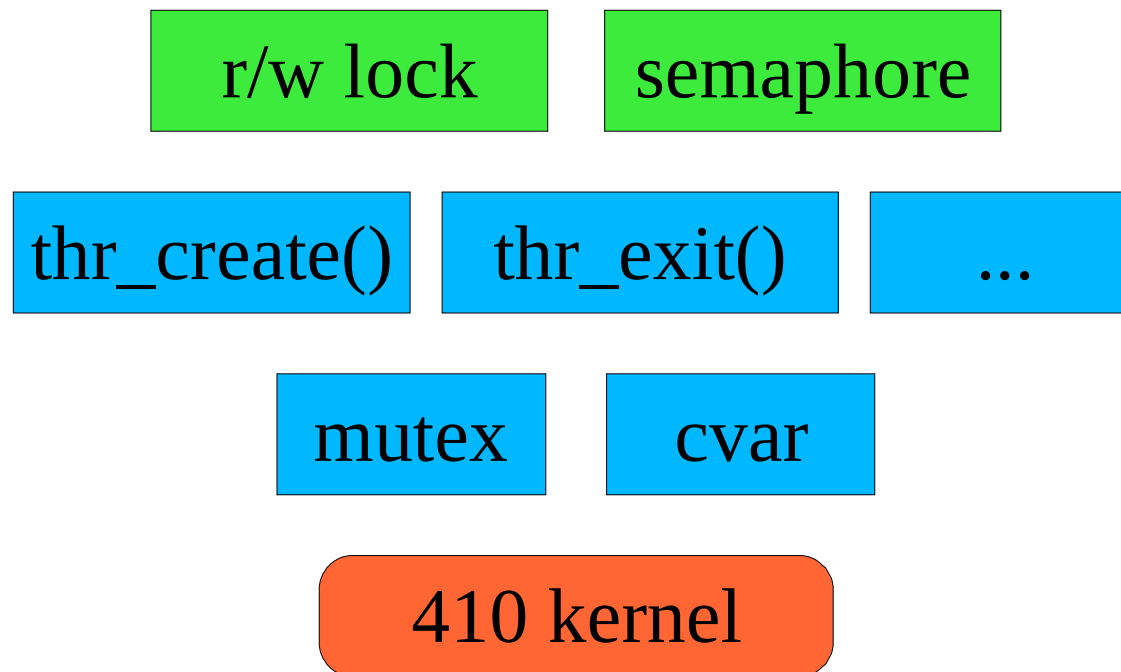
Mutex & Condition Variable



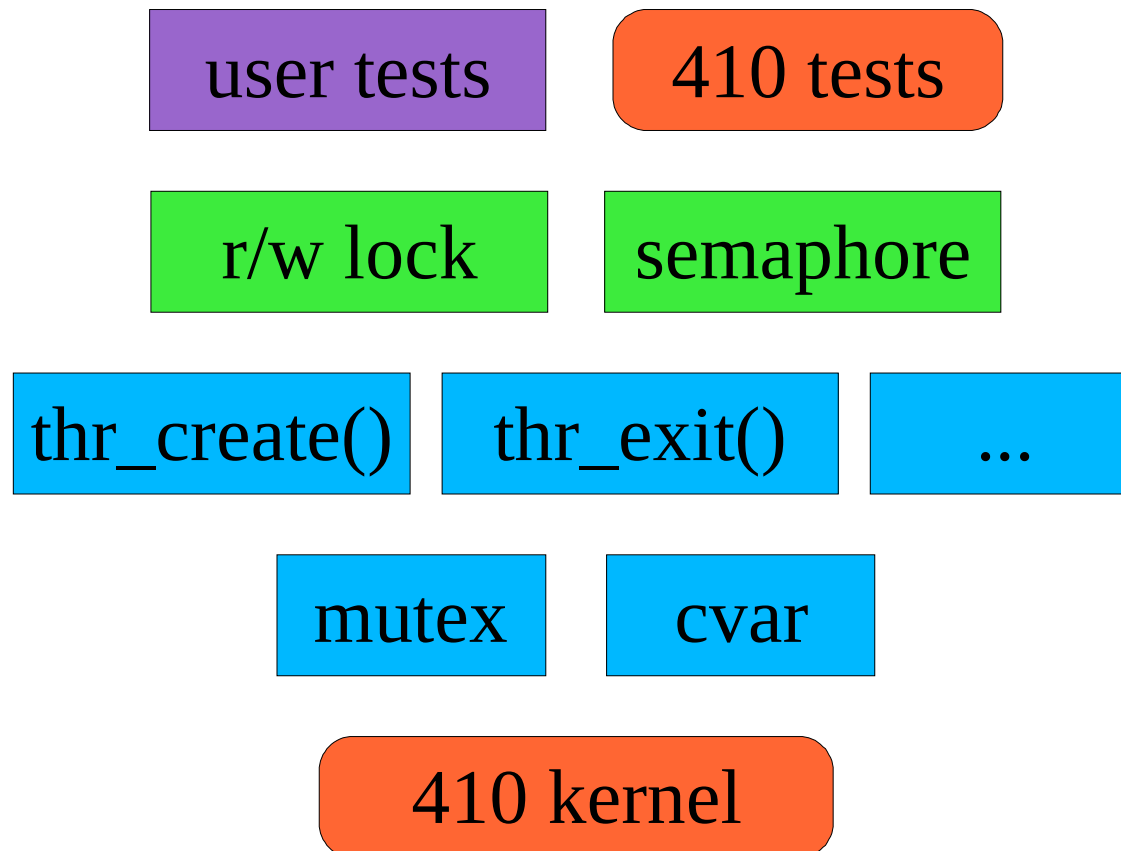
Remainder of Thread Library



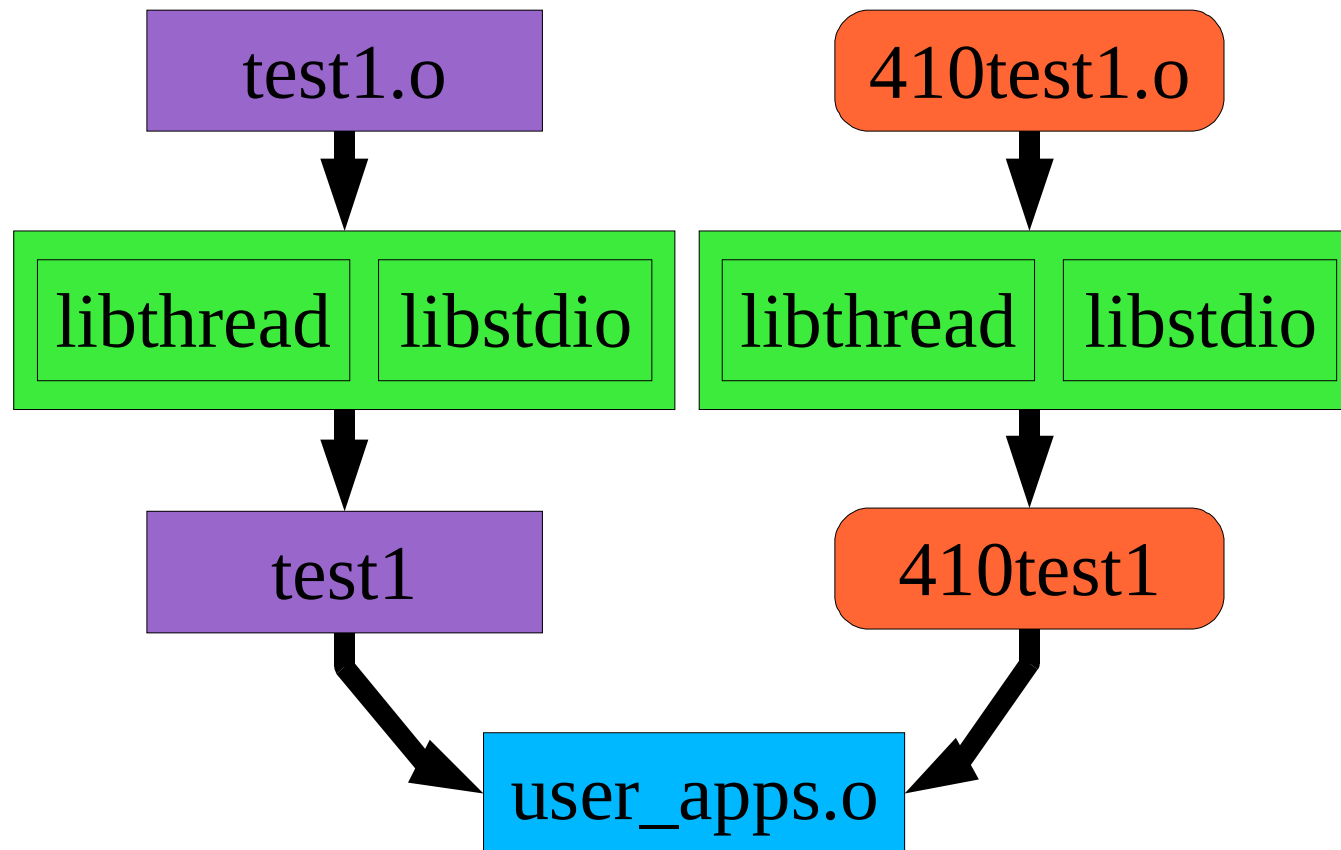
Supplemental Library Routines



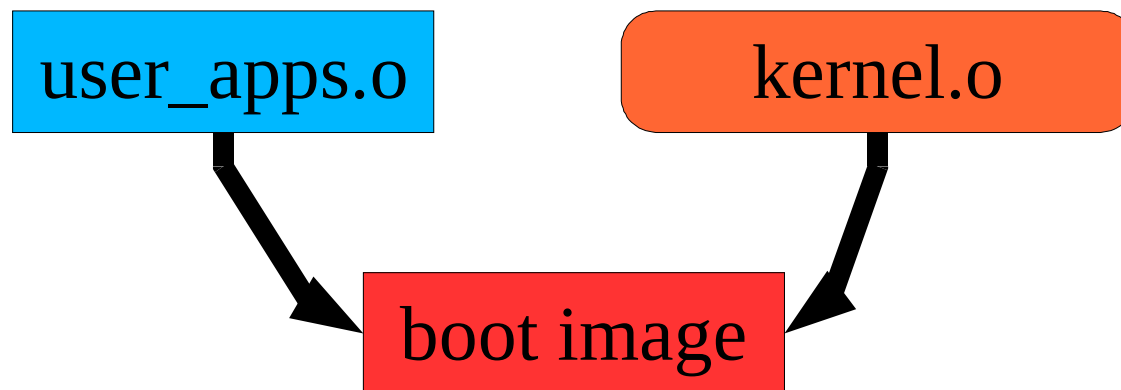
Tests (Yours & Ours)



Building a “RAM disk” image



Linking “RAM disk” to kernel



Misbehave

misbehave(int mode)

- Special debugging-support system call in our 410 kernel
- Adjusts “behavior” of system
 - Multiple legal behaviors (you will feel this during P3)
 - Each mode selects a particular mix
 - We will not document these
 - We expect you to not “document” them to classmates either
- Debug your thread library with one mode, then the next...
 - A dazzling array of flavors
 - 0...15
 - maybe even more
 - -1
- You will not be required to implement misbehave() in P3

threadinfo

```
simics> threadinfo 11
```

REGISTER DUMP FOLLOWS

```
CS = 0x00000043, EFLAGS = 0x00010246, SS = 0x0000004b  
EIP = 0x0100004a, ESP = 0xffffffffa0, EBP = 0xffffffffcc  
EDI = 0x00000000, ESI = 0x00000000, EAX = 0x31337000  
EBX = 0x00000000, ECX = 0x00000000, EDX = 0x01000c0a
```

What?

- Debugging information about thread 11
- The last instruction it executed in user space

Why would I want that?

- It might help with certain hard problems

Plea – Conceptual

This code is *tricky*

- Most of you have already written multi-threaded code
 - That can be tricky enough
- Writing the internals is harder
 - Get a part 99% done
 - Discover a “bug”...
 - ...which is really a misconception...
 - *Totally new design* to fix it

Make sure core parts are *solid*

- Better to skip readers/writers locks if not

Plea –Time

The first 90% will take the first 90% of the time

- The last 10% will take the *second 90% of the time*

“Code complete”

- Plan to spend *at least three days* debugging based on the tests we release
- If your thread library doesn't pass cyclone and agility_drill it won't pass a bunch of our tests either
 - Resultant grade is unlikely to exceed a C