ECE 598 – Advanced Operating Systems
Lecture 22

Vince Weaver

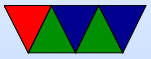
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

21 April 2015

Announcements

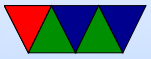
- HW#8 is still being determined



Homework 7 Review

- VGA Fonts
  Were once popular. You could change the built-in console font to an 8x16 bitmap font.
  These days, ungooglable. You just get a vga ttf font file for DOS emulation purposes
  My fontprint/ansi2gif programs use it.

- Game
  With a framebuffer and a keyboard you can write games!
  Need a non-blocking getchar and maybe a delay loop

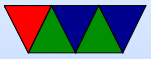


Slow on the Pi. Lack of cache? Non-caching framebuffer? Unsure

Even double-buffering doesn't seem to help much.

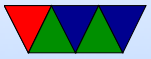
For a game, just need to erase old, draw new.

A game loop: Draw background (clear old), check keypress, move tems, collision detect, draw items, delay if needed, loop.



IPC in the news

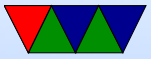
- kdbus – dbus into kernel to make it faster
  Desktop Bus, D-Bus
  allows communication on a desktop bus between apps and kernel
  example – incoming skype call can notify system, and apps like the audio adjust and mp3 player can pause music and set up microphone
  multicast
  example – battery low notification, apps can listen, save



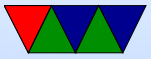
state, prepare to shut down.

- Kernel developers resist it
  Most because who has to maintain it?
  Also how well designed is it? can it be used by other tools?
  We already have a lot of IPC in the kernel, can it be made generic?
  It is faster, but what if user programs just bloat to negate this?

- Worse is better / the right thing

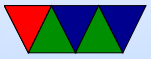


- New Jersey vs MIT
- Is it better to spend lots of time coming up with a perfect specification on paper, then implement it?
- Is quicker/easier to understand better than complex and perfect?
- Should you come up with something "good enough" and let it grow naturally?



Race Conditions

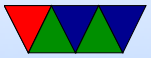
- Shared counter address
  RMW on ARM
  Thread A reads value into reg
  Context switch happens
  Thread B reads value into reg, increments, writes out
  Context switch back to A
  increments value, writes out
  What happened?
  What should value be?



Critical Sections

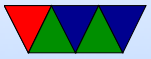
- Want mutual exclusion, only one can access structure at once

1. no two processes can be inside critical section at once
2. no assumption can be made about speed of CPU
3. no process not in critical section may block other processes
4. no process should wait forever



How to avoid

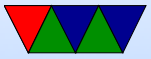
- Disable interrupts. Heavy handed, only works on single-core machines.
- Locks/mutex/semaphore



Mutex

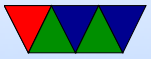
- mutex_lock: if unlocked (0), then it sets lock and returns
  if locked, returns 1, does not enter.
  what do we do if locked? Busy wait? (spinlock) reschedule (yield)?

- mutex_unlock: sets variable to zero



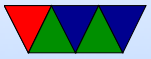
Semaphore

- Up/Down
- Wait in queue
- Blocking
- As lock frees, the job waiting is woken up



Locking Primitives

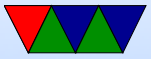
- fetch and add (bus lock for multiple cores), xadd (x86)
- test and set (atomically test value and set to 1)
- test and test and set
- compare-and-swap – Atomic swap instruction SWP (ARM before v6, deprecated)
  x86 CMPXCHG
  Does both load and store in one instruction!



Why bad? Longer interrupt latency (can't interrupt atomic op)
Especially bad in multi-core

- load-link/store conditional
  Load a value from memory
  Later store instruction to same memory address. Only succeeds if no other stores to that memory location in interim.
  ldrex/strex (ARMv6 and later)

- Transactional Memory



Locking Primitives

- can be shown to be equivalent

- how swap works:
  lock is 0 (free). r1=1; swap r1,lock
  now r1=0 (was free), lock=1 (in use)
  lock is 1 (not-free). r1=1, swap r1,lock
  now r1=1 (not-free), lock still==1 (in use)