ECE 598 – Advanced Operating Systems
Lecture 17

Vince Weaver

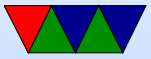
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 April 2015

Announcements

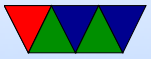
- HW#5 was posted.
- HW#6 will be back to coding.



HW#4 Review

1. Purpose of a filesystem
   Store your files, safely, and allow finding them by name.
   Organize data.

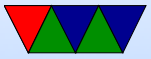
2. Filename in inode?
   No. This is because hard-links, more than one directory entry can link to a file.
   Could you still do name lookup if were in the inode?
   Yes, but it would be slower. (seeks are bad)



3. Ext2size

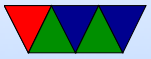
   (a) No indirect == 12 entries, 12kB
   (b) 1st indirect = 12kb+256kb = 268kB
   (c) 2nd indirect = 12kb+256kb + 256*256kB = 65MB
   (d) 3rd indirect = 12kb+256kb+256*256kb + 256*256*256kb = 16GB
   (e) Overhead = 0 + 1 + (1+256) + (1+256+256*256) = 64.5MB

4. Pick a fs. As discussed various are available. Some that people picked:



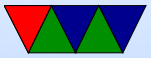
sysfs, afs, ramfs, coda, squashfs, reiserfs, ntfs, 9p, jfs

5. EXT2 better than fat?
   filesysize, long filenames, better performance
   Fat better than ext2?
   Simpler to implement, comaptible to more OSes



ISPASS Recap

- IEEE International Symposium on Performance Analysis of Systems and Software
- What's a conference like.
- Philadelphia
- Not much directly related to Operating Systems



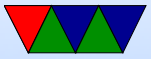
Hardware Performance Counters – ARM1176

- Three registers
- cp15 c15

```
MRC p15, 0, <Rd>, c15, c12, 0
MCR p15, 0, <Rd>, c15, c12, 0
```

- Performance Monitor Control Register

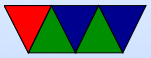
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
SBZ				EvtCount0								EvtCount1						



  - EvtCount0/1 – event to measure in Counters 0 or 1
  - X – Enable external bus?
  - CCR/CR1/CR0 – Cycle/R0/R1 register overflow
  - ECC/EC1/EC0 – enable overflow interrupt
  - D – divide cycle count by 64
  - C – reset cycle count reg
  - P – reset R0 and R1
  - E – enable all counters

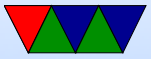
- Can set V bit to allow user mode access to registers

- Cycle count register, `MRC p15, 0, <Rd>, c15, c12,`



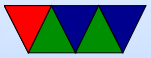
1

- Register 0, `MRC p15, 0, <Rd>, c15, c12, 2`
- Register 1, `MRC p15, 0, <Rd>, c15, c12, 3`

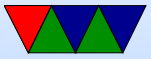


Other features

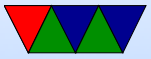
- Overflow interrupts, why useful? Counts greater than 32 bit
  Profiling
- User-space reading
- Multiplexing
- User/Kernel split
- Hardware watchdog



- NMI interrupt

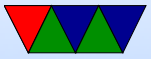


Kernel Interfaces



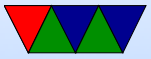
Most Simple – Raw Hardware

- Let userspace program MSRs directly
- Usually requires at least some level of kernel driver, but very small.



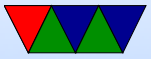
perfctr like

- ioctl() interface to start/stop



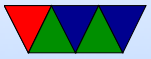
perfmon2

- Initially 12 system calls
- Why not a mux? Frowned upon.
- As much as possible done in userspace. Why?
  Event scheduling, generic events, multiplexing.
- The exported interface was a thin layer over the underlying PMU hardware

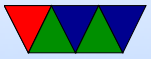


perf_event

- Everything done in kernel. Event scheduling, multiplexing, generic events, etc. why?
- perf_event_open() syscall returns a fd per event. Why might that be bad?
- attr, pid, cpu, group, flags
- attr large struct with 40+ conflicting options
- pid, lets you attach to process



- cpu, pick which cpu to monitor on
- group, lets you have group leaders and events grouped together to be read at once
- flags, allows future expansion. Lots of probles with syscalls without flags option, leads to thing like mmap2 etc.



Types of monitoring

- aggregate
- statistical sampling
- self-monitoring