ECE 531 – Advanced Operating Systems
Lecture 13

Vince Weaver

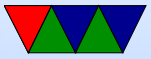
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

12 October 2023

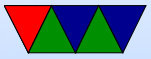
Announcements

- Homework #6 will be coming
- Midterm exam next class, Tuesday, October 17th
- Will try to have outstanding homeworks graded by then



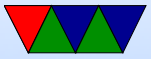
Project Preview

- Project pdf posted to course website
- Can work in groups
- First deadline is project topic, due October 30th
- More details as we get closer



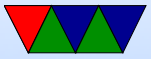
HW#5 Notes – Code

- Shell to userspace
- Avoid using sizeof() where you mean strlen()



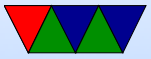
HW#5 Notes – Questions

- Nonblocking getchar – what happens if in kernel mode waiting for `serial_read()` to finish and a timer interrupt comes in?
- Why run in userspace? – mostly so kernel can protect system from rogue apps
- Changing back to kernel mode – syscall, interrupt, reset(?)
- What is an ABI
- Security implications of kernel writing to a user-supplied



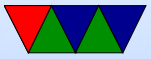
pointer. Linux for example uses `copy_to_user()` that checks to see if area pointed to actually belongs go user process in question. Can we do that? Does the kernel know what memory belongs to a process?

- System call of choice
  manpages, from section "2"
  many operate on file descriptors
  chmod, inotify, exit, fork, truncate, futex, stat, wait
  surprised no one said perf_event_open



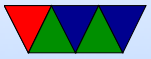
HW#4 Notes – Code

- The clock / timer used might vary based on CPU throttling or scaling, should use one of the many other timers available. Which one? They aren't all documented well. Might make a good project topic.



HW#4 Notes – Questions

- FIQ vs IRQ difference? FIQ banks some registers, so is faster (no saving), higher priority, only one so don't have to search for source.
- `BASIC_PENDING` bit 19 is interrupt 57 which is UART. Manual is unclear, says it's in the GPU interrupt table but that's probably a typo.
- How to change modes? Write to the mode field of CPSR register.
  Can either manually change the mode in the CPSR

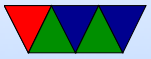


register (previous notes) or in theory the CPSID type instructions can be used to enable interrupts but also to change modes.

Can we trigger a hardware interrupt to get us into the hardware interrupt mode?

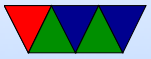
What about jumping directly to the interrupt vector?

- Subtract 4 because it offsets by four when saving the PC. Historical reasons? Probably artifact of pipelined processor where instruction mid-pipeline before interrupt comes in and PC already incremented. Computer architecture frozen in spec and then you're stuck with it.



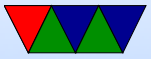
HW#3 Notes – Code

- Wasn't as picky this time, but comment code!
- Serial port: most got value right
  forgot to warn about floating point (say you wanted this to be parameterizable)
  Can you use floating point in kernel?
- printk: instead of /10, print remainder plus '0' instead /16 (which converts to shift) and two cases. 0-9 same as before, but A-F (you can just add 'A'-10 which I think is 55)

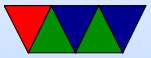


Beauty of ASCII. Often complicated use of ternary operator
technically upper vs lowercase %X vs %x
Can also use lookup table.
Be careful shifting, what if print 0xfffffff? Shift right? Be sure unsigned! Also in C, shift right by 32?

- Be sure print hardware info (r1)
- Be careful copying code from other places! A lot of the hex printing code is a bit obscure but a few implementations were almost byte-for-byte same as others. Try doing things yourself, not just doing a

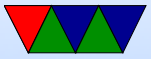


google search and cut-and-paste (or using an AI, or copying your friend, or whatever)



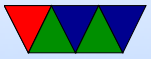
Midterm Review

- Closed book/notes/computer but can bring one piece of notebook paper (front only) with notes on it
- Questions will be similar to those from homeworks
- Topics
  - Benefits of an OS / Downsides of an OS
  - Serial communication: why are we using it? What does 9600 7E1 mean? How does hardware and software flow control work?
  - Boot process (firmware/bootloader/OS), how it works

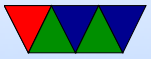


on Pi

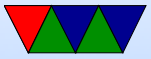
- MMIO interface: accessing hardware registers. Using inline assembly, the different starting addresses for I/O on various Pis. Using the interface to blink GPIOs
- Interrupts: how they switch processor mode, why FIQ is different from IRQ mode. How to switch back from userspace.
- System calls
- ABI
- Context switch / Scheduler. What is saved? Why does it have to be fast?



- Note: no memory or Virtual Memory as we haven't done that homework yet

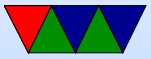


Brief History of Memory Handling in Operating Systems



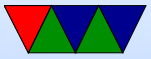
Mono-Programming

- Simple mono-programming: just OS and one program in memory at once (like DOS)
- Linear physical memory, assume you have all (or maybe up to a limit set by OS)
- Hassle of DOS 640k low memory, games

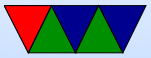


Fixed Multi-Programming

- Multiprogramming: let you run multiple tasks at once.
- Fixed Partitions of memory available. Jobs queued. When spot frees up job can run. Can have complex scheduling rules out which size and priority to give to jobs. Older mainframes (OS/MFT) used this.
- Relocations a problem
- Memory protection. Permissions on pages.

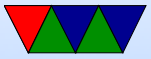


- Solution to both protection and permission in segments (with base offset and range that are valid to access)

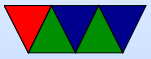


Swapping

- Timesharing systems. All jobs not fit in RAM?
- Swapping: bring in each program in entirety, run it a while, then when done writing all back out to disk.
- Paging: virtual memory.

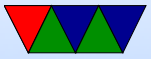


Memory Allocation in Linux



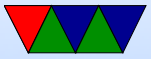
Buddy Allocator

- Used by Linux
- Pick a low size, say 4k, and a high size, say 1MB
- When allocate, round up to the next power of two
- Search for free area that size. If not, scale up. If you find one, split it into chunks until you reach the size being looked for. Give it.
- When freeing, not only free but see if neighboring blocks also free, if so, re-join them to bigger sized memory.



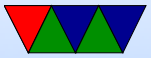
Buddy Allocator example

- Want to allocate 7000 bytes
- Gets rounded up to power of two, 8192 (8k)
- Look for free 8k block. If found just hand it out. If not, bump to 16k try again
- No 16k free, bump to 32k
- 32k found! Break it up to four 8k chunks, hand out one
- Later if that 8k chunk is freed, if nearby chunks are also free, merge them together to create larger chunk
- This is designed to limit fragmentation



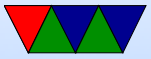
Linux – SLAB/SLOB/SLUB

- Have cache of commonly allocated structs
- Don't completely clear/free them when done, but leave them pre-initialized



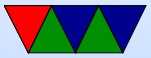
Linux – dynamic allocation

- Most code will try to do things on stack if possible
- Kernel stacks are small (why? Need to be contiguous, fragmentation, etc)
- Back in the day they tried to fit in one 4k page, not always work
  8k. 16k now?
- Part of the problem was large structs being allocated, but especially deep callchains. Sometimes be fine in common case but then some obscure thing calls



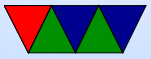
filesystem/network/network-card/etc and it all adds up

- Can allocate memory
  - kmalloc()
  - get_free_pages()
  - vmalloc() – allocated virtual memory, avoids fragmentation



Linux – Memory Zones

- Not all memory is equal
- Not all is in reach of DMA
- 32-bit processes can't access memory above 4G
- Normal vs HIMEM (historical on Linux?)
- NUMA – sometimes want allocations to be close to CPU core



Linux – Memory stats

- /proc/buddyinfo
- /proc/zoneinfo