CS 471/571 Operating Systems
2018 fall - Syllabus and Information

Table of Contents:
Links to - ***table of contents*****,** ***class notes*****,** ***video lectures*****,** ***slides/pictures***
textbook/notes for xv6**,** Jeff's xv6 for this class, OSTEP

Instructor/class Directories: cd ~jkinne/public_html/cs471-f2018/
Course website and information:
https://docs.google.com/document/d/1w2bGyWU5YWTlb4hvEJlla5cAlKsbaOcKoACBGC8F-Is/edit?usp=sharing

Note - information at the top of this document changes (the Study Guide and Notes). Information after the Notes (so, the syllabus) doesn't normally change once the semester starts.

Study Guide

- *Use this area to keep track of what has been covered in class. It can either be a list of topics, or more detailed notes that you've compiled*
- Plan for the semester - see course outline and learning outcomes in Syllabus below
- Operating system should provide
    - *Abstraction / virtualization* - so programming does not require low level details of dealing with physical drives, hardware interrupts, etc.
    - *Sharing / protection* - enable programs to share the physical resources in a fair way that also keeps them from interfering with each other
- *Physical hardware* - needs to be involved. Many of the things an OS wants to do cannot be done (or would be really difficult) without the right feature set in the CPU, etc.
- Abstraction / virtualization in the OS
    - *CPU* - abstraction is a "process" that includes the state of the CPU and memory as it runs, and it's resources (open files, etc.). *Context switch* - switch from one process to another. *Interrupt* - something hardware-wise needs servicing, so wake up interrupt handle to deal with it.
    - *Memory* - abstraction is "virtual memory" with each process having its own separate "copy" of memory. Also, breaking memory into "pages" that each can have different access rights (some owned by OS, others by process, etc.).
    - *Storage (HDD/SSD)* - abstraction is the file system, typically a tree-based directory structure of each FS.
- **Terms** - see http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/
- **xv6** - developed at MIT, interface is some part of the unix VI, basically a simple version of unix used for teaching at MIT, and lots of other places.
    - Documentation for qemu emulator, and commands in the qemu monitor
    - Git cheat sheet

- mkdir ~/xv6-mine/
  cd ~/xv6-mine
  git clone https://github.com/kinnejeff/xv6-isu-f2018.git
  cd ~/xv6-mine/xv6-isu-f2018/
  make
  make qemu-nox
- To run xv6 graphically-ish, run make qemu
- Note that the plain make builds xv6, and the make qemu-* starts qemu with xv6 loaded in.
- To print process list, ctrl-p.
- To exit xv6 (which is running inside of the qemu emulator), type ctrl-a x
- To go to qemu monitor, ctrl-a c, to go back ctrl-a c
- Making a new user utility
  - Create a file new_utility.c with your program
  - Edit the Makefile to include new_utility.c EXTRA variable, and _new_utility in the UPROGS variable
- Making a new system call mySysCall
  - usys.S -> add a new line SYSCALL(mySysCall) to the end of file
  - syscall.h -> new line at end of file #define SYS_mySysCall 23 // next #
  - user.h -> function prototype for mySysCall
  - syscall.c -> add extern int sys_mySysCall(void); line
  - syscall.c -> add [SYS_mySysCall] sys_mySysCall, line
  - In some file, define the sys_mySysCall function. Sysproc.c, or somewhere else.
  - How to do something with files, or memory, or the CPU - find the existing xv6 system call that is most similar to what you want do.
- Process table - array/DS with information about all processes. Rows in this table are struct proc's.

- **What the hardware gives us** -
  - - in the x86 architecture, and what we can do with it
  - *Memory address translation*
    - Convert from virtual addresses to physical addresses (using page tabled and page directory - two level tree)
    - Conversion is done in hardware
    - Kernel controls memory pages - good for user-level programs that don't need to worry about that, good for the OS to be able to keep user-level programs from reading memory they shouldn't (e.g., seg fault)
    - Memory page - block of memory that is all together in physical memory and in virtual memory. In xv6, this 4096 bytes.
    - Memory address translation: given the virtual address page (first 20 bits of virtual memory address), convert that into physical address page (first 20 bits of physical memory address).

- 32 bit memory address. Last 12 bits specify which byte within the page. Note that 12 bits gives 4096 different bytes. First 20 bits specify which page.
- How many different pages are there in VM? 20 bits. Around 1 million pages. Could have a lookup table of all 1 million possible VM pages.
- Xv6 - translation from VM to PM pages is a two-level tree. Root of tree is page directory. 10 bits in VM address to specify a row in the page directory. 1024 rows in the page directory. Entry in page directory points at a page table, page table is 1024 rows. Entry in page table gives 20 bits of PM address. How much memory does the two-level tree take?
    - Cool - be lazy, initially store in page directory that none of the memory is present/valid. First use of VM address wakes up the OS, and OS can find a spot for that VM to live. It's likely that process doesn't use most of the page directory entries; for those - we don't need to store a page table!
    - Note - page directory is 1024 rows, 4 bytes each, so 4096 bytes. Also, page table is 4096 bytes.
    - Virtual memory address examples
        - Main: 0x00000000
        - Global: 0x00000B78
        - Stack: 0x00002FEB
        - Heap: 0x0000AFF0
            - Heh - this is classic interview question type stuff
            - 0000 0000 00*00 0000 1010* **1111 1111 0000**
            - First 10 bits is index into page directory: 0
            - Next 10 bits is index in page table: 10
            - Last 12 bits is offset in page: 4080
            - Go to the process page directory, and get the 0th entry. The 0th entry gives a physical memory address for a page table. Go to that page table and get 10th row out of the page table. That 10th row has a physical memory address of a page (4096 bytes). Go to that page and get the 4080th byte.
            - How memory accesses does this take? (Hopefully 1 normally, with cachesing). But without cacheing - a read of the PDE (32 bits), a read of the PTE (32 bits), a read of the actual address we want (1 byte or whatever).
            - That's bad - 3x performance hit. So TLB - translation lookaside buffer. If TLB keeps the PDE and PTE in cache, then you have cache reads for that and 1 memory read.
- Key idea - page tables are created "on the fly", "on demand".

                - Process has an assembly instruction like
                  Store the value 1234 in address 0xB2AF
                  - row 0 in PD, row 11 in PT, offset 2AF
                    - On first access, need create a new PT for row 0 of PD.
                    - Create new PT, initialize present bits to 0
                    - In new PT, set row 11 to be a new page we allocate
        - *Interrupts*
            - In kernel mode can set interrupt handlers - specify where in memory the code is to handle device I/O, timer interrupt (for time slicing), etc.
            - Also specify interrupt handle for system calls
            - The first program to run when the computer boots sets up all the interrupt handlers - so user-level programs don't have to worry about this, and keep user-level programs from messing things up
            - Same setup used for exceptions (divide by 0, etc.).
        - *Locks*
            - Atomic CPU instructions to do test-and-set or similar as an atomic operation
            - Needed to avoid race conditions when either multiple threads/processes executing at once (multi-core) or are time-sliced in adversarial ways (can happen even on single-core)
- When writing OS code …
    - Careful not to let bad user processes cause us to do something bad. Must check parameters thoroughly, paranoidly
    - Careful about synchronization - we might get interrupted while executing, and some other process (a system call for that process) executes before we finish. Or, multiple processes accessing at same exact time.
- When a context switch happens, when a new process runs
    - Save state of the CPU for old process. Also the memory translation.
    - Load up the CPU registers for new proces. Alos the memory translation.
- Unix review …
    - printf(2, "hello\n") - 2 is fd for stderr. Also, 0 is stdin, 1 is stdout.

Notes (most recent first)

- Vocab for the future
    - "Working set"
- Coming up soon …
    - *When applicable, FYI to the students of what is coming next …*
    - Working with xv6 - utilities. Along the way (keep notes) - key unix/OS concepts. Eventually - quizzes about the key unix/OS concepts.
    - Requested videos
        - bit flags, how disks work, pointer stuff, time and what the stuff means
- Come back to at some point …
    - *When applicable, make note of anything I'd like to come back to at some point*

    - Compare/contrast xv6 memory management with: linux, DOS, windows 95, windows NT, mac os X, java VM. Do it for one of those, put together some slides or nodes, some terms to know, some simple problems to be able to solve, and come discuss with me. Value is a 6/3 HW or QUIZ.
    - Where is TLB in cache hierarchy? What about performance in the real world? How often do you have cache misses? What does OSTEP says?
    - Later - meltdown and spectre security vulnerabilities
- 12/12
    - Happy paper exam
    - Computer exam, due Sunday AOE, is at http://cs.indstate.edu/~jkinne/cs471-f2018/exam2_computer.txt
        - Right now it's 2 problems. I'll likely add a third tonight.
- 12/7
    - What you should know from OSTEP storage - http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/storage-terms.txt
    - Sample quiz 7 by Monday.
    - Final exam will be roughly (+/- 5% on each of these) -
      20% functions every citizen should know,
      10% unix/xv6 terms and such,
      30% OSTEP virtualization and virtual memory in xv6,
      20% OSTEP concurrency and pthreads,
      20% OSTEP storage
    - Ext2 optional assignment released by Monday if we're going to do that
        - Something along the lines of the Ext2 assignment(s) at http://cs.indstate.edu/CS471/
- 12/5
    - TBD - Quiz 7 - practice over storage. OSTEP chapters - 36-41, probably also 42-45 at a high level. For now, <= 10%
    - TBD - Hw6 - storage, due 12/16.
    - Quiz6 - you give it to me for quiz grade. 10 points.
- 12/3
    - Storage - goal is the basic structure of a reasonable file system (xv6 one)
    - Hw5. Concurrent hash table - ?
    -
- 11/30
    - Quiz 6 on OSTEP chapters 26-32. Practice quiz.
- 11/28
    - Plan for the final exam - 2 hour on paper final on Wednesday Dec 12, take-home computer honors-system final released on Dec 12 and due Dec 16 AOE.
- 11/26
    - Exams - fix blackboard
    -

    - Quiz 6 on 11/28 - OSTEP concurrency chapters (26-32). See http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/terms-concurrency.txt for what you need to know
    - HW 5 concurrency due 12/3. In progress making it up. See http://cs.indstate.edu/~jkinne/cs471-f2018/HW/hw5.txt
    -
    - Quiz 7 on 12/5 - OSTEP storage chapters (36-40 for sure) and *some reference on ext2* . Sample quiz by 12/3.
    - HW 6 storage due 12/10.
    -
    - How much of this on final exam - TBD. Sample final exam ready for 12/5.
    -
    - Optional review sessions - 11/30, 12/7, probably.
- 11/23
    - Computer exam grades are in. Check ~/EXAM1_COMPUTER/grade.txt to see where I got your score from. If you had late work that I did not find, let me know.
- 11/14
    - On computer exam
- 11/12
    - On paper exam
- 11/7
    - Yes, class
    - New hw's? Plan on having those to work on over Thanksgiving break.
    -
    - Sample exam
        - http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/exam1_sample.txt
        - How about on computer part?
            - Yes
            - 3 possible types of questions
                - Add a new user program to your xv6.
                - Add a new system call to your xv6, have the user program use it.
                - Regular program that uses some of the functions every citizen should know.
            - http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/exam1_computer_sample.txt
- 11/5
    - ~~Yes, class.~~ No class.
    - objdump -d -S, and compile with -g
    - Keep reading concurrency.
    - Review things we've done to see if you want more lecture or assignments in some area, to help prepare for the exam.
- 10/31

    - Hw4c - don't do anything except one memory update in your innermost loop. Don't call gettimeofday or anything like that. It would mess up your timing. Take a look at my results.
    - **New reading** - concurrency chapters in OSTEP (26, 27)
    - **New code, hw5**
        - Figure 26.6 (the 20 million program)
            - Copy/paste and run it, verify it behaves as claimed.
            - Your figure 26.6 code - put a lock around the counter +1 part.
            - Timing for the buggy version and the correct version.
            - More than 2 threads.
        - problems at end of chapter 27. Note that those require downloading some code from the OSTEP homework (linked from OSTEP, or directly here).
            - Get them in your account, make sure can compile and run.
            - Some modifications.
        - My first mildly interesting or useful multi-threaded code.
        - Thinking questions
            - For Figure 26.6, what is the max and min possible values that might be produced (instead of 20,000,000).
            - And others?
- 10/29
    - Why exam? (1) you study and learn, (2) you probably didn't cheat/copy/work-together
    - Quiz 5 returned (was actually 36 questions, calculated out of 33)
    - Quiz 4 - correction put in BB (your previous score * 60 / 45).
    - HW 4a, memory-user.c first version
        - People that had much less than 8GB with malloc return NULL, was anyone else logged in? Other programs running?
    - Grades - all up to date except for hw4c and letter grade. That will be wrapped up tonight. Ask me about it Wednesday...
- 10/24
    - **Big quiz 5 - 10 pointer**. Rahul is here to give quiz.
    - Jeff makes this quiz on 10/23 (morning), with preference for things already in the hw4d.txt's. Jeff - copy those over to .../STUDY/ before making the quiz.
- 10/22
    - Let's look at the hw4.*'s in class
        - Something else that is fair game on the big quiz - questions with numbers. For example, beginning of 20.1 in OSTEP. Note - first time you see this is "big quiz", will also happen with exam.
    - Quiz 4 returned. Note - I made corrections from Rahul's points (I gave you a few back). I had two questions with incorrect answers (PDE and PTE indices); corrected in the _answers.txt file. Highest score was ?
- 10/19

    - Work on stuff.
- 10/17
    - I'm around tomorrow except 10-11, not after 3pm. Friday 8-10, 2-4 in science.
    - Yes, we meet. Status check on hw3, hw4.
    - Quiz on OSTEP readings - 10/24
    - Hw 4a - you can try ~jkinne/…./HW/m
    - Hw 4c - 10/22
    - HW 4d - 10/22
    - Next up after VM - concurrency and pthreads (not xv6)
- 10/15
    - You met and discussed things.
    - Note - hw 4a and 4b due 10/16 - so you can come ask questions if you want
- 10/10
    - chat/Q&A system?
    - Turning off address space randomization, and possibly other things
    - New stuff - plan for the next 2 weeks
        - Memory chapters in OSTEP (13-23) - solidify some of the concepts in your brain, some good coding exercises that are not in OSTEP.
        - HW questions from OSTEP chapters 13, 14, 19, 21
        - HW to start working on - http://cs.indstate.edu/~jkinne/cs471-f2018/HW/hw4.txt
        - Note - no videos this time. We're going to try to do this part without me providing videos and see how it goes. Let's use some chat/Q&A system instead; suggestions?
            - Why? For people that have less access to good internet.
- 10/5, 10/8 - no class, nothing new
- 10/3
    - Working on things. Grade hw3b for those who are done.
    - Note Jeff's mistake in free.c (else if on flags).
- 10/1
    - No class
- 9/28
    - No class as usual.
    - Hw3b due
- 9/26
    - Quiz 4 - like quiz 3 but not matching
        - And also, draw figures - 2-1, 2-2, 2-3
        - See new xv6 terms in STUDY directory, chapter 2
- 9/24
    - Yes we'll meet. Work on your things, get things graded that need being graded.
    - Hw3a due
- 9/21
    - No class. Work on hw3, come show me what you have.

- 9/19
    - Hw3b due? Not a chance, we'll see what we think.
    - Meet today - you betcha. We'll go over what's supposed to happen in hw3, record some of it, have some fun.
        - Recording, it's public, I can pause.
    - Once you've finished everything, see come back to at some point for bonus points
    - How about an exam? Note - interim grades put in by Oct 2. No, not yet.
- 9/17
    - You meet without me to discuss hw 3a, hw 3b.
- 9/14
    - No class
    - Look at your quiz 3 and notes on any answers of mine that could be improved (even if just about misunderstandings).
    - Fridays 2-4pm - I will often be in the second floor of the Science Building where it overlooks the fountain. If I am I'll leave a note on my door, and you can come over there to ask me questions, etc.
- 9/12/2018
    -
      git commit
      git commit my_usage.c

- In the vim-ish editor that comes up
  i for insert
  Comment on what the udpate is
  esc
  :w enter
  :q enter
  git push
- **HW 2b** - show and tell, graded today.
- **HW 3b** - xv6 memory something more, 10 points, due 9/28
  - System call page_dir_dump that prints information about the page directory to the console. Note - each process has its own page directory, so you'll be printing information about the process that calls page_dir_dump
  - See http://cs.indstate.edu/~jkinne/cs471-f2018/HW/xv6-hw3/README
  - Points - +6 for -t or -p, +4 for the other, +1 for -f out of 10
  - Note - for -f, look for kmem.freelist … and only print the first 5 next's
- **HW 3a** - xv6 system call, 10 points, due 9/14
  - Something to do with memory. Not a big project yet, something smaller.
  - struct system_info {
    int num_procs; // total number of processes
    int uvm_used; // total amount of memory for all user processes
    int num_cpus; // number of CPUs, 2
    };
  - System call ***system_load*** that takes as a parameter a pointer to a struct system_info, and fills it in. Note - do something similar to Jeff's usage system call (in terms of defining a new structure, passing a pointer to that). Note - do something like allocproc in terms of accessing the process table. For the process count, count anything that does not have state UNUSED. For the memory count, += sz for all process that have state not equal to UNUSED.
  - Utility program free.c that uses the system_load system call and prints out the total amount of memory used. Make the utility handle the -h parameter. Also print the total # of processes and # of CPUs.
    - For -h, use K=1000, M=1000000, G=1000000000, only use G if the number is > 2*G, only use M if number is > 2*M, only use K if number is > 2*K.
  - Do this in your ~/xv6-mine, and when you're done with it, push your changes to your github account
  - My output when running free is
    #procs: 3, size(uvm) 40960, ncpu 2
  - My output when running free -h is
    #procs: 3, size(uvm) 40KB, ncpu 2
- 9/10/2018

    - Check your quiz 3, and if there are any mistakes let me know.
    - **Hw2c - quiz 2 corrections - checkpoint!!!!!!!!**. If I have a record of you finishing this, you have 5/5 in BB for this. If you don't have it in BB, then check with me or Rahul again.
    - New videos up - making a new xv6 system call, talking about hw 3a and process table, talking about hw 3b and virtual memory page.
    - Reading - memory chapter (2) in xv6 textbook, memory chapters in OSTEP (13-18)
    - Hw1's - I show you mine, maybe it helps.
    - Hw 2a, 2b - vote on whether you just tell me where your file is, or you create your own xv6 branch on github and send me the link to it.
        - If the latter, who is willing to give it a try to make sure it works first? A and J.
    - Remember
        - When allocating space for strings, +1 for the NULL character.
- 9/7/2018
    - No meeting, we'll never meet on Fridays.
- 9/5/2018
    - Yes, let's meet.
    - **HW grading**
        - I'm going to be very picky. See comments in the google doc for each assignment. I don't want to make a judgement call about what is okay or not. If it doesn't do what my program does or doesn't handle an important boundary case, 0 credit. Don't panic, but do watch the video for the assignment again to see what your program should do. And test both it and my program on different types of test files.
        - Also, you need proper style and comments, or 0 credit.
        - Tomorrow morning is on time for hw1c, hw1d.
    - Note virtual and physical addresses of devices - Figure 1-2, page 22.
    - **HW 2c** - 5 points, **checkpoint assignment** due 9/12
        - You complete this for full credit by 9/12 or you fail the class.
        - Take your graded Quiz 2, and make corrections to all of your mistakes. Get your corrections signed off by Rahul (see his lab hours) or Jeff by 4pm on 9/12.
        - Rahul - make note in /u1/junk/lab-assi
        - Note - if you had a perfect score on Quiz 3, then just show that to Jeff or Rahul.
    - Assigned reading - see Quiz 3.
    - **Quiz 3** - 5 points, today
        - keywords/terms from chapters 0-1, appendix A and first part of appendix B in the xv6 book. Supplement with chapters 4-6 from OSTEP. First matching, then fill in the blank. Terms to know - http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/terms-xv6-part1.txt

    - Latest video
        - Note - see in the study guide the important features of the hardware that are available to us (and really needed to do what we want) …
            - For example, see usys.S in xv6
        - More fun with xv6 …
        - Need for “atomic” operations - classic example
        - Virtual memory - proof that it’s happening
        - Faults - proof the xv6 is in control
- 9/3/2018
    - Labor day
    - HW’s - fix your HW files to have comments. You put something at the top saying what HW it was for, the file you started with, etc. Comments throughout as appropriate.
- 8/31/2018
    - Do we meet today? No. I give you more assignments.
    - **HW 2b** - 5 points, due 9/7/2018
        - Do what you planned to do.
        - Note - 0 points if your program doesn’t pass the “style test” - you should have comments for things, use good variable names, proper indenting/spacing, etc.
        - Create a file hw2b.txt in your ~/HW/ directory telling me the full path and name for your utility program.
            - Or, create your own git branch, and put a link to that in your ~/HW/hw2b.txt file … Vote on that?
    - **HW 2a** - 2 points, due 9/5/2018
        - Make a proposal for a program to make for xv6. It can be something useful or something fun. Something about as difficult as the primes program would be fine.
        - Get approval from me, and then put a text file
          ~/HW/hw2a.txt
          that says what you are planning to do (because I may forget)
- 8/29/2018
    - **HW 1f** - 2.5 points, due 9/4/2018
        - With someone else in our class (group of 2 or 3), come see me in my office some time by Tuesday. We can say hello, I can ask how you are doing, and hopefully you’ll feel comfortable coming to ask me questions when you have them.
    - New video - what is built in to xv6 and writing a utility for xv6.
        - Functions available to us - listed in user.h
        - Utilities available to us - those in printout from ls while in xv6
        - Steps to create a new utility
            - Do like one of the existing utilities, e.g., echo, so ...
            - Create new file myUtility.c

            - Make changes in - Makefile
        - Other fun things - make a game (higher/lower number game)
        - But, a little more interesting - something that requires using the system calls.
- New video - working with code in xv6, in particular writing utilities that use the xv6 system calls.
    - Xv6 can be compiled and run on the CS systems. Jeff's version is hosted on github.
    - To grab a copy first create a github account. Then, be logged on, and run
      mkdir ~/xv6-mine/
      cd ~/xv6-mine
      git clone https://github.com/kinnejeff/xv6-isu-f2018.git
    - Git cheat sheet
    - To compile, do
      cd ~/xv6-mine/xv6-isu-f2018/
      make
      make qemu-nox
    - To exit xv6 (which is running inside of the qemu emulator), type ctrl-a x
    - To run xv6 graphically-ish, run
      cd ~/xv6-mine/xv6-isu-f2018/
      make
      make qemu
    - Note that the plain make builds xv6, and the make qemu-* starts qemu with xv6 loaded in.
    - Documentation for qemu emulator, and commands in the qemu monitor
- New video - high level overview of what the OS is supposed to do, what things look like for unix process, xv6 system calls.
- Quiz 2 - 5 points, same as quiz1, but fill in the blank.
- **HW1b** - if it didn't look right, I gave you a 0. Get it fixed today or tomorrow so I can give you the points.
- **HW1c, HW1d, HW1e** - I won't auto grade these. When you think you're done with any of them come see me to take a look and let you know. One due per next FMW - see below. When you think you're stuck, or want me to look at something - come see me.
- **HW bonus point** - if you're finished with any of HW1c, HW1d, HW1e by the end of today and have it completely correct, I'll score it as 6/5 for that problem (they're each worth 5 points). And we can let the rest of the class know who they can ask for help other than me.
- Okay, so the plan for readings, assignments, etc. will be -
    - xv6 - is a unix-like OS developed at MIT for learning OS concepts and practicing writing OS code. It's widely used, so we'll be in good company by using it here. We'll get started with setting it up on CS later this week.

        - Readings - follow along xv6 documentation (specifically, the textbook/notes for xv6) for a while to understand the main tasks and concepts in OS's. Supplement with OSTEP as needed.
        - Assignments - writing utilities for xv6, then making small changes to xv6 kernel, then onto the "big 3" projects we have planned. The ext2 project we'll do on CS. Some of the others we may do in xv6.
        - That's the idea. I'll launch all that with a series of videos probably on 8/27. For now, finish up your HW1 stuff, and start reading the xv6 textbook/notes.
- 8/27/2018
    - We don't meet. I'm in my office.
    - Recommended reading - Introduction (chapter 2) of http://pages.cs.wisc.edu/~remzi/OSTEP/ - it is a pretty light read, and you'll know whether you like their writing style
    - **HW 1e** - 5 points, due 9/5
        - cp ~jkinne/public_html/cs471-f2018/CODE/run.c ~/HW/run_hw1e.c
        - run_hw1e.c should be modified to …
            - If there is a command-line argument then prepend this to the command when trying to run. So, the program could be run like ./run_hw1e /bin/
            - Before the call to execve, properly parse the command into the program name to run and the arguments. Note - it's enough to handle spaces. Don't worry about escape characters, what to do with quotes, etc.
            - If there is more than one command-line argument then treat them as a sequence of paths to try looking for the program to run in execve.
            - Functions that might be useful - strtok, realloc, stat
        - *Video explaining and giving a demo is in the video playlist.*
        - Grading …
            - 2/5 - one of the features asked for works correctly.
            - 3/5 - two of the features asked for work correctly.
    - **HW 1d** - 5 points, due 9/3
        - cp ~jkinne/public_html/cs471-f2018/CODE/head.c ~/HW/headTail_hw1d.c
        - headTail_hw1d.c should print the first 10 lines, then …, then the last 10 lines, then print *********, then print the total # of lines, words, and characters (separated by \t)
        - Hint - for counts, do the same as wc. wc counts \n.
        - Hint - functions you could use that might be helpful - getline, strtok (with delimiters " \t\r\n"), isspace
        - Test files to try - look in ~jkinne/public_html/cs471-f2018/HW/, also always try /u1/junk/kinne/shakespeare.txt
        - *Video explaining and giving a demo is in the video playlist.*

        - Grading
            - 0/5 if it doesn't conform to my spec in some way. Examples - missing a \n at the end of your output, doesn't give same output as mine on test1.txt or test2.txt or test3.txt
            - 0/5 as with all HW's if you don't comment appropriately and use proper indentation and white space.
            - 3/5 if yours works but doesn't handle files with < 20 lines properly. Or the wc part works but the printing lines doesn't.
            - 4/5 if yours works on all types of cases, but doesn't quite match my formatting.
    - **HW 1c** - 5 points, due 8/31
        - mkdir ~/HW/
        - cp ~jkinne/public_html/cs471-f2018/CODE/cp.c ~/HW/cp_hw1c.c
        - cp_hw1c.c - should work like
          ./cp_hw1c src dest bufferSize syncOrNo
          Where bufferSize is the size to use in the read and write calls, and syncOrNo is either O_SYNC or NORMAL. bufferSize is an integer at least 1. If syncOrNo is O_SYNC, then the O_SYNC flag should be given in the call to open.
        - Do the following testing on one of the machines y25, y26, …, y36
            - ssh gh101xy@cs.indstate.edu
              ssh gh101xy@y25
            - Create test files in /tmp that are 1KB, 1MB, 10MB
            - For each, run your cp_hw1c with bufferSize being each of 1, 1000, 1000000, and with syncOrNo being both O_SYNC and NORMAL.
            - That means for each test file you run 6 tests. For each one, run it with time and save your times (use the real time). Put the 6 times into a table. Do that for each of the test files.
            - And what does it look like?
        - *Video explaining and giving a demo is in the video playlist.*
        - Grading…
            - 0/5 - code does not properly handle an important boundary case. Ones people have gotten wrong - not handling too few command-line arguments, file size not multiple of buffer size.
            - 0/5 - code doesn't conform to my spec in some other way. Example - you're using 0/1 for argument instead of O_SYNC/NORMAL.
            - 0/5 - comments not up to the standards I said to do.
            - 0/5 - file not in right place, doesn't have correct name (capitalization matters) - cp_hw1c.c.
            - 4/5 - didn't give me the table with running times. Do it, it's kind of important !!!
- 8/24/2018

    - Note - this stuff is interesting/exciting. Really.
    - Link to youtube playlist above. First videos are up.
    - **Quiz 1**
        - 5 points - functions every citizen should know - know a one-liner description of what each does and what the return is. Quiz will be matching.
        - Note - “formatted print to stdin” -> “formatted print to stdout”
        - Answers to quiz1_sample are in http://cs.indstate.edu/~jkinne/cs471-f2018/STUDY/
    - **HW 1a**
        - **1 point - checkpoint assignment due 8/24/2018** - login with your cs471 login before end of the day on 8/24/2018, and chfn to put your name in.
        - Everyone except cs47101 is done.
    - **HW 1b**
        - 5 points - due 8/27/2018 - install VirtualBox on your laptop or home computer and a Linux VM installed (either by using a ready-made image [e.g. Kali], or installing from an iso file [e.g., lubuntu, mint, or slackware]).
        - mkdir ~/HW/
          ~/HW/hw1b.png - screen of virtualbox on your computer with linux running inside of it.
- 8/22/2018
    - Attendance - login with your cs471 account within the first 5 minutes of class.
    - Names
    - The plan - be like Exoo.
    - Decision - video lectures online, possibly skip some face to face meetings? Give it a try for a few weeks?
        - Let’s give it a shot.
    - The plan for now - read the unix paper
        - And questions on Friday / Monday! Good questions, not “I don’t understand”
    - Practice with functions-every-citizen-should-know, so we can get ready to do the ext2 project
        - whatis exit
          man 2 exit
        - Know the one liner description and what the return value means.
    - For review of connecting to the CS server, using the shell see appropriate videos (#s 2-7, 16, 23, 24) in https://www.youtube.com/playlist?list=PLXFP6J47Bp0dB2rRysTTkVVr4F_tDBnPR
        - Shell commands - ls, cd, mkdir, rm, rmdir, mv, grep, head, tail, less, top, ps, kill, rsync
        - Shell punctuation < > & | ; ctrl-c ctrl-d
        - See also unix linux cheatsheet, and others

- What you need to do in this course
    - Pay attention to lectures, ask questions when you don't understand something,
    - *A summary of different things that need to be done and what the normal schedule will be (e.g., an assignment due on Tuesday and a quiz on Thursday most weeks).*
    - *For online courses, definitely include all the different things that need to be done in the course*

Table of Contents

Syllabus in fall 2018 for CS 471/571 Operating Systems General Information

Contact Your Instructor

Name: *Jeff Kinne*
Email: *jkinne@cs.indstate.edu*
Phone: *812-237-2136*
Office: Root Hall, *A-140D*

Lecture, Exam, Office Hours

Lecture: *MWF 3-3:50pm in Root Hall A-017*

Exam: *Wednesday Dec 12 3pm*. Also check the Office of the Registrar's exam schedule

Instructor Office Hours: *10:30-noon TWR, and normally in roughly 9-4 with about half of any given day taken up with meetings*

GA Tutoring: See http://cs.indstate.edu/info/labs.html

Website:
https://docs.google.com/document/d/1w2bGyWU5YWTlb4hvEJlIa5cAlKsbaOcKoACBGC8F-Is/edit?usp=sharing

Prerequisites

A grade of C or better in CS 202 or consent of instructor.

Note - A 500 level course cannot be taken as part of the MS program if you have completed the same 400 level course at ISU. For example, if you completed CS 451 as an undergrad, you **cannot** take CS 551 as an MS student and count it towards the degree.

Recommended and/or Required Text

There is no required textbook. Readings will be assigned from online sources, including the following.

- The Unix Timesharing System - the original unix paper. Everything still is pretty much as it describes.
- Operating Systems: Three Easy Pieces - very nice textbook-style free book written by some experts in systems who are great teachers as well.
- The Little Book of Semaphores

- The ext2 File System

Course Announcements

Announcements regarding the course will be made both during class and via email to your @sycamores.indstate.edu email address. You should regularly check this email account or have it forwarded to an account that you check regularly.

Classroom conduct

You may not use cell phones, iPods/music players, etc. during class. You should be civil and respectful to both the instructor and your classmates, and you should arrive to class a few minutes before the scheduled lecture so you are ready for lecture to begin on time. You may use your computer during class if you are using it to follow along with the examples that are being discussed. You may not check email, facebook, work on other courses, etc. during class.

Course Description

The official description of this course from the catalog is

"Major topics include system structure, memory management, and process management. Hands-on experience using the department's minicomputer facilities are an important part of the course."

An Operating System is a complex computer program that mediates access to three essential resources: CPU, memory, and devices. The main goal of this class is gain an understanding of how these three objectives are met. A secondary goal is to understand how some of the ideas which originated as solutions to operating systems problems have contributed to solutions to other problems.

Course Outline

- Review - functions every citizen should know, basic data structures in C (1-2 weeks)
- Device Management, typically focusing on storage devices and filesystems (3 weeks)
    - ext2 file system in detail
    - Other filesystems at a high level
- Process/Thread Management (3 weeks)
    - Single CPU execution - time slicing, scheduling
    - Multi-core execution
    - Shared data - semaphores, other mechanisms, synchronization issues, race conditions
- Memory Management (3 weeks)
    - Memory organization, virtual memory, pages, page faults, CPU cache
- Other Topics, as time allows

We are likely to have one “project” for each of - device management, process/thread management, memory management - where you will complete code that we start in class to write a major part of the operating system.

Learning Outcomes

1. Source code for unix system calls - can trace through and explain the code, can make modifications.
2. Unix system calls - can write programs that use the calls to mimic unix utility commands.
3. Synchronization primitives - can explain proper use and write correct code that uses primitives to avoid race conditions or logical errors.
4. Common scheduling algorithms - can explain common algorithms and tradeoffs.
5. Process accounting - can explain in detail at least one process accounting system.
6. File systems - can explain in detail the layout of at least one modern file system, can explain basic properties and tradeoffs of a number of others.
7. Virtual memory - can explain fundamental concepts and their impact on performance, can explain in detail the mechanism of at least one modern memory management system.
8. Modern computer and operating system - can explain the basic architecture of modern computers and operating systems, including how an operating system ensures proper sharing of the following between different programs and users - CPU, disk, memory, other devices.

Expected Amount of Work

If you take this class seriously and get what you should out of it, some weeks you will likely be spending around **10 hours/week** or more on the class. The students who get A’s in their CS courses and have an easy time finding jobs do spend this much time on this course. Not everyone would need to spend this much time and not all weeks will be the same, but you should plan on putting in whatever time it takes.

Note - your classes should be more important than your part-time job.

Grading and Assignments

The students of this course have the following responsibilities: read assigned readings before lecture, attend lecture, complete homework assignments, take in-class quizzes, take exams, and complete projects.

Your grade in each of the following categories will be calculated. Your “total” grade will be the minimum of these.

1. **Exams** - we will have three exams, which will all be cumulative. The total exams grade will be max(exam3, .6*exam3 + .4*exam2, .5*exam3 + .3*exam2 + .2*exam1).
2. **Projects** - three of the homework assignments will be labeled as projects. The projects grade will be the simple average of the three project grades (each of the three projects is ⅓ of the project grade). Each project will receive a score for style, correctness, inclusion of required features, and inclusion of advanced features.
3. **HW** - each HW is given a number of points, and the total HW grade comes from just adding up all of the HW's (so HW's worth more points are worth more in the total HW grade)
4. **Quiz** - same as with HW's, but for in class quizzes.
5. **Attendance** - each lecture you should login to the machine. I have a script that runs to check who has logged in, and your total attendance will be the % of the time you were present and logged in within the first 5 minutes of the start of class.

Note that if you miss more than 20% of the lectures, you will receive an F for the course. This course has 45 lectures, so if you miss 9 lectures you fail. I will not give you a warning about this - if you are likely to miss more than a few lectures during the semester you should keep track for yourself how many you have missed.

CS Course Policies

Note that this course follows all standard CS course policies. In particular check the CS course policies related to - cheating/plagiarism, attendance, missing exams. See http://cs.indstate.edu/info/policies.html for details.

Late Homeworks

When an assignment is given, the assignment will state if late work will be accepted. For some assignments, a "full credit" date and "late credit" date will be given. By default, "late credit" is at most ½ of the points. If no "late credit" date is given for an assignment, then no late credit is given for the assignment.

Some key assignments will be labeled as "**checkpoint assignments**". For these assignments, if you do not complete the assignment correctly by the due date, **you fail the course**. These assignments are labeled as "checkpoint assignments" to make it clear to you that you cannot pass the course if you do not complete them on time.

Start Homeworks Early

We suggest attempting a homework assignment the day it is given, or the day after, so that if you have a problem you can ask early. If you continue to have problems in trying to complete the assignment, you will have time to ask again. Many of the homework assignments require thought and problem solving, which takes "time on the calendar" not just "time on the clock". By

that we mean that spending two hours on 3 consecutive days may be more productive than trying to spend 6 hours at once on the assignment.

Grade Cutoffs

We try to design homework assignments and exams so that a standard cutoff for grades will be close to what you deserve. After the first exam a grade will be created in Blackboard called "Letter Grade" that is what your letter grade would be if the semester ended today. Initially, I will likely assign the following grades: 93-100 A, 90-93 A-, 87-90 B+, 83-87 B, 80-83 B-, 77-80 C+, 73-77 C, 70-73 C-, 67-70 D+, 63-67 D, 60-63 D-, 0-60 F

Our goal is that the different grades have the following rough meaning.

A+/A
You can do *all* the assignments *on your own*.

B+/A-
You understand nearly everything, and should be all set to use this knowledge in other courses or in a job.

B-/B
Most things you understand very well and a few you might not (more towards the former for a B and more towards the latter for a C).

C/C+
Learned enough and have the minimum skills to move on in the subject.

D+/C-
You did put some effort in, and understand many things at a high level, but you haven't mastered the details well enough to be able to use this knowledge in the future.

D-
Students will normally *not* get an F if - you attend 80% of the lectures, complete some of the assignments up through the end of the course, and get nearly half of the problems on the final exam correct.

F
Normally, students that get an F simply stopped doing the required work at some point.

Blackboard

The course has a blackboard site. Click here to go to blackboard. You should see this course listed under your courses for the current term. The blackboard site is only used for giving you your grades (go to the course in blackboard, then click “My Tools”, and then “My Grades”). All course content, schedule, etc. is kept in this google doc (which you are currently viewing).

Academic Integrity

Follow the standard CS course policies in terms of what is and is not allowed on assignments: http://cs.indstate.edu/info/policies.html

Please ask the instructor if you have doubts about what is considered cheating in this course.

Special Needs / Student Disabilities

Standard language included in the syllabi for ISU courses.

Indiana State University recognizes that students with disabilities may have special needs that must be met to give them equal access to college programs and facilities. If you need course adaptations or accommodations because of a disability, please contact us as soon as possible in a confidential setting either after class or in my office. All conversations regarding your disability will be kept in strict confidence. Indiana State University's Student Support Services (SSS) office coordinates services for students with disabilities: documentation of a disability needs to be on file in that office before any accommodations can be provided. Student Support Services is located on the lower level of Normal Hall in the Center for Student Success and can be contacted at 812-237-2700, or you can visit the ISU website under A-Z, Disability Student Services and submit a Contact Form. Appointments to discuss accommodations with SSS staff members are encouraged.

Once a faculty member is notified by Student Support Services that a student is qualified to receive academic accommodations, a faculty member is obligated to provide or allow a reasonable classroom accommodation under ADA.

Disclosures Regarding Sexual Misconduct

Standard language included in the syllabi for ISU courses.

Indiana State University fosters a campus free of sexual misconduct including sexual harassment, sexual violence, intimate partner violence, and stalking and/or any form of sex or gender discrimination. If you disclose a potential violation of the sexual misconduct policy I will need to notify the Title IX Coordinator. Students who have experienced sexual misconduct are encouraged to contact confidential resources listed below. To make a report or the Title IX

Coordinator, visit the Equal Opportunity and Title IX website:
http://www.indstate.edu/equalopportunity-titleix/titleix.

The ISU Student Counseling Center – HMSU 7th Floor | 812-237-3939 | www.indstate.edu/cns
The ISU Victim Advocate – Trista Gibbons, trista.gibbons@indstate.edu
HMSU 7th Floor | 812-237-3939 (office) | 812-230-3803 (cell)
Campus Ministries - United Campus Ministries | 812-232-0186
http://www2.indstate.edu/sao/campusinistries.htm
www.unitedcampusministries.org | ucmminister2@gmail.com
321 N 7th St., Terre Haute, IN 47807

For more information on your rights and available resources
http://www.indstate.edu/equalopportunity-titleix/titleix