02159 Operating Systems, Fall 2014
Learning objectives

August 31, 2014

The learning objectives are very important. They, not the literature, define the course! Make sure you understand them!

The course assumes that you already are proficient with the following objectives:

- You can explain how the compiler, assembler and linker are used to create executables.
- Given reference material, you can implement C programs with pointers, pointer arithmetic, arrays, structs, memory management and low level I/O.
- You can explain in your own words how static, automatic and dynamic memory management can be used to handle memory in C programs.
- You can explain in your own words what a C pointer is, its relation to arrays in C and the result of pointer arithmetic.

The objectives for the 02159 operating systems course are:

1. You can list and describe three source code transformations the compiler can perform to achieve higher performance.
2. You can apply standard programming methodologies and tools such as test-driven development, build systems, debuggers.
3. You can explain the role of each component of a compilation toolchain used in system programming and how the components interact.
4. You can explain in your own words how memory leaks can occur in systems with dynamic memory management and give examples of two approaches to prevent them.
5. You can explain the difference between communicating with an I/O device using memory mapped I/O or I/O instructions.
6. You can explain in your own words the following terms: processor, register, program counter, cache, I/O device, instruction set architecture.

7. You can explain the basic workings of a processor and outline the steps involved in executing an instruction.
8. You can explain in your own words how interrupts are handled in the processor.
9. You can show with assembly code how the stack pointer, or a general purpose register, can be used to perform indirect addressing of data structures.
10. You can draw a figure showing the memory hierarchy of a modern computer with the processor, cache levels, memory and storage.
11. You can write pseudocode showing how to implement an interrupt handler.
12. You can demonstrate with pseudo code how the memory access pattern of a program can greatly influence cache performance.
13. You can explain the concept of privilege levels and show three examples of operations which should be permitted only at the most privileged level.
14. You can explain in your own words and in text and with your own commented examples, how message passing can be used to coordinate processes.
15. You can explain in your own words the differences between a monolithic operating system, a micro-kernel operating system, and an exo-kernel based operating system.
16. You can explain in your own words how multiple processes can share the same CPU.
17. You can explain in your own words how multiple processes can share a set of CPUs.
18. Assume a thread can be in three different states: running, blocked, ready. You can draw a diagram of your own showing the transitions between states and explain in text and in your own words what each transition means.
19. Given a schematic overview of the software stack of an operating system, you can label, in the diagram, key system parts. You can also explain in your own words and in text the purpose of those parts.
20. Given reference literature, you can write down in text and in your own words definitions or explanations of the following concepts: Process; Address space; Interprocess communication; System call; Daemon; Thread; Critical section; Mutual exclusion; Semaphore; Mutex; Monitor; Condition variable; Message passing; Kernel mode; User mode; Pre-emtiveness; Race condition; Scheduling algorithm; System time; Device driver.
21. You can explain in your own words, in text and with your own examples how race conditions can occur in systems with multiple processors or systems with interrupts.

22. You can explain in your own words, in text and with your own commented examples, how busy wait methods can achieve mutual exclusion in systems with multiple CPUs.
23. You can explain in your own words, in text and with your own commented examples, how blocking methods can achieve mutual exclusion in systems with multiple CPUs.
24. Given reference literature, you can write down in text and in your own words definitions or explanations of the following concepts: Virtual machine; Virtual machine monitor; Atomic action; Swapping; Virtual memory; Paging; File system; File attribute.
25. Given a skeleton operating system, you can implement operating system calls.
26. Given a skeleton operating system, you can implement code for process creation and termination.
27. Given a skeleton operating system, you can implement a scheduler.
28. Given a skeleton operating system, you can implement a memory management subsystem.
29. Given a skeleton operating system, you can implement an I/O device driver.
30. You can present your work in writing.
31. You can solve ill-structured, ill-formed, open problems related to operating systems.
32. You can, on your own, find information needed to solve problems.
33. You can use English as a working language.