USP= “UNIX System Programming, 2nd edition” by Haviland, Gray and Salama, Addison-Wesley 1999. The lecture number indicates when the assignment is DUE.


USP: Chapters 1, 2 and 9 through sec. 9.3.6 (Use of system calls related to files for projects 0 and 1.

Project 0: Program in UNIX to read/write on standard input and output; count characters and words. Read: “Guidelines for Programming Project 1 and others”, and How to Use turnin”.

Homework 1: Create a glossary for italicized terms found in the ULK readings.

Lecture 7 ULK: p.28-46 (Memory overview, Memory addressing and hardware)

OSCJ: Sections 4.1 to 4.4 (Process concept, scheduling, operations and cooperation); p.331-333 (Basic paging), p.336-337 (Basic hardware support for paging), p.355 (80x86 or IA-32 or “Intel Pentium” address translation hardware, segmentation and paging); and all of Chapter 3 (Operating system structures.)

Homework 2 questions:

1. Compare (a) Figure 9.6, “Paging Hardware” on p.332 of OSCJ with (b) Figure 9.21, “Intel 80x86 address translation” on p.355 of OSCJ and with (c) Figure 2-6, “Paging by 80x86 processors” on p.45 of ULK.
   (a) What important hardware register is missing from Figure 9.6? State its generic name and state which control register it is on the 80x86 (or IA-32).
   (b) Hardware support for segmentation is one feature of IA-32 architecture not included in the generic hardware described by Figure 9.6. Briefly describe another difference between the generic hardware with the IA-32 hardware.
   (c) Why are “logical addresses” different from “linear addresses” on the IA-32, but there is no difference between these and “virtual addresses” on generic hardware without hardware support for segments?
   (d) What IA-32 hardware register is loaded with a new value in order to carry out the action of changing the interpretation of virtual addresses? This action is one of the major operations executed to perform a context switch.

Lecture 12 ULK: p.28-46 (Memory overview, Memory addressing and hardware)

OSCJ: Sections 4.1 to 4.4 (Process concept, scheduling, operations and cooperation); p.331-333 (Basic paging), p.336-337 (Basic hardware support for paging), p.355 (80x86
or IA-32 or “Intel Pentium” address translation hardware, segmentation and paging); and all of Chapter 3 (Operating system structures.)

Homework 3 questions: Draw with pencil or graphics software two neat sequence diagrams like Figure 4.3 in AOS to illustrate each history:

1. One process makes a non-blocking system call to the operating system. The operating system returns directly to that process after handling the system call.

2. One process makes a blocking system call. The OS runs a second (ready) process. This second process is interrupted by the event caused by the I/O device where this event enables the blocked system call to complete. (You must show a column for the I/O device and indicate when the I/O device sends an interrupt that interrupts the second process.) The OS reschedules the interrupted 2nd process because it has high priority. After a timer interrupt, the OS makes the second process be idle (in the ready state) and makes the previously blocked system call return to the original process.

Lecture 9 Programming Project 1: (1) A character translation and character/word counting filter. (2) Combine (1) with opening named files, and explore/answer questions about their differences and configuration changes of the terminal driver. (3a) Make (1) a TCP server and test it with telnet. (3b) Write your own client for (3a) which acts like (2); try to find and explore shortcomings.

Reading: Haviland 9.1, 9.2, up to (9.3.6).

Main Goal: Gain experience with blocking and concurrent activities.

Lecture 12 Readings: OSCJ Chapter 5; Section 6.1, 6.8; begin Chapter 7 including 7.1, 7.2, 7.3, then jump to section 7.8, then go back to unread sections.


Lecture 14 Homework 4 Problem 1: Analyze all interleavings of 2 atomic steps performed by 2 threads.

Lecture 15 Homework 4 Problems 2-3: (2) Critical analysis of Peterson’s solution. (3) Critical analysis of solutions to the bounded buffer producer/consumer problem: (a) With semaphores; (b) With Java facilities (similar to monitors).