CS3844 Operating Systems

Dr. Walter Schilling

Winter, 2013-2014

For the midterm exam, you may bring one 8.5 x 11 inch sheet of paper with notes.

1. Week #1
   (a) Lecture #1 (An Introduction to Operating Systems)
      i. Draw an abstract view of a computer system.
      ii. Explain the difference between the user view and the system view of an operating system.
      iii. Explain the difference between the kernel, systems programs, and applications programs.
      iv. Draw a representation of a modern computer system.
   (b) Lecture #2 (System Calls)
      i. List the two modes of operation for a microprocessor
      ii. Define the concept of a trap
      iii. Explain how a system call is made
      iv. Describe various methods for handling parameters as they are passed to System calls.
      v. List some examples of system calls
      vi. Understand how a system call may manifest itself in x86 assembly language code.
      vii. Draw the C flow of C compilation from source code to object code.
      viii. Explain the purpose for the preprocessor, compiler, and linker within the C compilation model
      ix. Using the gcc compiler, generate the output for the preprocessor stage of compilation
      x. Explain the concept of a dependency.
      xi. Create a GNU Make file which automatically generates dependencies, creates preprocessed source code, and links a given C application.
   (c) Lecture #3 (Operating Systems Structures)
      i. Compare and contrast simple structured operating systems, layered operating systems, microkernels, and module based operating systems.
      ii. List the limitations of the MS-DOS operating system.
      iii. Draw a picture for a layered operating system.
      iv. List the advantages of a layered operating system.
      v. List the problems of designing a layered operating system.
      vi. Explain the fundamental purpose for the microkernel within a microkernel based operating system.
      vii. Explain the concept of a module based operating system
      viii. List the advantages and disadvantages of each approach to operating systems design.
      ix. Explain how to change the executing shell on a Linux machine.
      x. Explain the relationship between a shell and a process in Linux.
2. Week #2
   (a) Lecture #1 (Processes)
      i. Define the term process
      ii. Define the terms job, user programs, and tasks.
      iii. Explain the contents of the text section, data section, heap, and stack of a program
      iv. Draw a graphical representation of a process in memory
      v. Explain the concept of process state
      vi. Draw a state transition diagram for process states

vii. List the contents of a process control block
viii. Explain what the process scheduler is responsible for doing within the operating system.
ix. Explain the concept of process dispatching
x. List the processes executing on a Linux workstation.
xi. Explain how the kill program can be used to terminate an executing program.

(b) Lecture #2 (Process Context Switching)

i. Explain how a CPU Context switch occurs
ii. Explain how the hardware may impact the time necessary for a context switch (i.e. Sun Ultra Sparc)
iii. List reasons why a context switch would occur
iv. Explain why context switching can be bad
v. Compare and contrast IO Bound and CPU Bound processes
vi. Explain how the OS performs a context switch
vii. Construct code which switches the stack as part of an OS context switch
viii. Understand how process state is stored within a process control block
ix. Explain how to terminate a process

(c) Lecture #3 (No class - Rockwell Collins Trip)

3. Week #3

(a) Lecture #1 (Process Operations)

i. Explain the concept of the parent - child relationship between processes
ii. Explain the purpose for the UNIX fork, wait, and exec commands.
iii. Explain what happens when the exit method is called.
iv. Explain the concept of cascading termination
v. Explain the purpose for the UNIX fork, wait, and exec commands.
vi. Construct programs using the fork, wait, and exec unix commands

(b) Lecture #2 (Interprocess Communications)

i. Explain why processes should be allowed to operate in parallel
ii. List two methods for sharing information between processes
iii. Draw graphical representation for processes communicating using shared memory and message passing systems.
iv. Explain the flow necessary to use a shared memory partition
v. List and define the 4 main synchronization types that can be employed with message passing systems
vi. List the advantages of each mechanism for interprocess communication
vii. Construct a rudimentary program which uses pipes to communicate between processes.

4. Week #4

(a) Lecture #1 (Interprocess Communications (Sockets and RPC))

i. Define a socket
ii. Define the acronym RPC
iii. Explain how an RPC executes, specifically in regards to stubs and the concept of marshaling.
iv. Explain the difference between “big-endian” and “little-endian”.
v. Construct a simple application in Linux using RPC.
vi. Communicate between two programs using an RPC call.

(b) Lecture #2 (Threads)

i. Explain the concept of a thread
ii. Draw a representation of a single threaded process and a multi-threaded process.
iii. Compare and Contrast the advantages and disadvantages of threads versus processes
iv. Explain how multi-threaded program can be useful in a multi-core environment.
v. Explain the difference between kernel threads and user threads
vi. Explain the difference between many to one, one to one, and many to many models of thread behavior
vii. Using C, construct a simple multithreaded POSIX compliant application.

(c) Lecture #3 (Thread Problems)

i. Explain the interaction between threads and fork?
ii. Explain the difference between asynchronous and deferred cancelation.
iii. Explain the risks of improper termination of threads
iv. Define the concept of a UNIX Signal.
v. Explain the concept of a thread pool
vi. Explain the concept of a UNIX signal.
vii. Explain how a UNIX signal can be handled.
viii. Construct rudimentary programs which handle a UNIX signal.

5. Week #5

(a) Lecture #1 (Process Synchronization)

i. Define race condition.
ii. Define critical section.
iii. Explain the design ramifications of a preemptive kernel versus a non-preemptive kernel in terms of critical sections.
iv. Define mutual exclusion
v. Define an atomic operation
vi. Explain the problem of deadlocks and starvation
vii. Explain the problem of priority inversion and justify why priority inversion solves this problem
viii. List 4 mechanisms that can be used to prevent deadlocks.
ix. Explain the difference between a mutex and a semaphore.
x. Explain the concept of a counting semaphore.
xi. Perform basic synchronization using PThreads mutexes.
xii. Use semaphores to perform basic synchronization operations.

(b) Lecture #2 (Process Synchronization - Implementation)

i. Be able to construct a basic synchronization primitive in C source code.
ii. Explain the purpose for each entry within a semaphore structure.

(c) Lecture #3 (The Dining Philosophers)

i. Construct a resource allocation graph.
ii. List the conditions necessary for a deadlock to occur.
iii. Explain the dining philosophers problem and how it results in a potential deadlock.
iv. Construct a resource allocation graph from a given problem description.
v. Analyze a resource allocation graph to determine if a deadlock is present within the system.
vi. Analyze source code to determine if a deadlock is present based on execution traces.

6. Week #6

(a) Lecture #1 (Scheduling)

i. Explain the CPU and IO Burst cycle used for scheduling
ii. Recognize the distribution of CPU activities on a system
iii. Explain the relationship between an IO bound program and CPU bound program in terms of CPU bursts
iv. List the four reasons why the scheduler may be invoked
v. Compare and Contrast Pre-emptive and non-preemptive scheduling. What are the advantages of one system versus the other, and how is the operating system different based on the two approaches?
vi. Explain the purpose for the dispatcher and scheduler within the operating system.
vii. Define dispatch latency
viii. Define CPU utilization, Throughput, Turnaround time, Waiting time, Response time in terms of their impact on scheduling.

(b) Lecture #2 (FIFO Scheduling)

i. Explain the operation of a FIFO scheduler
ii. Calculate the average waiting time for a given set of processes using FCFS scheduling
iii. Construct a GANTT chart for a given set of processes

iv. Calculate the throughput for a given system.
v. Explain the convoy effect of FCFS scheduling
vi. List the advantages and disadvantages of FCFS scheduling.

(c) Lecture #3 (Midterm Exam)

i. Obtain a successful exam score.