

CS3841 Operating Systems

Dr. Walter Schilling

Fall, 2012

You may use 1 8.5 x 11 inch sheet of paper with notes and other supporting material for the exam.

1. Week #1

(a) Lecture #1 Working in C

- i. Draw the C flow of C compilation from source code to object code.
- ii. Explain the purpose for the preprocessor, compiler, and linker within the C compilation model
- iii. Using the gcc compiler, generate the output for the preprocessor stage of compilation
- iv. Explain the concept of a dependency.
- v. Create a GNU Make file which automatically generates dependencies, creates preprocessed source code, and links a given C application.

(b) Lecture #2 Introduction to Operating Systems

- i. Compare and Contrast the User View and System View of an operating system.
- ii. Explain the difference between user mode and kernel mode within an operating system.
- iii. Define the term Operating System
- iv. Draw a representation of a modern computer system.
- v. Draw the storage structure hierarchy for a computer system.
- vi. Explain the difference between a trap and an interrupt.
- vii. Explain, in the context of an operating system, multiprogramming.
- viii. Explain, in the context of an operating system, time sharing.
- ix. Understand and use the ls, man, cd, rm, cp, cat, more, less, tar, sort, kill, and ps commands.

(c) Lecture #3 Operating Systems Structures

- i. List and characterize operating systems services (User interface, program execution, IO, file system manipulation, communications, error detection, resource allocation, accounting, protection and security)
- ii. Compare and contrast the command interpreter and graphical user interface approaches to interface with the computer.
- iii. Compare and contrast approaches to command interpreter implementation
- iv. List various UNIX shells
- v. Explain how a system call is made
- vi. Explain the concept of a system call
- vii. Explain the usage of the malloc and free operations within the C programming language.
- viii. Construct simple C programs which use malloc and free to solve problems.
- ix. Implement Screen and File I/O in C, showing how the system calls are invoked

2. Week #2

(a) Lecture #1 Operating Systems Design and Virtual Machines

- 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 relationship between a layered architecture and a virtual machine.

- viii. List the benefits of using a virtual machine.
- ix. Define simulation in the context of virtual machines.
- x. Explain the construction and operation of the Java Virtual Machine and .Net virtual machines.

(b) Lecture #2 Processes

- i. Explain the flow of control when an operating system boots
- ii. Define the term process
- iii. Draw a graphical representation of a process in memory
- iv. Explain the concept of process state
- v. Draw a state transition diagram for process states
- vi. List the contents of a process control block
- vii. Explain what the process scheduler is responsible for doing within the operating system.
- viii. Explain the concept of process dispatching
- ix. Obtain information about the executing processes under Windows and Linux

(c) Lecture #3 Process Operations

- 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 the purpose for the UNIX fork, wait, and exec commands.
- vii. Construct programs using the fork, wait, and exec unix commands
- viii. Explain how a process is terminated.
- ix. Execute a UNIX command in the background using the shell
- x. Use the UNIX command shell to terminate a process

3. Week #3

(a) Lecture #1

- i. Explain why it is important to allow processes to execute in parallel.
- ii. List two methods for interprocess communication
- iii. Explain the difference between indirect and direct communication in terms of message passing.
- iv. Explain how UML sequence diagrams can be used to represent interprocess communications.
- v. List the advantages and disadvantages of using shared memory for interprocess communication.
- vi. List the advantages and disadvantages of using pipes for interprocess communication.
- vii. Construct a rudimentary program using shared memory.
- viii. Construct a rudimentary program using pipes.

(b) Lecture #2

- i. Define a socket
- ii. Explain the concept of loopback and recognize the ip address associated with loopback.
- iii. Define the acronym RPC
- iv. Explain how an RPC executes, specifically in regards to stubs and the concept of marshalling.
- v. Explain the difference between “big-endian” and “little-endian”.

(c) Lecture #3

- 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. List three commonly used thread libraries
- viii. Explain the concept of the join call relative to a thread
- ix. Implement multi-threaded software using Java and POSIX threads in C.

4. Week #4

(a) Lecture #1

- 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 challenges of signal handling in a multi-threaded environment.
- vi. Explain the concept of thread pools

(b) Lecture #2

- 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 five 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 CPU utilization, Throughput, Turnaround time, Waiting time, Response time in terms of their impact on scheduling.
- viii. Explain the operation of a FIFO scheduler
- ix. Explain the convoy effect of FCFS Scheduling

(c) Lecture #3

- i. Explain the algorithm for SJF Scheduling
- ii. Explain why exponential averaging can be used to estimate the shortest job burst.
- iii. Calculate the exponential average based on a series of CPU bursts and an initial estimate.
- iv. Explain priority scheduling.
- v. Using priority scheduling, draw a schedule for a set of jobs
- vi. Define starvation in terms of processor scheduling
- vii. Demonstrate how processor aging can solve the process of starvation
- viii. Explain round robin scheduling
- ix. Explain the relationship between quantum length and performance.
- x. For all scheduling algorithms
 - A. Draw GANTT Chart showing processing sequence
 - B. Calculate the average waiting time
- xi. Justify the design decisions for the Linux kernel based upon scheduling theory
- xii. Explain the concept of the UNIX nice command

5. Week #5

(a) Lecture #1

- i. Midterm Exam

1 Lab Outcomes

1. Lab 1: Getting used to Linux

- (a) Demonstrate an ability to use a Linux shell.
- (b) Use the man command to obtain documentation about Linux commands.
- (c) Explain how to list the contents of a directory in multiple forms.
- (d) Navigate the Linux file system by changing directories.
- (e) Manage the creation and deletion of new files and directories from within the command shell.
- (f) Capture the output of a Linux program executing to a file.
- (g) Manage the creation and extraction of zip files and tarballs using the command shell.

- (h) Construct a makefile which will automatically generate the project as well as allow for the clean building of source code.

2. Lab 2: Memory Management and Data Structures in C

- (a) Use malloc and free to manage the allocation and deallocation of dynamic memory.
- (b) Implement a doubly linked list in C.
- (c) Understand the purpose for the void pointer in C.
- (d) Apply appropriate casts to correctly use a void pointer.
- (e) Implement and use C struct to solve a software problem.
- (f) Use test cases to verify the correct operation of a constructed source code module.

3. Lab 3: Counting Words

- (a) Practice C development in a UNIX environment.
- (b) Construct software in C which uses File input and output routines.
- (c) Manage dynamic memory and heap allocation using C methods.
- (d) Use previously developed libraries as a part of a software development.
- (e) Practice the usage of UNIX piping to chain UNIX programs.