**CS 4345 (Spring 2019): Unit Test 1 Study Guide**

**Topics that you need to prepare:**

**Chapter 1:**

• interrupt and its role in OS functioning; interrupt categorization, and interrupt handling (Section 1.2.1)

A device controller informs the CPU that it is done with its operation by causing an interrupt. Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines.

Interrupt architecture must save the address of the interrupted instruction. A trap or exception is a software-generated interrupt caused either by an error or a user request. An operating system is interrupt driven.

I/O, trap, H/W, timer

The operating system preserves the state of the CPU by storing registers and the program counter. Determines which type of interrupt occurred:

\*polling

\*vectored interrupt system

Separate segments of code determine what action should be taken for each type of interrupt.

• you should be able to explain the concept of OS functioning “mode” (mode bit) and its purpose in OS (Section 1.4.2)

Duel-mode operation allows OS to protect itself and other system components.

User mode and kernel mode

Mode bit provided by hardware

Provides ability to distinguish when system us running user code or kernel code. Some instructions designated as privileged, only executable in kernel mode. System call changes mode to kernel, return from call resets it to user.

The mode bit whether the kernel or user mode is active. Kernal has access to important functions like hardware devices, control over interrupts, and certain instructions can only be executed while in kernel mode. While in user mode the CPU has limited capability giving a form of protection to critical resources.

**Chapter 2:**

• you should be able to identify and describe major tasks/services performed/provided by a general purpose OS (Section 2.1)

OS provide an environment for execution of programs and services to programs and users.

User interface – almost all os have a UI. Ex. Command-Line Interface, Graphics User Interface, Batch

Program execution – must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)

I/O operations – a running program may require I/O, which may involve a file or an I/O device

File-system manipulation – programs need to read and write files and directories, create and delete them, search them, list file information, permission management.

Communications – processes may exchange information, on the same computer or between computers over a network

Error detection – OS needs to be constantly aware of possible errors.

Make OS efficient:

Resource allocation - manages resources when multiple users or jobs are running concurrently.

Accounting – to keep track of which users use how much and what kinds of computer resources.

Protection and security -

• system call – what it is, how used, and why are they useful, etc. (Section 2.3)

A way for a program to request a service from the kernel, it is a way for programs to interact with the OS. System call provides the services of the OS to the user programs. System calls are the only entry points into the kernel. They allow programs to open, read, write to a file, allows a process to wait for another to complete, etc.

• should be able to describe, compare, and contrast the major structural designs (monolithic, layered, microkernel, modular) (Section 2.8)

Monolithic – written to provide the most functionality in the least space. Not divided into modules.

Layered – The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer(layer 0), is the hardware; the highest (layer N) is the user interface. With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.

Microkernel – Moves as much from the kernel into user space. Communication takes place between user modules using message passing.

Benefits: easier to extend a microkernel, easier to port the operating system to new architectures, more reliable, more secure.

Modular – Uses object-orented approach. Each core component is separate. Each talks to the others over known interfaces. Each is loadable as needed within the kernel. Similar to layers but with more flexibility.

**Chapter 3:**

• concept of process; how OS creates processes (Section 3.1, Section 3.3.1)

A process is a program in execution (process execution must progress in sequential fashion), which forms the basis of all computation.

Includes multiple parts:

-the program code: text section

-stack: temporary data

-data section: global variables

-heap: memory dynamically allocated

Program is passive, process is active. A process is created when a program is loaded into memory. Features: scheduling, creation and termination, and communication.

• should be able to identify different states that a process can go through, and the causes for the transitions among states (Section 3.1.2)

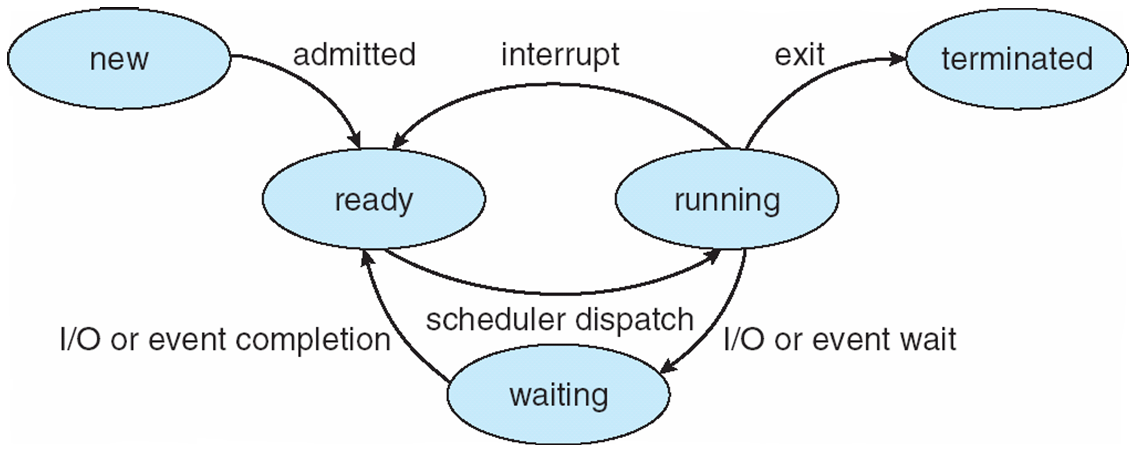
New: the process is being created

Running: instructions are being executed

Waiting: the process is waiting for some event to occur

Ready: the process is waiting to be assigned to a processor

Terminated: the process has finished execution



• should be able to explain inter process communication and its usefulness; focus on the two major models of IPC (Section 3.4)

Processes within a system may be independent or cooperating

Cooperating process can affect or be affected by other processes, including sharing data

Reasons for cooperating processes:

Information sharing

Computation speedup

Modularity

Convenience

Cooperating processes need IPC

Two models:  
 Shared memory

-An area of memory shared among the processes that wish to communicate.

-The communication is under the control of the users processes not the OS

-Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory

Message passing

-Mechanism for processes to communicate and to synchronize their actions

-Message system – processes communicate with each other without resorting to shared variable.

-IPC facility provides two operations:  
 send(message)

Receive(message)

-The message size is either fixed or variable

- Processes need to establish communication link

-Issues:

How many links establish?

Can link be associated with more than two processes?

How many kinks between pair?

What is the link capacity?

Fixed or variable?

Link unidirectional or bi-directional?