Study Set for Lecture 02: Operating System Structures

Due Sep 7 at 11:59pm **Points** 10 **Questions** 13
Available Sep 1 at 12pm - Dec 8 at 8pm 3 months **Time Limit** None
Allowed Attempts Unlimited

Instructions

Review lecture notes from lect02_Operating_System_Structures.pdf.

Then answer the questions from this study set and submit them to gain access to the further part of the course.

Take the Survey Again

Correct answers are hidden.

Score for this attempt: **10** out of 10
Submitted Sep 1 at 2:22pm
This attempt took 51 minutes.

Question 1

Explain what a bootstrap program and an OS loader are, and what's the difference between them.

Your Answer:

Operating systems are designed to run on a variety of machines.

The Bootstrap is a process that the computer must go through to be loaded and started. It is sometimes called boot for short.

The OS loader is a intermediary software that allows users to choose from several different operating systems that are present on the boot device.

The difference between the two is Bootstrap is a program that resides in the computer that is automatically executed by the processor when turning on the computer. The OS loader controls the board upon power-up and does not rely on the Linux kernel in any way. In contrast, the bootstrap primary purpose in life is to act as the glue between a board-level bootloader and the Linux kernel. It is the bootstrap responsibility to provide a proper context for the kernel to run in, as well as perform the necessary steps to decompress and relocate the kernel binary image.

Question 2

Explain what is a difference between a boot sequence and a boot menu.

Your Answer:

After power-on self-tests (POST), the code that is called bootstrap (or boot) loader tests and initializes the hardware and then following a **boot sequence** loads an operating system.

The **boot sequence** is the order of devices listed in BIOS that the computer will look for an operating system on.

On some computers, boot sequence can be overridden during the boot by invoking a **boot menu**, and then selecting a device to boot from disregarding the boot sequence.

● for example, it can be done on Dell computers by pressing the F12 key (as used in the lab)

Question 3

List and describe each of the services provided by an operating system to users.

Your Answer:

Program execution - The system 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 - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file information, permission management.

Communication – Processes may exchange information, on the same computer or between computers over a network ● Communications may be via shared memory or through message passing (packets moved by the OS)

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

● May occur in the CPU and memory hardware, in I/O devices, in user program

● For each type of error, OS should take the appropriate action to ensure correct and consistent computing

● Debugging facilities can greatly enhance the user’s and programmer’s abilities to efficiently use the system 7 COMP362 Operating Systems, Prof. AJ Bieszczad OS Services for Resource Sharing

Resource allocation

● When multiple users or multiple jobs run concurrently, resources must be allocated to each of them

●Some resources such as CPU cycles, main memory, and file storage need customized allocation algorithms (e.g., dynamically depending on the state)

●Others such as I/O devices may use generalized request and release algorithms

Accounting/Logging

● To keep track of which users use how much and what kinds of computer resources

Protection and security

● The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

●Protection is a mechanism

●i.e., involves ensuring that all access to system resources is controlled; for example, from erroneous code like runaway (“floating”) pointers

●Security is a policy

●i.e., access requires user authentication to defending resources from invalid access attempts; for example, from unauthorized access

Question 4

What is the purpose of the command interpreter (shell)? Why is a shell usually separate from the kernel?

Your Answer:

The command interpreter or the command-line interface is one of the ways a user can interface with the operating system. The command interpreter's main task is to understands and executes commands which it turns into system calls.

The kernel is the central module of an OS. Since the kernel is the core of OS it would be dangerous to have code which is prone to changes as part of the kernel. Any update or change made to the kernel need to be thought carefully.

Question 5

List and describe each of the OS services for resource sharing.

Your Answer:

Resource allocation

● When multiple users or multiple jobs run concurrently, resources must be allocated to each of them

-Some resources such as CPU cycles, main memory, and file storage need customized allocation algorithms (e.g., dynamically depending on the state)

-Others such as I/O devices may use generalized request and release algorithms

Accounting/Logging

● To keep track of which users use how much and what kinds of computer resources

Protection and security

● The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

-Protection is a mechanism

●i.e., involves ensuring that all access to system resources is controlled; for example, from erroneous code like runaway ("floating") pointers

-Security is a policy

●i.e., access requires user authentication to defending resources from invalid access attempts; for example, from unauthorized access

Question 6

What is a system call? What is it used for? How can it be implemented? Do application programs usually use system calls directly?

Your Answer:

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface(API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Why do you need System Calls in OS?

Following are situations which need system calls in OS:

- Reading and writing from files demand system calls.
- If a file system wants to create or delete files, system calls are required.
- System calls are used for the creation and management of new processes.
- Network connections need system calls for sending and receiving packets.
- Access to hardware devices like scanner, printer, need a system call.

Question 7

How can parameters be passed to system calls?

Your Answer:

There are three general methods used to pass parameters to the OS

●NOTE: This is a choice that the OS designers make, and not the users

●The developers of the APIs have to stick to the OS convention

1) Simplest: pass the **parameters in registers** (e.g., MS-DOS)

● In some cases, may be more parameters than registers

2) **Parameters placed, or pushed, onto the stack** by the program and popped off the stack by the operating system (e.g., NetBSD)

● This is just like a regular function call

3) **Parameters stored** in a block, or table, in memory, and address of the block passed as a parameter in a register (e.g., Linux, Solaris)

Question 8

List and describe common types of system calls.

Your Answer:

Process Control

These system calls deal with processes such as process creation, process termination etc.

File Management

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

Device Management

These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance

These system calls handle information and its transfer between the operating system and the user program.

Communication

These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

Some of the examples of all the above types of system calls in Windows and Unix are given as follows −

Types of System Calls	Windows	Linux

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

File Management	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		open()
		read()
		write()
		close()

Device Management	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		ioctl()
		read()
		write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		pipe()
		shmget()
		mmap()

Question 9

What is the purpose of system programs (utilities)?
Why are they a better solution than providing system or shell built-in functions.

Your Answer:

System programs provide a convenient environment for program development and execution.

Some are simply user interfaces to system calls; others are considerably more complex

1)File management

● Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

2) Status information

● Some ask the system for info - date, time, amount of available memory, disk space, number of users

● Others provide detailed performance, logging, and debugging information

● Typically, these programs format and print the output to the terminal or other output devices

●the text can be piped into another program rather than sent to the terminal

● Some systems implement a system database used to store and retrieve configuration information; for example, Registry in Windows

● Others, provide utilities to manipulate per-program configuration files; for example, plists in macOS 17 COMP362 Operating Systems, Prof. AJ Bieszczad System Programs (cont.)

3)File modification

● Text editors to create and modify files

●Special commands to search contents of files or perform transformations of the text

4) Programming-language support

● Compilers, assemblers, debuggers and interpreters sometimes provided

5) Program loading and execution

●Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language

6) Communication

●Provide the mechanism for creating virtual connections among processes, users, and computer systems

●Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

7)Common application programs

●(e.g., date, time, text processing, pattern matching, searching, calculators, formatters, etc.)

●Some applications may become recognized as system programs as they gain popularity and are included with the distribution of the OS

●e.g., many GNU utility programs

Question 10

List and describe the categories of system programs.

Your Answer:

Some are simply user interfaces to system calls; others are considerably more complex

1)File management

● Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

2) Status information

● Some ask the system for info - date, time, amount of available memory, disk space, number of users

● Others provide detailed performance, logging, and debugging information

● Typically, these programs format and print the output to the terminal or other output devices

●the text can be piped into another program rather than sent to the terminal

● Some systems implement a system database used to store and retrieve configuration information; for example, Registry in Windows

● Others, provide utilities to manipulate per-program configuration files; for example, plists in macOS 17 COMP362 Operating Systems, Prof. AJ Bieszczad System Programs (cont.)

3)File modification

● Text editors to create and modify files

●Special commands to search contents of files or perform transformations of the text

4) Programming-language support

● Compilers, assemblers, debuggers and interpreters sometimes provided

5) Program loading and execution

●Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language

6) Communication

●Provide the mechanism for creating virtual connections among processes, users, and computer systems

●Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

7)Common application programs

●(e.g., date, time, text processing, pattern matching, searching, calculators, formatters, etc.)

●Some applications may become recognized as system programs as they gain popularity and are included with the distribution of the OS

●e.g., many GNU utility programs

Question 11

Describe and evaluate the microkernel approach to operating system design.

Your Answer:

Kernel is the core part of an operating system which manages system resources. It also acts like a bridge between application and hardware of the computer. It is one of the first programs loaded on start-up (after the Bootloader).

Microkernel is one of the classification of the kernel. Being a kernel it manages all system resources. But in a **microkernel,** the **user services** and **kernel services** are implemented in different address space. The user services are kept in **user address space**, and kernel services are kept under **kernel address space**, thus also reduces the size of kernel and size of operating system as well.

Advantages of Microkernel –

- The architecture of this kernel is small and isolated hence it can function better.

- Expansion of the system is easier, it is simply added in the system application without disturbing the kernel.

Microkernel Architecture –
Since kernel is the core part of the operating system, so it is meant for handling the most important services only. Thus in this architecture only the most important services are inside kernel and rest of the OS services are present inside system application program. Thus users are able to interact with those not-so important services within the system application. And the microkernel is solely responsible for the most important services of operating system they are named as follows:

- Inter process-Communication
- Memory Management
- CPU-Scheduling

Question 12

Describe and evaluate the layered architecture of operating systems.

Your Answer:

This approach breaks up the operating system into different layers.

- This allows implementers to change the inner workings, and increases modularity.
- As long as the external interface of the routines don't change, developers have more freedom to change the inner workings of the routines.
- With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.

> - The main *advantage* is simplicity of construction and debugging.
> - The main *difficulty* is defining the various layers.
> - The main *disadvantage* is that the OS tends to be less efficient than other implementations.

Question 13

Describe and evaluate the architecture of operating system based on compiled and loadable modules.

Your Answer:

Most modern operating systems implement kernel modules

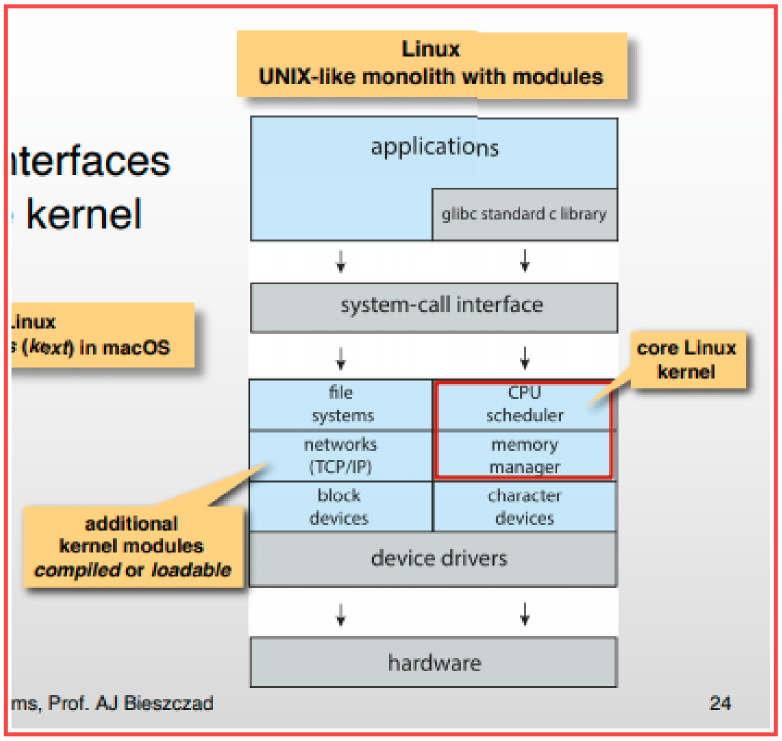
● Object-oriented approach

●Each core component is separate

●Each talks to the others over known interfaces

●Each is loadable as needed within the kernel

● Overall, like layers but more flexible



The basic goal remains however the same: keep what is loaded at boot-time minimal while still allowing the kernel to perform more complex functions. The basics of modular kernel are very close to what we find in implementation of *plugins* in applications or *dynamic libraries* in general.

Advantages

- The kernel doesn't have to load everything at boot time; it can be expanded as needed. This can decrease boot time, as some drivers won't be loaded unless the hardware they run is used (NOTE: This boot time decrease can be negligible depending on what drivers are modules, how they're loaded, etc.)
- The core kernel isn't as big
- If you need a new module, you don't have to recompile.

Disadvantages

- It may lose stability. If there is a module that does something bad, the kernel can crash, as modules should have full permissions.
- ...and therefore security is compromised. A module can do anything, so one could easily write an evil module to crash things. (Some OSs, like Linux (https://en.wikipedia.org/wiki/Linux), only allow modules to be loaded by the root user.)
- Coding can be more difficult, as the module cannot reference kernel procedures without kernel symbols.

What does a Modular Kernel look like ?

There are several components that can be identified in virtually every modular kernel:

The core
This is the collection of features in the kernel that are absolutely mandatory regardless of whether you have modules or not.

The modules loader
This is a part of the system that will be responsible of preparing a module file so that it can be used as if it was a part of the core itself.

The kernel symbols table
This contains additional information about the core and loaded modules that the module loader needs in order to *link* a new module to the existing kernel.

The dependencies tracking
As soon as you want to *unload* some module, you'll have to know whether you can do it or not. Especially, if a module *X* has requested

symbols from module *Z*, trying to unload *Z* while *X* is present in the system is likely to cause havoc.

Modules

Every part of the system you might want (or don't want) to have.

Survey Score: **10** out of 10