Operating System Structures

CHAPTER 2

Practice Exercises

2.1 What is the purpose of system calls?
Answer:
System calls allow user-level processes to request services of the operating system.

2.2 What are the five major activities of an operating system with regard to process management?
Answer:
The five major activities are:

a. The creation and deletion of both user and system processes

b. The suspension and resumption of processes

c. The provision of mechanisms for process synchronization

d. The provision of mechanisms for process communication

e. The provision of mechanisms for deadlock handling

2.3 What are the three major activities of an operating system with regard to memory management?
Answer:
The three major activities are:

a. Keep track of which parts of memory are currently being used and by whom.

b. Decide which processes are to be loaded into memory when memory space becomes available.

c. Allocate and deallocate memory space as needed.

2.4 What are the three major activities of an operating system with regard to secondary-storage management?
Answer:
The three major activities are:

- Free-space management.
- Storage allocation.
- Disk scheduling.

2.5 What is the purpose of the command interpreter? Why is it usually separate from the kernel?
Answer:
It reads commands from the user or from a file of commands and executes them, usually by turning them into one or more system calls. It is usually not part of the kernel since the command interpreter is subject to changes.

2.6 What system calls have to be executed by a command interpreter or shell in order to start a new process?
Answer:
In Unix systems, a *fork* system call followed by an *exec* system call need to be performed to start a new process. The *fork* call clones the currently executing process, while the *exec* call overlays a new process based on a different executable over the calling process.

2.7 What is the purpose of system programs?
Answer:
System programs can be thought of as bundles of useful system calls. They provide basic functionality to users so that users do not need to write their own programs to solve common problems.

2.8 What is the main advantage of the layered approach to system design? What are the disadvantages of using the layered approach?
Answer:
As in all cases of modular design, designing an operating system in a modular way has several advantages. The system is easier to debug and modify because changes affect only limited sections of the system rather than touching all sections of the operating system. Information is kept only where it is needed and is accessible only within a defined and restricted area, so any bugs affecting that data must be limited to a specific module or layer.

2.9 List five services provided by an operating system, and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.
Answer:
The five services are:

a. **Program execution**. The operating system loads the contents (or sections) of a file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

b. **I/O operations**. Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device- or controller-specific commands. User-level programs cannot be trusted to access only devices they should have access to and to access them only when they are otherwise unused.

c. **File-system manipulation**. There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

d. **Communications**. Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

e. **Error detection**. Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently process-independent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

2.10 Why do some systems store the operating system in firmware, while others store it on disk?

Answer:

For certain devices, such as handheld PDAs and cellular telephones, a disk with a file system may be not be available for the device. In this situation, the operating system must be stored in firmware.

2.11 How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?

Answer:

Consider a system that would like to run both Windows XP and three different distributions of Linux (e.g., RedHat, Debian, and Mandrake). Each operating system will be stored on disk. During system boot-up, a special program (which we will call the **boot manager**) will determine which operating system to boot into. This means that rather initially booting to an operating system, the boot manager will first run during system startup. It is this boot manager that is responsible for determining which system to boot into. Typically boot managers must be stored at certain locations of the hard disk to be recognized during system startup. Boot managers often provide the user with a selection of systems to boot into; boot managers are also typically designed to boot into a default operating system if no choice is selected by the user.

2.9 The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories and discuss how they differ.
Answer: One class of services provided by an operating system is to enforce protection between different processes running concurrently in the system. Processes are allowed to access only thosememory locations that are associated with their address spaces. Also, processes are not allowed to corrupt files associated with other users. A process is also not allowed to access devices directly without operating system intervention. The second class of services provided by an operating system is to provide new functionality that is not supported directly by the underlying hardware. Virtual memory and file systems are two such examples of new services provided by an operating system.

2.10 Describe three general methods for passing parameters to the operating system.
Answer:
a. Pass parameters in registers
b. Registers pass starting addresses of blocks of parameters
c. Parameters can be placed, or *pushed,* onto the *stack* by the program, and *popped* off the stack by the operating system.

2.11 Describe how you could obtain a statistical profile of the amount of time spent by a program executing different sections of its code. Discuss the importance of obtaining such a statistical profile.
Answer: One could issue periodic timer interrupts and monitor what instructions or what sections of code are currently executing when the interrupts are delivered. A statistical profile of which pieces of code were active should be consistent with the time spent by the program in different sections of its code. Once such a statistical profile has been obtained, the programmer could optimize those sections of code that are consuming more of the CPU resources.

2.12 What are the advantages and disadvantages of using the same systemcall interface for manipulating both files and devices?
Answer: Each device can be accessed as though it was a file in the file system. Since most of the kernel deals with devices through this file interface, it is relatively easy to add a new device driver by implementing the hardware-specific code to support this abstract file interface. Therefore, this benefits the development of both user program code, which can bewritten to access devices and files in the samemanner, and device driver code, which can be written to support a well-defined API. The disadvantage with using the same interface is that it might be difficult to capture the functionality of certain devices within the context of the file access API, thereby either resulting in a loss of functionality or a loss of performance. Some of this could be overcome by the use of ioctl operation that provides a general purpose interface for processes to invoke operations on devices.

2.13 What is the purpose of the command interpreter? Why is it usually separate from the kernel? **Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?**
Answer: It reads commands from the user or from a file of commands and executes them, usually by turning them into one or more system calls. It is usually not part of the kernel since the command interpreter is subject to changes. An user should be able to develop a new command interpreter using the system-call interface provided by the operating system. The command interpreter allows an user to create and manage processes and also determine ways by which they communicate (such as through pipes and files). As all of this functionality could be accessed by an user-level program using the system calls, it should be possible for the user to develop a new command-line interpreter.

2.14 DescribewhyAndroid uses ahead-of-time (AOT) rather than just-in-time (JIT) compilation.
Answer:

2.15 What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?
There are two common models of interprocess communication: The message - passing model and the shared memory model. The message passing model is useful for exchanging smaller amounts of data, is easier to implement and had no conflicts to avoid. However because of the implementation of the connection process it is slower than shared memory. Shared memory allows two or more processes to exchange information by reading and writing data in shared memory areas. It allows maximum speed and offers better convenience of communication however it has security and synchronization issues as well as the processes that use the shared memory model need to make sure that they are not writing to the same memory location.
There are two common models of interprocess communication:

1. **The message - passing model** The communicating processes exchange messages with one another to transfer information. A connection must be opened before communication can happen where a common mailbox allows the exchange of messages between the processes which can happen either directly or indirectly.

   Advantages

   - It is useful for exchanging smaller amounts of data
   - There are no conflicts to avoid
   - It is easier to implement

   Disadvantages

   - Because of the implementation of the connection process described above it is slower than its counterpart

2. **The shared-memory model**. Shared memory allows two or more processes to exchange information by reading and writing data in shared memory areas. The shared memory can be simultaneously accessed by multiple processes. Real- life examples would be the POSIX systems, as well as Windows operating systems.

   Advantages

   - Allows maximum speed
   - It offers better convenience of communication

   Disadvantages

   - It has security issues
   - The processes that use the shared memory model need to make sure that they are not writing to the same memory location.
   - Problems in the area of synchronization

2.17 Why is the separation of mechanism and policy desirable?
Answer: Mechanism and policymust be separate to ensure that systems are easy to modify. No two system installations are the same, so each installation may want to tune the operating system to suit its needs. With mechanism and policy separate, the policy may be changed at will while the mechanism stays unchanged. This arrangement provides a more flexible system.

More material for 2.17-

In operating systems **mechanisms** determine how to do something while **policies** determine what will be done. An example to understand the difference from real- life would be an office where everyday it is required from its employees to authenticate themselves. This would be a mechanism. A policy on the other hand would be showing their work card to the security guard in order to let them pass.
Separating mechanisms from policy is an important design principle in operating system which states that mechanisms should not dictate the policies. Lets look at the the timer construct. It is a mechanism created to ensure CPU protection, but deciding how long the timer is to be set for a particular user is a policy decision. However the timer might be set differently for different users based on some criteria. The separation of policy and mechanism is important for flexibility. Generally policies are more likely to change across platformsor time. If we were to let the mechanisms dictate or highly restrict the policies each particular change we want to make in a policy would need to change the underlying mechanisms meaning it would be harder and it would take longer to implement. So what programmer and system designers seek to develop is a system where the mechanism is not that affected by changes in policy. This means the developer wouldn't need to change the whole underlying mechanisms only redefine or correct certain parameters.
As a result by separating the mechanisms and policies we can make a more flexible system where modifications are easier to implement. This allows any system designer to develop mechanisms which will stay unchanged while the policies change to suit specific needs. Separating mechanism from policies is most commonly discussed in the context of security and resource allocation problems. Separating mechanisms from policies is one of the factors which distinguishes micro-kernels from monolithic ones.

2.18 It is sometimes difficult to achieve a layered approach if two components of the operating system are dependent on each other. Identify a scenario in which it is unclear how to layer two system components that require tight coupling of their functionalities.
Answer: The virtual memory subsystem and the storage subsystem are typically tightly-coupled and requires careful design in a layered system due to the following interactions. Many systems allow files to be mapped into the virtual memory space of an executing process. On the other hand, the virtualmemory subsystem typically uses the storage system to provide the backing store for pages that do not currently reside in memory. Also, updates to the filesystem are sometimes buffered in physical memory before it is flushed to disk, thereby requiring careful

coordination of the usage of memory between the virtual memory
subsystem and the filesystem.

2.19 What is the main advantage of the microkernel approach to system design?
How do user programs and system services interact in amicrokernel
architecture? What are the disadvantages of using the microkernel
approach?
Answer: Benefits typically include the following
(a) adding a new service does not require modifying the kernel,
(b) it is more secure as more operations are done in user mode than in kernel mode,
and (c) a simpler kernel design and functionality typically results in a more reliable
operating system.
User programs and system services interact in a microkernel architecture by using interprocess communication mechanisms such asmessaging. These messages are conveyed by the operating system. The primary disadvantage of the microkernel architecture are
the overheads associated with interprocess communication and the frequent
use of the operating system's messaging functions in order to
enable the user process and the system service to interact with each other.

The microkernel approach modularizes the kernel and structured the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result would be a smaller kernel. Generally microkernels provide minimal process and memory management in addition to a communication facility. Also this approach uses message passing in order to allow communication between the client program and the various services that are also running in user space. As a result the advantages of using the microkernel approach would be :

1. Extending the operating system is made easier.
2. The operating system is easier to port from one hardware design to another
3. There are fewer changes needed to be made when modifying the kernel
4. The microkernel provides more reliability because of the simpler kernel design and functionality
5. Since it uses message sharing to communicate and most services are running as user rather than kernel processes it offers also higher security.

The microkernel must provide communication between the client program and the various services that are also running in user space. In this case communication is provided through message passing. The client program and service never interact directly only by exchanging messages with the microkernel.

One of the disadvantages of using the microkernel approach are the system-function overheads which affect the performance of microkernels. As described above microkernels use message passing as a form to communicate and exchange information between the user process and the system services. This interprocess communication unfortunately comes with overheads because of the frequent use of the operating systems messaging functions.

2.20 What are the advantages of using loadable kernel modules?

With loadable kernel modules the idea of the design is for the kernel to provide core services while other services are implemented dynamically while the kernel is running. Here the kernel has a set of core components and the rest of additional services are linked in via modules. As an example we might build a scheduling algorithm directly into the kernel and then add support for different file systems through loadable modules. As a result we can add or remove functionality from the kernel while it is running. With loadable kernel modules we don't need to implement all the functionalities on the core system. The additional services are implemented only when they are needed either during boot up or when the OS is running. There is no need to either recompile or reboot the kernel when these new functionalities are added.

2.21 How are iOS and Android similar? How are they different?

iOS

iOS is a mobile operating system designed by Apple to run its smartphone and its tablet computer. iOS is structured on the Mac OS X operating system with added functionality pertinent to mobile devices. It's design has a layered approach with four basic layered stacks of software :

1. **Core OS** This layer holds the core of the OS implemented based on the Mac OS X OS.
2. **Core services** This layer holds support for cloud computing and databases
3. **Media Services** This layer holds support for media and graphics
4. **Cocoa Touch** This layer provides support for hardware features.

Android

The Android operating system was designed and developed for Android smartphones and tablet computers. Android runs on a variety of mobile platforms while it's code is open-sourced and programmed using Java programming language. Android's build is a layered stack of software that provides a rich set of frameworks for developing mobile applications while at the bottom of it's software is the Linux kernel.

From the above information about both systems we can make these comparisons between the two systems.
Similarities

1. Both are based on existing kernels. Android's OS core is build on the Linux kernel and the iOs on the Mac OS X
2. Both have architecture that uses software stacks.
3. Both provide frameworks for developers. In the iOS we can notice the implementation of Cocoa Touch which is an API for Objective-C that provides several frameworks for developing applications. In Android we can notice the presence of frameworks for developing web browsers (webkit), database support (SQLite), and multimedia.
4. Both are designed to run on smartphone and tablets

Differences

1. iOS applications are developed in Objective-C while Android is coded in Java.However, rather than using the standard Java API Google has designed a separate Android API for Java development.
2. iOS's code is closed-source while Android is open-source which has affected positively it's popularity
3. Since Android is programmed using Java it uses a virtual machine (Darkvil virtual machine) while iOS executes code natively

2.22 Explain why Java programs running on Android systems do not use the standard Java API and virtual machine.

Software designers for Android devices develop applications in the Java language however they do not use the standard Java API. Instead Google has designed a separate Android API for Java development. The Java class files are first compiled to Java bytecode and then translated into an executable file that runs on a specific virtual machine created for this Android Api. This is done because the normal API and the virtual machine are both created to be compatible with the desktop and server systems. However in the case of Android we need a new API and a virtual machine which should be compatible with devices with limited memory and CPU processing capabilities. So both the Android API for Java development and the virtual machine are optimized to work with smartphones and tablet systems.

2.23 The experimental Synthesis operating system has an assembler incorporated in the kernel. To optimize system-call performance, the kernel assembles routineswithin kernel space to minimize the path that the system call must take through the kernel. This approach is the antithesis of the layered approach, in which the path through the kernel is extended to make building the operating system easier. Discuss the pros and cons of the Synthesis approach to kernel design and system-performance optimization.

The experimental Synthesis operating system has an assembler incorporated in the kernel. To optimize system-call performance, the kernel assembles routines within kernel space to minimize the path that the system call must take through the kernel which is the antithesis of the layered approach.
Advantages

- The experimental Synthesis operating system achieves on-the-fly compilation thanks to the assembler incorporated on the kernel and it optimizes system-call performances.

Disadvantages

- Debugging inside the kernel is difficult
- This solution is very system specific which might cause issues with updates, portability and flexibility
- Also since it is very specific a new compiler must be written for each architecture