E/15/271

Assignment 2:

Answer the following questions:

1. What is the purpose of system calls?

   System calls allow user-level processes to request services of the operating system.

2. What is the purpose of the command interpreter? Why is it usually separate from the kernel?

   It reads commands from the user or from a file of commands and executes them, usually by turning them in to one or more system calls. It is usually not part of the kernel since the command interpreter is subject to changes.

3. What is the purpose of system programs?

   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.

4. What is the main advantage of layered approach to system design? What are the disadvantages of the layered approach?

   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.

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

   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.

6. 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.

   There are different ways to define different categories of operating system's services and functions. In general an operating system provides an environment for the execution of programs. In our case the difference is made based on the efficiency and usefulness of the operating system. One category of services makes the operating system helpful to the user, the other helpful to the system. So the two categories would be:

   1. The first category of functions makes the operating system useful and helpful to the user. The services provided in this category are:

      - User Interface
      - Program Execution
      - I/O operations
      - File-System Manipulation
      - Communications
      - Error Detection

   2. The other category of functions exists to ensure the efficient operation of the system itself. Thanks to these functions systems with multiple users can gain efficiency by sharing the computer resources among their users. The functions which are part of this category are :

      - Resource allocation
      - Accounting
      - Protection and security

7. Describe three general methods for passing parameters to the operating system.

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. Through a system call the programs interact with the operating system. Often some additional information is required when a system call is executed. The exact type and amount of information might vary according to the particular operating system and the particular system call. This additional information that we pass is nothing more than a variable and we name it a parameter. For example to get input, we may need to specify the file from which we need to read. Parameters in this context generally are passed to system calls in the same way that parameters are passed to other functions in programming. There are three ways to pass parameters:

1. **Registers** The first method is to pass the parameters in registers. Parameter passing is easy if the parameters are passed in registers as this is the most direct way to pass parameters.

2. **Block** When there may be more parameters than registers the parameters are generally stored in a block or table in memory and the address of the block is passed as a parameter in a register. So in this case the passed parameter to the registers will be the addresses of block of parameters.

3. **Stack** Parameters also can be place or pushed onto a stack by the program and popped off the stack by the operating system. Since this method does not limit the number or length of parameters it is preferred from some OS.

8. What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?

Advantages

If we use the same system calls for both file and device manipulation then the devices can be accessed as though they are a file in the file system. Actually the *open()* and *close()* system calls are similar to the *request()* and *release()* system calls used for file manipulation. The similarity between I/O devices and files is why many operating systems including UNIX merge the two into a combined file/device structure. In real life on Linux and Unix systems device files (or special files) are used to represent real physical devices used for Input/Output operations (ex.printer). These device files would appear on the file system just like any other ordinary file. In such cases the kernel deals with devices through the file interface so it is easy to add a new device driver. As an example the *open()* system call would take on a parameter which would be the hardware-specific code to implement this special file interface. So the user program code can be written to access devices and files in the same manner. On the other hand the system call interface can be very beneficial also for the driver code because it can be rewritten in order to support well defined API's(library calls). It is always better if we use API's rather than the core system calls for the following reasons :

- API can support multiple versions of the operating system.
- System calls differ from platform to platform.
- Since API's are nothing more than a library of system calls they are easier to implement than manually writing several lines of code to perform the same function with system calls.
- The API usually provides more useful functionality than the system call directly.

Disvantages

On the other hand when using the same interface it might be difficult to capture the functionality of certain devices. Since we are using the file access API's the system might not capture all the functionalities which can result in a loss of functionality or a loss of performance. **ioctl** Input/Output control is a system call supported by most Unix/Linus system for device-specific input/output operations and other operations which cannot be expressed by regular system calls. This system call can help overcome loss of performance and functionality up to a degree. Windows's version of this system call in the Win32Api is **DeviceIoControl**

9. What are the two models of inter process communication? What are the strengths and weaknesses of the two approaches?

   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

10. Why is the separation of mechanism and policy desirable?

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 platforms or 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.

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.

11. What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

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.

As a summary,

Microkernels have several advantages, such as making it easier to extend the operating system, making it easier to port it from one hardware design to another, needing less changes when modifying the kernel, being more reliable and secure. Its disadvantages are related to the overheads coming with the communication system and the frequent usage of messages.

12. 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.

13. 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.