S.THINESH

E/15/366

**CO327 – Operating Systems**

**Assignment – 2**

1. What is the purpose of system calls?

System calls provide an interface between the process and the operating system. System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do. In handling the trap, the operating system will enter in the kernel mode, where it has access to privileged instructions, and can perform the desired service on the behalf of user-level process. It is because of the critical nature of operations that the operating system itself does them every time they are needed. For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.

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

* Command interpreters have a large range of commands and queries available for different operations. Also, it is much faster to type than to click as is done using graphical user interfaces.
* There are some systems that don’t have enough resources to support graphical user interfaces. In those cases, command interpreters can be used.
* Scientists and engineers often used command interpreters in scientific environments. Technically advanced users also prefer command interpreters as compared to graphical user interfaces.
* People with visual disabilities use command interpreters as they cannot work with graphical user interfaces. Commands and instructions can be displayed using braille displays in command interpreters.

1. What is the purpose of system programs?

System programs provide basic functioning to users so that they do not need to write their own environment for program development (editors, compilers) and program execution (shells). In some sense, they are bundles of useful system calls.

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

Advantages of layered operating systems are:

It is decomposable and therefore effects separation of concerns and different abstraction levels

It allows good maintenance, where you can make changes without affecting layer interfaces

Disadvantages of layered operating systems are:

It is difficult to exactly assign of functionalities to the correct and appropriate layer

Because of having too many layers, performance of the system isdegraded

1. 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 not be available for the device, In this situation the operating system must be stored in firmware.

1. 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?

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 those memory locations that are associated with their address spaces. Also, processes are not allowed to corrupt files associated with other users. A process is 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.

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

Register, Block, Stack

1. pass the parameters in registers
2. Parameters stored in a block,or table,in memory,and address of block passed as a parameter in a register
3. Parameters placed,or pushed, onto the stack by the program and popped off the stack by the operating system
4. What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?

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 be written to access devices and files in the same manner, and device-driver code, which can be written to support a well-defined application program interface (API).

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 resulting in either a loss of functionality or a loss of performance. Some of this could be overcome by the use of the ioctl operation that provides a general-purpose interface for processes to invoke operations on devices.

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

|  |  |
| --- | --- |
|  | |
| The two models of interprocess communication are shared-memory model and message-passing model. | | |
|  |  | |
|  | Shared memory model | |
|  |  | |
|  | Shared memory model allows maximum speed and convenience of communication, as it can be performed at memory transfer speeds when it occurs inside a computer. | |
|  |  | |
|  | The processes are responsible for ensuring that they are not writing into the same memory location simultaneously. Hence, its main disadvantage lies in the areas of protection and synchronization between the processes. | |
|  |  | |
|  | Message passing Model | |
|  |  | |
|  | This model is useful for exchanging smaller amounts of data. It is much easier to implement compared to the Shared memory model. | |
|  |  | |
|  | This process has the overhead of opening a connection and closing a connection when the communication is over. This is possible only when the recipient name is known by the process. | |
|  | |  |
|  | |  |

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

In this course, we shall distinguish between policy and mechanism. Policies are ways to choose which activities to perform. Mechanisms are the implementations that enforce policies, and often depend to some extent on the hardware on which the operating system runs. For instance, a processes may be granted resources using the first come, first serve policy. This policy may be implemented using a queue of requests. Often the kernel provides mechanisms that are used to implement policies in servers.

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

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 could interact through message passing or shared memory

the disadvantages is : suffer from performance decreases due to increased system function overhead (context switching increased).

1. What are the advantages of using loadable kernel modules?

Loadable kernel modules have several advantages over monolithic "blobs" of code in the kernel:

Device drivers don't have to be hard-coded into the kernel. For example, if a new chip-set comes out that powers many webcams, that kernel module can simply be loaded instead of recompiling the kernel with the new module.

Features can be added to the kernel after initial release: If a kernel is missing several system calls (like creating a Named Pipe, or setting file permissions), then a loadable kernel module can be added later.

In Embedded systems (such as Windows CE), the central part of the kernel may exist only on read-only media (such as an EEPROM). Loadable kernel modules loaded from easily-writable FLASH or other media can extend the OS.

Loadable System Calls can extend the functionality of the kernel to use-cases not originally envisioned in its design. For example, consider that Linux started as a small project for x86-class, single-processor PCs. A Loadable System Call module could be installed that allows for the Linux kernel to be run within a -Cluster-: System calls for Remote Procedure Calls (RPCs) could easily be loaded in. A loadable driver that supports high-end Infiniband hardware could

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

Instead of using the standard Java API, google has designed a separate Android API for Java development. Android also uses the specially designed Dalvik virtual machine instead of the standard virtual machine or JVM. In android, the java classes are first compiled to Java bytecode which is then translated into an executable file to run on the Dalvik virtual machine. This Dalvik virtual machine is specially designed for Android which comes optimized for mobile devices with limited memory and CPU processing capability. Hence, the Java programs running on Android don't use the standard Java API or the virtual machine.