

Operating System Concepts for System Programmers

System programming is about developing programs directly on top of hardware or writing programs on top of operating system by using its system calls. System programmers who wish to write programs by using system calls of operating system should have basic concepts about the OS.

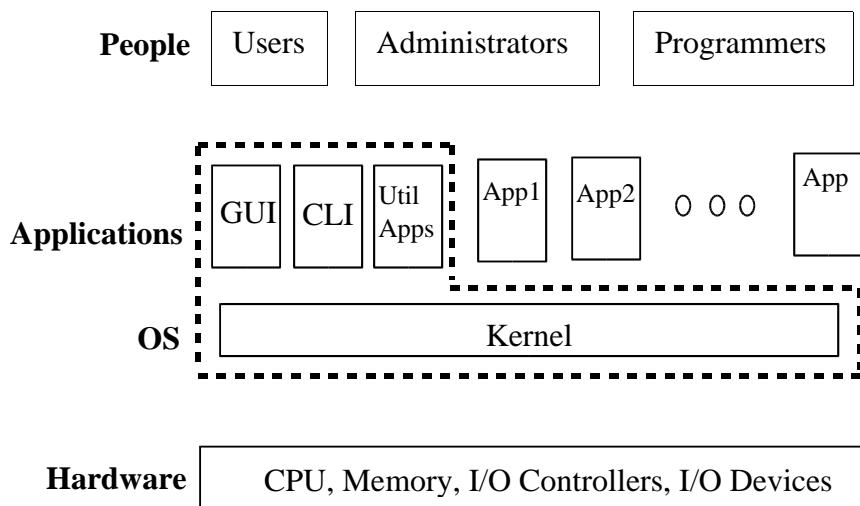
1. What is Operating System?

Operating system consists of a set of programs (software) that manages the hardware and provides an environment to run multiple application programs. The OS shares the hardware (CPU, Memory, I/O controllers and I/O devices) among multiple application programs.

2. Components of an Operating System

Operating system (OS) is not a single program; it is a set of programs. However the real core of an operating system is called Kernel. The second important program of an OS is User Interface (UI) program. Old operating systems used to have only Command Line user Interface (CLI) program called Shell. But all modern operating systems are coming with Graphical User Interfaces (GUI) called Windows. All the other programs that come with the OS are utility application programs.

Finally there are lots of applications, which we separately buy (which is legal) or get pirated copy of application (illegal) or download freeware/shareware applications from the Internet. This is illustrated in the following figure.



The set of programs that comes with OS, and which we call OS is shown inside the dashed lines. The most important part of the OS, is the Kernel. But just kernel alone is not enough. So OS comes with two types of user interface programs. One is graphics based and other is command (or text) based. Besides these user interface programs, OS also includes many utility programs, to create and list file, directories etc..

Even though GUI, CLI and utility programs come with OS, still they are considered as applications. One major difference between applications and kernel is that, Kernel runs in kernel mode or Supervisor mode of the CPU, where as all applications (that comes with OS also) will run in user mode of the CPU.

It is very difficult to distinguish between applications you separately buy and applications that come along with OS. For example windows bundles lot of applications as part of the OS. Examples are Internet explorer (Browser), Outlook express (E-mail client), Media player etc..

Finally people use OS for different purposes. Ordinary users will use it to run the applications. Administrators are responsible for maintaining the OS and other applications. And programmers are responsible for the development of new programs. We may also classify the programmers into System Programmers and Applications programmers. System programmers directly deal with the services (system calls) provided by the OS. Where as application programmers use the APIs (Application Programming Interface functions) provided by the application environments such as Database, Web, Graphics, and distributed application environments. These application environments in turn run on top of the OS.

What normal user sees in an operating system is its user interface that could be either CLI based or GUI based, and application programs that comes along with the OS. For normal users, OS provides an interface to run their favorite applications. Not just one, but multiple applications at a time.

But what system programmer sees in an operating system are its kernel, the system calls provided by the kernel, and libraries that comes with the operating system. The programmer wants to develop some quality useful applications for this operating system. So she/he should know the services (system calls) provided by the kernel. So that programs can use these kernel services.

You are reading this because you want to become a system programmer. So as a system programmer we will concentrate on the kernel. For you OS means it is kernel. Its user interface and application programs are not important for you, but they are very important for all normal users.

Following are the typical services provided by the Kernel to the system applications:

- File Services
- I/O Services
- Multi-processing services
- Multi-threading services
- Memory allocation and mapping services
- Signal/Event services
- Inter Process Communication and Synchronization services
- Time services
- Network communication services

All most all applications need to use all or some of the above services provided by the operating system.

Review Questions

What is Operating System?

What are the various programs that belong to the operating system?

What are the two types of user interfaces provided by the OS, and what are the differences between these two types of interfaces?

Is it possible to differentiate the applications that are part of OS, and applications, which are not part of OS?

What is the main difference between Kernel and Application programs?

How do you classify the people using the operating system?

What are the differences between Application programs and System programs?

What are the types of services provided by the kernel?

What are the components present in the computer hardware?

3. Booting of Operating System

Once you power-on a computer, the process of loading the operating system and applications is called booting process. Following are the various steps involved while booting the OS:

The BIOS program present in the non-volatile (ROM/Flash) memory of a computer will run first. In fact whenever a CPU is powered on, it tries to fetch (read) instructions from a fixed start address. The computer hardware designers will ensure that ROM containing BIOS program will match with this start address.

The BIOS program initializes various I/O controllers and I/O devices. Finally it identifies the boot device (as stored in battery backup memory (CMOS memory)). Typical boot devices are floppy, Hard disk, CD, USB flash stick and finally Network.

The BIOS program reads the first sector (called boot sector) from the boot device and loads these 512 bytes of first sector into the memory (RAM). Next it jumps to that memory location. In this way what ever program that is present in the boot sector of boot device will get executed. This 512 byte, boot sector program is called primary boot loader.

The primary boot loader program present in the boot sector program is a small program (512 bytes). So it this program can not load the operating system. So instead it loads slightly larger program (called secondary boot loader program) into the memory and jumps to that program. This secondary boot loader program is bigger than primary boot loader and capable of loading the operating system kernel into memory. Secondary boot loads the OS kernel into memory and jumps to the OS kernel.

In this way kernel starts execution. This kernel will initialize all its data structures, initializes the hardware and starts the first Application program. This application program is responsible for starting all the required server applications and user interface applications.

Review Questions

1. When you power-on a computer, which program runs first?
2. What is boot device and what are the possible boot devices in a computer?
3. What is primary boot loader program and what this program will do?
4. What is secondary boot loader program and will it do?
5. When kernel starts execution what will it do?

4. Kernel Execution and Responsibilities

As we saw in the previous section, during booting process, kernel will get loaded into memory and kernel starts the first application program. This first application in turn starts all other server and user interface applications. All these applications are started and running in user mode. So now question is that, when kernel will run again. Kernel will run under the following cases.

- Whenever application running in the user mode invokes a System call (kernel calls) the kernel code will run on behalf of that application
- Whenever interrupt occurs, the application running in use mode will stop and Kernel will run the Interrupt Service Routine (ISR) and other functions associated with that ISR.
- Whenever application causes any exception, then also kernel will get control and runs the exception handling functions associated with that exception

Kernel has got the following main responsibilities for the programs:

- Loading a program into memory by allocating required memory
- Providing a CPU time slice (i.e scheduling) for the above program to run
- Providing the services (system calls) so that the program can use these services
- Once the program is finished, unloading the application from memory and freeing the memory

Kernel performs the scheduling during the following events:

- During timer interrupt from a timer device
- During system call

Every computer's hardware will have a timer chip that can generate periodic interrupts to the CPU. These timer interrupts are called system ticks. Typical period for these timer interrupts are 10 milliseconds to 1 millisecond.

Whenever timer interrupt occurs, control will switch from application to the kernel and kernel will execute timer ISR and associated processing functions. During this timer processing kernel will verify whether the running application has completed its time slice or not. If completed the kernel will perform a task switch to run another application, which is waiting, in the ready queue. This application, which exhausts its time slice, will be added to the end of the ready queue.

Some times application running in user mode, wants to read data from a device like keyboard. To read, the application invokes kernel's read() system call. Kernel will verify whether the keyboard device has got any data or not. If no data is available with keyboard, the kernel will put the application reading keyboard into wait queue and schedules another application waiting in the ready queue.

Review Questions

While application program is running, what are the events that cause kernel to run?
What are the main responsibilities of the kernel for the application programs?

What are the events in which kernel will perform the scheduling activity?
What is the role of the timer chip hardware?

5. Kernel Services

As studied in the previous section, following are the typical services provided by the Kernel to the application programs.

- File Services
- I/O Services
- Multi-processing services
- Multi-threading services
- Memory allocation and mapping services
- Signal services
- Inter Process Communication and Synchronization services
- Time services
- Network communication services

5.1 File Services

Using the file services one can write a program to do the following things:

- Opening an existing file
- Creating new a file
- Writing data to a file sequentially or to selected areas of file
- Reading data from a file sequentially or from selected areas of a file
- Reading the attributes (details such size, owner, permissions etc..) of a file
- Deleting a file
- Changing the attributes of file
- Opening an existing directory
- Creating a new directory
- Read the contents of a directory
- Closing the file or directory

5.2 I/O Services

Using the file services one can write a program to do the following things:

- Opening an I/O device
- Reading from I/O device
- Writing to I/O device
- Reading the settings of an I/O device
- Modifying the settings of an I/O device
- Closing the I/O device

Typical I/O devices are devices connected to serial port such as Modem, Dumb Terminal; devices connected to parallel port such as Printer, Devices attached to the USB ports, Speakers and Microphone.

Settings of a device vary from device to device. For serial devices, baudrate is one setting, which can be read or modified. For the speaker volume could be one setting, which can be modified.

5.3 Multi-processing Services

Multi processing services of kernel will provide the following functionalities:

- Creating a new process
- Loading a new executable code into a process
- Scheduling the processes during timer interrupt, and during system calls
- Wait for a process till its completes its work
- Exiting a process

5.4 Multi-threading Services

Multi threading services of kernel will provide the following functionalities:

- Creating a new thread
- Exiting a thread
- Waiting for the completion of a thread

5.5 Memory allocation and mapping services

Memory allocation and mapping services of kernel will provide the following functionalities:

- Dynamic memory allocation
- Freeing the dynamic memory
- Mapping a file or device into the memory space of a process

5.6 Signal services

Signal services of kernel will provide the following functionalities:

- Catching a signal by assigning a signal handler
- Sending a signal to other processes
- Masking and unmasking of signals
- Reading the pending signals

5.7 IPC services

IPC services of kernel will provide the following functionalities:

- Creating a communication object such as pipe, FIFO, message queue and shared memory
- Writing to the communication objects created above
- Reading from the communication objects
- Deleting the communication objects

- Changing the properties of some communication objects
- Creating synchronization objects such a semaphores
- Posting tokens to a semaphore
- Waiting for tokens from a semaphore

5.8 Time services

Time services of kernel will provide the following functionalities:

- Getting the time of day
- Sleeping or delaying for a specified period
- Starting an Alarm

5.9 Network communication services

Network communication services of kernel will provide the following functionalities:

- Creating a network communication object such as socket
- Establishing a connection to the remote socket
- Sending data to the remote socket
- Reading data from the remote socket
- Reading the attributes of a socket
- Modifying the attributes of a socket
- Closing the socket

All the above services of a kernel are available in the form of C functions called System calls. The System programmer must be able to use all these system calls as when required.

Review Questions

What are the various groups of services provided by the kernel?
List the functionalities provided under each service group.

6. Sample Operating Systems

Following sub sections will describe the some operating systems to give an idea of operating systems with various complexities. We can classify operating systems based on the complexity involved as below:

- OS that supports only a single Task (thread) Application
- OS that supports multi-task (or multi-thread) Applications
- OS that supports multiple processes, each process with separate virtual memory space
- OS with multiple processes with each process having multiple threads

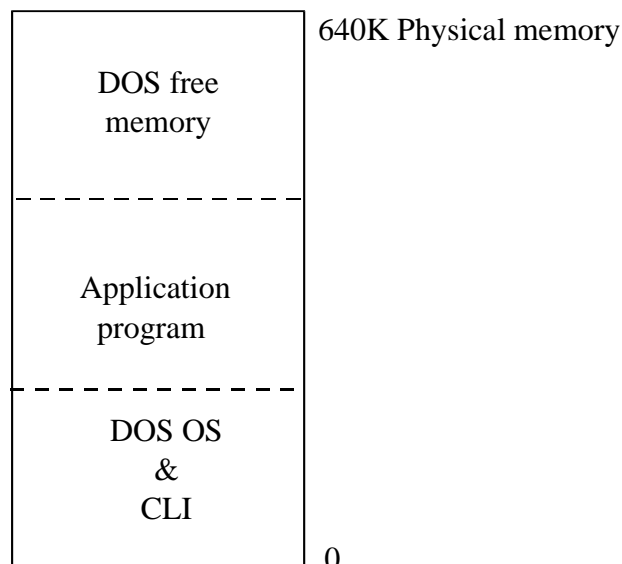
6.1 Simple Operating System (DOS)

DOS is the simplest operating system. It does not support multi tasking or virtual memory. When we boot DOS OS, the DOS kernel will get loaded into the lower part of the memory. After DOS got initialized, it will start the Command Line Interpreter (CLI) program.

DOS is designed to run on 8086 CPU. So it can address only 1 MB of RAM. But in original PC motherboard only 640K of RAM is available. Remaining address range is used for the BIOS ROM and Display memory. So total memory available to the DOS OS is only 640 KB only. DOS and CLI are loaded during booting time and occupy the lower portion of this 640 KB memory.

When user enters a command at DOS prompt. The DOS will load this application into the memory just above the DOS and jumps to that application. The application will keep running till completion. This application may call DOS system calls to use the DOS services. When application finishes control will go back to the CLI program. Now CLI program is ready to accept one more program (or command) from the user.

DOS will not (or need not) do any scheduling activity, as there is only one application running at a time. So all the CPU time will be allocated to the single application.



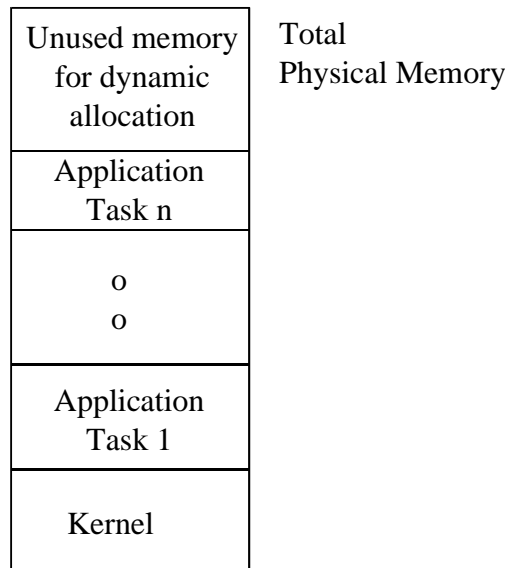
The role of DOS operating system is to provide services to the application program. DOS OS provides the following services to the application program.

- File Services
- I/O Services
- Memory allocation services
- Time services

6.2 Embedded or Real Time Operating Systems (VxWorks)

When compared to DOS, most of the real time operating systems are developed for 32 bit processors and can use all the physical memory present on the board. Embedded operating systems also do not support virtual memory. This will use only physical memory addresses.

But all RTOSes support multi-tasking (also called multi-threading). RTOS uses priority based pre-emptive scheduling to schedule the application tasks.

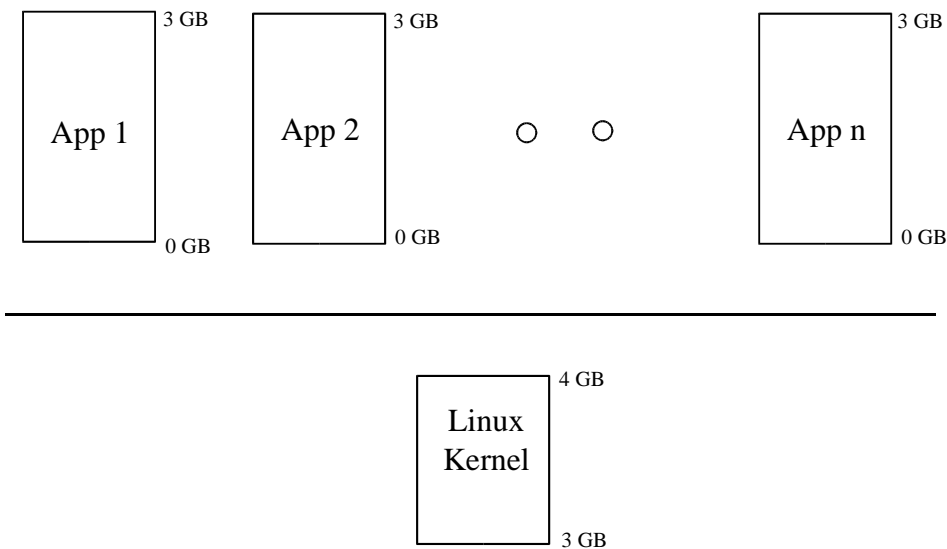


The application programs developed on the RTOS will have multiple tasks. All the tasks are loaded into the physical memory at different addresses. RTOS schedules these tasks and provides CPU time based on their priorities. The RTOS will provide the following services.

- File Services
- I/O Services
- Multi tasking (also called multi-threading) services
- Memory allocation services
- Inter Task Communication and Synchronization services
- Time services
- Network communication services

6.3 Multi-processing Operating System (Linux)

Operating systems such as Unix, Linux and Windows support multiple processes by providing separate virtual memory for each process. One application program runs in one process by using its virtual memory.



Because of virtual memory support, the operating system provides a separate virtual memory space of maximum 3 GB for each process. Every process feels that it is using all the 3GB of memory for itself. But in fact, the total physical memory present in the hardware will be much less than 3GB and typically around 128MB to 512MB. OS manages this physical memory and provides an illusion for every process, as if it is using 3GB.

The basic concept behind the virtual memory is simple. Even though the process using, for example 1 GB, it access only few locations at a time. So only those locations are mapped to the physical memory. While program is using some other locations, the old memory locations could be unmapped. In this way even though the process is using overall 1GB, only few physical locations are used at any given time.

Another advantage with virtual memory is that, each process is isolated from other processes. So one process cannot access the memory of other process. In fact a process is not even aware of the presence other processes. Each process thinks as if it is running on the CPU all by itself. But fact is that, kernel is providing a small time slice for each process, to run on the CPU.

Each application running in the virtual memory can call kernel system call to use the kernel services. The Linux kernel is typically mapped to 3GB to 4GB of virtual memory space.

In this way kernel is allocating CPU time and physical memory to the multiple processes.

Multi-Threading

All the modern operating systems support multiple threads. In the virtual memory of a single process multiple threads can execute. All the threads running in a process will share the virtual memory and other resources of that process.

Multiple threads running in a process are similar to the multiple tasks running in an RTOS. In RTOS multiple tasks are running by sharing the same physical memory. In the same way threads in a process share the virtual memory space of that process.

In this way one process with multiple threads is equivalent to one RTOS environment.

Review Questions

- What are the differences between DOS and RTOS?
- What are the differences between RTOS and Linux?
- Which OS supports virtual memory?
- What is virtual memory?