**CHAPTER-1**

**INTRODUCTION**

* 1. **WHERE DOES AN OPERATING SYSTEM FIT IN?**

An operating system is the layer of a computer system between the hardware and the user programs (user software). An operating system does what all software does it implements some desired functionality by building on the functionality available in lower levels.

An operating system is built directly on the hardware interface and provides an interface between the hardware and the user programs.

A computer system, as it comes “out of the box,” is not easy to use it requires an operating system and application software to run on it. An application software package, as it comes “out of the box” (or “out of the shrink wrap”), will not run directly on the computer hardware it requires an operating system to run it. An operating system is the layer of software that nearly every application software package excepts to be present. The operating system is the first program run on a computer when the computer boots up. There can be only one operating system running on a computer.

**1.1.1 SYSTEM LEVELS:**

The layered structure of the levels of operating system is comprises of hardware, the software, and the operating system. The hardware consists of things like program counter, registers, interrupts, disks, terminals, etc. The hardware interface consists of everything you need to know about the hardware in order to write programs that will execute on the hardware.

User programs

Operating system

Hardware

Operating system interface

Hardware interface

Levels in a computer system

An operating system shares characteristics with both hardware and software. An operating system is software, that is, it is a program that is compiled, linked, and run on a computer. However, it is like hardware in that only one copy of the operating system runs on a computer and it extends the capabilities of the hardware. The services of the operating system are invoked with a special “trap” or “system call” instruction that is used just like any other hardware instruction.

The functions of an operating system can be viewed in two ways:

* As a resource manager.
* As an implementor of virtual computers.

By virtual computer we mean an implementation of the functionality of a computer in software. The hardware system manages the hardware resources of the computer system. The main hardware resources in a computer system are:

* Processor: - The processor is the part of the computer system capable of executing instructions.
* Memory: - The memory contains all instructions and data used by a processor.
* Input/Output(I/O) controllers: - Processors that know how to transfer data between memory and devices.
* Disk devices: - Disks provide long-term storage for data.
* Other devices: - Hardware components that accepts data from or generate data for I/O controllers. Examples are terminals, networks, tape drives, etc.

Processor

Devices

Memory

IO Controller

Disk

Primary resources (Computing)

Secondary resources

(Interfaces and long-term storage)

Fig: The hardware resources in a computer system

Resources management is:

* Transforming - Creating a new resource from an existing resource. The created resource will act as a substitute for the existing resource but will be easier to use.
* Multiplexing – Creating the illusion of several resources from one resource.
* Scheduling – Deciding which programs should get each resource and when they should get it.

**Transformation:**

Operating systems transform physical resources into virtual resources to avoid the difficulties associated with using hardware resources. A virtual resource provides the essential functionality of the hardware resource, but is easier to use because the details of the hardware interface are hidden.

**Multiplexing:**

When there are more virtual computers than physical resources, the operating system needs to make sure that the virtual computers can share the physical resources. The sharing of physical resources is called Multiplexing.

**Spooling:**

To create the illusion of multiple printers, the operating system can implement a virtual printer using a disk file. In this solution, each virtual computer has its own “printer file”. Every time the application prints a character, the operating system simply appends the character to the end of the printer file which belongs to the virtual computer being used for that application. When the application finishes, the operating system queues the application’s printer file. The software that drives the physical printer only prints files this solution was called Spooling.

**Space-division multiplexing:**

In space-division multiplexing, a resource is divided into smaller versions of itself and each virtual computer is given a part of the resource. Space-division multiplexing is used in multiplexing primary and secondary storage (memory and disks). Although not commonly used, the term space-sharing for space-division multiplexing.

**1.2.3 VIRTUAL COMPUTERS:**

An operating system creates software copies of the processor (the capability to execute instructions) and the memory (the capability to store information). It also transforms the disk devices into a file system and the other input-output devices into more abstract (less detailed) and easily used devices. Thus an operating system:

* Creates multiple processes (simulated processors) out of the single processor and allocates them to programs. It accomplishes this by time-multiplexing the processor.
* Creates multiple address spaces (memory for a process to run in) out of the memory and allocates them to processes. It accomplishes this by space-multiplexing the processor.
* Implements a file system and input/output(I/O) system so that processes can easily use and share the disks. It accomplishes this by space-multiplexing the I/O channels.

A virtual computer is a computer implemented in software using the hardware resources of a physical computer. It has the four basic components (Processor, Memory, I/O, and Disk) as the physical computer. Each virtual resource will be a transferred and multiplexed version of physical resource.

**1.3.1 VIRTUAL PROCESSOR:**

The virtual processor has nearly the same interface to the user as the physical processor, that is, it has nearly the same set of instructions. One reason for this is that it is implemented by using the physical processor directly. The operating system will rapidly switch the services of the physical processor between the virtual computers to implement the virtual processors.

The operating system removes some of the physical processor instructions and adds some other operations. The instructions it removes are the ones that control the physical resources of the computer (processor, memory, I/O). These are the instructions that affect access of memory, provide direct access to the devices, and change the protection state of the processor. In return for the instructions, the operating system adds instructions that allow the virtual processor to request virtual resources from the operating system. These instructions are called system calls. System call allow a programmer to:

* Create new virtual computers;
* Communicate with other virtual computers;
* Allocate memory as needed;
* Do I O; and
* Access a sophisticated file system.

**1.3.2 VIRTUAL PRIMARY MEMORY:**

The memory of the virtual computer is very similar to hardware memory, that is, it is a long sequence of cells with sequential, numerical names. The operating system will divide up the physical memory into parts and give a part to each virtual computer. The operating system usually creates, from the hardware the illusion that the memory seen by each virtual computer is named with numerical names started at 0. The operating system may also create the illusion that the virtual computer as more memory than the physical computer has physical memory. This is called Virtual memory.

**1.3.3 VIRTUAL SECONDARY MEMORY:**

Secondary storage provides long-term storage for data. This is done physically on disk blocks and virtually in disk files. The virtual computer sees a file system consisting of hierarchically named files that can be of any size and can be read and written in any size units. It considerable amount of operating system code is required to create this illusion.

**1.3.4 VIRTUAL I/O:**

The I/O operations of the virtual computer are software driven different than I/O operations of the physical computer. The physical computer has devices with complex control and status registers. For example, the disks are basically large sequence of physical blocks with no structure.

**CHAPTER 2**

**THE HARDWARE INTERFACE**

Introduction:

Hardware interface is an architecture used to interconnect two devices together. The general architecture of computers are of two types namely CISC and RISC. An example of an RISC architecture is CRA-1. This system provides a CPU (also known as processor) for executing program, and a disk system (disk controller and disk) permanent storage of programs and data. These components are interconnected using a system bus as shown below.

Disk

Disk Controllers

CPU

Memory

CRA-1 organization

**2.1 THE CPU:**

**2.1.1 GENERAL-PURPOSE REGISTERS:**

The CPU provides 32 general-purpose registers, each 32 bits long. In assembly language, these registers are referenced using names r0 through r31. As is common with RISC machines, r0 always holds the value 0. The other registers are for general use, but a few have conventional uses in most programs.

* R1- return values from procedures.
* R8- first parameter to a function call (or system call)
* R9- second parameter to a function call (or system call)
* R10- third parameter to a function call (or system call)
* R11- fourth parameter to a function call (or system call)
* R29- the frame pointer.
* R30- the stack pointer.
* R31- the return address from a procedure call.

Procedure calls send their arguments in registers r8, r9, r10 and r11 and return values in r1.

**2.1.2 CONTROL REGISTERS:**

The control registers hold values that control the execution of the processor. The control registers are all 32 bits long.

* Ia- The instruction address register contains the address of the next instruction.
* Psw- The program status word has two significant bits. Bit 0 (the low-order bit) is 1 if the processor is in user mode, and 0 if the processor is in system mode. Bit 1 is 1 if interrupts are enabled, and 0 if interrupts are masked.
* Base- The memory base register is added to all addresses when system is in user mode.
* Bound- The memory bound register is the address limit.
* Iia- The interrupt instruction address register stores the value of the ia register before an interruput.
* Ipsw- The interrupt program status word stores the value of the psw register before an interrupt. When an interrupt occurs, the psw register is saved in the ipsw register before it is loaded with 0.
* Ip- The interrupt parameter register contains the data about the last interrupt (if any is necessary).
* Iva- The interrupt vector address register is location in memory where the interrupt vector table is located.

**2.1.3 PROCESSOR MODES:**

The processor can either be in system mode (if psw bit 0 is 0) are in user mode (if psw bit 0 is 1) in system mode, all instructions are legal and all address are physical addresses. In user mode, the instructions that modify the control registers are not legal and will cause a program error interrupt (with ip of 1).

In a load/store architecture, there are two main instructions that access memory, one to load a register from memory and another to store a register into memory. We will assume that the load and store instructions can load into a store from a control register as well as a general register.

The system call instruction is executed by a user process in order to request a system service. This instruction causes an interrupt which puts the processor into system mode and transfers to the system call interrupt handler in the operating system. The arguments to the system call (including the system call number) are assumed to be in the user registers at the time of the system call.

**2.2 MEMORY AND ADDRESSING:**

The smallest addressable unit will be an 8-bit byte and addresses will be 32 bits long. When the processor is in system mode, all addresses are physical addresses, which means that the address generated by the instruction is sent directly to the memory by placing it on the system bus as a memory address. Physical addresses are also called absolute addresses.

When the processor is in user mode, all addresses are logical addresses. When an instruction generates a logical address, the processor will first check to be sure that the logical address is less than the bound register. If it is not, then program error interrupt (with ip of 2) is generated. Otherwise, the base register is added to the logical address to generate a physical address. This physical address is then placed on the system bus as a memory address.

In user mode, a program is restricted to a portion of the physical addresses. It cannot access any physical address less than the value of the base register (since it is added to every address) and it cannot access physical address greater than (or equal to) the sum of base and bound register (since this limit is checked for each logical address). This is a sample form of memory protection.The physical address space consists of all the physical address that a program can generate.

The memory address space is the part of a physical address space where we can place memory. In the CRA-1, the memory address space is from 0 to 0xEFFFFFFF. Only a part of the memory address space will have actual memory responding to those addresses.

**2.3 INTERRUPTS:**

An interrupt is an immediate transfer of control caused by an event in the system. There are several events that can cause interrupt in CRA-1.

1. System call - A system call instruction was executed.
2. Timer – The internal timer counted down to zero.
3. Disk – The disk device finished a transfer.
4. Program error – The program made an error. The ip register indicates exactly what happened.

When an interrupt occurs, the CRA-1 does following:

1. The psw register is saved in ipsw register.
2. The psw is set to 0.
3. The parameters of the interrupt (if it is a program error interrupt) is put in the ip register.
4. The ia register is saved in the iia register.
5. The interrupt number (system call=0, timer=1, disk=2, program=3) is multiplexed by 4 and added to the contents of the iva register to get the interrupt to vector slot. The address in that slot is loaded in the ia register.

**2.4 I/O DEVICES:**

The CRA-1 uses memory-mapped I/O, meaning that communication between the I/O devices and a processor is done through physical memory locations in the I/O address space. Each I/O device will occupy some location in the I/O address space. That is, it will respond when those addresses are placed on the bus the processor can write those locations to send commands and information to the I/O device. Memory-mapped I/O makes it easy to write device drivers in a high-level language as long as the high-level language can load and store from arbitrary addresses.

**2.4.1 DISK CONTROLLERS:**

The CRA-1 has one disk controller and it controls one fixed disk. Disk commands are initiated by storing them in the disk controller control register. Disk blocks (also called sectors) contain 4096 data bytes. These are only two commands: load disk block from disk into memory, and store disk block from memory to the disk. the disk block is specified by the disk block number, which is 20 bits long. This allows for about a million blocks on the disk, and the total disk size of 4gigabytes.

The address to load into or store from in memory is specified by the disk memory address register, which should be loaded before the disk control register is loaded with a command.

When a disk transfer command is initiated, the disk controller goes into a busy state and remains busy until the transfer is completed. When the transfer completes, the disk controller can optionally causes a disk interrupt to inform the CPU that the disk is no longer is busy.