Operating Systems

1. Multiplexing and De-Multiplexing

Multiplexing: Multiplexing is a term used in the field of communications and computer networking. It generally refers to the process and technique of transmitting multiple analog or digital input signals or data streams over a single channel. Since multiplexing can integrate multiple low-speed channels into one high-speed channel for transmission, the high-speed channel is effectively utilized. By using multiplexing, communication carriers can avoid maintaining multiple lines, therefore, operating costs are effectively saved. Multiplexer is a device which performs the multiplexing process. It is a hardware component that combines multiple analog or digital input signals into a single line of transmission.

In Communication System it increases the efficiency of the communication system by allowing the transmission of data, such as audio and video data transmission.

In Computer Memory it keeps up a vast amount of memory in the computers and decrease the number of copper lines necessary to connect the memory to other parts of the computer as well.

In Telephone Network: integrate the multiple audio signals on a single line of transmission.

De-Multiplexing: Demultiplexing is a term relative to multiplexing. It is the reverse of the multiplexing process. Demultiplex is a process reconverting a signal containing multiple analog or digital signal streams back into the original separate and unrelated signals. Although demultiplexing is the reverse of the multiplexing process, it is not the opposite of multiplexing. The opposite of multiplexing is inverse multiplexing, which breaks one data stream into several related data streams. Thus, the difference between demultiplexing and inverse multiplexing is that the output streams of demultiplexing are unrelated, while the output streams of inverse multiplexing are related. Demultiplexer is a device that performs the reverse process of multiplexer.

In Communication System it receives the output signals from the multiplexer and converts them back to the original form at the receiver end.

In Arithmetic Logic Unit, the output of the arithmetic logic unit is fed as an input to the Demux, and the o/p of the Demux is connected to a multiple register.

In Serial to Parallel Converter, the serial to parallel converter is used to reform parallel data. In this method, serial data are given as an input to the Demux, and a counter is attached to the Demux to sense the data signal at the Demux's o/p. When all data signals are stored, the output of the Demux can be read out in parallel.

2. Time and space multiplexing

Time Multiplexing: Time multiplexing means a resource is not divided into units. Instead, a program is allocated exclusive control of the entire resource for a short period of time. After that time has elapsed, the resource is deallocated from the program and allocated to another. Time-multiplexing is used with the processor resource.

Space Multiplexing: Space-multiplexing means that the resource can be divided into two or more distinct units of the resource. Different executing programs can be allocated exclusive control of different units of a resource at the same time. Memory and disks are examples of space-multiplexed resources.

3. System Resources

In a computer system, system resources are the components that provide its inherent capabilities and contribute to its overall performance. System memory, cache memory, hard disk space, IRQs and DMA channels are examples. In an operating system, system resources are internal tables and pointers set up to keep track of running applications. They may be limited by hardware resources but are often as not arbitrary limitations within the software itself.

PCs have four types of system resources Interrupt Request lines, DMA channels, I/O ports, and memory ranges. Many system components and peripherals require one or more of these resources, which raises the twin problems of resource availability and resource conflicts. Resource availability is particularly important about IRQs, which are in high demand, and of which only 16 exist. Resource conflicts can occur when two devices are assigned the same resource, in which case one or both devices may not function and may function unpredictably.

Resource conflicts may occur even with plentiful resources, such as I/O ports, where many are available and only a few are in use.

4. Batch Systems

Batch Operating system is one of the important types of operating system. The users who using a batch operating system do not interact with the computer directly. Each user prepares its job on an off-line device like punch cards and submits it to the computer operator. To speed up the processing, jobs with similar needs are batched together and run as a group. The programmers exit their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems that occur with Batch Systems are as follows:

- There is a lack of interaction between the user and the job.
- CPU is being often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- It is difficult to provide the desired priority.

Batch processing is a method of running high-volume, repetitive data jobs. The batch method allows users to process data when computing resources are available, and with little or no user interaction. With batch processing, users collect and store data, and then process the data during an event known as a "batch window." Batch processing improves efficiency by setting processing priorities and completing data jobs at a time that makes the most sense. The batch processing method was first used in the 19th century by Herman Hollerith, an American inventor who created the first tabulating machine. This device became the precursor to the modern computer, capable of counting and sorting data organized in the form of punched cards. The cards and the information they contained could then be collected and processed together in batches. This innovation allowed large amounts of data to be processed more quickly and accurately than by manual entry methods.

Batch processing plays a critical role in helping companies and organizations manage large amounts of data efficiently. It is especially suited for handling frequent, repetitive tasks such as accounting processes. In every industry and for every job, the basics of batch processing remain the same. The essential parameters include:

- who is submitting the job?

- which program will run?
- the location of the input and outputs
- when the job should be run.

5. Overheads Generation

In computers, overhead refers to the processing time required by system software, which includes the operating system and any utility that supports application programs. CPU overhead measures the amount of work on a computer's central processing unit can perform and the percentage of that capacity that's used by individual computing tasks. Some of these tasks involve supporting the functions of the computer's operating system. These include communication with internal components, such as hard drives and graphics processing units, with connective functionality such as networking support, and with external components like your computer's monitor. Other tasks that require CPU overhead provide processor power for applications. In total, these tasks must require less than the processor's overall capacity.

Ongoing Task: Your operating system must power your screen with the elements of your graphical user interface, or GUI, and talk to your internal and external hardware. Many of these tasks continue for as long as you keep your computer powered on and in an active state. Each task requires at least a little bit of processor power, and thus contributes to processor overhead. If you run a virtual machine a simulated computer entirely based in software. your computer must support your primary operating system as well as your emulated operating system.

Momentary Task: Some computing tasks last only long enough to print a document, burn a disk, send a message, or play an alert sound. Although your computer's ongoing operating processes may be responsible for some of these temporary duties, others call on software modules that take care of business and then shut down. Still more tasks remain active only as long as you keep a particular application running. If the combined CPU overhead of these processes exceeds the power of your CPU, your computer can grind to a halt.

Unexpected Overheads: Operating system or software bugs can draw too many CPU clock cycles and overtax your system unexpectedly. If you install a new operating system or a new application version and notice your system slowing dramatically, you may want to check in at online discussion and support venues to see if other users experience the same problems. If you see many reports

of troubles like your own, it's a good bet that you'll see a software or OS update from the manufacturer to correct the problem.

6. Software-defined networking concepts for Operating Systems

Software-Defined Networking (SDN) is a network architecture approach that enables the network to be intelligently and centrally controlled, or programmed, using software applications. This helps operators manage the entire network consistently and holistically, regardless of the underlying network technology. SDN enables the programming of network behavior in a centrally controlled manner through software applications using open APIs. By opening traditionally closed network platforms and implementing a common SDN control layer, operators can manage the entire network and its devices consistently, regardless of the complexity of the underlying network technology.

In SDN (like anything virtualized), the software is decoupled from the hardware. SDN moves the control plane that determines where to send traffic to software and leaves the data plane that forwards the traffic in the hardware. This allows network administrators who use software-defined networking to program and control the entire network via a single pane of glass instead of on a device-by-device basis.

There are three parts to a typical SDN architecture, which may be in different physical locations:

Applications, which communicate resource requests or information about the network as a whole

Controllers, which use the information from applications to decide how to route a data packet

Networking devices, which receive information from the controller about where to move the data

Physical or virtual networking devices move the data through the network. In some cases, virtual switches, which may be embedded in either the software or the hardware, take over the responsibilities of physical switches and consolidate their functions into a single, intelligent switch. The switch checks the integrity of both the data packets and their virtual machine destinations and moves the packets along.

7. Scheduling

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

There are three types of process scheduler.

1. **Long Term or job scheduler :**
   It brings the new process to the 'Ready State'. It controls **Degree of Multi-programming**, i.e., number of process present in ready state at any point of time. It is important that the long-term scheduler make a careful selection of both IO and CPU bound process. IO bound tasks are which use much of their time in input and output operations while CPU bound processes are which spend their time on CPU. The job scheduler increases efficiency by maintaining a balance between the two.

2. **Short term or CPU scheduler :**
   It is responsible for selecting one process from ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes.
   Dispatcher is responsible for loading the process selected by Short-term scheduler on the CPU (Ready to Running State) Context switching is done by dispatcher only. A dispatcher does the following:
   a. Switching context.
   b. Switching to user mode.
   c. Jumping to the proper location in the newly loaded program.
3. **Medium-term scheduler :**
   It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.

8. Memory Read and Write Operations

A memory unit stores binary information in groups of bits called words. Data input lines provide the information to be stored into the memory, Data output lines carry the information out from the memory. The control lines Read and write specifies the direction of transfer of data. Basically, in the memory organization, there are 2^l memory locations indexing from 0 to $2^l - 1$ where l is the address buses. We can describe the memory in terms of the bytes using the following formula:

$$N = 2^l Bytes$$

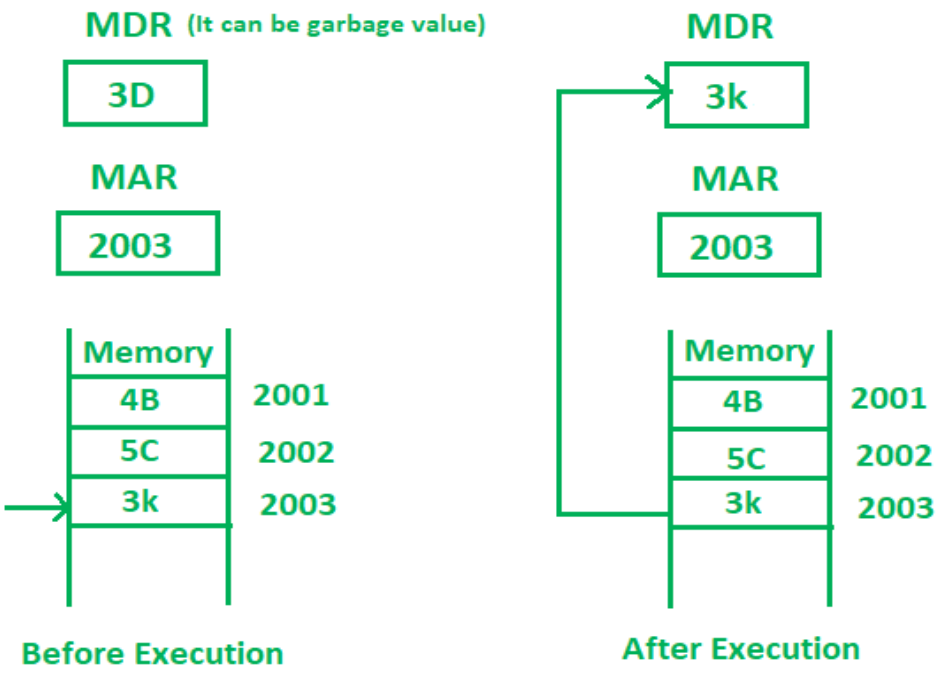
Where,
l is the total address buses
N is the memory in bytes

For example, some storage can be described below in terms of bytes using the above formula:

1kB= 2^{10} Bytes
64 kB = 2^6 x 2^{10} Bytes
= 2^{16} Bytes
4 GB = 2^2 x 2^{10}(kB) x 2^{10}(MB) x 2^{10} (GB)
= 2^{32} Bytes

Memory Address Register (MAR) is the address register which is used to store the address of the memory location where the operation is being performed.
Memory Data Register (MDR) is the data register which is used to store the data on which the operation is being performed.

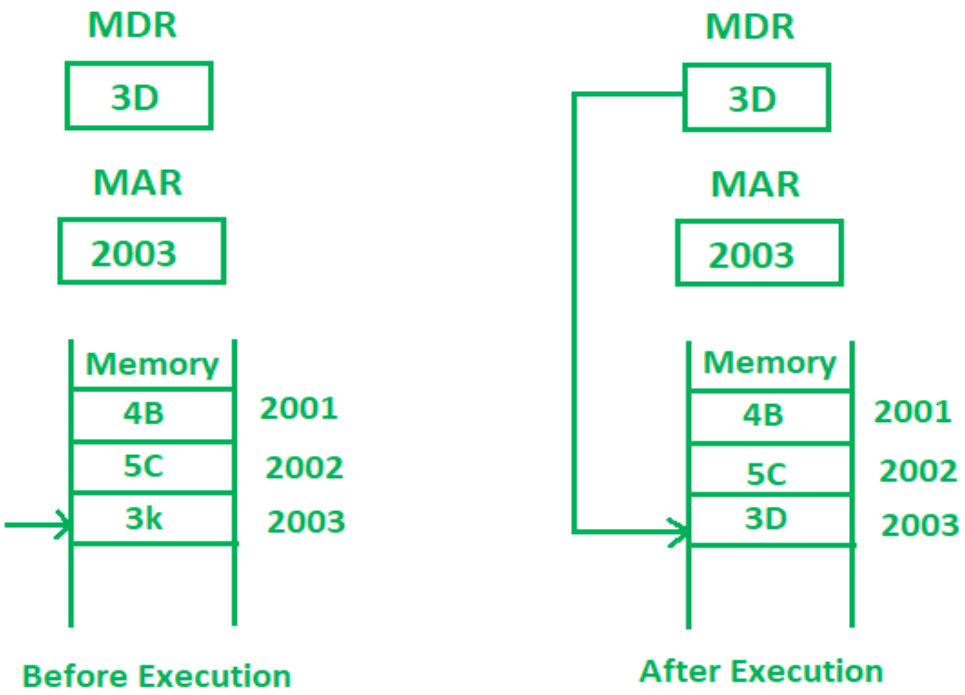
1. **Memory Read Operation:**
   Memory read operation transfers the desired word to address lines and activates the read control line.Description of memory read operation is given below:



Memory Read Operation

In the above diagram initially, MDR can contain any garbage value and MAR is containing 2003 memory address. After the execution of read instruction, the data of memory location 2003 will be read and the MDR will get updated by the value of the 2003 memory location (3D).

2. **Memory Write Operation:**
   Memory write operation transfers the address of the desired word to the address lines, transfers the data bits to be stored in memory to the data input lines. Then it activates the write control line. Description of the write operation is given below:



Memory Write Operation

In the above diagram, the MAR contains 2003 and MDR contains 3D. After the execution of write instruction 3D will be written at 2003 memory location.

9. Data, address, and control information for read and write operations

A bus is a pathway for digital signals to rapidly move data. There are three internal buses associated with processors: the data bus, address bus, and control bus. Together, these three make up the "system bus." The system bus is an internal bus, intended to connect the processor with internal hardware devices, and is also called the "local" bus, Front Side Bus, or is sometimes loosely referred to as the "memory bus."

Data moving in and out of the data bus is bi-directional, since the processor reads and writes data, however, the others are uni-directional, since the processor always determines when and what it will read from or write to. The address bus carries addressing signals from the processor to memory, I/O (or peripherals), and other addressable devices around the processor. Control signals move out of the processor, but not in to it.

The data bus "width" of an MCU is typically 8-, 16-, 32- or 64-bits, although MCUs of just a 4-bit data bus or greater than 64-bit width are possible. The

width of the data bus reflects the maximum amount of data that can be processed and delivered at one time. A 64-bit processor has a 64-bit data bus and can communicate 64-bits of data at a time, and whether the data is read or written is determined by the control bus. The physical location of the data in memory is carried by the address bus. An internal hardware component, having received the address from the address bus and about to receive the data, enables a buffer to allow the flow of signals to or from the location that was designated by the address bus. The address bus carries only the information regarding the address, and is synchronized with the data bus to accomplish read/write tasks from the processor. The address bus is only as wide as is necessary to address all memory in the system.

Other communication buses also communicate with the processor but are external to the system, such as Universal Serial Bus, RS-232, Controller Area Network (CAN), eSATA, and others. External peripherals may be set up to use the internal bus, and this was common with computers that used "expansion cards" to connect products to the internal bus. However, with one card per device this became untenable for the long term, and other bus communication systems such as USB were developed.

A system bus can be "extended" to communicate with other computers via a chassis called a backplane. Internal buses have very rapid throughput and low latency. Several computers can be rack-mounted in a single backplane for very fast communication between computers.

10. CPU, memory registers, instruction register, instruction pointers, control unit, arithmetic logic unit

CPU:

Central processing unit (CPU), principal part of any digital computer system, generally composed of the main memory, control unit, and arithmetic-logic unit. It constitutes the physical heart of the entire computer system; to it is linked various peripheral equipment, including input/output
devices and auxiliary storage units. In modern computers, the CPU is contained on an integrated circuit chip called a microprocessor.

The control unit of the central processing unit regulates and integrates the operations of the computer. It selects and retrieves instructions from the main memory in proper sequence and interprets them so as to activate the other functional elements of the system at the appropriate moment to perform their

respective operations. All input data are transferred via the main memory to the arithmetic-logic unit for processing, which involves the four basic arithmetic functions (i.e., addition, subtraction, multiplication, and division) and certain logic operations such as the comparing of data and the selection of the desired problem-solving procedure or a viable alternative based on predetermined decision criteria.

Memory registers:

Register memory is the smallest and fastest memory in a computer. It is not a part of the main memory and is located in the CPU in the form of registers, which are the smallest data holding elements. A register temporarily holds frequently used data, instructions, and memory address that are to be used by CPU. They hold instructions that are currently processed by the CPU. All data is required to pass through registers before it can be processed. So, they are used by CPU to process the data entered by the users.

Instruction registers:

In computing, the instruction register (IR) or current instruction register (CIR) is the part of a CPU's control unit that holds the instruction currently being executed or decoded. In simple processors, each instruction to be executed is loaded into the instruction register, which holds it while it is decoded, prepared and ultimately executed, which can take several steps.

Some of the complicated processors use a pipeline of instruction registers where each stage of the pipeline does part of the decoding, preparation or execution and then passes it to the next stage for its step. Modern processors can even do some of the steps out of order as decoding on several instructions is done in parallel.

Decoding the op-code in the instruction register includes determining the instruction, determining where its operands are in memory, retrieving the operands from memory, allocating processor resources to execute the command (in super scalar processors), etc.

The output of the IR is available to control circuits, which generate the timing signals that control the various processing elements involved in executing the instruction.

In the instruction cycle, the instruction is loaded into the instruction register after the processor fetches it from the memory location pointed to by the program counter.

Instruction pointers:

In most processors, the instruction pointer is incremented after fetching an instruction, and holds the memory address of ("points to") the next instruction that would be executed. (In a processor where the incrementation precedes the fetch, instruction pointer points to the current instruction being executed.)

Control unit:

A **control unit** or **CU** is circuitry that directs operations within a computer's processor. It lets the computer's logic unit, memory, and both input and output devices know how to respond to instructions received from a program. Examples of devices that utilize control units include CPUs and GPUs.

A control unit works by receiving input information that it converts into control signals, which are then sent to the central processor. The computer's processor then tells the attached hardware what operations to carry out. The functions that a control unit performs are dependent on the type of CPU, due to the variance of architecture between different manufacturers.

Arithmetic Logic unit:

In computing, an arithmetic logic unit (ALU) is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers. This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. It is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs).

The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed; the ALU's output is the result of the performed operation. In many designs, the ALU also has status inputs or outputs, or both, which convey information about a previous operation or the current operation, respectively, between the ALU and external status registers.

11. Interrupts and exceptions

Interrupt is one of the classes of Exception. There are 4 classes of Exception- interrupt, trap, fault and abort. Though, interrupt belongs to exception still there are many differences between them.

In any computer, during its normal execution of a program, there could be events that can cause the CPU to temporarily halt. Events like this are called interrupts. Interrupts can be caused by either software or hardware faults. Hardware interrupts are called Interrupts, while software interrupts are called Exceptions. Once an interrupt is raised, the control is transferred to a special sub-routine called Interrupt Service Routine (ISR), that can handle the conditions that are raised by the interrupt.

What is Trap, Fault and Abort ?

1. **Trap –**
   It is typically a type of synchronous interrupt caused by an exceptional condition (e.g., breakpoint, division by zero, invalid memory access).
2. **Fault –**
   Fault exception is used in a client application to catch contractually-specified SOAP faults. By the simple exception message, you can't identify the reason of the exception, that's why a Fault Exception is useful.
3. **Abort –**
   It is a type of exception occurs when an instruction fetch causes an error.

What is Interrupt ?

The term Interrupt is usually reserved for hardware interrupts. They are program control interruptions caused by external hardware events. Here, external means external to the CPU. Hardware interrupts usually come from many different sources such as timer chip, peripheral devices (keyboards, mouse, etc.), I/O ports (serial, parallel, etc.), disk drives, CMOS clock, expansion cards (sound card, video card, etc). That means hardware interrupts almost never occur due to some event related to the executing program.

Example –

An event like a key press on the keyboard by the user, or an internal hardware timer timing out can raise this kind of interrupt and can inform the CPU that a certain device needs some attention. In a situation like that the CPU will stop whatever it was doing (i.e. pauses the current program), provides the service required by the device and will get back to the normal program. When hardware interrupts occur and the CPU starts the ISR, other hardware interrupts are disabled (e.g. in 80×86 machines). If you need other hardware interrupts to occur while the ISR is running, you need to do that explicitly by clearing the interrupt flag (with sti instruction). In 80×86 machines, clearing the interrupt flag will only affect hardware interrupts.

What is Exception ?

Exception is a software interrupt, which can be identified as a special handler routine. Exception can be identified as an automatically occurring trap.

Generally, there are no specific instructions associated with exceptions (traps are generated using a specific instruction). So, an exception occurs due to an "exceptional" condition that occurs during program execution.

Example –

Division by zero, execution of an illegal opcode or memory related fault could cause exceptions. Whenever an exception is raised, the CPU temporarily suspends the program it was executing and starts the ISR. ISR will contain what to do with the exception. It may correct the problem or if it is not possible it may abort the program gracefully by printing a suitable error message. Although a specific instruction does not cause an exception, an exception will always be caused by an instruction. For example, the division by zero error can only occur during the execution of the division instruction.

Difference between Interrupt and Exception :

Interrupt	Exception
These are Hardware interrupts.	These are Software Interrupts.
Occurrences of hardware interrupts usually disable other hardware interrupts.	This is not a true case in terms of Exception.
These are asynchronous external requests for service (like keyboard or printer needs service).	These are synchronous internal requests for service based upon abnormal events (think of illegal instructions, illegal address, overflow etc).
These are normal events and shouldn't interfere with the normal running of a computer.	These are abnormal events and often result in the termination of a program.

Interrupts vs Exceptions

- Varying terminology but for Intel
- Interrupt (synchronous, device generated)
- Maskable device-generated, associated with IRQs (interrupt request lines) may be temporarily disabled (still pending)
- Nonmaskable some critical hardware failures
- Exceptions (asynchronous)
- Processor-detected
- Faults correctable (restartable) e.g. page fault
- Traps no reexecution needed e.g. breakpoint
- Aborts severe error process usually terminated (by signal)
- Programmed exceptions (software interrupts)
- int (system call), int3 (breakpoint)
- into (overflow), bounds (address check)

12. Device status table

It keeps a track record of device interrupts. A record in a device status table can have three status namely.

- Not Functioning
- Idle
- Busy

If a device is busy the new request upcoming request is saved in queue. Similarly the idle and busy status represents the two different states of a device.

13. Process Creation

A process creation is a set of sequenced programs executed to achieve a particular objective. A process normally consists of three main stages.

1. Process initiation
2. Process execution
3. Process termination

A process can be executed due to one of the following reasons:

1. Submit a new batch/job.
2. Can already running process.
3. Facilitating the termination of an existing process.
4. Pooling the running processes to accommodate hyphen resources.

Process are created by using the init thread. Similarly the fork() procedure is used to schedule the execution of a new process. The fork() procedure is used to create the relation between the new processes (child processes) with the existing process (parent process). This create new structures (data structure) which is also handled by the operating system.

A child process inherits memory, stack, environment, root directory and resources. The wait system call is used to enable the child program to return an output to the parent program. If a child programs wants to terminate a parent program we use a call procedure called zombie. The zombie call is not mentioned in the process table.

Exit () terminate a program, de allocate memory, de assign resources and values.

Wait() enable output to parent program

Init () initiates a task

Execv () child process run another program.

14. Process Control Block

It is a list of memory locations from min (read as 0) to max (read as n) where a process can read and write. It might contains

1. Executable program
2. Program data
3. Stack

These programs are connected to perform a similar task or objective.