Dr. N. SHAKEELA

Guest Lecturer

School of Computer Science, Engineering & Applications

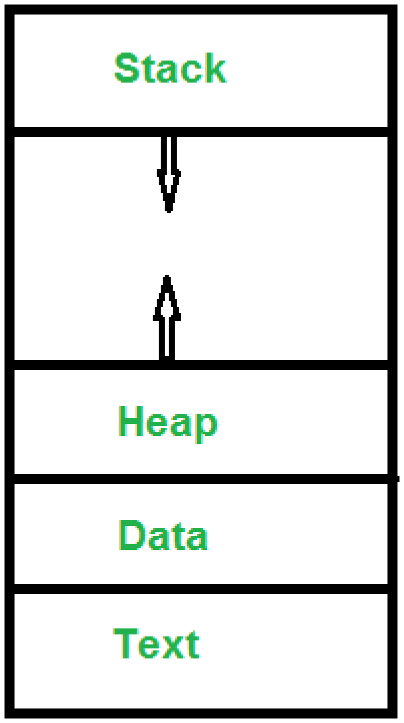
Bharathidasan University

Trichy-24

Introduction of Process Management

Program vs Process
A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.
A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

What does a process look like in memory?



Text Section**:*A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the ***Program Counter.
***Stack**:* The Stack contains the temporary data, such as function parameters, returns addresses, and local variables.
***Data Section**:* Contains the global variable.
***Heap Section**:* Dynamically allocated memory to process during its run time.

Attributes or Characteristics of a Process
A process has following attributes.

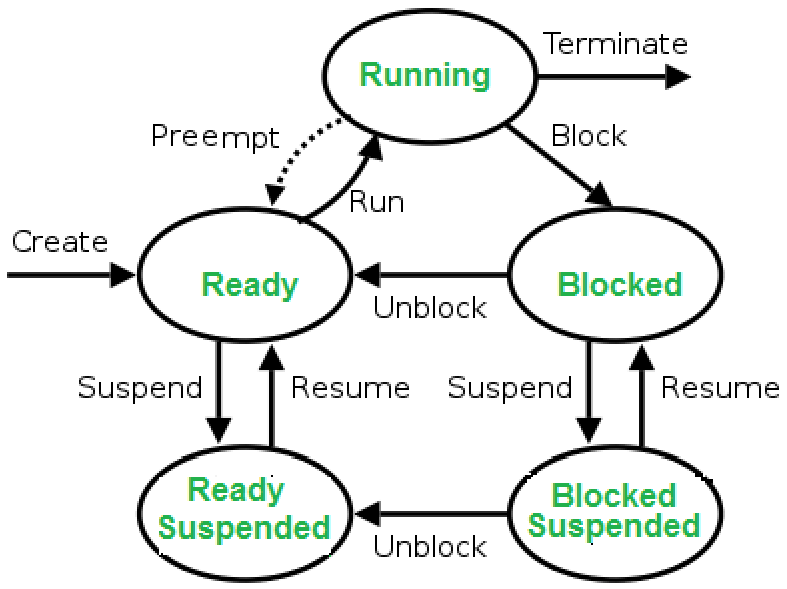
```
1. Process Id:    A unique identifier assigned by the operating system
2. Process State: Can be ready, running, etc.
```

```
3. CPU registers: Like the Program Counter (CPU registers must be saved and
                  restored when a process is swapped in and out of CPU)
5. Accounts information:
6. I/O status information: For example, devices allocated to the process,
                           open files, etc
8. CPU scheduling information: For example, Priority (Different processes
                               may have different priorities, for example
                               a short process may be assigned a low priority
                               in the shortest job first scheduling)
```

All of the above attributes of a process are also known as the ***context of the process***.
Every process has its own process control block(PCB), i.e each process will have a unique PCB. All of the above attributes are part of the PCB.

States of Process:
A process is in one of the following states:

```
1. New: Newly Created Process (or) being-created process.

2. Ready: After creation process moves to Ready state, i.e. the
          process is ready for execution.

3. Run: Currently running process in CPU (only one process at
        a time can be under execution in a single processor).

4. Wait (or Block): When a process requests I/O access.

5. Complete (or Terminated): The process completed its execution.

6. Suspended Ready: When the ready queue becomes full, some processes
                    are moved to suspended ready state

7. Suspended Block: When waiting queue becomes full.
```



Context Switching
The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from running state to ready state.

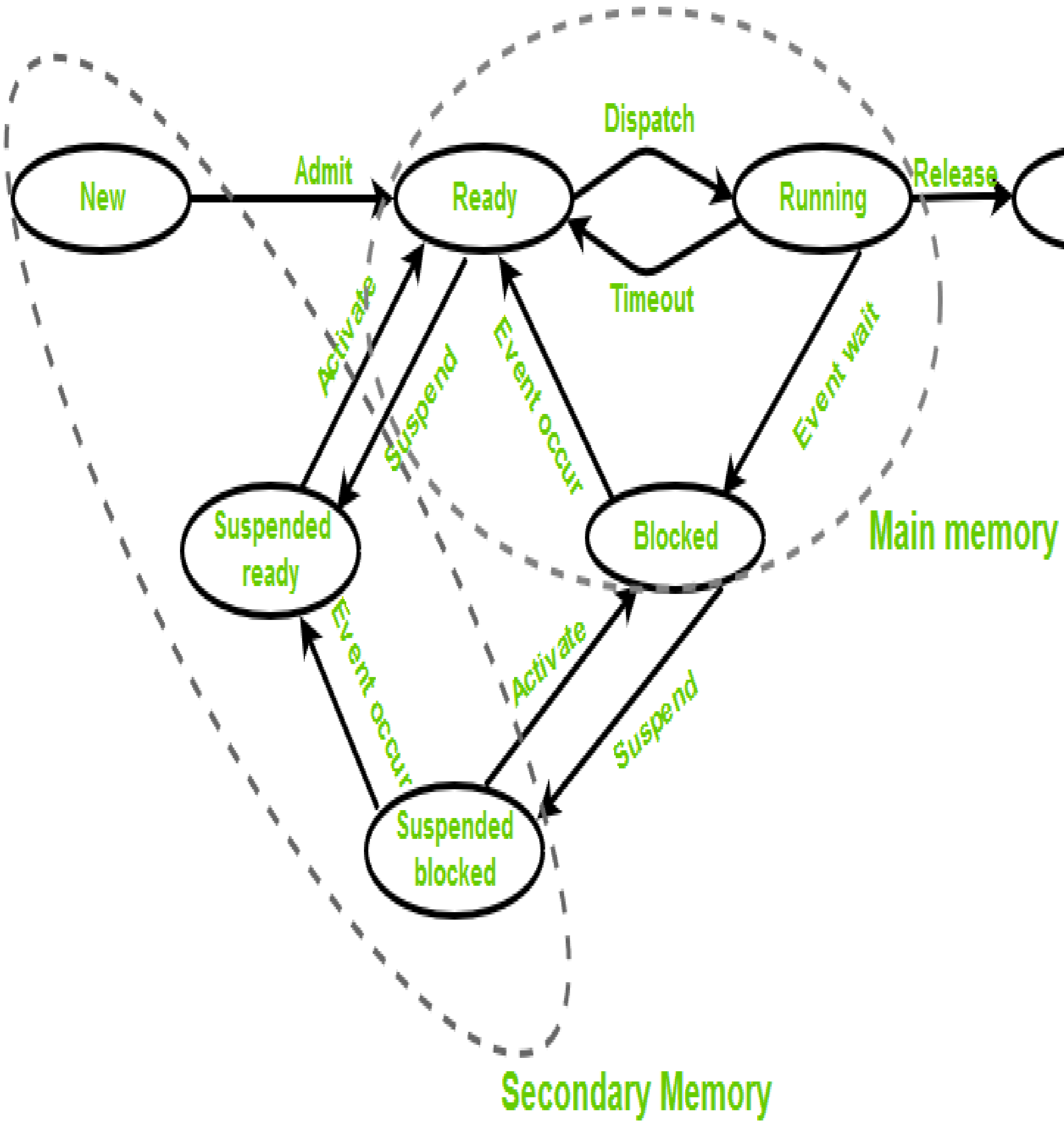
When does context switching happen?
1. When a high-priority process comes to ready state (i.e. with higher priority than the running process)
2. An Interrupt occurs
3. User and kernel mode switch
4. Preemptive CPU scheduling used.

Context Switch vs Mode Switch
A mode switch occurs when CPU privilege level is changed, for example when a system call is made or a fault occurs. The kernel works in more a privileged mode than a standard user task. If a user process wants to access things which are only accessible to the kernel, a mode switch must occur. The currently executing process need not be changed during a mode switch.

CPU-Bound vs I/O-Bound Processes:
A CPU-bound process requires more CPU time or spends more time in the running state.
An I/O-bound process requires more I/O time and less CPU time. An I/O-bound process spends more time in the waiting state.

States of a Process in Operating Systems

States of a process are as following:



- **New (Create) –** In this step, the process is about to be created but not yet created, it is the program which is present in secondary memory that will be picked up by OS to create the process.

- **Ready –** New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue for ready processes.
- **Run –** The process is chosen by CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
- **Blocked or wait –** Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or wait state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
- **Terminated or completed –** Process is killed as well as PCB is deleted.
- **Suspend ready –** Process that was initially in the ready state but were swapped out of main memory and placed onto external storage by scheduler are said to be in suspend ready state. The process will transition back to ready state whenever the process is again brought onto the main memory.
- **Suspend wait or suspend blocked –** Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory.
  When work is finished it may go to suspend ready.

Types of schedulers:

1. **Long term – performance –** Makes a decision about how many processes should be made to stay in the ready state, this decides the degree of multiprogramming. Once a decision is taken it lasts for a long time hence called long term scheduler.
2. **Short term – Context switching time –** Short term scheduler will decide which process to be executed next and then it will call dispatcher. A dispatcher is a software that moves process from ready to run and vice versa. In other words, it is context switching.
3. **Medium term – Swapping time –** Suspension decision is taken by medium term scheduler. Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.

Multiprogramming – We have many processes ready to run. There are two types of multiprogramming:

1. **Pre-emption –** Process is forcefully removed from CPU. Pre-emption is also called as time sharing or multitasking.
2. **Non pre-emption –** Processes are not removed until they complete the execution.

Degree of multiprogramming –

The number of processes that can reside in the ready state at maximum decides the degree of multiprogramming, e.g., if the degree of programming = 100, this means 100 processes can reside in the ready state

Process Schedulers in Operating System

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.

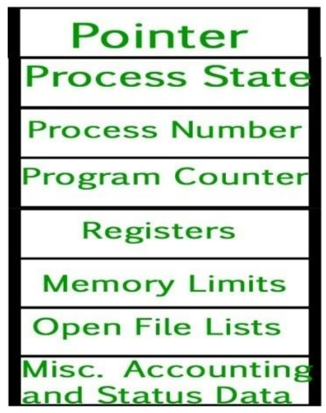
2. **Short term or CPU scheduler :**
   It is responsible for selecting one process from ready state for scheduling it on the running state. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes.
   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:
   1. Switching context.
   2. Switching to user mode.
   3. 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

Process Table and Process Control Block (PCB)

While creating a process the operating system performs several operations. To identify the processes, it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB) is used to track the process's execution status. Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc. All these information is required and must be saved when the process is switched from one state to another. When the process makes a transition from one state to another, the operating system must update information in the process's PCB.

A process control block (PCB) contains information about the process, i.e. registers, priority, etc. The process table is an array of PCB's, that means logically contains a PCB for all of the current processes in the system.

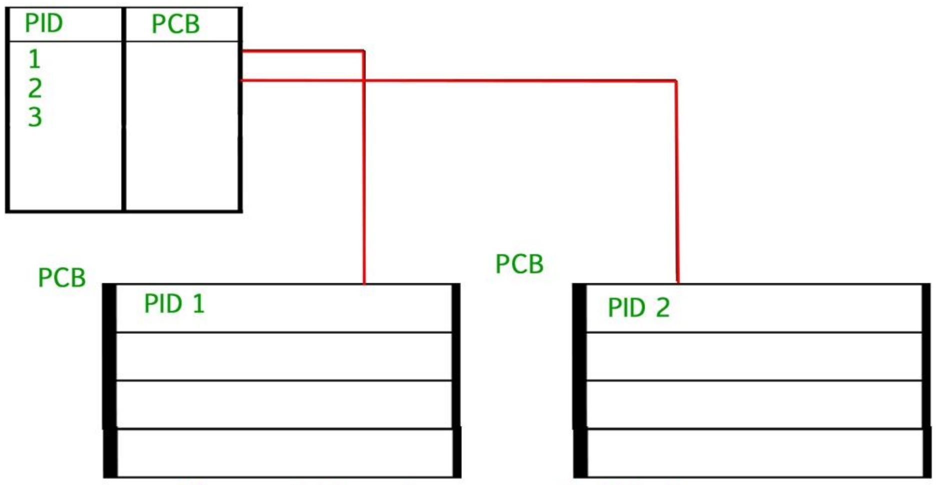


Process Control Block

- **Pointer –** It is a stack pointer which is required to be saved when the process is switched from one state to another to retain the current position of the process.
- **Process state –** It stores the respective state of the process.
- **Process number –** Every process is assigned with a unique id known as process ID or PID which stores the process identifier.
- **Program counter –** It stores the counter which contains the address of the next instruction that is to be executed for the process.
- **Register –** These are the CPU registers which includes: accumulator, base, registers and general purpose registers.
- **Memory limits –** This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.
- **Open files list –** This information includes the list of files opened for a process.

Miscellaneous accounting and status data – This field includes information about the amount of CPU used, time constraints, jobs or process number, etc.
The process control block stores the register content also known as execution content of the processor when it was blocked from running. This execution content architecture enables the operating system to restore a process's execution context when the process returns to the running state. When the process makes a transition from one state to another,

the operating system updates its information in the process's PCB. The operating system maintains pointers to each process's PCB in a process table so that it can access the PCB quickly.



Process table and process control block

Interrupts

Interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high priority process requiring interruption of the current working process. One of the bus control lines is dedicated for this purpose and is called the *Interrupt Service Routine (ISR).*
When a device raises an interrupt at lets say process i, the processor first completes the execution of instruction i. Then it loads the Program Counter (PC) with the address of the first instruction of the ISR. Before loading the Program Counter with the address, the address of the interrupted instruction is moved to a temporary location. Therefore, after handling the interrupt the processor can continue with process i+1.

While the processor is handling the interrupts, it must inform the device that its request has been recognized so that it stops sending the interrupt request signal. Also, saving the registers so that the interrupted process can be restored in the future, increases the delay between the time an interrupt is received and the start of the execution of the ISR. This is called Interrupt Lattency.

Hardware Interrupts:
In a hardware interrupt, all the devices are connected to the Interrupt Request Line. A single request line is used for all the n devices.
Sequence of events involved in handling an IRQ:

1. Devices raise an IRQ.
2. Processor interrupts the program currently being executed.
3. Device is informed that its request has been recognized and the device deactivates the request signal.
4. The requested action is performed.
5. Interrupt is enabled and the interrupted program is resumed.

Handling Multiple Devices:
When more than one device raises an interrupt request signal, then additional information is needed to decide which which device to be considered first. The following methods are used to decide which device to select: Polling, Vectored Interrupts, and Interrupt Nesting. These are explained as following below.

1. **Polling:**
   In polling, the first device encountered with with IRQ bit set is the device that is to be serviced first. It is easy to implement but a lot of time is wasted by interrogating the IRQ bit of all devices.
2. **Vectored Interrupts:**
   In vectored interrupts, a device requesting an interrupt identifies itself directly by sending a special code to the processor over the bus. This enables the processor to identify the device that generated the interrupt. The

special code can be the starting address of the ISR or where the ISR is located in memory, and is called the interrupt vector.

3. **Interrupt Nesting:**
   In this method, I/O device is organized in a priority structure. Therefore, interrupt request from a higher priority device is recognized where as request from a lower priority device is not. To implement this each process/device (even the processor). Processor accepts interrupts only from devices/processes having priority more than it.

Processors priority is encoded in a few bits of PS (Process Status register). It can be changed by program instructions that write into the PS. Processor is in supervised mode only while executing OS routines. It switches to user mode before executing application programs.

Thread in Operating System

Last Updated: 16-08-2019

What is a Thread?
A thread is a path of execution within a process. A process can contain multiple threads.

Why Multithreading?
A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

Process vs Thread?
The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.
Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Advantages of Thread over Process
1. Responsiveness: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.
2. Faster context switch: Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.
3. Effective utilization of multiprocessor system: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processor. This will make process execution faster.
4. Resource sharing: Resources like code, data, and files can be shared among all threads within a process.
Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.
5. Communication: Communication between multiple threads is easier, as the threads shares common address space. while in process we have to follow some specific communication technique for communication between two process.

6. Enhanced throughput of the system: If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system..

Threads and its types in Operating System

Thread is a single sequence stream within a process. Threads have same properties as of the process so they are called as light weight processes. Threads are executed one after another but gives the illusion as if they are executing in parallel. Each thread has different states. Each thread has

1. A program counter
2. A register set
3. A stack space

Threads are not independent of each other as they share the code, data, OS resources etc.

Similarity between Threads and Processes –

- Only one thread or process is active at a time
- Within process both execute sequentiall
- Both can create children

Differences between Threads and Processes –

- Threads are not independent, processes are.
- Threads are designed to assist each other, processes may or may not do it

Types of Threads:

1. **User Level thread (ULT) –**
   Is implemented in the user level library, they are not created using the system calls. Thread switching does not need to call OS and to cause interrupt to Kernel. Kernel doesn't know about the user level thread and manages them as if they were single-threaded processes.
   Advantages of ULT –
   - Can be implemented on an OS that does't support multithreading.
   - Simple representation since thread has only program counter, register set, stack space.
   - Simple to create since no intervention of kernel.
   - Thread switching is fast since no OS calls need to be made.

   Disadvantages of ULT –
   - No or less co-ordination among the threads and Kernel.
   - If one thread causes a page fault, the entire process blocks.
2. **Kernel Level Thread (KLT) –**
   Kernel knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table (a master one) that keeps track of all the threads in the system. In addition kernel also maintains the traditional process table to keep track of the processes. OS kernel provides system call to create and manage threads.
   Advantages of KLT –
   - Since kernel has full knowledge about the threads in the system, scheduler may decide to give more time to processes having large number of threads.
   - Good for applications that frequently block.

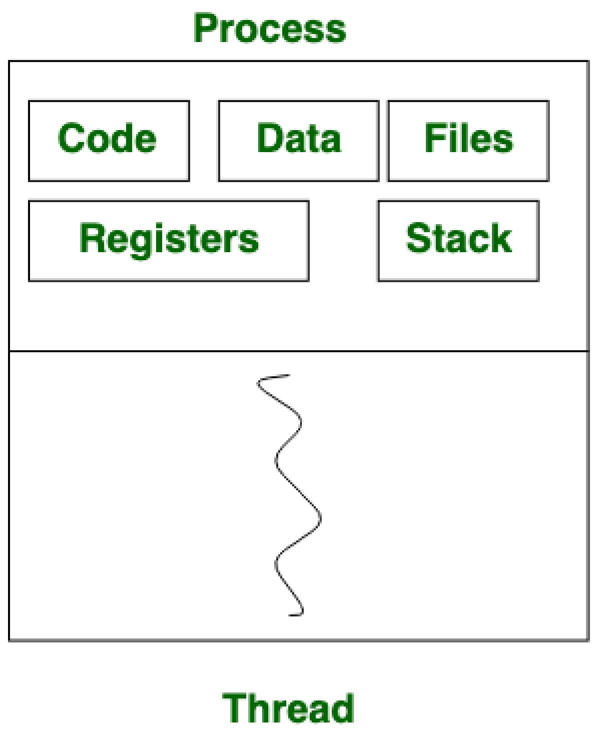
   Disadvantages of KLT –
   - Slow and inefficient.
   - It requires thread control block so it is an overhead.

Difference between Process and Thread

Last Updated: 17-09-2020

Process:
Process means any program is in execution. Process control block controls the operation of any process. Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc. A process can creates other processes which are known as **Child Processes**. Process takes more time to terminate and it is isolated means it does not share memory with any other process.

Thread:
Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread have 3 states: running, ready, and blocked.
Thread takes less time to terminate as compared to process and like process threads do not isolate.



Difference between Process and Thread:

S.NO	PROCESS	THREAD
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Process consume more resources.	Thread consume less resources.
7.	Process is isolated.	Threads share memory.

8.	Process is called heavy weight process.	Thread is called light weight process.
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.
10.	If one server process is blocked no other server process can execute until the first process unblocked.	Second thread in the same task could run, while one server thread is blocked.
11.	Process has its own Process Control Block, Stack and Address Space.	Thread has Parents' PCB, its own Thread Control Block and Stack and common Address space.