**Ques 2**: Study the process table data structures used by Linux OS. List the fields used with their purposes**.**

**Ans:** Linux operating system has two main structures for each process: the PCB, which is a structure of type **struct task\_struct**, and another one of type **struct thread\_info**.

* + The PCB **struct task\_struct** holds most of the information of the process.
  + The structure **struct thread\_ info** contains low level information of the process.

The address of the **task\_struct** and of the **thread\_info** of a process is unique, so it is often used in the kernel to identify processes

The structure **thread\_info** is part of **union thread\_union** that holds the low-level information of the process (**struct thread\_info**) and the kernel stack of that process.  
Linux Tasks

**A process, or task, in Linux is represented by a task\_struct data structure. The task\_struct data structure contains information in a number of categories**:

* [**State**](#L529):It contains the execution state of the process (executing, ready, suspended, stopped, zombie. The fact that this variable is declared as **volatile** is used to tell the compiler that the variable can be changed asynchronously (e.g. from an interrupt handler).
* **Scheduling information**: Information needed by Linux to schedule processes. A process can be normal or real time and has a priority. Real-time processes are scheduled before normal processes, and within each category, relative priorities can be used. A counter keeps track of the amount of time a process is allowed to execute.
* **prio**: Dynamic priority of the process. This is used by the scheduler to decide which is the next process to be run.
* **static\_ prio**: Static or nominal priority of the process.
* **run\_ list**: Pointers to the previous and next processes in the run queue where the process is located.
* [**array**](#L539): Pointer to the structure of the active priorities in the run queue of the process.
* [**activated**](#L543):Condition code used to wake the process.
* [**policy**](#L545):Schedule policy used by the process.SCHED\_NORMAL, SCHED\_RR, or SCHED\_FIFO
* [**cpus\_allowed**](#L546): Bitmap indicating which CPUs can run the process.
* [**first\_time\_slice**](#L547):If it is 1, signals that the process has not yet consumed its first quantum.
* [**rt\_priority**](#L595):Real time priority of the process, if it has one.
* [**tasks**](#L553): Pointers to the previous and next processes in the process table.
* [**mm**](#L561): Is a pointer to a data structure used by the memory manager containing information on the memory map of the process. It must be highlighted that a it is a pointer and can point to a structure shared by several processes (e.g. the threads of a UNIX process).
* [**active\_mm**](#L561): contains information about the active memory map.Its value is usually the same as the field **mm**, but in the kernel threads. Since they do not have user memory assigned, they use the memory map of the last user process.
* **Identifiers**: Each process has a unique process identifier and also has user and group identifiers. A group identifier is used to assign resource access privileges to a group of processes.
* [**pid**](#L571): Contains the process identifier.
* [**tpid**](#L572): Contains the process group identifier. It is the same as the group leader's **pid**.
* **uid, euid, suid ,fsuid :**Owner of the process, real (**uid**), and effective (**euid**), and other specific attributes.
* [**gid,egid,sgid,fsgid**](#L607): Owner group of the process, real (**gid**), and effective (**egid**), and other specific attributes.
* **Interprocess communication**: Linux supports the IPC mechanisms found in UNIX SVR4.
* IPC\_CREAT—This flag indicates that a new segment should be created.This permits creating a new segment while specifying a key value.
* IPC\_EXCL—This flag, which is always used with IPC\_CREAT, causes shmget to fail if a segment key is specified that already exists.Therefore, it arranges for the calling process to have an “exclusive” segment. If this flag is not given and the key of an existing segment is used, shmget returns the existing segment instead of creating a new one.
* Mode flags—This value is made of 9 bits indicating permissions granted to owner, group, and world to control access to the segment. Execution bits are ignored. An easy way to specify permissions is to use the constants defined in <sys/stat.h> and documented in the section 2 stat man page. For example, S\_IRUSR and S\_IWUSR specify read and write permissions for the owner of the shared memory segment, and S\_IROTH and S\_IWOTH specify read and write permissions for others.
* SHM\_RND indicates that the address specified for the second parameter should be rounded down to a multiple of the page size. If you don’t specify this flag, you must page-align the second argument to shmat yourself.
* SHM\_RDONLY indicates that the segment will be only read, not written.
* Sem\_num is the semaphore number in the semaphore set on which the operationis performed.
* Sem\_op is an integer that specifies the semaphore operation. If sem\_op is a positive number, that number is added to the semaphore value immediately. If sem\_op is a negative number, the absolute value of that number is subtracted from the semaphore value. If this would make the semaphore value negative, the call blocks until the semaphore value becomes as large as the absolute value of sem\_op (because some other process increments it). If sem\_op is zero, the operation blocks until the semaphore value becomes zero.
* Sem\_flg is a flag value. Specify IPC\_NOWAIT to prevent the operation from blocking; if the operation would have blocked, the call to semop fails instead. If you specify SEM\_UNDO, Linux automatically undoes the operation on the semaphore when the process exits.
* **Links**: Each process includes a link to its parent process, links to its siblings (processes with the same parent), and links to all of its children.
* [**real\_parent**](#L578): Pointer to the real parent process or to the init process if the parent is dead.
* [**parent**](#L579): Pointer to the original parent. Usually equal to **real\_parent**.
* [**children**](#L584) : List of processes created by this process.
* [**sibling**](#L585): List of processes with the same parent process as this one.This relations are managed through the macros **SET\_LINKS REMOVE\_LINKS**
* **group\_leader**:Pointer to the leader process of the group.
* **Times and timers**: Includes process creation time and the amount of processor time so far consumed by the process. A process may also have associated one or more interval timers. A process defines an interval timer by means of a system call; as a result a signal is sent to the process when the timer expires. A timer may be single use or periodic.
* [**sleep\_avg**](#L541): Average time in state *sleeping* of the process.
* [**timestamp**](#L542):Time when the process entered for the last time into the run queue or of the last context switch it was involved in.
* [**last\_ran**](#L542): When the process lost th CPU for the last time.
* [**time\_slice**](#L547): Remaining *quantum* time for the process, measured in clock ticks.
* **File system**: Includes pointers to any files opened by this process, as well as pointers to the current and the root directories for this process.
* **Address space**: Defines the virtual address space assigned to this process.

Some of the other fields are:

* [**flags**](#L532): contains the detailed state of the process in the kernel. It indicates the [**key points**](#L714)of the life-cycle of a process. Each bit indicates a possible event and they are not mutually exclusive.
* [**ptrace**](#L533): set of [**flags**](#L52)used when a process is being traced.
* **lock \_depth :** It indicates how many times has been acquired the kernel lock. This lock protects some of the kernel data structures.
* [**exit\_state**](#L565);[**exit\_code,exit\_signal**](#L566): contains the exit value of a process, in the case that it finished calling the **exit** system call. If it is terminated by a signal, it will contain the number of the signal.