2. SCHEDULING:
Scheduling is the job of allocating CPU time to different tasks within an operating system.

SCHEDULING IN LINUX:
Like all UNIX systems, ~~ ~~ LINUX supports preemptive multitasking. In such a system, the process scheduler decides which process runs and when. Making these decision in a way that ~~ ~~ balances fairness and performance across many different workloads is one of the more complicated challenges in modern operating systems.

PROCESS SCHEDULING:
-> Linux has two ~~spe~~ separate process - scheduling algorithms. One is a time - sharing algorithm for fair, preemptive scheduling among multple processes.
-> The other is designed for real- time tasks, where absolute priorities are more important than fairness.
->
-> The scheduling algorithm used for routine time-sharing tasks received a major overhaul with the version 2.6 of the kernel.
-> This algorithm does not provide adequate support for SMP systems, does not scale well as the number of tasks on the system grows, and does not maintain fairness among interactive tasks, particularly on systems such as desktops and mobile devices.
-> The Linux scheduler is a preemptive, priority - based algorithm with two ~~sp~~ separate priority ranges: a real-time range from 0 to 99 and a nice value ranging from -20 to 19.
-> Smaller nice values indicate higher priorities. Thus by increasing the nice value, you are decreasing your priority and being "nice" to the rest of the system.

→ CFS consequently relies on a second configuration variable, the minimum granularity, which is a minimum length of time any process is allotted the processor.

REAL - TIME SCHEDULING!

Linux's real-time scheduling algorithm is significantly simpler than the fair scheduling employed for standard time-sharing processes. Linux implements the two real-time scheduling ; they are

① First - come, first - served (FCFS)
2) Round - robin. (RR)

- In both cases, each process has a priority in addition to its scheduling class. The scheduler always runs the process with the highest priority. Among processes of equal priority, it runs the process that has been waiting longest.

- The only difference betn FCFS and round-robin scheduling is that FCFS processes continue to run until ~~the~~ they either ~~exit~~ or block, whereas a round-robin process will be preempted. after a while and will be moved to the end of the of the scheduling queue

- Linux's real-time scheduling is soft - rather than hard - real time. The scheduler offers strict guarantees about the relative priorities of real-time processes, but the kernel does not offer any guarantees about how quickly a real-time process ~~es~~ will be scheduled once that process becomes runnable.

- In contrast, a hard real-time system can guarantee a minimum latency betn when a process becomes runnable and when it actually runs.

-> CFS is a significant departure from the traditional UNIX process scheduler.

-> In the latter, the core variables in the scheduling algorithm are priority and time slice.

TIME SLICE:

The time slice is the length of time - the slice of the processor of the processor - that a process is afforded.

-> The traditional UNIX systems give processes a fixed time slice,

-> ~~The~~ A process may for the length of its time slice, and higher priority processes run before lower-priority processes.

CFS introduced a new scheduling algorithm called fair scheduling that eliminates time slices in the traditional sense.

-> Insted of time slices, all processes are alloted a proportion of the processor's time.

-> ~~If there is N~~

-> CFS says that if there are N runnable processes, then each should be afforded 1/N of the processor's time.

-> CFS then adjusts the allotment by weighting each

-> Processes with the default nice value have a weigh of 1 - their priority is unchanged.

-> To calculate the actual length of time a process runs, CFS relies on a configurable variable called target latency; which is the interval of time during which every runnable task should run at least once.

→ CFS consequently relies on a second configuration variable, the minimum granularity, which is a minimum length of time any process is allotted the processor.

REAL-TIME SCHEDULING:

Linux's real-time scheduling algorithm is significantly simpler than the fair scheduling employ-ed for standard time-sharing processes. Linux implements the two real-time scheduling; they are

1. First-come, first-served (FCFS)
2. Round-robin. (RR)

- In both cases, each process has a priority in addition to its scheduling class. The scheduler always runs the process with the highest priority. Among processes of equal priority, it runs the process that has been waiting longest.
- The only difference betn FCFS and round-robin scheduling is that FCFS processes continue to run until ~~the~~ they either ~~exit~~ or block, whereas a round-robin process will be preempted. after a while and will be moved to the end of the of the scheduling queue
- Linux's real-time scheduling is soft- rather than hard-real time. The scheduler offers strict guarantees about the relative priorities of real-time processes, but the kernel does not offer any guarantees about how quickly a real-time process will be scheduled once that process becomes runnable.
- In contrast, a hard real-time system can guarantee a minimum latency betn when a process becomes runnable and when it actually runs.