Process priorities (Nice values)

- A nice value is the priority level that a process has. This value is used in the scheduler of the OS to be run in the time scheduled.
  This value has to be fair (each process gets a share of the CPU), responsiveness (Doesn't need to wait for long periods before it uses CPU). Traditionally this value goes from -20 to +19, 0 is the default value, -20 is the higher priority and +19 is the lowest. This value can be changed by the administrator into a code using a syscall.

Realtime process scheduling

- A real-time process scheduling must have these requirements:

  - Must provide a maximum response time according to its function.
  - A high priority process should be able to maintain exclusive access to the CPU until it completes or relinquishes the CPU.
  - Should be able to control the precise order in which component processes are scheduled.
- Some of the most efficient scheduling algorithms are: RoundRobin Scheduling, FIFO scheduling, Batch scheduling.
- We can change the priority with some syscalls inside the code if we need to.
- We need to be careful with some processes to prevent locking up the system, to give a high privilege to child processes.

Realtime process scheduling API

- For an API, scheduling is similar, but has a little changes, for example, the priority is from 1 to 99.
- Not all the schedulers are available

CPU affinity

- A process can be run in different CPU, this impacts directly on the efficiency and the performance. If a process changes of CPU it will also have to send the data involved, cache, and information of the processes that has involved.

In conclusion:

The default kernel scheduling algorithm employs a round-robin time-sharing policy. By default, all processes have equal access to the CPU under this policy, but we can set a process's nice value to a number in the range –20 (high priority) to +19 (low priority) to cause the scheduler to favor or disfavor that process. However, even if we give a process the lowest priority, it is not completely starved of the CPU.

Linux also implements the POSIX realtime scheduling extensions. These allow an application to precisely control the allocation of the CPU to processes. Processes operating under the two realtime scheduling policies, SCHED_RR (round-robin) and SCHED_FIFO (first-in, first-out), always have priority over processes operating under nonrealtime policies. Realtime processes have priorities in the range 1 (low) to 99 (high). As long as it is runnable, a higher-priority process completely excludes lower-priority processes from the CPU. A process operating under the SCHED_FIFO policy maintains exclusive access to the CPU until either it terminates, it voluntarily relinquishes the CPU, or it is preempted because a higher-priority process became runnable. Similar rules apply to the SCHED_RR policy, with the addition that if multiple processes are running at the same priority, then the CPU is shared among these processes in a round-robin fashion.

A process's CPU affinity mask can be used to restrict the process to running on a subset of the CPUs available on a multiprocessor system. This can improve the performance of certain types of applications.