**PART 1**

The nice command is used to run a program with modified scheduling priority / “nicenesses”. Nicenesses range at least from -20 (resulting in the most favorable scheduling) through 19 (the least favorable). The default behavior is to increase the niceness by 10. **10 = default value for adjustment.**

NAME

nice - run a program with modified scheduling priority

SYNOPSIS

nice [OPTION] [COMMAND [ARG]...]

DESCRIPTION

Run COMMAND with an adjusted niceness, which affects process schedul-

ing. With no COMMAND, print the current niceness. Nicenesses range

from -20 (most favorable scheduling) to 19 (least favorable).

-n, --adjustment=N

add integer N to the niceness (**default 10**)

**PART 2**

**Scheduling Algorithms:**

*Windows XP-*

XP uses a quantum-based, preemptive priority scheduling algorithm.

Instead of scheduling processes, XP schedules threads. This algorithm can also be considered a multiple feedback-queue algorithm, since the preemptive priority algorithm implements multiple queues. Higher-priority threads moving to the ready state, terminating threads, exhausting the time quantum, and threads performing a blocking system call can result in preemption. A thread with a priority of 0 is at the lowest priority, and 31 is at the highest priority. Variable class of priorities range from 1 to 15, while real-time class of priorities range from 16 to 31.

Threads in the real-time class have fixed priorities, and the thread that is currently running has the highest priority level. The idle thread is run if no ready thread exists. A thread’s priority is lowered when its time quantum runs out. When a thread’s state changes from waiting to ready, its priority is increased. Priority boosts are also given to processes that are currently interacting with the user.

*UNIX (Linux)-*

UNIX uses a preemptive priority algorithm consisting of two process scheduling algorithms: Time-sharing Algorithm and Real-time Algorithm. A process’s scheduling class defines which scheduling algorithm to apply. The process’ current priority is determined from the process’ nice value set by the user (ranging from -20 to 19, going from highest to lowest).

Time-sharing: Linux uses a prioritized credit based algorithm. The credit rule: Credit = credit/2 + priority factors in both process history and its priority. The credit system also auto-prioritizes interactive or I/O bound processes.

Real-time scheduling classes: Real-time processes have performance guarantees from the operating system, and have fixed priorities. Linux uses FIFO and robin round system. The scheduler runs the process with the highest priority. If the priority is equal, it runs the process waiting the longest. FIFO process continues to run either they exit or block.

*Mac OSX-*

Mac OS 9 used cooperative scheduling algorithm where one process controls multiple cooperative threads, and also provides preemptive scheduling for MP tasks. All process manager processes are scheduled cooperatively using a round robin algorithm. Each process on OS 9 has its own copy of the Thread Manager that schedules the threads belonging to a process cooperatively.

Mac OS X uses a multilevel feedback queue with four thread priority bands which are labeled: normal, system high priority, kernel mode only and real-time. The threads in OS X are scheduled preemptively but can also be cooperatively scheduled.

Mac threads represent the lowest level threading on the system. POSIX threads (pthreads) are layered on top of Mach thread. Cocoa threads (NSThreads) are layered directly on top of pthreads. Carbon MP tasks are layered on top of pthreads and Carbon Thread Manager cooperative threads are also layered on top of pthreads.