Unix SVR4 (Open Solaris and illumos distributions) CPU Scheduling

Ali Jameel Abdulrazzaq Al-Rawi

Submitted to:

Prof. Dr. Hasan Huseyin Balik

outline

- ✓ Definition the Unix SVR4
- ✓ Definition the OpenSolaris
- ✓ Definition the Illumos
- ✓ Scheduling review
- ✓ Unix SVR4 Scheduling
- ✓ SVR4 priority classes
- ✓ CPU Scheduling

UNIX SVR4

- The Unix operating system is a set of programs that act as a link between the computer and the user.
- The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the **operating system** or **the kernel**.
- Users communicate with the kernel through a program known as the **shell**. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.
- There are various Unix variants available in the market. Solaris Unix, AIX, HP Unix and BSD are a few examples. Linux is also a flavor of Unix which is freely available.

UNIX SVR4

- Unix was originally developed in 1969 by a group of AT&T employees (**Ken Thompson, Dennis Ritchie, and Douglas McIlroy**) and in 1983. Four major versions of System V were released, numbered 1, 2, 3, and 4. System V Release 4, or SVR4, was commercially the most successful version.
- Several people can use a Unix computer at the same time; hence Unix is called a **multiuser system**.
- A user can also run multiple programs at the same time; hence Unix is a **multitasking environment**.

OpenSolaris

OpenSolaris is an open source operating system ,which is developed and sponsored by Sun Microsystems Inc. and a community of contributors . It is based on an development version of an enterprise Solaris OS, which was open sourced by Sun in 2005. More precisely , the OpenSolaris now represents an operating system , a community and a source base and in 2010, Oracle decided to discontinue open development of the core software, and replaced the OpenSolaris distribution model with the proprietary Solaris Express

illumos

- **Illumos** is a free and open-source unix operating system it is derives from opensolaris which in turn derives from SVR4 unix and berkeley software distribution (BSD) illumos comprises a kernel , device drivers , system libraries and utility software for system administration .
- This core is now the base for many different open-sourced opensolaris distribution. Illumos is a consolidation of software that forms the core of an operating system.

illumos

- Finally OpenSolaris and its derivatives are the only SVR4 descendants that are open-source software. Core system software continues to be developed as illumos used in illumos distributions such as SmartOS, OpenIndiana and others.

Scheduling review

- Scheduling mechanism is the part of the process manager that handles the removal of the running process of CPU and the selection of another process on the basis of a particular strategy.
- Scheduler chooses one from the ready threads to use the CPU when it is available.
- Scheduling policy determines when it is time for a thread to be removed from the CPU and which ready thread should be allocated the CPU next .

UNIX SVR4 Scheduling

- The scheduling algorithm used in UNIX SVR4 is a complete overhaul of the scheduling algorithm used in earlier UNIX systems
- Goal: give preference in the following order
  - Highest preference to real-time processes.
  - Next-highest to kernel-mode processes.
  - Lowest preference to other user-mode processes (time-shared processes).
- Major enhancements
  - The addition of a preemptable static priority scheduler.
  - The introduction of (160) priority levels divided into three priority classes.
  - The introduction of preemption points in the kernel (points where the kernel can safely interrupt and schedule a new process).

UNIX SVR4 Scheduling (cont.)

- Implementation
  - A dispatch queue is associated with each priority level; processes at that level are executed round-robin (*dispq*)
  - A bit-map vector, *dqactmap*, contains one bit for each priority level; the bit is set to one for any priority level with a nonempty queue .
  - When a process leaves the Running state (block, time-slice expiration, or preemption), the dispatcher checks dqactmap and dispatches a ready process from the highest-priority nonempty queue.

Dispatch Queues

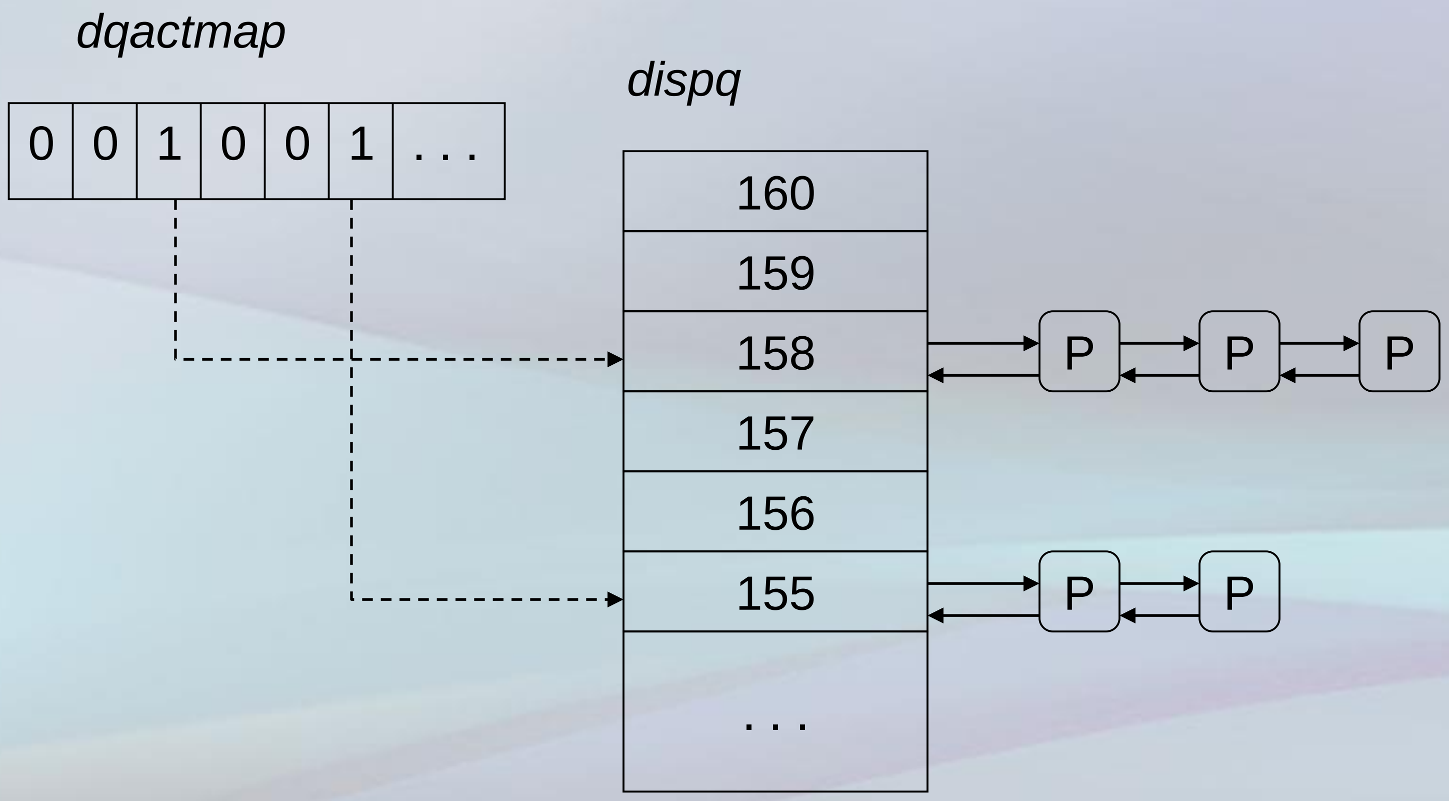


Figure (1) SVR4 dispatch queues

UNIX SVR4 Scheduling (cont.)

- When a defined preemption point is reached, the kernel
  - Checks the flag *kprunrun*
  - If set this indicates that (at least one real-time process is in the Ready state), preempts the current process if it is of lower priority than the highest-priority real-time ready process
- A real-time process has a fixed priority and a fixed quantum
- A time-sharing process has a variable priority and a variable time (100ms for priority 0, 10ms for 59)

UNIX SVR4 Scheduling (cont.)

- quantum
  - The priority is reduced every time the process uses up the time quantum and is raised when the process blocks on an event or resource
  - The time quantum changes with the priority

SVR4 Priority Classes

Priority classes and levels

- **Real-time (159-100)**
  - Guaranteed execution before kernel or time-sharing processes
  - Preemption points can be used to preempt kernel or user processes
  - Lowest-priority processes: user applications other than real-time
- **Kernel (99-60)**
  - Guaranteed execution before time-sharing processes
- **Time-shared (59-0)**
  - Lowest-priority processes: user applications other than real-time

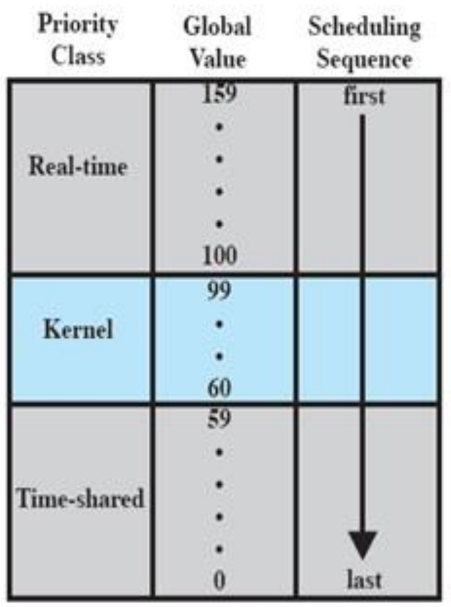
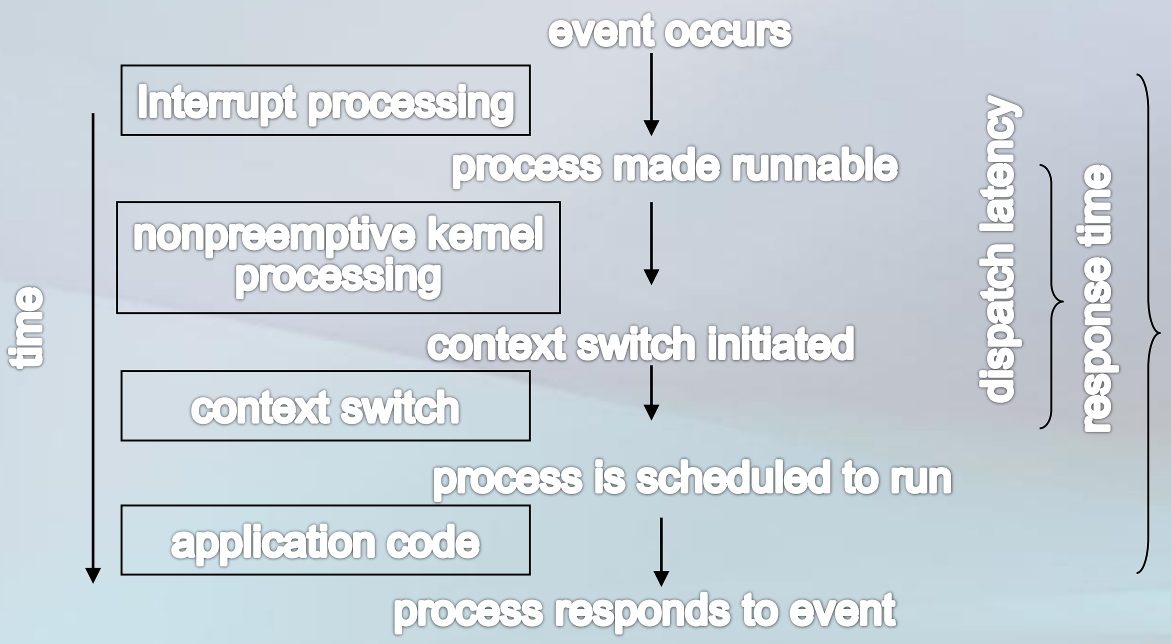


Figure (2) SVR4 Priority Classes

SVR4 Priority Classes

- **Real-time class**



- ❖ Soft real-time capabilities are needed for quality of service sensitive applications - video, audio, multimedia, virtual reality
- ❖ require bounded dispatch latency, and response time

- ❖ dispatch latency - time from the moment the process becomes runnable to the moment it begins to run
- ❖ response time = interrupt processing + dispatch latency + real-time process execution

SVR4 Priority Classes

- **Time-sharing class**
  - ➢ changes process priorities dynamically
  - ➢ round-robin scheduling with the same priority
  - ➢ event driven scheduling
    - ❖ reduces process priority each time it uses up its time slice
    - ❖ boosts the priority of the process if it blocks on an event or resource, or if it takes a long time to use up it quantum

CPU Scheduling

- The CPU is a resource that must be shared by all processes. The part of the kernel that apportions CPU time between processes is called the scheduler. The traditional UNIX scheduler uses preemptive round-robin scheduling. Processes of equal priority are scheduled in a round-robin manner, each running for a fixed quantum of time(typically 100 milliseconds).
- If a higher priority process becomes runnable, it will preempt the current process (unless the current process is running in kernel mode), even if the current process has not used up its time quantum. In traditional UNIX systems, the process priority is determined by two factors-the nice value and the usage factor.

CPU Scheduling

- When a process receives some CPU time, the kernel reduces its priority. This scheme prevents starvation of any process, since eventually the priority of any process that is waiting to run will rise high enough for it to be scheduled. A process executing in kernel mode may relinquish the CPU if it must block for a resource or event. When it becomes runnable again, it is assigned a kernel priority.
- In 4.3BSD, for instance, the kernel priorities range from 0 to 49, and user priorities from 50 to 127. While user priorities vary with CPU usage, kernel priorities are fixed, and depend on the reason for sleeping. Because of this, kernel priorities are also known as sleep priorities. Table 2-2 lists the sleep priorities in 4.3BSD UNIX.

The end of This presentation

Thank you For Watching It

☺