Great Place To Work. Certified FEB 2023 – FEB 2024 INDIA

UPES

UNIVERSITY OF TOMORROW

nirf RANKING 2023

#RANKED 52 IN INDIA

#University Category

QS WORLD UNIVERSITY RANKINGS

NO.1 PVT. UNIVERSITY IN ACADEMIC REPUTATION IN INDIA

ACCREDITED WITH GRADE A NAAC

ACCREDITED **GRADE 'A'** BY NAAC

E-LEAD

PERFECT SCORE OF **150/150** AS A TESTAMENT TO EXECEPTIONAL E-LEARNING METHODS

Unit 2 : Process and Thread Management

Lecture 8

Submitted by:

Dr Khushboo Jain

School of Computer Science

UPES, Dehradun

India

UPES
UNIVERSITY OF TOMORROW

Table of Contents

Case study: Process Management in Linux

1. Linux History
2. Design Principles
3. Process Management

UPES
UNIVERSITY OF TOMORROW

Learning & Course Outcomes

LO1: Explain the fundamental concepts of process management in Linux,
LO2: Demonstrate the ability to use Linux commands and system calls to manage processes, such as ps, top, kill, fork(), exec(), and wait().

CO2: Evaluate and analyze process and thread scheduling techniques, discerning their benefits and challenges.

UPES
UNIVERSITY OF TOMORROW

Linux History

- Linux is a modern, free operating system based on UNIX standards
- First developed as a small but self-contained kernel in 1991 by Linus Torvalds, with the major design goal of UNIX compatibility
- Its history has been one of collaboration by many users from all around the world, corresponding almost exclusively over the Internet
- It has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms
- The core Linux operating system kernel is entirely original, but it can run much existing free UNIX software, resulting in an entire UNIX- compatible operating system free from proprietary code
- Many, varying Linux Distributions including the kernel, applications, and management tools

UPES
UNIVERSITY OF TOMORROW

The Linux Kernel

- Version 0.01 (May 1991) had no networking, ran only on 80386- compatible Intel processors and on PC hardware, had extremely limited device-drive support, and supported only the Minix file system
- Linux 1.0 (March 1994) included these new features:
  - Support for UNIX's standard TCP/IP networking protocols
  - BSD-compatible socket interface for networking programming
  - Device-driver support for running IP over an Ethernet
  - Enhanced file system
  - Support for a range of SCSI controllers for high-performance disk access
  - Extra hardware support
- Version 1.2 (March 1995) was the final PC-only Linux kernel

UPES
UNIVERSITY OF TOMORROW

Linux 2.0

- Released in June 1996, 2.0 added two major new capabilities:
  - Support for multiple architectures, including a fully 64-bit native Alpha port
  - Support for multiprocessor architectures
- Other new features included:
  - Improved memory-management code
  - Improved TCP/IP performance
  - Support for internal kernel threads, for handling dependencies between loadable modules, and for automatic loading of modules on demand
  - Standardized configuration interface
- Available for Motorola 68000-series processors, Sun Sparc systems, and for PC and PowerMac systems
- 2.4 and 2.6 increased SMP support, added journaling file system, preemptive kernel, 64-bit memory support

Linux Systems

- Linux uses many tools developed as part of Berkeley's BSD operating system, MIT's X Window System, and the Free Software Foundation's GNU project
- The min system libraries were started by the GNU project, with improvements provided by the Linux community
- Linux networking-administration tools were derived from 4.3BSD code; recent BSD derivatives such as Free BSD have borrowed code from Linux in return
- The Linux system is maintained by a loose network of developers collaborating over the Internet, with a small number of public ftp sites acting as de facto standard repositories

Linux Distribution

- Standard, precompiled sets of packages, or distributions, include the basic Linux system, system installation and management utilities, and ready-to-install packages of common UNIX tools
- The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management
- Early distributions included SLS and Slackware
- Red Hat and Debian are popular distributions from commercial and noncommercial sources, respectively
- The RPM Package file format permits compatibility among the various Linux distributions

Linux Licensing

- The Linux kernel is distributed under the GNU General Public License (GPL), the terms of which are set out by the Free Software Foundation

- Anyone using Linux, or creating their own derivative of Linux, may not make the derived product proprietary; software released under the GPL may not be redistributed as a binary-only product

Design Principles

- Linux is a multiuser, multitasking system with a full set of UNIX- compatible tools
- Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
- Main design goals are speed, efficiency, and standardization
- Linux is designed to be compliant with the relevant POSIX documents; at least two Linux distributions have achieved official POSIX certification
- The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior

Process Management

- UNIX process management separates the creation of processes and the running of a new program into two distinct operations.
  - The fork system call creates a new process
  - A new program is run after a call to execve
- Under UNIX, a process encompasses all the information that the operating system must maintain to track the context of a single execution of a single program
- Under Linux, process properties fall into three groups: the process's identity, environment, and context

Process Identity

- Process ID (PID). The unique identifier for the process; used to specify processes to the operating system when an application makes a system call to signal, modify, or wait for another process
- Credentials. Each process must have an associated user ID and one or more group IDs that determine the process's rights to access system resources and files
- Personality. Not traditionally found on UNIX systems, but under Linux each process has an associated personality identifier that can slightly modify the semantics of certain system calls
  - Used primarily by emulation libraries to request that system calls be compatible with certain specific flavors of UNIX

Process Environment

- The process's environment is inherited from its parent, and is composed of two null-terminated vectors:
  - The argument vector lists the command-line arguments used to invoke the running program; conventionally starts with the name of the program itself
  - The environment vector is a list of "NAME=VALUE" pairs that associates named environment variables with arbitrary textual values
- Passing environment variables among processes and inheriting variables by a process's children are flexible means of passing information to components of the user-mode system software
- The environment-variable mechanism provides a customization of the operating system that can be set on a per-process basis, rather than being configured for the system as a whole

Process Context

- The (constantly changing) state of a running program at any point in time
- The scheduling context is the most important part of the process context; it is the information that the scheduler needs to suspend and restart the process
- The kernel maintains accounting information about the resources currently being consumed by each process, and the total resources consumed by the process in its lifetime so far
- The file table is an array of pointers to kernel file structures
  - When making file I/O system calls, processes refer to files by their index into this table
- Whereas the file table lists the existing open files, the file-system context applies to requests to open new files
  - The current root and default directories to be used for new file searches are stored here
- The signal-handler table defines the routine in the process's address space to be called when specific signals arrive
- The virtual-memory context of a process describes the full contents of the its private address space

Process and Threads

- Linux uses the same internal representation for processes and threads; a thread is simply a new process that happens to share the same address space as its parent
- A distinction is only made when a new thread is created by the clone
- system call
  - fork creates a new process with its own entirely new process context
  - clone creates a new process with its own identity, but that is allowed to share the data structures of its parent
- Using clone gives an application fine-grained control over exactly what is shared between two threads

Stages of a Process in Linux

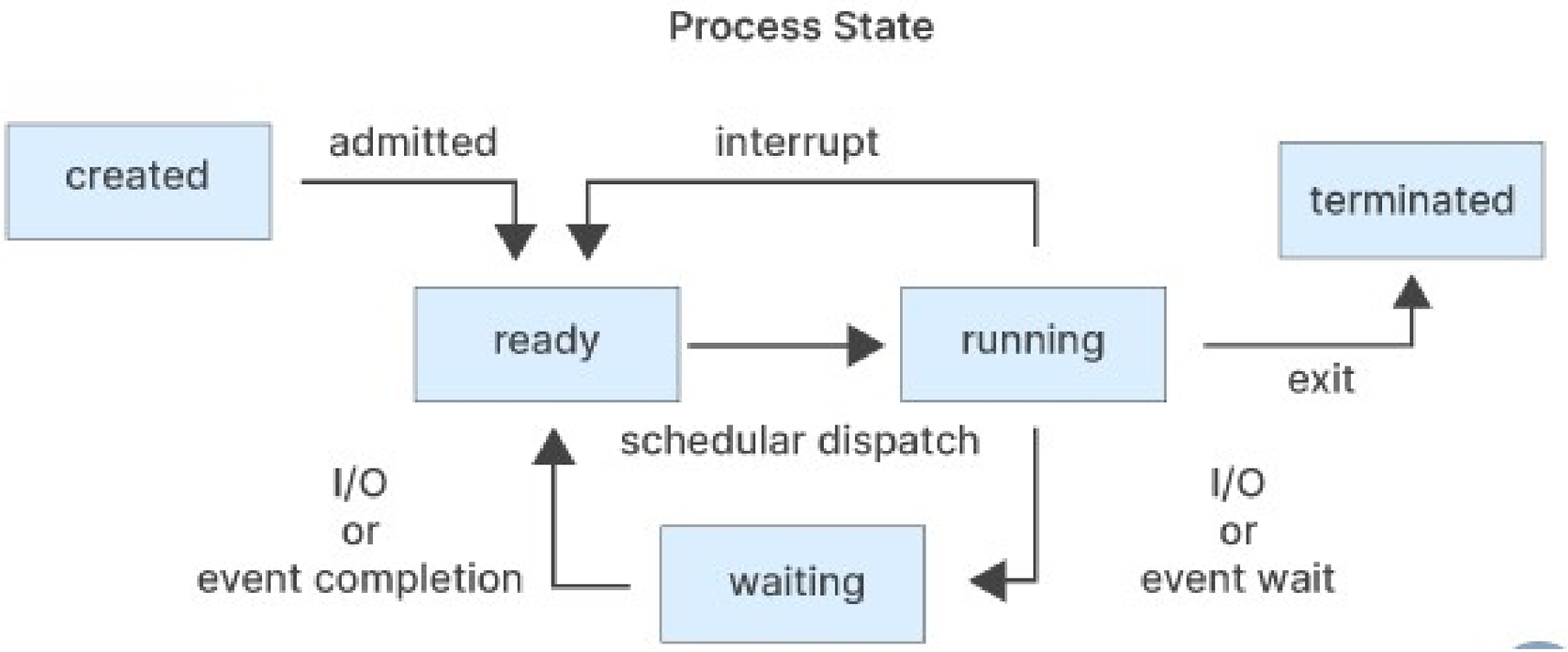

Fig 1. Stages of a Process in Linux [1]

Stages of a Process in Linux

1. **Created:** A process is created when a program is executed. At this stage, the process is in a "created" state, and its data structures are initialized.
2. **Ready:** The process enters the "ready" state when it is waiting to be assigned to a processor by the Linux scheduler. At this stage, the process is waiting for its turn to execute.
3. **Running:** The process enters the "running" state when it is assigned to a processor and is actively executing its instructions.
4. **Waiting:** The process enters the "waiting" state when it is waiting for some event to occur, such as input/output completion, a signal, or a timer. At this stage, the process is not actively executing its instructions.
5. **Terminated:** The process enters the "terminated" state when it has completed its execution or has been terminated by a signal. At this stage, the process data structures are removed, and its resources are freed.
6. **Zombie:** A process enters the "zombie" state when it has completed its execution but its parent process has not yet read its exit status. At this stage, the process details still have an entry in the process table, but it does not execute any instructions. The zombie process is removed from the process table when its parent process reads its exit status.

UPES
UNIVERSITY OF TOMORROW

Types of Process Management in Linux

- **Foreground Processes:**
  - Interactive processes requiring user input.
  - Executed in the foreground and associated with a terminal.
  - Managed using:
    - Shell commands (start, stop, pause, resume).
    - "ps" command for process information.
    - Signals (e.g., "Ctrl+C").
    - Job control commands (e.g., "bg", "fg", "kill").
- **Background Processes (Non-Interactive):**
  - Run in the background without user interaction.
  - Used for system services, daemons, and long-running tasks.
  - Managed using:
    - "ps" command for process information.
    - "top" command for real-time monitoring.
    - Methods for starting (e.g., "&" symbol, "nohup" command).
    - "kill" command for termination.

UPES
UNIVERSITY OF TOMORROW

Commands Used to Manage Processes in Linux

1. **ps:** This command is used to display information about running processes. The "ps" command can be used to list all processes or filter the list based on various criteria, such as the user who started the process, the process ID (PID), and the process status.
2. **top:** This command is used to display a real-time view of system processes. The "top" command provides information about the processes running on the system, including their resource usages, such as CPU and memory.
3. **kill:** This command is used to terminate a process. The "kill" command can be used with the process ID (PID) of the process or with a signal number to request a specific action.
4. **nice:** This command is used to adjust the priority of a process. Higher-priority processes get more CPU time than lower-priority processes. The "nice" command can be used to increase or decrease the priority of a process, which affects its CPU usage.
5. **renice:** This command is used to change or adjust the priority of a running process, which affects its CPU usage.

Commands Used to Manage Processes in Linux

6. **pkill:** This command is used to send a signal to a process to request it to terminate. The "pkill" command can be used with a current process name or a regular expression to match multiple processes.
7. **top:** This command is used to display a real-time view of system processes. The "top" command provides information about the processes running on the system, including their resource usages, such as CPU and memory.
8. **jobs:** This command is used to display a list of background jobs running in the current shell session.
9. **fg:** This command is used to move a background process to the foreground. The "fg" command can be used with the job ID of the background process.
10. **bg:** This command is used to move a suspended process to the background. The "bg" command can be used with the job ID of the suspended process.

UPES
UNIVERSITY OF TOMORROW

Summary

- **Introduction to Process Management:**
  - Linux manages processes, which are instances of running programs.
  - Processes can be categorized into foreground and background processes.
- **Foreground Processes:**
  - Interactive processes requiring user input.
  - Managed using shell commands, "ps" command, signals, and job control commands.
- **Background Processes (Non-Interactive):**
  - Run in the background without user interaction.
  - Used for system services, daemons, and long-running tasks.
  - Managed using "ps" command, "top" command, and methods for starting and terminating processes.

UPES
UNIVERSITY OF TOMORROW

Reference Material

[1]. Silberschatz, A. & Galvin, P. (2009) Operating System Concepts. 8th ed. NJ: John Wiley & Sons, Inc.

- Download Link- https://www.mbit.edu.in/wp-content/uploads/2020/05/Operating_System_Concepts_8th_EditionA4.pdf

[2]. NPTEL Video Lecture: https://onlinecourses.nptel.ac.in/noc24_cs80/preview

[3]. Red-hat Link: https://www.redhat.com/sysadmin/linux-command-basics-7-commands-process-management

UPES
UNIVERSITY OF TOMORROW

MCQ’s

1 Which of the following is NOT a type of process in Linux?

a) Foreground process
b) Background process
c) Daemon process
d) Kernel process

2. What are foreground processes in Linux?

a) Processes that run in the background without user interaction
b) Processes associated with a terminal and require user input
c) Processes that manage system services
d) Processes executed by the kernel

MCQ’s

3. Which command is commonly used to list all running processes in Linux?
    a) ls
    b) ps
    c) top
    d) cat

4. How are background processes typically started in Linux?
    a) Using the "bg" command
    b) By appending "&" at the end of a command
    c) Using the "start" command
    d) By redirecting output to a file

UPES
UNIVERSITY OF TOMORROW

MCQ's

5. Which command is used to terminate a process in Linux?
    a) end
    b) terminate
    c) kill
    d) Stop

6. What is job control in Linux?
    a) A method for controlling system resources
    b) A feature for managing foreground processes
    c) A tool for listing background processes
    d) A command for managing kernel processes

MCQ's

7. Which of the following tools provides a real-time view of system processes in Linux?
a) ls
b) ps
c) top
d) Grep

8. What is the purpose of signals in Linux process management?
a) To send messages between processes
b) To request a process to take a specific action
c) To stop the execution of a process
d) To limit the resources used by a process

MCQ’s

9. What is the role of the shell in Linux process management?
a) To provide a graphical user interface for process management
b) To execute commands and manage processes
c) To control hardware resources
d) To monitor system performance

MCQ’s Answers

Answers:

1. d) Kernel process
2. b) Processes associated with a terminal and require user input
3. b) ps
4. b) By appending "&" at the end of a command
5. c) kill
6. b) A feature for managing foreground processes
7. c) top
8. b) To request a process to take a specific action
9. b) To execute commands and manage processes

What's Next

Unit-3
Inter Process Communication (IPC)
IPC mechanisms
Shared Memory
Message Passing

UPES
UNIVERSITY OF TOMORROW

Thank You

UPES
UNIVERSITY OF TOMORROW