# Linux and Shell Programming

# Cyber Security Specialization Core-I

Dr Vimal Kumar (Course Instructor),

Assistant r. Professor-SCSET

Bennett University Greater Noida

# Outline

- Objectives of the Course
- Course Outcomes
- Course Structure & Credits
- Book Resources
- Assessment Components
- Industrial Certifications
- What you will loose for not attending the classes/Labs?
- Q & A

# Objective

Essential ideas behind the open-source operating system approach

Programming. knowledge of Linux and shell script

Understand the backbone of cybersecurity

Course involves

- *Basic Linux commands*
- *Shell scripting*
- *File structure and management*
- *Processes*
- *Inter-process communication*
- *Socket programming*
- *Security*

# COURSE OUTCOMES (CO)

1

**CO1:** To articulate Linux commands that are used to manipulate system operations at an admin level.

2

**CO2:** To write Shell Programming using Linux commands.

3

**CO3:** To design and write applications to manipulate internal kernel-level Linux File systems.

# Course Structure & Credits

| L | T | P | C |
|---|---|---|---|
| 2 | 0 | 4 | 4 |

- Theory Modules: 03
- Theory Lectures: 28 Hours
- Lab Sessions: 56 Hours (28+28 Self Lab)
- Continuous Lab Assessments: 20
- Marks per Lab: 02
- The Linux Kernel Project: 27/11/2024 (Tentative)
- Probable Industry Talks: 01

Resources will be available on LMS

# Text & Reference Books

## Textbooks

- M. Ebrahim and A Mallett, Mastering Linux Shell Scripting: A Practical Guide to Linux Command-Line, Bash Scripting, and She (2 ed.), Packt Publication, 2018. ISBN 978-1788990554.
- R. Blum and C. Bresnahan, Linux Command Line and Shell Scripting Bible (3 ed.), Wiley, 2016. ISBN 978-1118983843.

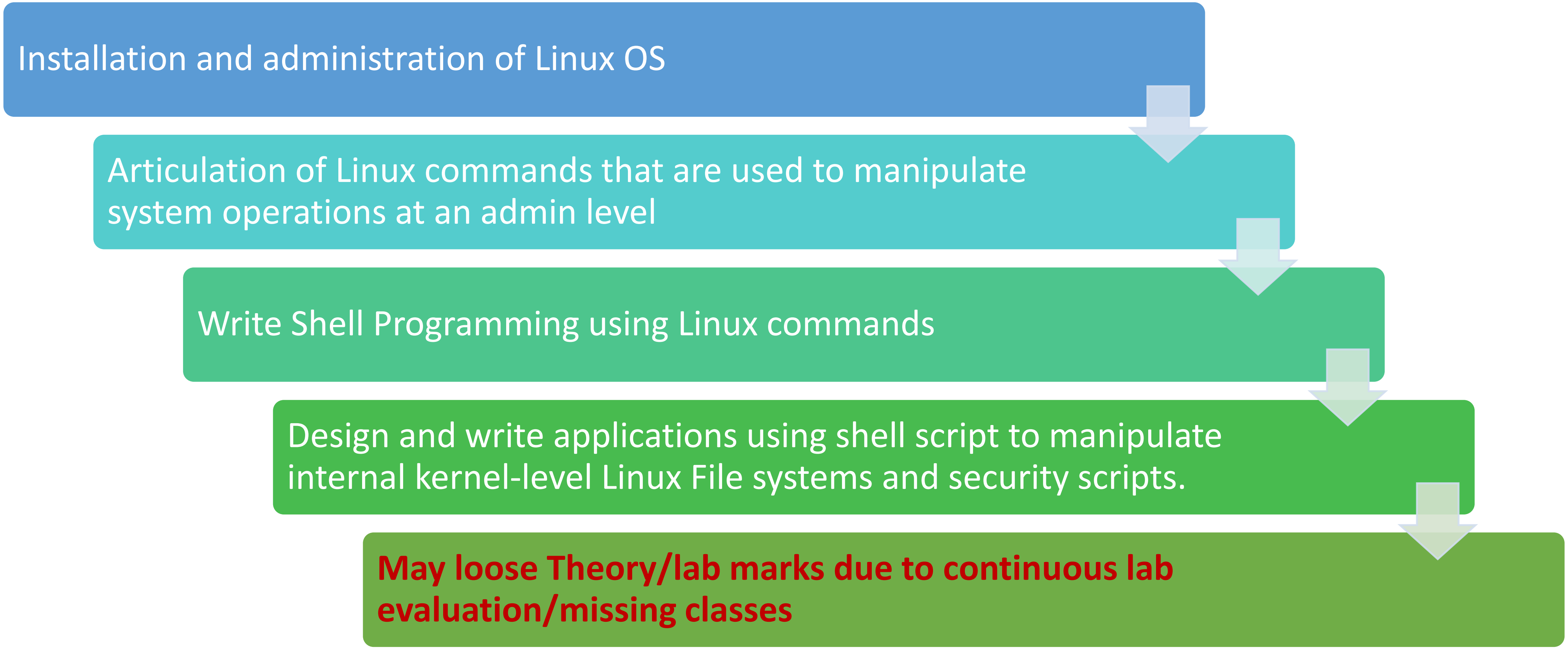
## Reference Books

- W.R. Stevens, UNIX Network Programming (3 ed.), PHI Publications, 2017. ISBN 978-8120307490.

# Assessment

| S. No. | Component | Marks | Date of Evaluation (Tentative) | |
|---|---|---|---|---|
| 1 | Mid Term | 10 | As per schedule notified by CoE | |
| 2 | End Term Examination | 40 | As per schedule notified by CoE | |
| 3 | MOOC Certification from the notified list | 10 | 27/11/2024 | |
| 4 | Lab Continuous Evaluation | 20 | As per timetable. 10 Best Labs will be considered, 2 marks of each Lab | |
| 5 | Linux Kernel Project/ Hackathon | 20 | Evaluation-1 (5 Marks) | Hackathon (15 Marks) |
| | | | 4th Week of August 24 | 29/11/2024 |
| Total | | 100 | | |

# What you will loose if not attended the classes/Labs?



Installation and administration of Linux OS

Articulation of Linux commands that are used to manipulate system operations at an admin level

Write Shell Programming using Linux commands

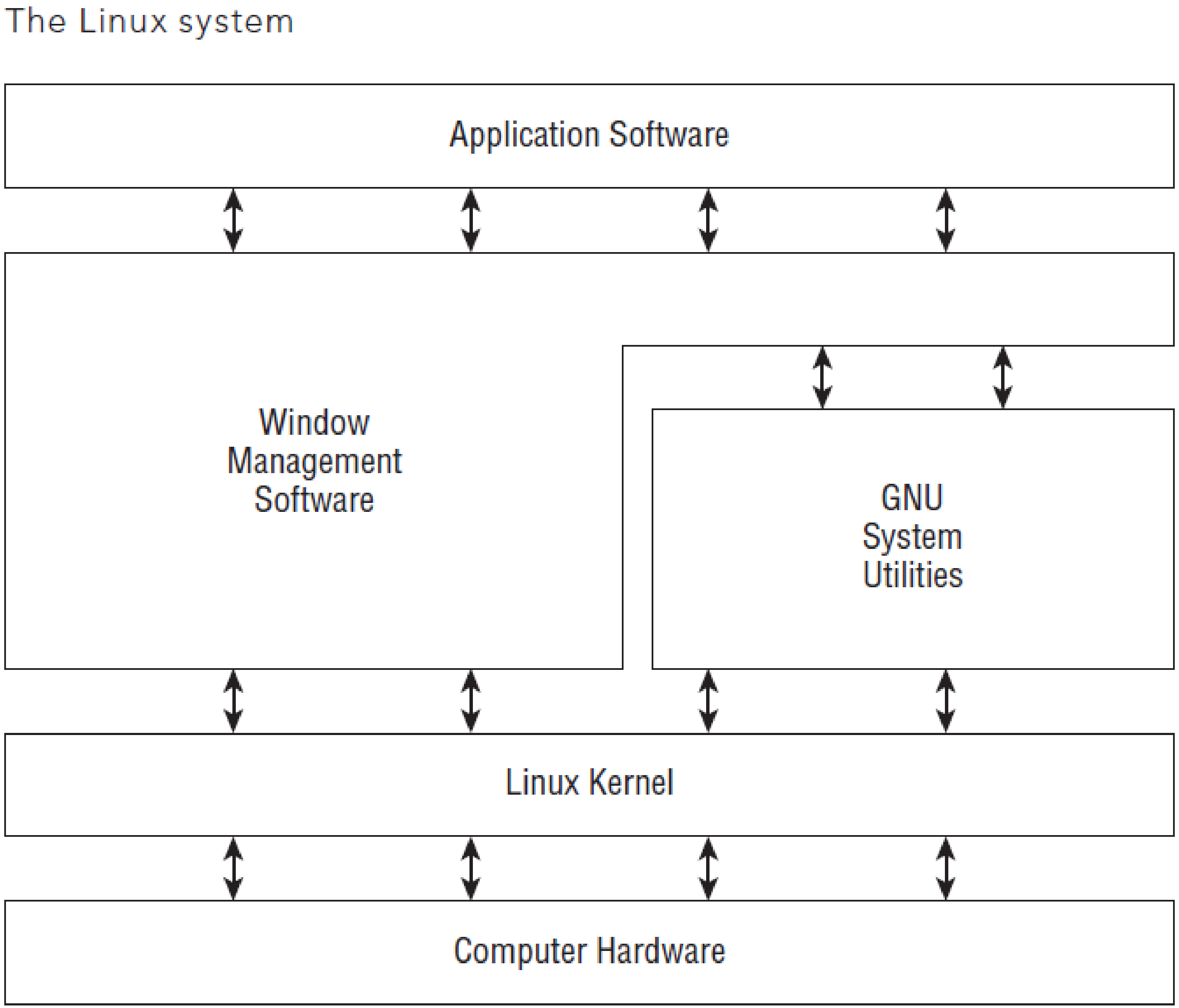
Design and write applications using shell script to manipulate internal kernel-level Linux File systems and security scripts.

**May loose Theory/lab marks due to continuous lab evaluation/missing classes**

# What is Linux?

- An Open source OS, written by ***Linus Torvalds***.

- Four parts of Linux
  - Kernel
  - Utilities
  - Graphical Desktop
  - Applications Software



The Linux system

Application Software

Window Management Software

GNU System Utilities

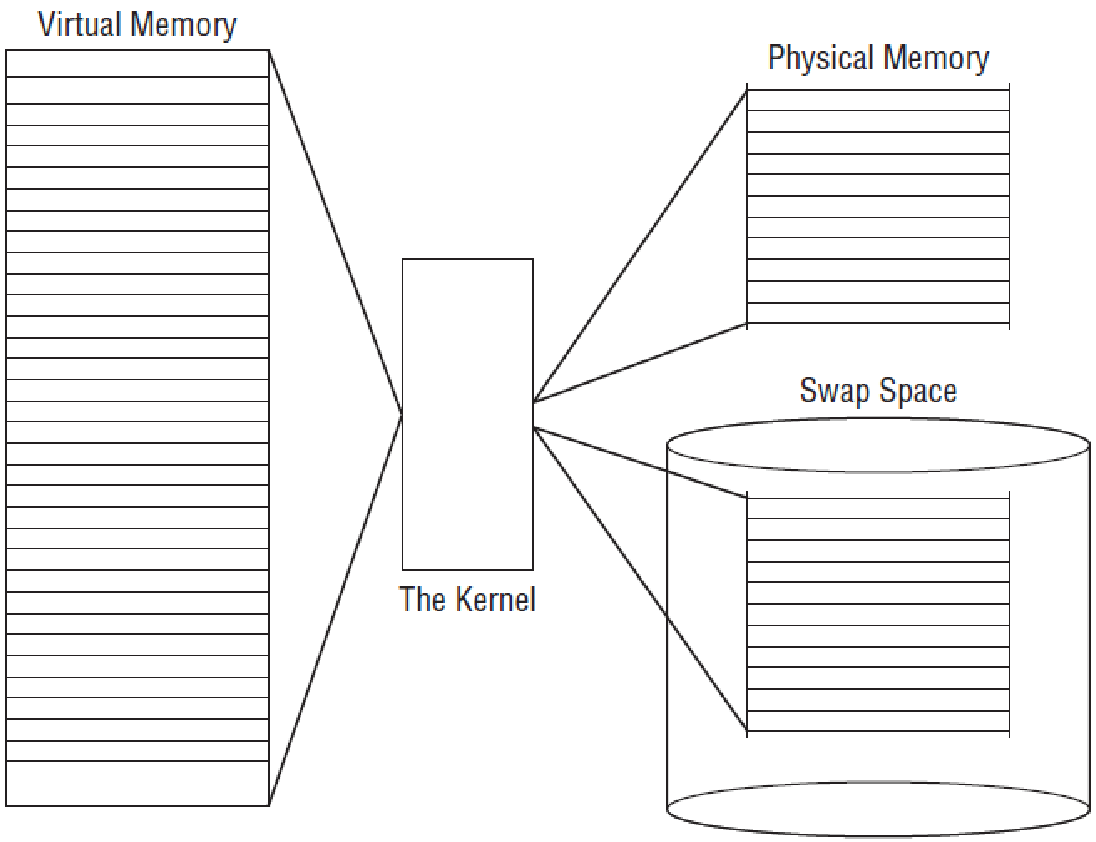
Linux Kernel

Computer Hardware

# System Memory Management

- Virtual Memory using swap space
- Memory locations grouped into blocks are called pages
- Kernel keeps track of used pages through page tables
- Swap-out
- Swap-in

The Linux system memory map



# Software Program Management

- Running program is a process, a foreground or a background process
- Kernel controls the processes
- init process---first process created by kernel
- When the kernel starts, it loads the init process into virtual memory
- Some Linux implementations contain a table of processes to start automatically on bootup.
- On Linux systems, this table is usually located in the special file /etc/inittabs.

# Software Program Management

- The popular Ubuntu Linux distribution utilize the /etc/init.d folder, which contains scripts for starting and stopping individual applications at boot time.
- The scripts are started via entries under the /etc/rcX.d folders, where X is a run level.
- The init utilizes run levels. There are 5 init run levels in the Linux.

- At run level 1&2, only the basic system processes are started, along with one console terminal process. This is called single-user (admin) mode.
- At run level 3&4, most application software, such as network support software, is started.
- At level 5, the GUI X-windows is started

# Hardware Management

- Any device that Linux must communicate with needs device driver code to be inserted into kernel
- Driver code allows the kernel to pass data back and forth to the device
- Two methods are used to insert device driver code into kernel
  - Drivers compiled into the kernel
  - **Drivers' modules are added to the kernel**
- The Linux system identifies hardware devices as special files, called device files.
- There are three classifications of device files:
  - **Character:** Character device files are for devices that can only handle data one character at a time. Examples: modems, and terminals
  - **Block:** Block files are for devices that can handle data in large blocks at a time, such as disk drives
  - **Network:** The network file types are used for devices that use packets to send and receive data. Examples are network cards and a special loopback device

# Hardware Management

Linux creates special files, called **nodes**, for each device on the system.

All communication with the device is performed through the device **node**.

Each **node** has a unique number pair that identifies it to the Linux kernel.

The number pair includes a major and a minor device number.

Similar devices are grouped into the same major device number.

The minor device number is used to identify a specific device within the major device group.

# File System Management

- The Linux kernel can support different types of filesystems to read and write data to and from hard drives.
- Table 1-1 lists the standard filesystems that a Linux system can use to read and write data.

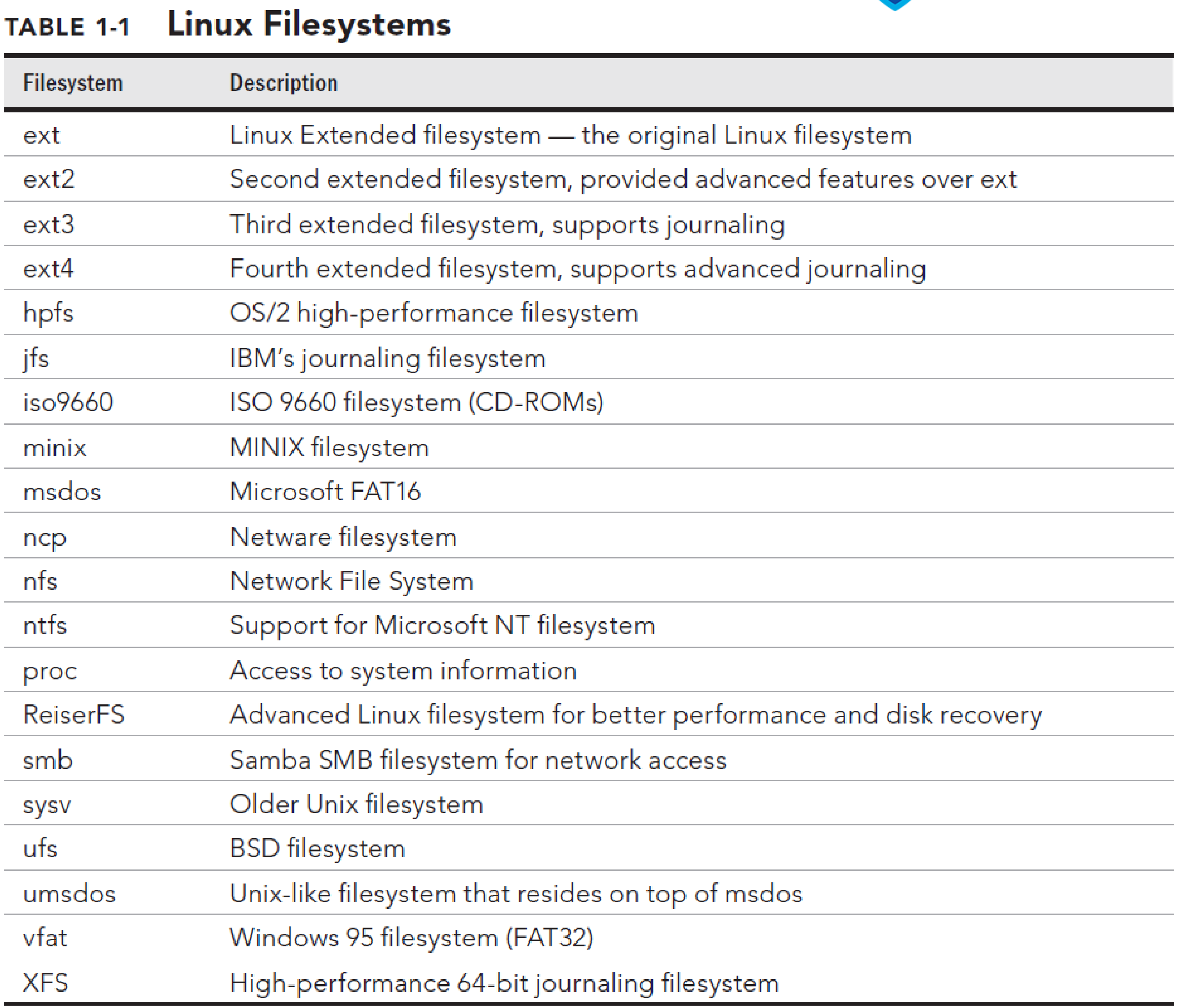


**TABLE 1-1** **Linux Filesystems**

| Filesystem | Description |
|---|---|
| ext | Linux Extended filesystem — the original Linux filesystem |
| ext2 | Second extended filesystem, provided advanced features over ext |
| ext3 | Third extended filesystem, supports journaling |
| ext4 | Fourth extended filesystem, supports advanced journaling |
| hpfs | OS/2 high-performance filesystem |
| jfs | IBM's journaling filesystem |
| iso9660 | ISO 9660 filesystem (CD-ROMs) |
| minix | MINIX filesystem |
| msdos | Microsoft FAT16 |
| ncp | Netware filesystem |
| nfs | Network File System |
| ntfs | Support for Microsoft NT filesystem |
| proc | Access to system information |
| ReiserFS | Advanced Linux filesystem for better performance and disk recovery |
| smb | Samba SMB filesystem for network access |
| sysv | Older Unix filesystem |
| ufs | BSD filesystem |
| umsdos | Unix-like filesystem that resides on top of msdos |
| vfat | Windows 95 filesystem (FAT32) |
| XFS | High-performance 64-bit journaling filesystem |

# The GNU Utilities

The core bundle of utilities supplied for Linux systems iscalled the ***coreutils*** package.

The GNU coreutils package consists of three parts:

- Utilities for handling files
- Utilities for manipulating text
- Utilities for managing processes

# The Shell Utility

Shell provides a way for users to start programs, manage fi les on the filesystem, and manage processes running on the Linux system.

The core of the shell is the command prompt.

The command prompt is the interactive part of the shell.

It allows you to enter text commands, and then it interprets the commands and executes them in the kernel.

You can also group shell commands into files to execute as a program. Those files are called *shell scripts*.

Table 2 shows the Linux Shells

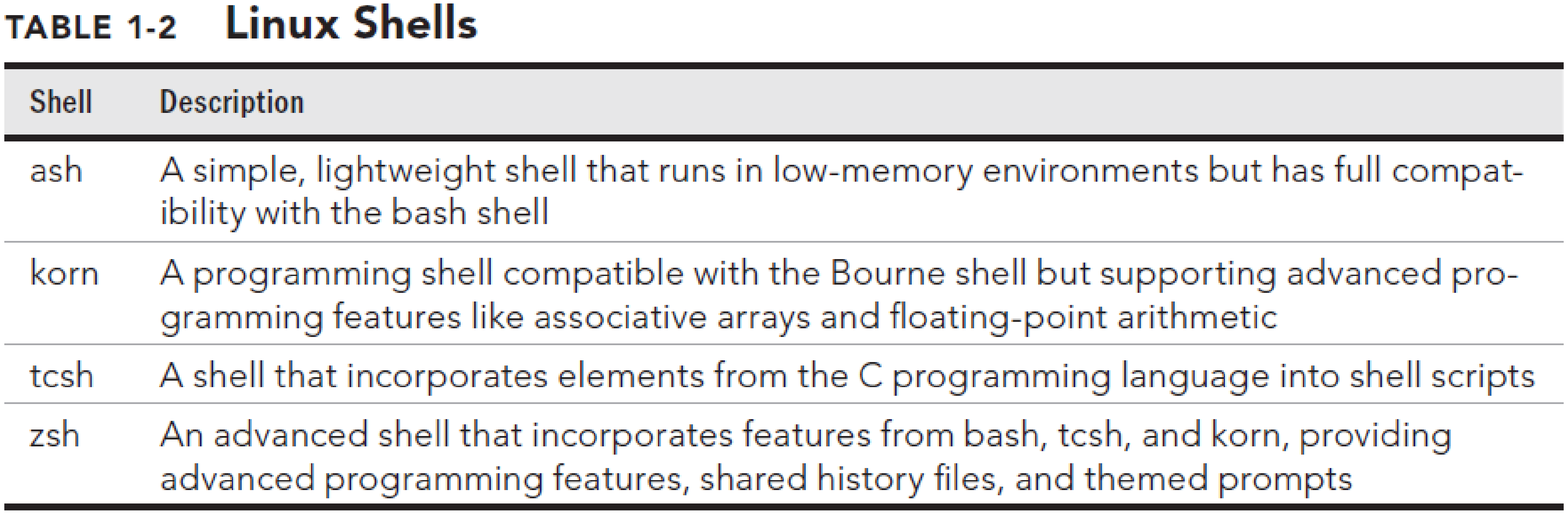
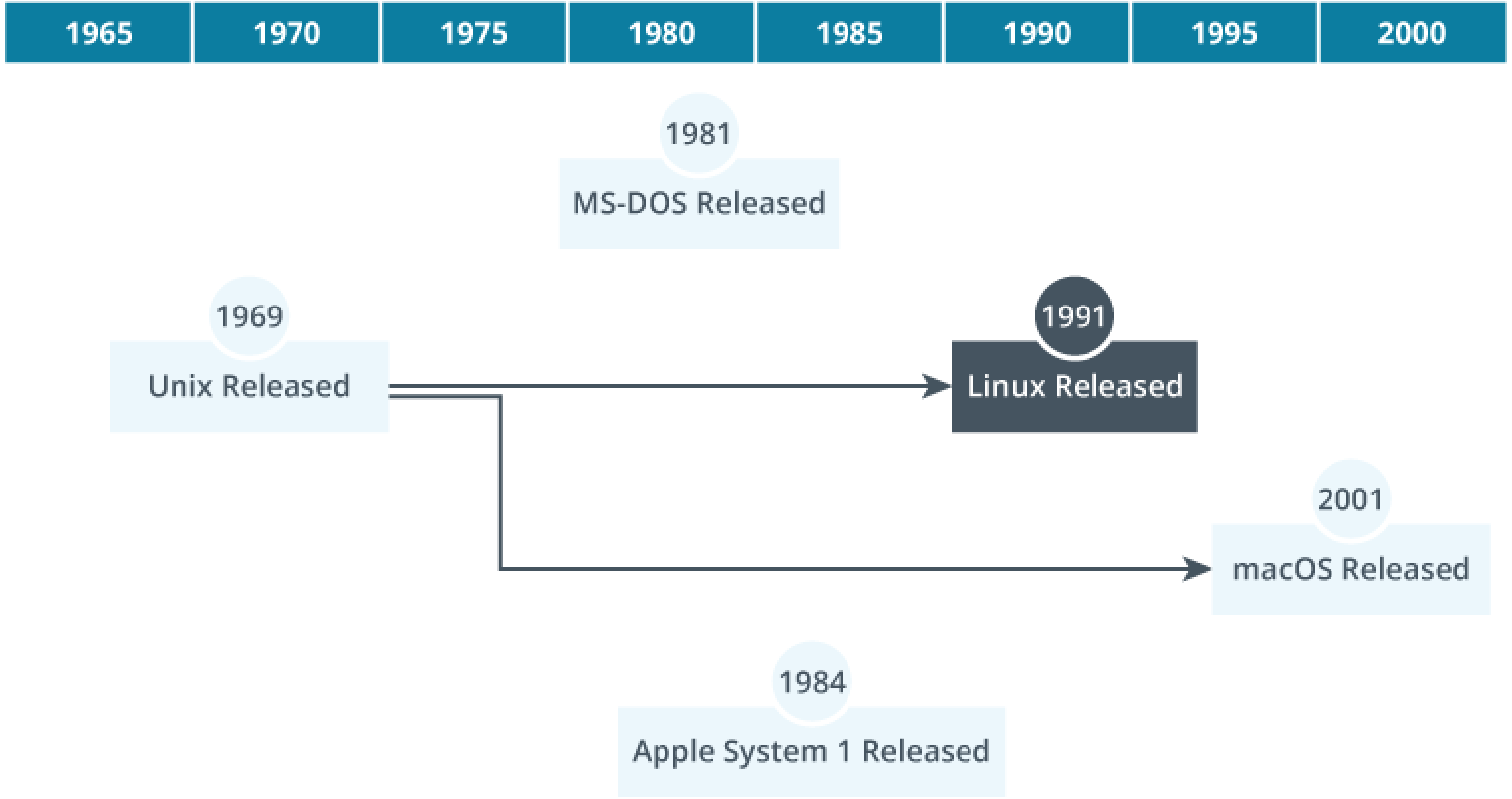


**TABLE 1-2 Linux Shells**

| Shell | Description |
|---|---|
| ash | A simple, lightweight shell that runs in low-memory environments but has full compatibility with the bash shell |
| korn | A programming shell compatible with the Bourne shell but supporting advanced programming features like associative arrays and floating-point arithmetic |
| tcsh | A shell that incorporates elements from the C programming language into shell scripts |
| zsh | An advanced shell that incorporates features from bash, tcsh, and korn, providing advanced programming features, shared history files, and themed prompts |

# Timeline



1965 1970 1975 1980 1985 1990 1995 2000

1981
MS-DOS Released

1969
Unix Released

1991
Linux Released

2001
macOS Released

1984
Apple System 1 Released

# Thanks

Q & A