Index

Introduction to OS

What is an Operating system?

Operating system is a mega computer program having millions of lines of code written in several languages. It is a system software, a software designed to provide platform to other software, that manages a computer's hardware and acts as an intermediary between the computer user and the computer hardware.

Types of operating systems:

Single- and multi-tasking

A single-tasking system can only run one program at a time, while a multi-tasking operating system allows more than one program to be running in concurrency. This is achieved by time-sharing, where the available processor time is divided between multiple processes. These processes are each interrupted repeatedly in time slices by a task-scheduling subsystem of the operating system. Multi-tasking may be characterized in preemptive and co-operative types. In preemptive multitasking, the operating system slices the CPU time and dedicates a slot to each of the programs. Unix-like operating systems, such as Solaris and Linux—as well as non-Unix-like, such as AmigaOS—support preemptive multitasking. Cooperative multitasking is achieved by relying on each process to provide time to the other processes in a defined manner. 16-bit versions of Microsoft Windows used cooperative multi-tasking. 32-bit versions of both Windows NT and Win9x, used preemptive multi-tasking.

Single- and multi-user

Single-user operating systems have no facilities to distinguish users, but may allow multiple programs to run in tandem. A multi-user operating system extends the basic concept of multi-tasking with facilities that identify processes and resources, such as disk space, belonging to multiple users, and the system permits multiple users to interact with the system at the same time. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources to multiple users.

Distributed

A distributed operating system manages a group of distinct computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they form a distributed system.

Templated

In an OS, distributed and cloud computing context, templating refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines. The technique is used both in virtualization and cloud computing management, and is common in large server warehouses.[8]

Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

Real-time

A real-time operating system is an operating system that guarantees to process events or data by a specific moment in time. A real-time operating system may be single- or multi-tasking, but when multitasking, it uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

Library

A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of libraries and composed with the application and configuration code to construct a unikernel: a specialized, single address space, machine image that can be deployed to cloud or embedded environments.

Introduction to Linux

What is Linux?

People those who are not very aware of technical terminologies often find Linux to be something really glorious coming from the Neptune. But really Linux is nothing but an operating system like Windows. The kernel of which was first released by Linus Torvalds in 1991 who also invented the Git. Linux has an open source license which means you can download the source code, modify it and submit changes to it.

Why to use Linux?

The main reason one would choose Linux is its efficiency in terms of managing memory. Linux is really fast. I have been using Ubuntu – a Linux distribution – for two years. It is pretty evident from my usage experience that Linux is faster than Windows. Linux offers multi-tasking, multi-threading, multi-processing, security, portability and reliability features.

What is a Linux Terminal?

Terminal is an application program in Linux that turns up when a user presses Ctrl+T and that in turn loads another program inside it called Shell which is a command line interpreter program that takes command from keyboard and passes it to the OS.

There are different shells in the market such as:

Bourne shell – Bourne shell The most basic shell available on all UNIX systems
Korn Shell – Based on the Bourne shell with enhancements
C Shell – similar to the C programming language in syntax
Bash Shell – Bourne Again Shell combines the advantages of the Korn Shell and the C Shell. The default on most Linux distributions.
tcsh – similar to the C Shell.

Linux commands:

# prompt – denotes administrative level
$ prompt – denotes user level

In Linux:

‘/’= Directory operator
‘\’ = Escape operator

In DOS:

‘/’ = Command argument

Create:

useradd <username> - add user
passwd <username> - creation of user password
userdel <username> - Delete the username

Switching:

su <username> - switch to another user

su – switch to root account
su- – switch to root and log in with root’s environment
cat> <filename> - create new file
cat<filename> – show the content of the filename
cat -n <filename> - show the content with the line numbers

file <filename> - to know what type of file it is
head <filename> - display first 10 lines of a text filename
tail <filename> - display last 10 lines of a text filename

cp <sourcefile> <destinationfile> - Move or rename file
mv <sourcefile> <destinationfile> - move or rename file
rm <filename> - delete file
mkdir<dirname> – create directory
pwd – show present working directory
cd – change directory
cd ~ – shortcut for home directory
cd.. – back to the previous directory
vi <filename> - create file using vi text editor
ls -l – display with long list

ls	-al – display long list with hidden files start with '.'
ls	-ld – long list of directory (only directory detail)
find	.-name*<filename extension> - search from current directory and sub directories for the filename extension
ps	– show currently executing processor
w	(or who) – display info about log in user
id	– display user and groupid
man	<any command> - displays information about the command

Introduction to Vi editor

Vi can potentially run on any computer with a monitor or vdu screen. On Linux systems, vi is a mode of the even more powerful editor, vim. Vim is context sensitive. If it recognises the type of file it is editing it will highlight parts of the text in colour to show you features of the content. For example, if the file is a C program, it can highlight reserved words, strings, blocks and many more.

In general, the description here of vi also applies to vim in vi mode. The editor vi can be used in terminal emulation windows on UNIX or Linux workstations, or on terminal emulation windows (like telnet or PuTTY or SSH Client) connected remotely to UNIX or Linux from other systems such as Windows PCs.

It is a terminal-screen based editor, i.e. part of file that's being edited is displayed in the terminal emulation window. You edit the file by editing the characters in the terminal. When you are happy with the edits, you write them away to the file. Up to that point, the file on disk is untouched.

Running Vi

Vi may be used to create or edit a file. Like all programs on UNIX and Linux you can start vi from the Shell by giving its name. Give the name of the file to be edited as an argument to the command. For example. to edit myfile.c use the following command.

```
vi myfile.c
```

If myfile.c exists it is read by vi. If myfile.c doesn't exist it is created. The terminal screen acts as a window on the file and as many lines as can fit into the terminal emulation will be displayed. To begin with the first few lines of the file are displayed. The cursor is positioned at the top left corner of the terminal.

Introduction to Shell scripting

A Shell script can be defined as - "a series of command(s) stored in a plain text file". A shell script is similar to a batch file in MS-DOS, but it is much more powerful compared to a batch file.

Shell scripts are a fundamental part of the UNIX and Linux programming environment.

Each shell script consists of

Shell keywords such as if..else, do..while.

Shell commands such as pwd, test, echo, continue, type.

Linux binary commands such as w, who, free etc..

Text processing utilities such as grep, awk, cut.

Functions - add frequent actions together via functions. For example, /etc/init.d/functions file contains functions to be used by most or all system shell scripts in the /etc/init.d directory.

Control flow statments such as if..then..else or shell loops to perform repeated actions.

Each script has purpose

Specific purpose - For example, backup file system and database to NAS server.

Act like a command - Each shell script is executed like any other command under Linux.

Script code usability - Shell scripts can be extended from existing scripts. Also, you can use functions files to package frequently used tasks.