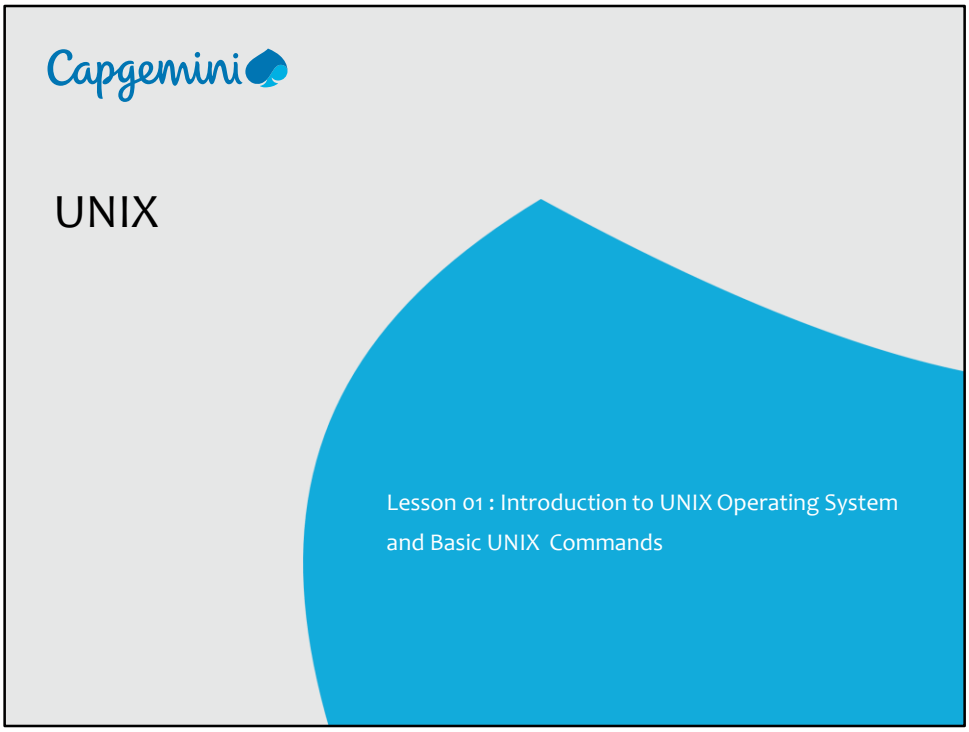
Instructor Notes:



Instructor Notes:

Lesson Objectives

In this lesson, you will learn:

- Operating System
  - Functions of Operating System
  - History of UNIX
  - Features of UNIX
  - UNIX System Architecture
- Basic UNIX Commands

Instructor Notes:

1.1: Operating System

Overview

An Operating System (OS) is the software that manages the sharing of the resources of a computer and provides programmers with an interface that is used to access those resources.

Operating System:
Overview:

An **Operating System (OS)** is the software that manages the sharing of the resources of a computer and provides programmers with an interface used to access those resources.

An Operating System processes **system data** and **user input**, and responds by allocating and managing tasks and internal system resources as a service to users and programs of the system. At the foundation of all system software, an Operating System performs basic tasks such as controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating networking and managing file systems. Most Operating Systems come with an application that provides a user interface for managing the operating system, such as a command line interpreter or graphical user interface. The operating system forms a platform for other system software and for application software.

The most commonly used contemporary desktop and laptop (notebook) OS is **Microsoft Windows**. More powerful servers often employ Linux, FreeBSD, and other Unix-like systems. However, these Unix-like operating systems, especially Mac OS X, are also used on personal computers.

Instructor Notes:

1.1: Operating System

Functions of an Operating System

Following are some of the important functions of an OS:

- Process Management
- Main-Memory Management
- Secondary-Storage Management
- I/O System Management
- File Management
- Protection System
- Networking
- Command-Interpreter System

Functions of an OS:

Following are the various services provided by the Operating system:

1. **Process Management:** It involves creation and deletion of user and system processes, deadlock handling, and so on.
2. **Main-Memory Management:** It involves keeping track of the parts of memory that are being used, allocating/deallocating memory space as required, etc.
3. **Secondary-Storage Management:** It involves free-space management, disk scheduling, storage allocation.
4. **I/O System Management:** It deals with hardware specific drivers for devices, keeps it all hidden from the rest of the system.
5. **File Management:** It involves creating/deleting files and directories, backup, and so on.
6. **Protection System:** It involves controlling access to programs, processes, or users
7. **Networking:** It generalizes the network access.
8. **Command-Interpreter System:** It provides an interface between the user and the OS.

Instructor Notes:

1.2: History of UNIX

History

UNIX evolved at AT&T Bell Labs in the late sixties.

The writers of Unix are Ken Thomson, Rudd Canaday, Doug McIilroy, Joe Ossanna, and Dennis Ritchie.

It was originally written as OS for PDP-7 and later for PDP-11.

Liberal licensing: Various versions.

System V in 1983 - Unification of all variants.

History of UNIX:

Unix had a modest beginning at AT&T Bell Laboratories in late sixties, when AT&T withdrew its team from the MULTICS project. The immediate motivation towards development of Unix was a Space Travel game that Thompson had developed for PDP-7. Finding the game slow on the PDP machine, he requested for a DEC-10 machine. The request was rejected. This led to the idea of designing a new operating system for PDP-7.

As a result, a multitasking system supporting two users was developed. It had an elegant file system, a command interpreter, and a set of utilities. Initially called UNICS (as a pun on MULTICS), by 1970, it came to be known as Unix. Ken Thompson, along with Rudd Canaday, Doug McIlroy, joe Ossanna, and Dennis Ritchie, were the writers of this operating system. Ritchie helped the system to be moved to PDP-11 system in 1970. Ritchie and Kerningham also wrote a compiler for "C".

Unix was originally written in Assembly language. In 1973, Ritchie and Thompson rewrote the Unix kernel in "C". This was a revolutionary step, as earlier the operating systems were written in assembly languages. The idea of writing it in "C" was so that the Operating system could now be ported (the speed, in effect, was traded off). It also made the system easier to maintain and adapt to particular requirements.

Around 1974, the system was licensed to universities for educational purposes, and was later available for commercial use. The licensing by AT&T was very liberal – the source code was made available to universities, industries, and government organizations. This led to development of various versions of Unix as everybody added their own utilities. The important ones amongst them include BSD (Berkeley Software Distribution from University of California, Berkeley), SunOS (from Sun Microsystems).

Instructor Notes:

1.3: Features of Unix

Features

UNIX OS exhibits the following features:

- It is a simple User Interface.
- It is Multi-User and Multiprocessing System.
- It is a Time Sharing Operating System.
- It is written in "C" (HLL).
- It has a consistent file format - the Byte Stream.
- It is a hierarchical file system.
- It supports Languages such as FORTRAN, BASIC, PASCAL, Ada, COBOL, LISP, PROLOG, C, C++, and so on.

Features of UNIX:

UNIX is a **timesharing operating system** written in "C". It is a **multi-user** as well as a **multiprocessing system**. Many users can work with multiple tasks at the same time. **Round Robin Scheduling** with fixed time slices is the algorithm that is used by the CPU for scheduling tasks.

UNIX, as it is written in "C", is portable. It is available on a variety of platforms ranging from a 8088 based PC to a parallel processing system like Cray 2.

Everything in UNIX is a **file** – even if it is a directory or a device. The file format is consistent. A file is nothing but a **stream of bytes** – it is not considered as containing fixed length records. UNIX follows a hierarchical file system. All files are related with root at the top of the hierarchy – it follows an inverted tree structure.

The kernel is a set of "C" program that controls the resources of a computer and allocates them amongst its users. It lets users run their programs, controls peripheral devices, and provides a file system to manage programs and data. It is loaded into memory when system is booted. It is often called as "the" operating system.

The approach in UNIX is that there is no need to have separate utilities to solve each and every complex problem. Instead, by combining several commands (each of which is capable of doing a simple job), it is possible to solve bigger problems. Pipes and filters allow building solutions from simple commands – they also make UNIX powerful and flexible.

Instructor Notes:

Services

Services Provided by UNIX:

- Process Management:
  - It involves Creation, Termination, Suspension, and Communication between processes.
- File Management:
  - It involves aspects related to files like creation and deletion, file security, and so on.

iGae Sensitive

Services provided by UNIX:

Amongst the various services that UNIX operating system provides are Process Management (Creation, Termination, Suspension and communication between processes) and File Management (creation and deletion of files, allowing for dynamic growth of files, security of files etc). Files and processes are two major entities in Unix. A file resides in memory – it is static in nature – as against a process, which is alive and exists in time.

UNIX has a very simple user interface: programmers have written it and it is meant for programmers, hence there are absolutely no "frills". However, graphical interfaces are becoming available in latter releases.

UNIX includes editors and compilers (for various languages like C, C++, Pascal, Fortran, Prolog, Lisp, Java, and so on). It has several utilities like calculators, graphics and statistical packages, and tools for document preparation.

UNIX includes an online help facility, which is very useful to look at the syntax involving several different options for the many commands of UNIX.

File Management:

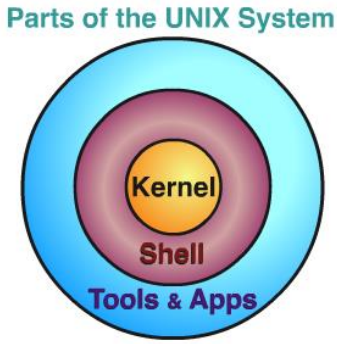
It helps to create or delete files. It assigns read, write, and execute permissions for users, groups, and others to restrict access to file.

Instructor Notes:

1.4: UNIX System Architecture

UNIX System Architecture

Following is a pictorial representation of the UNIX system:



UNIX System Architecture:
The UNIX system is functionally organized at three levels:

- The **kernel**, which schedules tasks and manages storage
- The **shell**, which connects and interprets users' commands, calls programs from memory, and executes them
- The **tools** and **applications** that offer additional functionality to the operating system

There is a policy of division of labour that is followed between the kernel and the shell on the Unix systems. The **kernel** interacts with the machine hardware, and the **shell** interacts with the User.

The kernel:
The heart of the operating system, the kernel controls the hardware and turns part of the system on and off at the programmer's command. Suppose you ask the computer to list all the files in a directory. Then the kernel tells the computer to read all the files in that directory from the disk and display them on your screen.

The shell:
The **shell** is technically a **UNIX** command that interprets the user's requests to execute commands and programs. The **shell** acts as the main interface for the system, communicating with the **kernel**. The shell is also programmable in UNIX. There are many different types of shells like **Bourne Shell**, **Korn Shell**, and **C Shell,** most notably the command driven **Bourne Shell** and the **C Shell**, and menu-driven shells that make it easier for beginners to use. Whatever shell is used, its purpose remains the same -- to act as an interpreter between the user and the computer.
We will see more about shell later.

Tools and applications:
There are hundreds of tools available to UNIX users, although some have been written by third party vendors for specific applications. Typically, tools are grouped into categories for certain functions, such as word processing, business applications, or programming.

Instructor Notes:

1.5: Basic Unix Command

Logging In and Out Commands

Logging In and Out:

- Logon name and password are required.
- Successful logon places user in home directory.

Basic Unix Commands:

In order to work with Unix, one needs a login name and password, which the system administrator needs to assign. With these in hand, one can "log on" to a Unix System and start a "session" with Unix by typing out the login name and password at the prompt. Once the session is through, one needs to "log off" from the system.

Logging In and Out:

Once the terminal is connected to a Unix system, it displays the "login:" message on screen. The login name, given by the system administrator needs to be typed here. If the password has been set, the system will prompt for the password (printing will be turned off when password is typed).

In case of valid **user name** and **password**, the user will get logged on to the system. The system displays a prompt, typically a **$ sign** (it can be different as it is possible to change prompts) – which indicates that the system is ready to accept commands. There can be some messages and other notifications before the prompt.

On successful logon, user is automatically placed in home directory, the name of home directory being usually the same as the login name.
If there is an error in validating the user name and password, then the system requests the user to re-enter the same. Usually due to security considerations, system allows only a fixed number of attempts to re-enter information.

Instructor Notes:

man Command

man command:

- The on line help provided by the **man** command includes brief description, options, and examples.
- Example:

```
$man <command>
```

man Command:

Online help using the man command:

Using the **man** command, it is possible to get a brief description about a command including all its options as well as some suitable examples.

Example: To get help on **passwd** command, use the following syntax:

```
$ man passwd
```

Instructor Notes:

cal Command

cal command:

- The **cal** command is used to display calendar from the year 1 to 9999.
- Example:

```
$cal 9 2001
```

- The above syntax can be used to print the calendar for the 9th month of the year 2001.

The calendar – cal command:

Any calendar from the year 1 to 9999 (either for a month or complete year), can be displayed using this command.

An example is shown in the above slide.

```
September 2001
      Su Mo Tu We Th Fr Sa
                        1
       2  3  4  5  6  7  8
       9 10 11 12 13 14 15
      16 17 18 19 20 21 22
      23 24 25 26 27 28 29
      30
```

Example: Using the cal command, observe the calendar for September 2001!

Instructor Notes:

date Command

date command:

- The **date** command is used to see current date and time.
- Date can be displayed in different formats
- Example:

```
$ date
```

- **Output:** Fri Apr 6 11:14:46 IST 2001

```
$ date "+%T"          -- %t is used to display only time
```

- **Output:** 11:15:20

```
$ date "+ %d %h"      -- To display date and month name
```

- **Output:** 6 Apr

Displaying System Date – date:
Unix has an internal clock, which actually stores the number of seconds elapsed from 1 Jan. 1970; and it is used for time stamping. A number of options are separately available to retrieve various components of the date.

Table: Commands used with date command

Option	Description
m	Month
h	Month Name
d	Day of month
y	Year (Last 2 digits)
H	Hour
M	Minutes
S	Second
T	Time in hh:mm:ss
A	Day of week (Sun to Sat)
R	Time in AM/PM

Instructor Notes:

lp Command

lp command:

- The **lp** command is used for printing files.
  - Example:

```
$lp myfile.txt
```

```
$lp –n 10 myfile.txt
```

```
$lpq
```

```
$lprm -Pps99 11042
```

lp command – for printing files:

Jobs can be queued up for printing by using the spooling facility of Unix. Several users can print their files without any conflict. This command can be used to print the file, it returns the job number that can later be used to check the status of job.

The command is used as follows:

lp –n 10 mytextfile.txt

-n option sets number of copies to print from 1 to 100.

lpq

Prints jobs in queue

```
$lp file1.txt
```

lprm -Pps99 11042

the lprm command lets you cancel the printing of a file. To remove a print job, first use the lpq command to find the job number of the print request you want to cancel. Then use the lprm command to kill the job by specifying the job number.

Instructor Notes:

nl Command

nl command:

- The **nl** command is used to print file contents along with line numbers.
- Options:
  - -w : width of the number
  - -v : Indicate first line number
  - -i : increment line number by
- Example:

```
$ nl myfile.txt
1  line one
2  line two
```

Line Numbering – nl :
This command elaborates schemes for numbering lines.
Options:

- -w : width of the number
- -v : Indicate first line number
- -i : increment line number by

Example: Using the nl command:

```
$nl -w2 -v40 -i7 file1.txt
40      one
47      two
54      three
61      four
```

Instructor Notes:

tty Command

tty Command:

- Unix treats a terminal also as a file. In order to display the device name of a terminal, the **tty** (teletype) command is used.
- print the file name of the terminal connected to standard input
- Example: Using tty command

```
$ tty
 /dev/ttyp3
```

tty Commands:

Unix treats terminal also as a file. In order to display the device name of a terminal, the command used is tty (**t**ele**ty**pe).

The above slide shows an example on using the **tty** command.

In order to display the current terminal related settings, the **stty** command can be used.

The output of the stty command depends on the Unix implementation.

This command can also be used to change some settings. For example, in order to use **<Ctrl-a>** instead of **<Ctrl-d>** as end of file indicator.

Instructor Notes:

who Command

who Command:

- To list all users who are currently logged in
- Example:

```
$who
```

- **Output:**

```
ssdesh   ttyp0    Mar 29 09:00
root     tty01    Mar 29 10:32
root     tty03    Mar 29 10:37
```

$who am I Command:

- To see the current user

Login details – who command:

Unix maintains an account of all current users of system, the list of which can be printed using this command. The command can also be used to get one's own login details.

Example 1: Using the who command

```
$ who
```

```
ssdesh      ttyp0    Mar 29 09:00
root        tty01    Mar 29 10:32
root        tty03    Mar 29 10:37
root        tty11    Mar 29 09:52
pmaint      tty12    Mar 29 10:00
deshpavn    ttyp1    Mar 29 10:38
munageka    ttyp2    Mar 28 16:55
deshpavn    ttyp3    Mar 29 10:49
```

Example 2:

```
deshpavn   ttyp3      Mar 29 10:49
```

```
$ who am i
```

Instructor Notes:

Summary

In this lesson, you have learnt:

- UNIX is multi-user, multiprocessor, time sharing operating system.
- It uses hierarchical file system.
- The UNIX system is functionally organized at three levels: Kernel, shell, tools and applications.

Instructor Notes:

Review Questions

Question 1: ___ controls system hardware.

Question 2: The kernel interacts with the machine hardware, and the shell interacts with the User.

- True / False

Question 3: ___ command displays details of all users currently logged in.