

Unit-1 Syllabus: Introduction to UNIX

- What is UNIX
- UNIX Components
- Using UNIX
- Commands in UNIX
- Some Basic Commands
- Command Substitution
- Giving Multiple Commands.
- Brief History

Text Books:

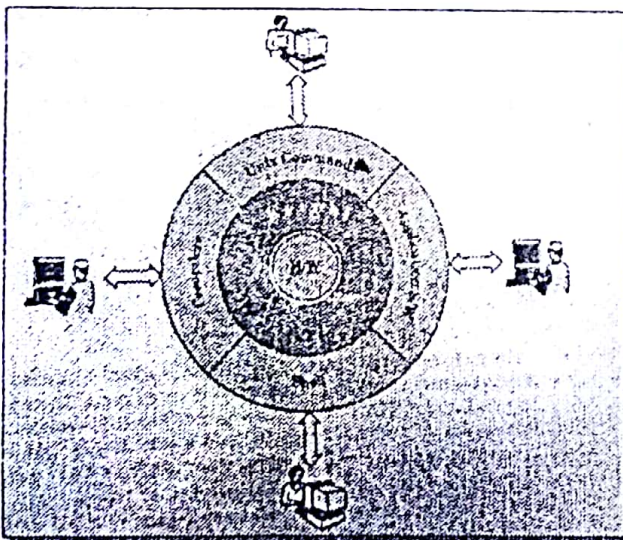
1. The UNIX programming Environment by Brian W. Kernighan & Rob Pike, Pearson.
2. Introduction to UNIX Shell Programming by M.G.Venkateshmurthy, Pearson.

1. WHAT IS UNIX?

- UNIX is
 - An Operating System
 - A Multi-tasking Operating System
 - A Multi-user Operating System
 - Highly Portable
 - Provides Security at various levels
 - Treats Everything including memory and I/O devices
 - Uses Text-based Terminal

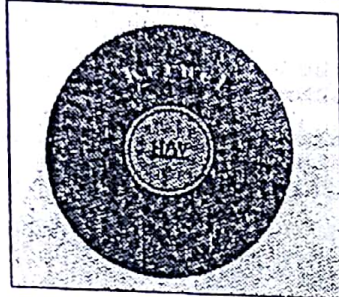
2. UNIX COMPONENTS:

- It has 3 major Components
 - i) The Kernel
 - ii) The Shell
 - iii) The File System



i) The Kernel:

- ✓ - Kernel is the Heart of any Unix Operating System.
- ✓ - Every UNIX has just one Kernel.
- ✓ - It is a small piece of code that is embedded on the Computer Hardware.
- ✓ - It is a collection of programs that are mostly written in C.
- It is the only component that can communicate directly with the Hardware.



- ✓ - There are 2 types of Kernels
 - Monolithic Kernels
 - Micro kernels

ii) The Shell:

- ✓ - A Shell is a Command Interpreter or a Processor.
- ✓ - It acts as an agent or interface between the Users and the Kernel (Hardware).
- ✓ - Every UNIX has at least one shell.
- ✓ - It is similar to command.com in the MS-DOS environment.
- Shell has certain programming capability of its own, called Shell Programs or Shell Scripts.

Types of Shells:

- The Shells are 4 types
 - a) The Bourne Shell (sh)
 - b) The C Shell (csh)
 - c) The Korn Shell (ksh)
 - d) The Bourne-Again Shell (bash)

a) The Bourne Shell (sh):

- ② - This is the most common shell available on Unix Systems. ✓
- ① - The first major shell developed by Stephen Bourne at AT&T Bell Labs. ✓
- ③ - This shell is widely used. ✓
- This shell is distributed as the standard shell on almost all Unix Systems.

b) The C Shell (csh):

- ④ - It is developed by Bill Joy at UCB as part of the BSD release.
- ✓ - Its syntax and usage is very similar to the C programming language.
- ✓ - This shell is not available on all machines.
- Shell scripts written in the C shell are not compatible with the Bourne Shell.
- A version of it called tcsh is available free of cost under Linux.

c) The Korn Shell (ksh):

- ✓ - This shell was developed by David Korn at AT&T Bell labs.
- ✓ - Basically It is built on the Bourne Shell.
- ✓ - It also incorporates certain features of the C shell.
- ✓ - At present it is one of the widely used shells.
- ✓ - It can run Bourne shell scripts without any modification.
- One of its versions, the Public Domain Korn Shell (pdksh), comes with Linux free of cost.

d) The Bourne-Again Shell:

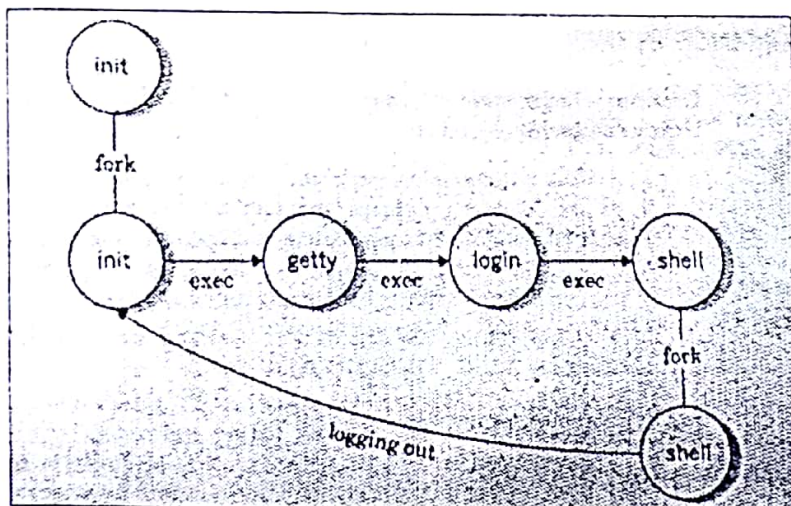
- ✓ - It was developed by B Fox and C Ramey at Free Software Foundation.
- ✓ - Certain Linux operating system variants come with this shell as its default shell.
- ✓ - This is clearly a free ware shell.

iii) The File System:

- ✓ - Another major component of UNIX system.
- ✓ - UNIX treats everything including hardware devices – as a file.
- Files in UNIX are organized in an inverted tree-like hierarchical structure.
- All files are stored in a File System as in the form of a tree-like hierarchical structure.

3. USING UNIX:

- ✓ - To use UNIX, one has to get into the UNIX environment.
- ✓ - The process of getting into the UNIX environment is known as Logging into the System.
- ✓ - The Login process is depicted as,



The Sequence of events in a Complete Login process is given as,

- ✓ - As soon as the system is booted, a daemon called **init** gets started along with some other daemons.
- Daemons are background processes that get started at the boot time and continue to run as long as the system is up.
- This **init** daemon spawns a process called **getty** for every terminal.
- The user enters a login name at the **getty**'s login prompt on the terminal.
- **getty** executes the login program with the login name as the argument.

- login requests for a password and validates it against /etc/passwd.
- A file called passwd file under /etc directory contains a line for every user containing the user's login name, numerical user id, encrypted password, home directory, and other such information.
- login sets up the TERM environment variable and runs a shell.
- The shell executes the appropriate startup files like .profile.
- The shell then prints a prompt, usually a \$ or a % symbol and waits for further input.
- This indicates successful entry made into a UNIX environment with a proper shell.
- When the user completes the session with the system, he comes out of the UNIX environment.
- The process of coming out of the UNIX environment is known as Logging out.
- As soon as the user logs out, the control return to init daemon, which in turn spawns a new getty on the corresponding terminal.
- This facilitates a new user to login to the system.

The shell prompt:

- Successful login into a UNIX system is indicated by the appearance of a prompt called the shell prompt or system prompt on the terminal.
- The character that appears as a prompt depends on the shell used.
- The list of default prompts given in a table as,

Prompt	Shell
\$ (dollar)	Bourne and Korn Shells (sh, bash, ksh)
% (percent)	C Shells (csh, tcsh)
# (hash)	Any shell as root

4. COMMANDS IN UNIX:

- UNIX has a Large number of commands.
- A UNIX command is a program written to perform certain specific action.
- All UNIX commands are written using Lower-case letters.
- Almost all UNIX commands are Cryptic.
- UNIX commands can have zero, one or more number of arguments associated with them.
- UNIX commands can also have Format Specifiers (indicated by +) as well as Options (indicated by -) associated with them.
- A current UNIX command can be killed by using either <delete> or <ctrl+u> command.
- There are two types of UNIX Commands available.
 - i) External Commands
 - ii) Internal Commands

i) External Commands:

- A Command with an independent existence in the form of a separate file is called an External Command.
- For example, programs for commands such as cat and ls, exist independently in a directory called the /bin directory.
- When such commands are given, the shell reaches these command files with the help of a system variable called the PATH variable and executes them.
- Most of the UNIX commands are External Commands.

ii) Internal Commands:

- A Command that does not have an independent existence is called an Internal Command.
- These commands also called as Built-in Commands.
- The routine for internal commands will be a part of another program or routine.
- For example, the echo command is an internal command as its routine will be a part of the shell's routine, sh.
- cd and mkdir are also examples of internal commands.

5. SOME BASIC COMMANDS:

- ~~a)~~ echo
- b) tput
- c) tty
- ~~d)~~ who
- e) uname
- f) date
- g) cal
- h) calendar
- ~~i)~~ passwd
- j) lock
- k) banner
- ~~l)~~ cat
- ~~m)~~ bc
- ~~n)~~ spell and ispell

a) The echo command:

- This command is used to display messages.
- It takes zero, one or more arguments.
- Arguments may be given either as a series of individual symbols or as a string within a pair of double quotes (" ").

Examples:

i) \$echo

#A Blank line is displayed

\$

ii) \$echo Hello World

Hello World

\$

iii) \$echo Hello

World

Hello World

\$

- The echo command

iv) \$echo "Hello World"

Hello World

\$

v) \$echo "Hello World"

Hello World

\$

b) The tput command:

- It is used to control the movement of the cursor on the screen.
- Along with clear argument clears screen.

Example:

1) \$tput clear

- Along with cup argument and certain coordinate values is used to position the cursor.

Example:

\$tput cup 10 20

- To find the number of rows and columns on the current terminal we use lines and cols arguments along with tput command.

Example:

\$tput lines

45

\$tput cols

120

\$

c) The tty command:

- In UNIX, every terminal is associated with a device file.
- All the device files will be present in the /dev directory.
- A user can know the name of his device file on which he is working by using the tty command.

Examples:

- In UNIX, the tty command gives the result as,

\$tty

/dev/tty01

\$

- In UNIX, the tty command gives the result as,

\$tty

/dev/pts/0

\$

d) The who command:

- The user can know login details of all the current users by using the who command.
- It has three column format.

Example:

\$who

```

root console july 2 10:30
aaa tty01 july 2 10:40
bbb tty02 july 2 10:55

```

\$

- Some options like -H, -u can be used with this command

-H option provides headers for columns.

-u option provides more details like idle time, PID and comments.

Example:

\$who -Hu

NAME	LINE	TIME	IDLE	PID	COMMENTS
root	console	july 2 10:30	0:05	22	
aaa	tty01	july 2 10:40	0:10	23	
bbb	tty02	july 2 10:55	0:15	24	

\$

The who am I command:

- The self-login details of a user can be obtained as a single line output using am and i arguments along with the who command.

Example:

\$who am i

```

aaa tty01 july 2 10:40

```

\$

e) The uname command:

- It gives the name of the UNIX system being used by the user.
- It uses certain options like r,v,m,a.

Examples:

i) \$uname

Linux

\$

ii) \$uname -r

2.4.18 - 3

#release details ✓

\$

iii) \$uname -m

i686

#machine details ✓

\$

f) The date command:

- Using this command, the user can display the current date along with the time nearest to the second

Example:

\$date

Sat Jan 10 11:58:00 IST 2004

\$

- It allows the use of format specifiers

- Each format specifier is preceded by a + symbol followed by the % operator

- +m is used to display the month in the numeric form

Example:

\$date +%m

09

\$

- +h is used to display the name of the month

Example:

\$date +%h

Sep

\$

g) The cal command:

This command is used to print the calendar of a specific month or a specific year

Example:

i) \$cal

MARCH 2018						
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	31
\$						

ii) \$cal 09 1949

September 1949						
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	
\$						

iii) \$cal 2002

2002																				
Jan							Feb							Mar						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4														
6	7	8	9	10	11	12	3	4	5	6	7	8	9	3	4	5	6	7	8	9
13	14	15	16	17	18	19	10	11	12	13	14	15	16	10	11	12	13	14	15	16
20	21	22	23	24	25	26	17	18	19	20	21	22	23	17	18	19	20	21	22	23
27	28	29	30	31			24	25	26	27	28			24	25	26	27	28	29	30
April							May							Jun						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4					1	2	3	30						
7	8	9	10	11	12	13	5	6	7	8	9	10	11	2	3	4	5	6	7	8
14	15	16	17	18	19	20	12	13	14	15	16	17	18	9	10	11	12	13	14	15
21	22	23	24	25	26	27	19	20	21	22	23	24	25	16	17	18	19	20	21	22
28	29	30					26	27	28	29	30	31		23	24	25	26	27	28	29
July							Aug							Sep						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4														
7	8	9	10	11	12	13	4	5	6	7	8	9	10	1	2	3	4	5	6	7
14	15	16	17	18	19	20	11	12	13	14	15	16	17	8	9	10	11	12	13	14
21	22	23	24	25	26	27	18	19	20	21	22	23	24	15	16	17	18	19	20	21
28	29	30	31				25	26	27	28	29	30	31	22	23	24	25	26	27	28
Oct							Nov							Dec						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4														
6	7	8	9	10	11	12	3	4	5	6	7	8	9	1	2	3	4	5	6	7
13	14	15	16	17	18	19	10	11	12	13	14	15	16	8	9	10	11	12	13	14
20	21	22	23	24	25	26	17	18	19	20	21	22	23	15	16	17	18	19	20	21
27	28	29	30	31			24	25	26	27	28	29	30	22	23	24	25	26	27	28

h) The calendar command:

- It is like an engagement diary that contains text information and offers a remainder service based on a file.
- The file must be in the present working directory/ home directory.
- This file is created and managed by the user with the help of an editor (like vi) on the system.
- This command works on today and tomorrow dates concept

Example:

```
$calendar
```

```
Jun 11, 2018 college opening day
```

```
$
```

i) The passwd command

- UNIX is a multi-user OS in which there is always a security threat.
- The security of an individual user can be given using password.
- During the addition of new users, the system administrator permits or authorizes the new user by assigning a unique password to him or her.
- A user can change his or her password using the passwd command.

Example:

```
$passwd
```

```
old password: *****
```

```
new password: *****
```

```
new password: *****
```

```
$
```

j) The lock command:

- It is used for locking a session for any required amount of time.

Example:

```
password:*****
```

```
re-enter password:*****
```

```
terminal locked by 501 0 min ago
```

- One can lock a terminal for a maximum period of 60 minutes.
- A numeric option may be used to lock the terminal for any period ranging between 1 and 60 minutes

Example:

```
$lock -45
```

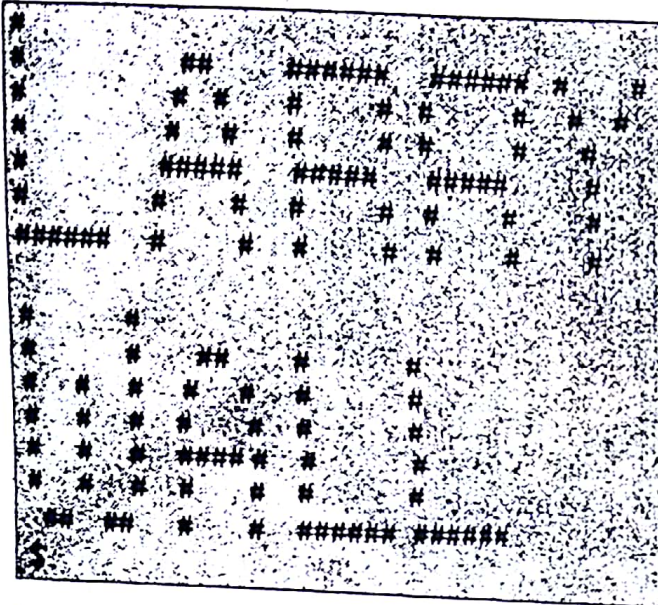
```
#locks for 45 minutes
```

k) The banner Command

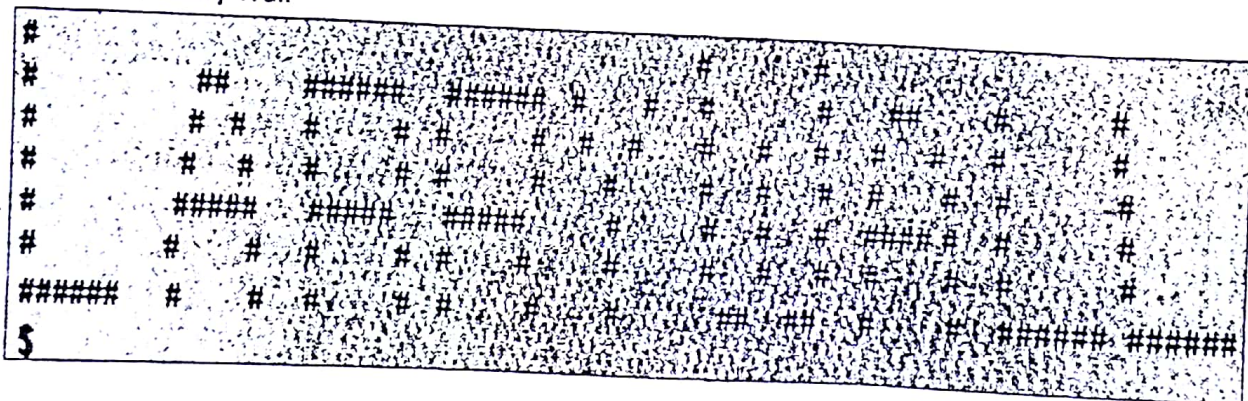
- It is available on SCO UNIX (But not in Linux). ✓
- It is used to display banners and posters.

Example:

i) \$banner Larry Wall



ii) \$banner "Larry Wall"



l) The cat command:

- It is used for creating small UNIX files

Example:

```
$cat > newfile
```

```
Hi
```

```
Hell
```

```
Welcome
```

```
$
```

- The input operation is terminated by using <ctrl+d> command

m) The bc command:

- The bc command is both a calculator and a small language for writing numerical programs
- Math functions in bc command are

Function	Acronym
Cosine	c(n)
Sine	s(n)
Tan	t(n)
Arctan	a(n)
natural log	l(n)
exponential function	e(n)
square root	sqrt(n)
exponent	^

- The bc command is used as a calculator

Example:

```

1. $bc
   sqrt 55
   7
   quit
   $

2. $bc
   scale = 4
   sqrt 55
   7.4161
   quit
   $

```

Example: control construct

```

$bc
for (i=1; i<=4; i=i+1) i^2
1
4
9
16
quit
$

```

n) The spell and ispell commands:

- The spell command displays a list of misspelled words in the document used as argument.

Example:

```
$ cat spell.txt
This is an example.
I am testing the spel command.
Als I am testing the ispell comand.
```

\$ cat > sample

- By using spell command, we can get the misspelled words

```
$ spell spell.txt
Als
comand
exmple
spel
```

\$ spell sample.

- The ispell command is an interactive spell-check program

Example:

```
This is an exmple.
I am testing the spel command.
Als I am testing the ispell comand.
```

\$ ispell sample ✓

- 2
- 1) mine ✓
 - 2) example ✓
 - 3) exemplar ✓
 - 4) exampled ✓

- 5) examples ✓
- 6) expol ✓
- 7) ampule ✓
- 8) example's ✓

- l) Ignore
- r) Replace
- a) Add

- l) Ignore all ✓
- R) Replace all
- X) Exit

6. COMMAND SUBSTITUTION:

- In UNIX, It is possible to run a command within a command.
- The date command can be run within the echo command by writing a command line.

Example:

\$echo Today the date is 'date'

Today the date is Fri Jul 6 9:40:00 IST 2018

- It shows the command (date) is executed within another command (echo).
- The command is given here between single quotes (' ').

- In Korn shell, the command substitution is accomplished by using a \$ sign followed by the command within a pair of parenthesis.

Example:

\$echo Today the date is \$(date)

Today the date is Fri Jul 6 9:40:00 IST 2018

7. GIVING MULTIPLE COMMANDS:

- Normally a single command is given to the shell at its prompt.
- There are many situations when more than one command is given in a single command line.
- One of the ways of giving multiple commands is to use a semicolon (;) between successive commands.

Example:

\$echo "Giving multiple commands"; date; who

- Commands given in this way does not mutually interact with each other in any manner.
- They are executed independently one after the other, from left to right as they appear in the command line.

IMPORTANT QUESTIONS:

1. What is UNIX? What are UNIX Components? Explain with a neat diagram how to enter into UNIX environment
2. What is a UNIX command? What are rules to write UNIX commands? Write its types. And explain the following commands

a) echo b) who c) passwd d) cat e) bc f) ~~spell and ispell~~