Lecture 9: Structure of UNIX

Overview of the lecture:

- UNIX Architecture
  - Kernel
  - Shell
  - Utilities
- UNIX File System
  - UNIX Files
  - Types of Files
  - File Naming Conventions
- UNIX Directories
  - Standard directories of a UNIX system
  - Path names: - Absolute and Relative
- Basic Commands for Working with Directories
  - Pwd mkdir cd Is
  - Metacharacters and Wildcard characters
  - rmdir

STRUCTURE OF UNIX OS

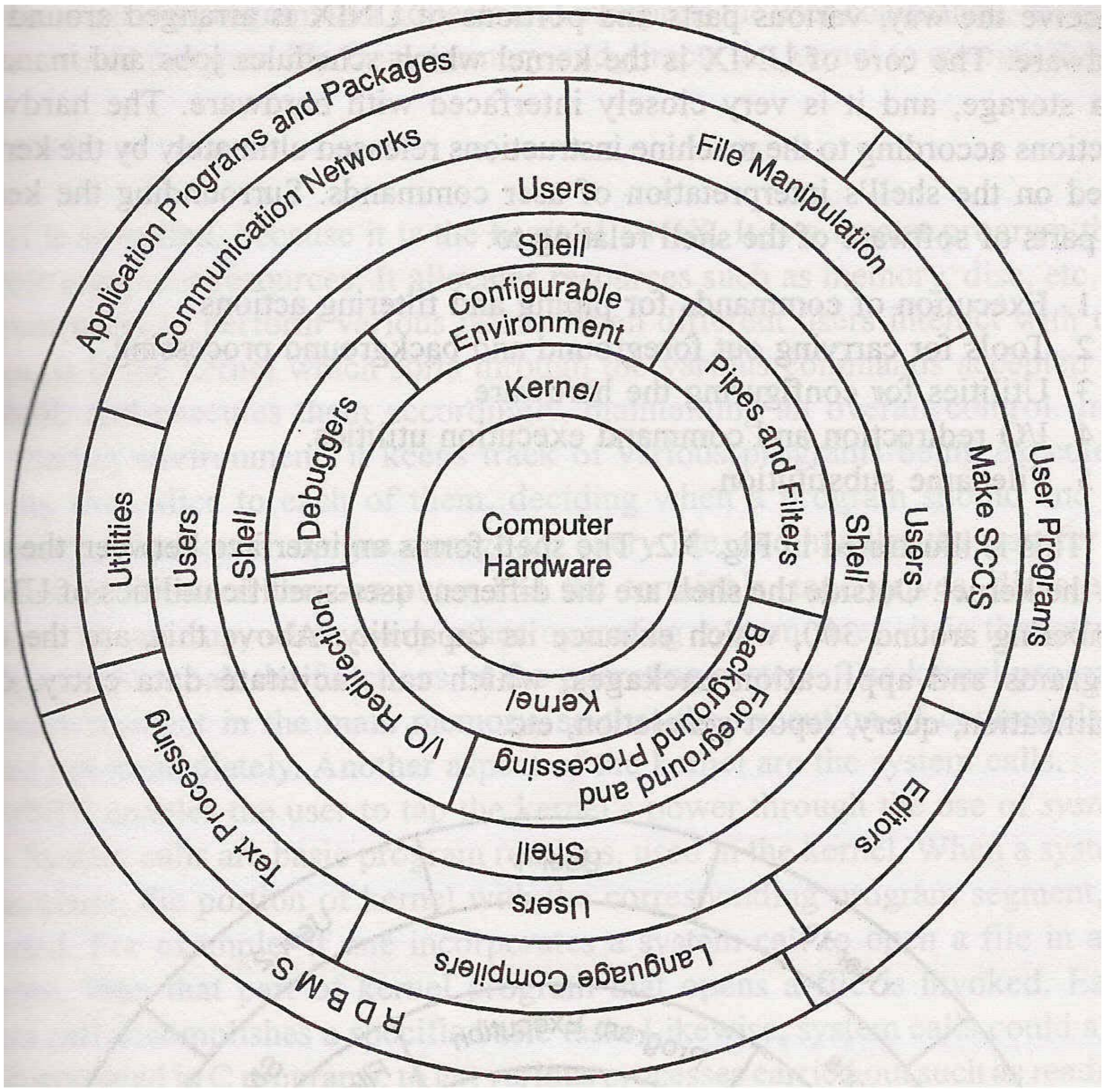


Figure: Conceptual structure of UNIX system software layers

UNIX Architecture

- The Kernel
- The Shell
- UNIX Utility programs

UNIX has evolved over the past thirty years from its conceptual stage into a powerful and effective operating system and the credit goes to the fundamental structure of **UNIX,** which is very robust due to its layered approach comprising of the:

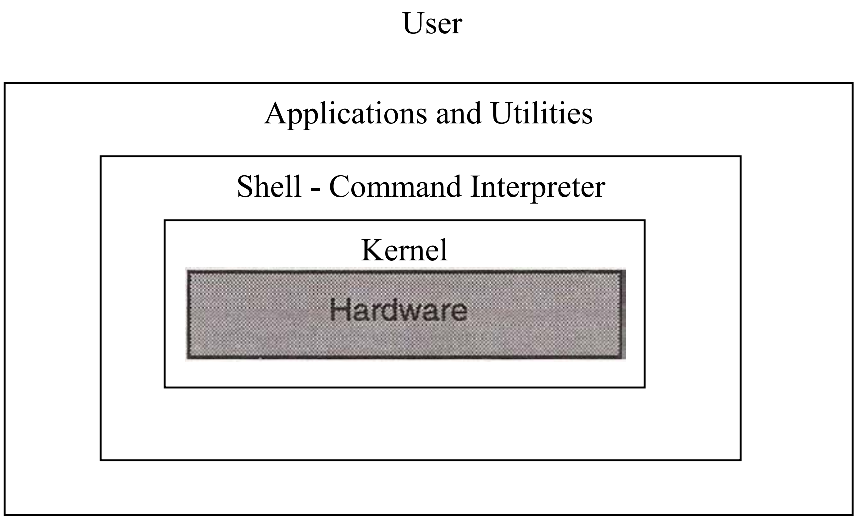
1. Kernel

2. Shell

3. Utilities and applications

One of the most powerful and attractive features of **UNIX** is its **file system,** which manages data, stored on the computer's secondary storage devices. The file system facilitates organizing stored information in a very logical way, and manipulating it as and when required.

Fig: UNIX Architecture



The Kernel

- Core of the UNIX system
- Interacts directly with the hardware
- Insulates other parts of UNIX from hardware
- Performs all low level functions
- Parts of kernel deals with I/O devices, called Device Drivers
- All programs and applications interact with the kernel

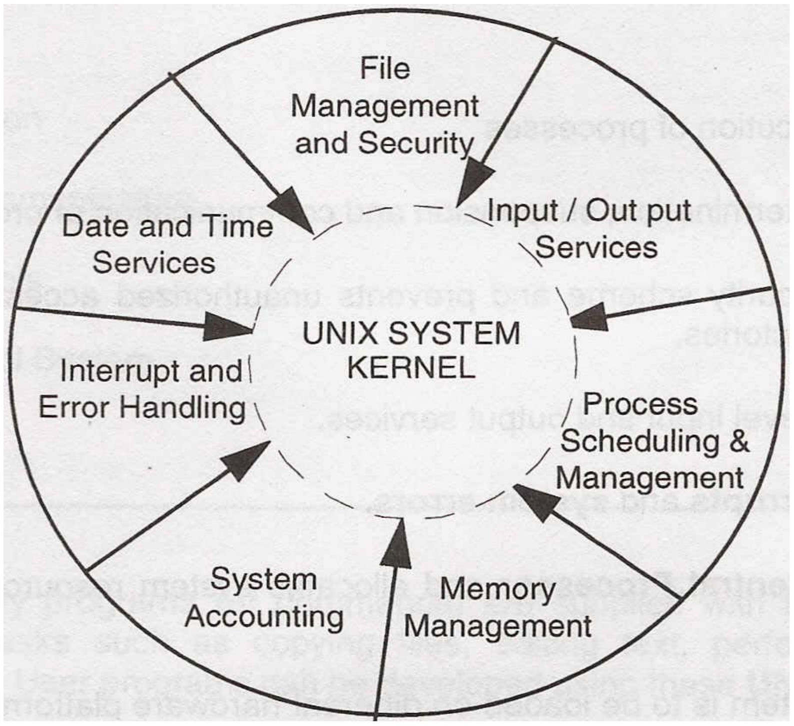


Fig: UNIX Kernel

The kernel is the **Core** of the **UNIX** system, which controls the system hardware and performs various low level functions. Viewing the operating system as a set of layers, the kernel usually refers to the operating system itself.

However, the kernel does not directly deal with the user. Instead, it starts a separate program, called **shell,** which actually interacts with the user and interprets the commands. The **utilities** and **applications** add special capabilities to the operating system.

Kernel functions

- Memory management
- Process scheduling
- File management and security
- Interrupt handling and error reporting
- Input / Output services
- Date and Time services
- System accounting

The Kernel:

- Controls the execution of processes
- Allows creation, termination, suspension and communication of process.
- Enforces the security scheme and prevents unauthorized access to stored information like files and directories.
- Handles all low level input and output services.
- Handles the **interrupts** and **system errors.**
- Schedules the **Central Processor** and allocates system resources to each user of the system.

If the **UNIX** operating system is to be loaded on different hardware platforms, the kernel has to be customized accordingly. .

All applications and utilities including the shell, interact with the kernel by invoking well structured routines of kernel by means of messages called **system** calls.

The Shell

- Is the command interpreter of UNIX
- Interface between the kernel and the user
- Provides powerful programming capabilities

The **shell** is the interface between a user and the operating system. When any user logs in, a copy of the shell program is started up, displaying a **prompt** sign such as '$' (dollar), which informs the user that the shell is waiting to accept a command.

In UNIX, there are three types of shells:

1. The Bourne Shell
2. The C Shell
3. The Korn Shell

From the figures you can conceive the way; various parts and portions of UNIX are arranged around the hardware. The core of UNIX is the kernel, which schedules jobs and manages data storage, and it is very closely interfaced with hardware. The hardware functions according to the machine instructions released ultimately by the kernel, based on the shell's interpretation of user commands. Surrounding the kernel are parts of software of the shell relating to:

1. Execution of commands for piping and filtering actions.
2. Tools for carrying out foreground and background processing.
3. Utilities for configuring the hardware.
4. I/O redirection and command execution utilities.

5. Filename substitution.

The shell forms an interface between the user and the kernel. Outside the shell are the different user-specific utilities of UNIX, numbering around 300, which enhance its capability. Above this, are the user programs and application packages, which can facilitate data entry, data modification, query, report generation, etc.

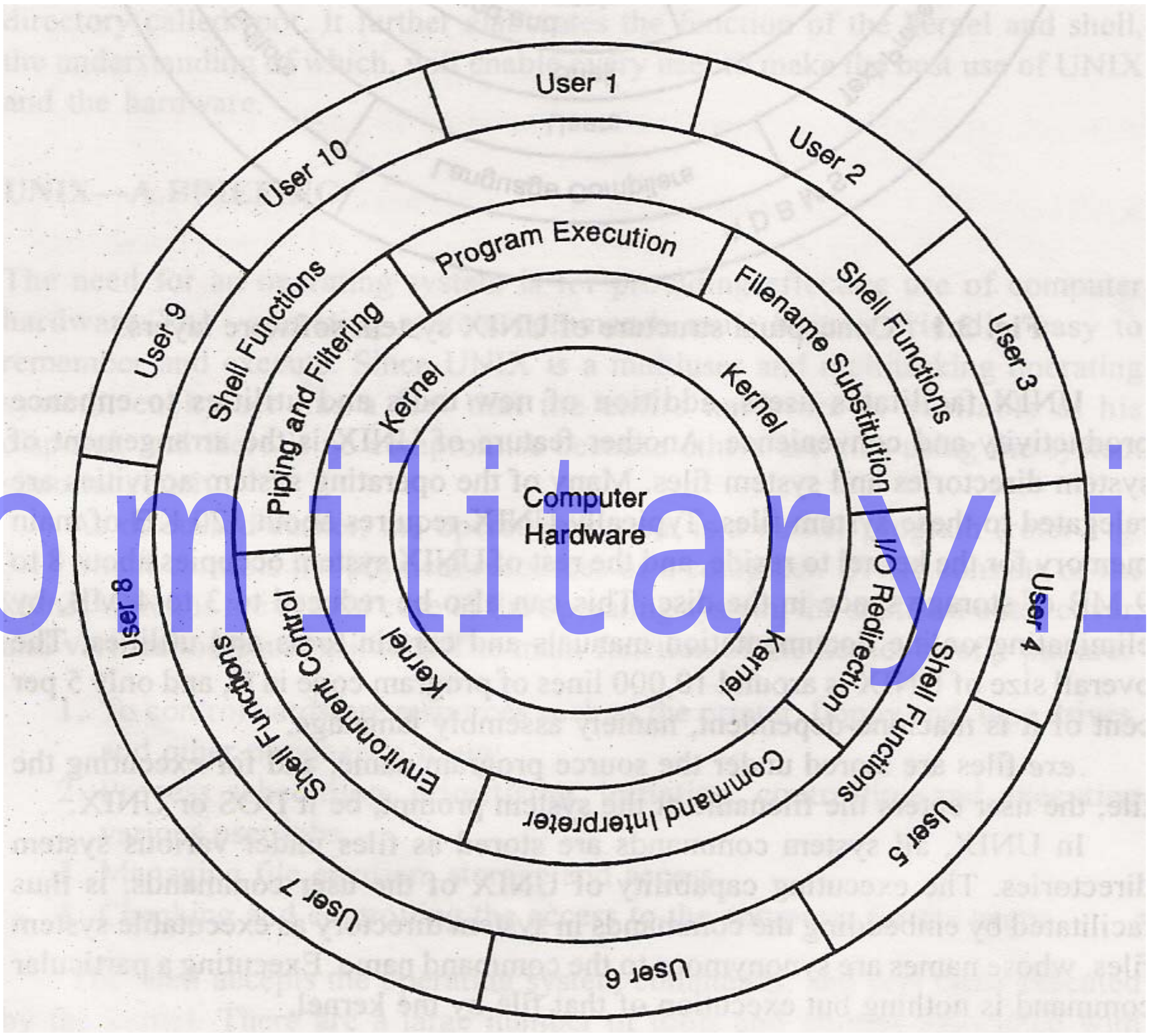


Figure: Shell functions

The shell is a program that collects and interprets the user commands, invokes the concerned program file from memory and directs the kernel to execute them.

SHELL

As the name *shell* suggests, the shell envelops the kernel. Externally, the UNIX operating system is made up of two parts. One part is a large set of system programs, each one corresponding to a command, and another part is the shell, which interprets, manages and coordinates the execution of these programs. Most of the users are unaware of the various complexities of the operating system or the hardware. The user sees the s4ell command as one peculating *down* the UNIX system, and the kernel, looking *up* to the user through the shell- for the next set of process. This relationship and interface of the kernel with different users is shown in the figure below. Every user, once accepted by the kernel as authorized, is issued a shell prompt. Each user has his own shell (a separate and personal copy for each user) and the commands issued by him is counted and numbered serially. A separate shell procedure is reserved for each user.

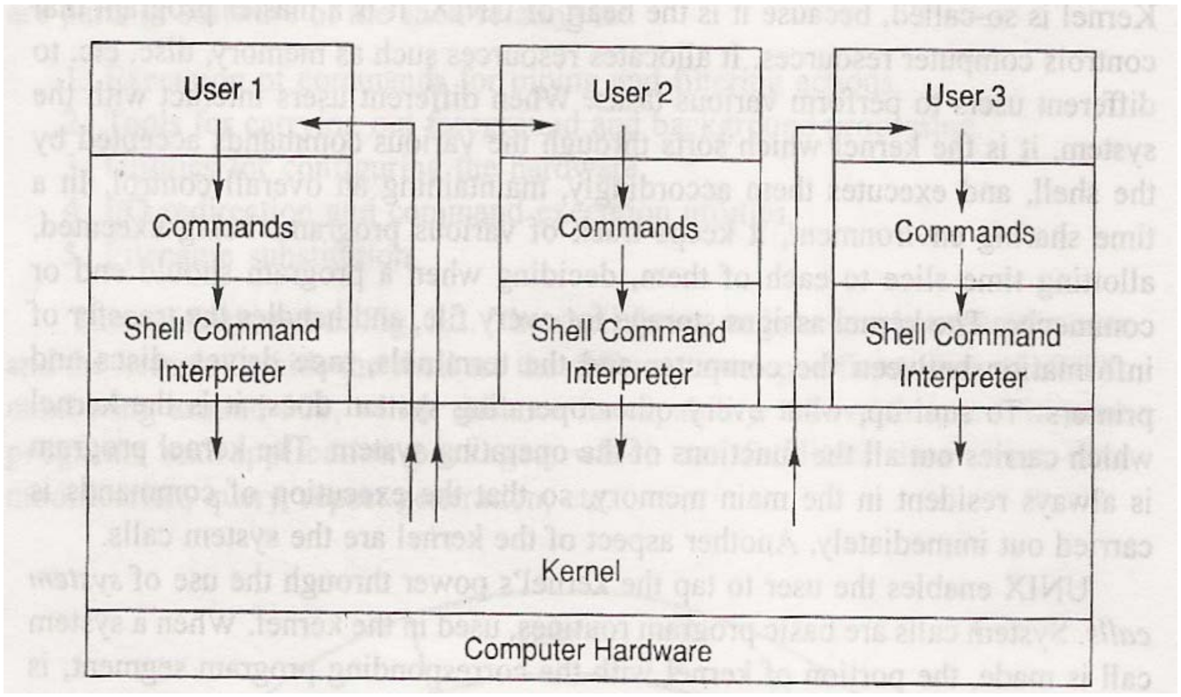
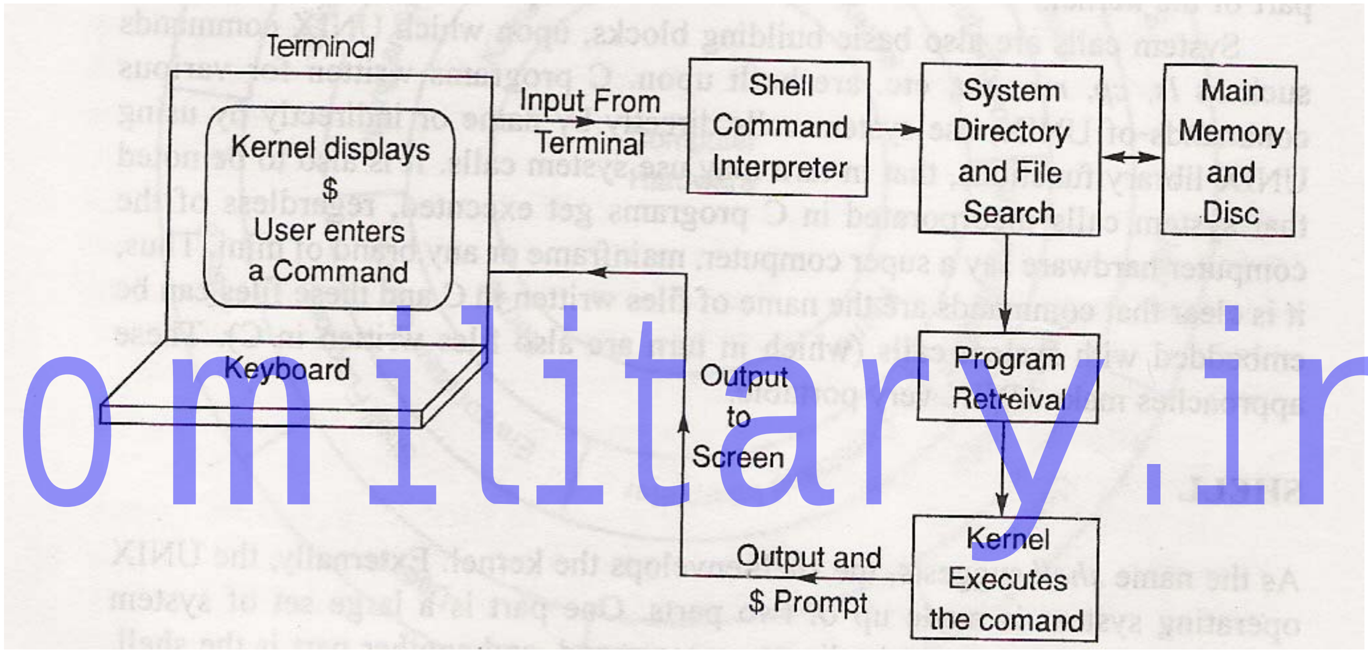

User 1
User 2
User 3
Commands
Commands
Commands
Shell Command Interpreter
Shell Command Interpreter
Shell Command Interpreter
Kernel
Computer Hardware

Figure: UNIX interaction with the users

COMMAND EXECUTION

Refer figure below, which illustrates the cycle of command execution. To begin with, the kernel displays a shell prompt after authorized login. The shell waits for the input from the user, decodes the command line and searches for the program. If the program is found, the shell retrieves and submits it to the kernel



to execute the program. The kernel delivers the output. If the command is not found, it signals the kernel to display *command not found* at the terminal. The shell accepts the kernel's reply, and in both cases displays the next prompt. This cycle continues until the user with <CTRL> D or logout terminates it. Once the shell encounters end of input, it instructs the kernel to log out the user, and displays the login message again.

The system directories where the command file programs are stored are the */bin* and */usr/bin.* On receipt of the command, which is nothing but the executable filenames in the system directory, the shell locates them and transfers control to the kernel to execute. These filenames can be verified by the command *pwd* and *Is,* after changing to the directories *Ibin* and */usrlbin.*

SHELL ENVIRONMENT

A separate shell process is apportioned to each user as stated earlier. The separate shell once created provides an environment within which the user operates. The shell environment consists of the following to ensure proper interaction between the user and the system:

1. the name of the users, and a working directory (a directory allotted to the user by the operating system).
2. a list of system directories to be searched to locate the commands given by the user.
3. the file access permission modes.
4. a record of users identification (user id) and group identification number (group id).

FOREGROUND AND BACKGROUND PROCESSING

The shell can execute commands in the *foreground* and also in the *background.* Foreground commands execute sequentially (one command cannot commence until execution of the preceding command has completed execution). Foreground commands like editing a program or processing a text involves interaction with the terminal and the input. Background processing jobs related commands are accompanied by & (ampersand) symbol. Jobs like compiling, printing can be carried out as background jobs, since they do not require any interaction with the user. There may be several background commands executed at a time. This truly makes UNIX a multitasking system.

Review Questions (lecture 8 & 9):

1. List the main functions of the kernel, and shell, respectively.
2. Explain the relation between UNIX commands and UNIX system filenames.
3. Define and explain *system calls* in UNIX, and explain how UNIX is made portable across various platforms.
4. Sketch the cyclic process of executions of commands issued from the terminal, and various roles played by the shell, kernel and hardware.
5. What does the shell environment contain with reference to every specific user interacting with UNIX.
6. What does background processing mean?
7. What do you mean by tools of UNIX, list the classification of tools and utilities?
8. Sketch the UNIX directory system listing the system directories and the functions of the files under them.
9. Sketch and explain the conceptual structure of UNIX system software layers.
10. Explain with a sketch, how a common user looks upon his interaction with hardware.

II. Fill in the blanks in the following statements:

(i) The two main constituent parts of the UNIX operating system are __________ and -_______.

(ii) The overall size of UNIX in terms of lines of C program code is ______and machine dependent code is ___________.

(iii) Shell commands are the name of ________________ directories.

(iv) User commands are interpreted by_______________

(v) The different type of shells available in UNIX are __________, __________ and _____________and the commands to be used to switch from one to the other are _____________ and_______________.

(vi) UNIX commands are decoded by __________and executed by _______________.

(vii) The topmost directory in UNIX is called as ________________.

(viii) Commands of background jobs to be issued, are proceeded by the symbol, _______.