Your UNIX: The Ultimate Guide—Solutions to Exercises

Solutions To Exercises

Chapter 1

1.1 The operating system (OS) allocates the CPU, memory and registers to be used by programs in execution. It also offers a set of programs as additional services for performing routine functions like file copying, directory removal and so forth.

1.2 It makes a call to the operating system to do the job on its behalf.

1.3 No, when a program needs to perform an operation that doesn't need the services of the CPU (like doing a disk I/O operation), the process representing the program vacates the CPU while the operation is in progress.

1.4 Multiprogramming means that multiple programs can be concurrently present in memory. Multiuser operation implies that these multiple programs can be run by multiple users. Multitasking means that a single user can also run multiple programs.

1.5 UNIX design allows commands to be interconnected so one command takes input from another. This is possible only if the commands are noninteractive. Header information of one command has no meaning for another, the reason why most UNIX commands don't produce headers.

1.6 System calls are routines built into the kernel that handle all calls made to the OS. They are available as functions that can be invoked by C programs designed to run on a UNIX system.

1.7 False, they must use the system calls specified by POSIX.

1.8 *[Ctrl-d]*, **`logout`** and **`exit`**. The last one will always work.

1.9 `README` and `readme` exist as separate files; UNIX filenames are case-sensitive.

1.10 (i) **`tty`** shows the filename of your terminal which also shows up against your username in the **`who`** output. (ii) Clears the screen. (iii) **`id`** shows the username and the group the user belongs to. (iv) **`echo $$`** shows the PID of your shell which also appears in the **`ps`** output.

1.11 The shell. Most systems offer the Bourne, C, Korn and Bash shells.

1.12 The user-id is associated with all files and processes that you create. When you create a file, your user-id is its owner. When you run a program, your user-id is the owner of the process associated with the program. When you send a mail to someone, the mail header shows your user-id as the sender.

1.13 The AT&T and Berkeley schools. Two major standards bodies, POSIX and The Open Group, have developed the Single UNIX Specification that serves as a single reference for all developers.

1.14 The **`cd`** command returns you to the directory where you were placed on login irrespective of where you are currently.

1.15 Because the shell scans the command for special characters and recreates a simplified command structure that is understood by the kernel.

1.16 They are all represented by files.

Chapter 2

2.1 The command is not executed. The shell ignores all text following the #, so the # could serve as a comment character in shell scripts.

2.2 (i) UNIX commands are generally not interactive; Windows programs are. (ii) UNIX commands don't need to have any specific extensions in their filenames; Windows must have `.EXE` or `.COM`. (iii) UNIX is sensitive to case; Windows is not.

2.3 The current directory doesn't exist in `PATH`.

2.4 (i) By changing the value of `PATH` to include that directory. (ii) By using a pathname of the command.

2.5 In `/sbin` and `/usr/sbin`.

2.6 **`cd`** is an internal command of the shell.

2.7 The one in `/bin` is an external command all right, but it's the shell builtin that is normally executed.

2.8 An option is also an argument. No arguments, the `>`, `<` and the words following them are interpreted by the