

LAB Instructions

Security and Architecture of Distributed Systems

Academic year 2005-06

A computer that runs an operating system Linux can be used simultaneously by many users, being thus a multi-user system. Also, an user can run simultaneously several programs on such a system. Thus, Linux is also a multi-process (or multi-tasking) system. To each user it is assigned an *user account* identified by a name. Each account has a password associated with it.

Connecting to the system

Taking into consideration the above short introduction, in order to use a system running Linux, the first step is to create a session. Thus, when you see a prompt line as:

```
login as:
```

you must insert the name of the user account, followed by the password:

```
login as: mario
mario@hostname's password:
```

The operation of inserting the password is without echo (i.e. you will not see anything on the screen no matter how many characters you insert). If either the name and the password are incorrect you will be asked to insert the password again, after an error message:

```
login as: mario
mario@hostname's password:
Access denied
mario@hostname's password:
```

If you were authenticated, the system will print out a number of information, such as the date and hour of the last connection of the user and the version of the

operating system. After this you should be able to see the command interpreter (or “shell”) that waits for your Linux command to execute:

```
mario@hostname:~$ exit
```

In the above case, the result of the command exist is to terminate the session.

Getting help

This section presents shortly the most important command that allow you to obtain detailed information about a Linux command, a function or a file. The Linux command that allows you to access the online manual pages is “man” followed by the command name. For example:

```
$ man man
```

provides information about the command man and is about three pages long. To exit, press key Q (or q).

If you want only a short description of the command you can use the command:

```
$ whatis
```

that search for the data in the whatis database.

Exercise: Enter these commands at the command prompt, and try to interpret the output.

```
$ man ls (you may need to press q to quit)
```

```
$ man who
```

If you don't remember exactly the name of the command that you need to use, try the command “apropos”. For example, to obtain a list of commands which have something to do with "copy", you can execute the next command from the command line:

```
$ apropos copy
```

Another way to obtain information is to use the command named “help”, that provides only shell built-in commands.

Exercise: Use a command that allows you to find out the syntax of the command “help”. Next use “help” to find out information on a “cd” command and interpret the output:

```
$ help cd
```

Most Linux commands can be run with the “--help” option. For example, this command will give you concise help on the Linux “cp” (copy) command:

```
$ cp --help | less
```

Exercise: Use the “--help” option with the next commands: “whoami”, “who”, “passwd”. Next run the command “whoami” and interpret the output. If you run “passwd” what happens ?

Working with directories and files

From the user’s point of view, Linux file system has a tree structure. The root of the system is expressed with /. The root directory usually contains only directories, but the other directories contain directories as well as files. In the root directory there exist a set of directories that contain information more or less standard: /bin, /dev, /etc, /home, /lib, /tmp, /usr

To find out the current directory the next command is used:

```
$ pwd
```

This command will print out the absolute path name of the current directory. Initially the current directory is contained in the environment variable HOME. Environment variables can be viewed with the command:

```
$ env
```

To change the current directory you can use the command “cd”, the syntax of this command being:

```
$ cd new-directory
```

If you want to select the parent directory of the current directory use:

```
$ cd ..
```

To view the content of a directory you need to use the command “ls”.

Exercise: Run the “ls” command in the current directory and interpret the output: how are the files listed?. Run helping commands to find out options of the “ls” command.

Run the “ls” command with an option that allows you to find out the dimension of each directory or file. Which is this option?

Exercise: Run commands that allows you to create a directory (“mkdir”), copy files (“cp”) and move files (“mv”). Create a test directory in your current directory, and copy one file from your current directory in the newly created directory. Move the file from the newly created directory back to your current directory.

To remove a file you need to use the command:

```
$ rm [options] file ...
```

Exercise: Which is the command to remove a directory? Is it also “rm”

Exercise: Explore the filesystem tree using the command “cd”, “ls”, “pwd” and “cat”

Important note: Pay attention when you use the remove commands. Be sure that you **really** want to remove that file or directory.

Editing and compiling C programs in Linux

The “classical” editor for Unix is vi, but in the next exercises a simpler editor will be used, that has enough features to allow the editing of C programs. This editor is named joe.

The syntax for the joe editor is:

```
joe [global-options] [ [local-options] file-name ] ...
```

The most important command is:

```
^K H
```

The character ^ expresses the character CTRL (or control) and for the above command you need to press simultaneously the keys CTRL and K (or keep CTRL pressed and press K), followed by pressing H.

Exercise: Use joe to insert a simple text in a file you created at the above exercises.

If you want to exist the editor without saving the modifications performed you need to press ^C, and if you want to save when exiting you need to press ^K X. If you want only to save and continue editing you need to use ^K D

Compiling C programs

To compile the C programs you can use the utility gcc. The complete syntax is:

```
gcc [-c|-S|-E] [-std=standard]  
    [-g] [-pg] [-Olevel]  
    [-Wwarn...] [-pedantic]  
    [-Idir...] [-Ldir...]  
    [-Dmacro[=defn]...] [-Umacro]  
    [-foption...] [-mmachine-option...]  
    [-o outfile] infile ...
```

Most of the time you will use the form:

```
gcc -o outfile infile ...
```

This command has as effect to compile the input file, the result is put in the outfile. To compile the file test.c you can use:

```
$ gcc -o test test.c
```

If the compiling runs without errors you will find the executable file in the file named test. If you don't specify the output name the executable will be put in a file named a.out, file that be overwritten without any warning.

Debugging programs in UniX

There are many debuggers available for UNIX operating systems, one of the most powerful one is gdb (GNU Debugger).

Unfortunately it is not very easy to use, therefore the programmer should look carefully at the documentation. For better results the developer should also know details about OS and how programs are formatted in memory.

To start the debugger simply use the following command:

```
$ gdb [ -core core_dump_file] program_filename
```

then you will use the gdb command prompt. To run the loaded program use the run command:

```
> run
```

the program will start its execution until an error is caught. In case of errors you can look at the stack of the running process by using the command:

```
> backtrace
```