



OPERATING SYSTEMS

Božo Krstajić, PhD, University of Montenegro Podgorica

bozok@cg.ac.yu



The Bash shell

The **shell** is the traditional interface used by UNIX and GNU/Linux.

In contrast to the X Window System it is an interface that works with **commands**.

In the beginning this can be a bit difficult, but the shell is very powerful.

The default shell on Slackware Linux is **Bash**.

Bash means "Bourne-Again SHell",

Slackware Linux also provides some other shells, but Bash is the main topic of this chapter.



The Bash shell

The most important job for the shell is to execute your commands.

Let's look at a simple example.

Most UNIX variants have a command named *whoami*, which shows as which user you are logged in.

Try typing *whoami*, and press the `<Enter>` after that. The `<Enter>` tells the shell that it should execute the command that you have typed on the current line.



Browsing through shell commands

It often happens that you have to execute commands that you executed earlier.

Fortunately, you do not have to type them all over again.

You can browse through the history of executed commands with the up and down arrows.

Besides that it is also possible to search for a command. Press <Control> + <r> and start typing the command you want to execute.



Completion

Completion is one of the most useful functionalities of Unix-like shells.

Suppose that you have a directory with two files named `websites` and `recipe`. And suppose you want to **cat** the file `websites` (cat shows the contents of a file), by specifying `websites` as a parameter to `cat`.

Normally you would type “`cat websites`”, and execute the command.

Try typing “`cat w`”, and hit the <Tab> key. Bash will automatically expand what you typed to “`cat websites`”.



Wildcards (“joker sign”)

Most shells, including Bash, support wildcards. Wildcards are special characters that can be used to do pattern matching.

The table listed below displays some commonly used wildcards. We are going to look at several examples to give a general idea how wildcards work.

- | | |
|----|---------------------------------------|
| * | A string of characters |
| ? | A single character |
| [] | A character in an array of characters |

Standard input and output

When a command is not getting data from a file, it will open a special pseudo-file named ***stdin***, and wait for data to appear on it.

The same principle can be applied for command output, when there is no explicit reason for saving output to a file, the pseudo-file ***stdout*** will be opened for output of data.

Usually, keyboard input will be used for *stdin*, and *stdout* output will be printed to the terminal.





Redirections

The shell allows you to take use of stdin and stdout using the "<" and ">". Data is **redirected** in which way the sharp bracket points.

In the following example we will redirect the output of ls command to a file named list:

```
ls > list
```

We can also use redirection to provide input to a command:

```
cat < list
```

Pipes

You can also connect the input and output of commands using so-called **pipes**.

A pipe between commands can be made with the “|” character. Two or more combined commands are called a **pipeline**.

```
$ cat /usr/share/dict/american-english | grep "aba" | wc -l
```

So, combined three commands to count the number of words containing the phrase “aba” in this particular dictionary.

