Contents

Glossary

Shell: User interface for Unix-like operating systems

Bash: Default shell on Linux systems

Introduction

A shell is a powerful user interface for Unix-like operating systems. It is an interactive scripting language.

- It can interpret commands and run other programs.
- It enables access to files, utilities, and applications.
- It can be used to automate tasks.

Linux shell commands are used for navigating and working with files and directories. You can also use them for file compression and archiving. In this course, we will learn;

- Characteristics of Linux commands and shell scripting
- Explore the different Linux commands and their outputs
- Bash scripting
- How to schedule jobs using crontab
- How to work with filters, pipes, and variables

Common Linux/Unix Shell Commands

Applications of shell commands include:

- Getting information
- Navigating and working with files and directories
- Printing file and string contents
- File compression and archiving
- Performing network operations
- Monitoring performance and status of the system, its components and applications
- Running batch jobs, such as ETL operations

Running Linux on a Windows Machine

- Dual booth with a partition
- Install Linux on a virtual machine

- Use a Linux emulator (For example Git Bash)
- Windows subsystem for Linux (WSL)

Shell Scripting Basics

Interactive vs Traditional Programming

Interactive programming, also known as live coding, refers to any **computer programming** language that allows the creator to make changes to the program while it is already running. With interactive programming, the designer can make changes to the code without having to run the program over again.

In traditional programming, the coder first writes out the program and then saves it. He then runs the program on the computer. If an error occurs, it's back to the drawing board to type out new code and run the program all over again.

Scripting vs Programming Languages

Basically, all scripting languages are programming languages. The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted. For example, normally, a C program needs to be compiled before running whereas normally, a scripting language like JavaScript or PHP need not be compiled.

Shell Script

A shell script is an executable text file in which the first line usually has the form of an interpreter directive. The interpreter directive is also known as a ‘shebang’ directive, and has the following form: 'pound, bang, interpreter' plus an optional argument.

Interpreter is an absolute path to an executable program, and the optional argument is a string representing a single argument.

Shell scripts are scripts that invoke a shell program.

Interpreter Directive (shebang)

Invoking Interpreters:

#!bin/sh	Invokes Bourne Shell
#!bin/bash	Invokes Bash Shell
#!usr/bin/env python3	Invokes Python Interpreter

Create Shell Script:

touch hello_world.sh	Create empty text file
echo ‘#!/bin/bash’ >> hello_world.sh	Turn text file into bash script by shebang
echo ‘Hello World’ >> hello_world.sh	Redirect Hello World output to Bash script
ls –l hello_world.sh	Check if the file is executable
chmod +x hello_world.sh	Make it executable
./hello_world.sh	Run the Bash script

Filter, Pipes and Variables

- Describe and use pipes and filters
- Explain and set shell and environment variables

Filters are shell commands or programs, which: take their input from standard input, normally the keyboard, and return their output to standard output, which is normally the terminal. We can think of a filter as a transformer, a program that transforms input data into output data. There are many examples, including,

- wc
- cat
- more
- head
- sort
- grep etc.

Pipe commands allows you to chain together sequence of filter commands.

Command 1 | Command 2 → Output of command 1 becomes the input of the command 2 and so on.

Create, See Content and Delete of Shell Variable:

GREETINGS=”Hello”	Create a shell variable
echo $GREETINGS	See its content
unset GREETINGS	Delete Shell variable
set	List all Shell variables

Environmental variables are just like shell variables, except they have extended scope: They persist in any child processes spawned by the shell in which they originate. You can extend any shell variable to an environment variable by applying the ‘export’ command to it.

Environmental Variable:

export GREETINGS	Extend Shell variable to Environmental variable
env	List all Environment Variables
export GREETINGS	Extend GREETINGS
env | grep “GRE”	Check if it is extended

Useful Features of the Bash Shell

- Metacharacters
- Quoting
- I/O redirection
- Command substitution
- Command line arguments
- Batch vs. concurrent modes of execution

Metacharacters

Common Metacharacters:

#	Precedes a comment
;	Command separator
*	Filename expansion wildcard
?	Single character wildcard in filename expansion

Quoting

It is used to interpret or escape metacharacter meaning.

Quoting:

\	Escape special character interpretation
" "	Interpret literally, but evaluate metacharacters
' '	Interpret literally

Backslash removes the meaning of the special character that follows it.

A pair of single quotes removes special meanings of all special characters within them (except another single quote).

A pair of double quotes removes special meanings of all special characters within them except another double quote, variable substitution and command substitution.

```
echo "Current user name: $USERNAME"

      >> Current user name: Bengü Atıcı

echo ’Current user name: $USERNAME’

      >> Current user name: $USERNAME
```

I/O Redirection

Linux sends the output of a command to standard output (display) and any error generated is sent to standard error (display).

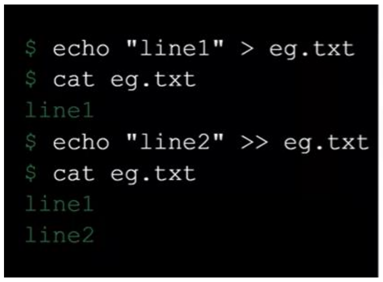
Similarly, the input required by a command is received from standard input (keyboard).

If we need to change these defaults, shell provides a feature called I/O Redirection.

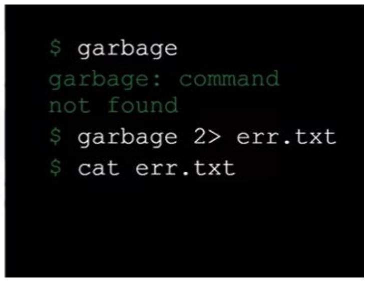
When you redirect using > the contents of the target file are overwritten if the file already exists.

I/O Redirection:

>	Redirect output to file (Output redirection)
>>	Append output to file
2>	Redirect standard error to file
2>>	Append standard error to file
<	Redirect file contents to standard input (Input redirection)



```
$ echo "line1" > eg.txt
$ cat eg.txt
line1
$ echo "line2" >> eg.txt
$ cat eg.txt
line1
line2
```



```
$ garbage
garbage: command
not found
$ garbage 2> err.txt
$ cat err.txt
```

Append Output to File:

echo “Some text” > test.txt	Create test.txt file and append the text
date >> test.txt	Append date to test.txt
cat bengu.txt	Check out the content of test.txt
tr "[a-z]" "[A-Z]" < test.txt	Display the content in all uppercase

Command Substitution

Command substitution is a feature of the shell, which helps save the output generated by a command in a variable.

It can also be used to nest multiple commands , so that the innermost command's output can be used by outer commands.

The inner command is enclosed in $() and will execute first.

Command substitution replace command with its output.

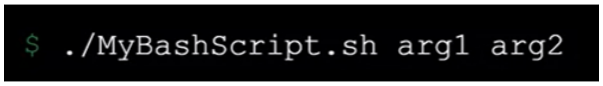
$(command) or *‘command’*

Create Variable and Assign Its Value as pwd:

here=$(pwd)	Create variable
echo $here	Print its value

Command Line Arguments

Command line arguments are arguments used by a program that are specified on the command line. In particular, they provide a way to pass arguments to a shell script, which is itself a program.



```
$ ./MyBashScript.sh arg1 arg2
```

Batch vs. Concurrent Modes of Execution

In batch mode, commands run sequentially.

Command 1 ; Command 2

In concurrent mode, commands run parallel.

Command 1 & Command 2

Scheduling Jobs Using Cron

- Schedule cron jobs with crontab
- Understand the cron syntax
- View and remove cron jobs

The cron utility on Linux and Unix-like operating systems allows you to schedule certain jobs to run automatically at certain times.

❖ Cron is the general name of the tool that runs scheduled jobs consisting of shell commands or shell scripts.
❖ Crond is the service that interprets "crontab files" every minute and submits the corresponding jobs to cron at scheduled times.
❖ A crontab, short for "cron table," is a file containing jobs and schedule data. Crontab is also a command that invokes a text editor to allow you to edit a crontab file.

In Windows, there is *Task Scheduler* to schedule jobs.

```
m h dom mon dow command
```

(minute – hour - day of month – month - day of week)

All five positions must have either a numeric entry or an asterisk, which is a wildcard (*) symbol that means any. Extra spaces are ignored. You can use as many spaces as between positions you want.

Schedule cron Jobs with crontab:

crontab -e	Opens editor
0 2 * * 0 /cron_scripts/backup_data.sh	Runs "backup" data shell script to run at 2 AM on Sundays

To save the job, first type "control x" to exit the editor, and then enter "y" to save your changes. The jobs are now in production! Running crontab with the "l" option returns a list of all cron jobs and their schedules. To remove a job, simply invoke the crontab editor, delete the corresponding line in the crontab file, and save the changes.