Edith Cowan University
School of Science

CSI6203 Scripting Languages

Module 1

Introduction to Scripting Languages and Command-Line Interfaces

Contents

- What and why of scripting
- Linux and Shells
- Bash basics
- Text editors
- Creating and executing scripts
- Variables
- Version control

Learning Objectives

After finishing this module, you should be able to:

- Understand the concepts of Unix-Like operating systems and Linux
- Use the bash shell to perform simple tasks on your computer
- Write and execute simple bash scripts
- Use version control to keep track of your projects and manage changes

What and why of scripting

What is a script?

- Scripting is a form of computer programming
- A script is a series of commands given to a computer to automate a specific job
- Scripts can be used to automatically do many tasks at once or perform a series of operations in a specific order

What is a script?

- Other forms of programming often focus on creating applications designed to be used by non-technical users

- Applications are often designed to be user friendly and involve complex large-scale programming architectures

What is a script?

- Scripts are instead designed to allow technical users to automatically run other programs, schedule complex tasks and automate workflows

- Scripts are usually designed to be executed in a command-line interface

- Scripts are written using sets of short, simple commands rather than complex programming structures and design patterns

What is a script?

- Computers may be controlled through text-based commands

- Any task that can be done on a computer using a mouse and graphical user interface can usually also be done using a text-based command line interface

- Some scripts work by executing these text-based commands in order

- Other scripts use a different programming language (Such as python, perl or ruby)

Linux and Shells

Operating Systems

- There are many different operating systems currently in use today
- Windows is popular in consumer devices such as laptop and desktop PCs
- Most other devices (such as servers, networking equipment, mobiles, embedded devices) use Unix-Like operating systems.

Operating Systems

- Unix was an operating system popular in the early days of computing. Many modern operating systems are either clones of Unix or have been adapted from Unix
- For example. MacOS is a Unix-Like operating system based on BSD
- Many Unix-Like operating systems use a kernel called “Linux” to control their systems

Linux

- In this unit we will be using a Linux-based operating system for scripting.

Shells

- The command interpreter that converts text based commands into instructions for an operating system is called a “Shell”

- Different operating systems use different languages for their command-line interface (CLI)

- Windows has two CLI shells.
  - The Command Prompt (cmd)
  - PowerShell

- Unix-Like operating systems can use many different shells
  - sh
  - bash
  - fish
  - csh
  - zsh

Bash basics

The Bash Shell

- The original shell in use in Unix-like systems is called “sh” or “the Bourne shell”
  - Developed in the 70s by a guy named Stephen Bourne

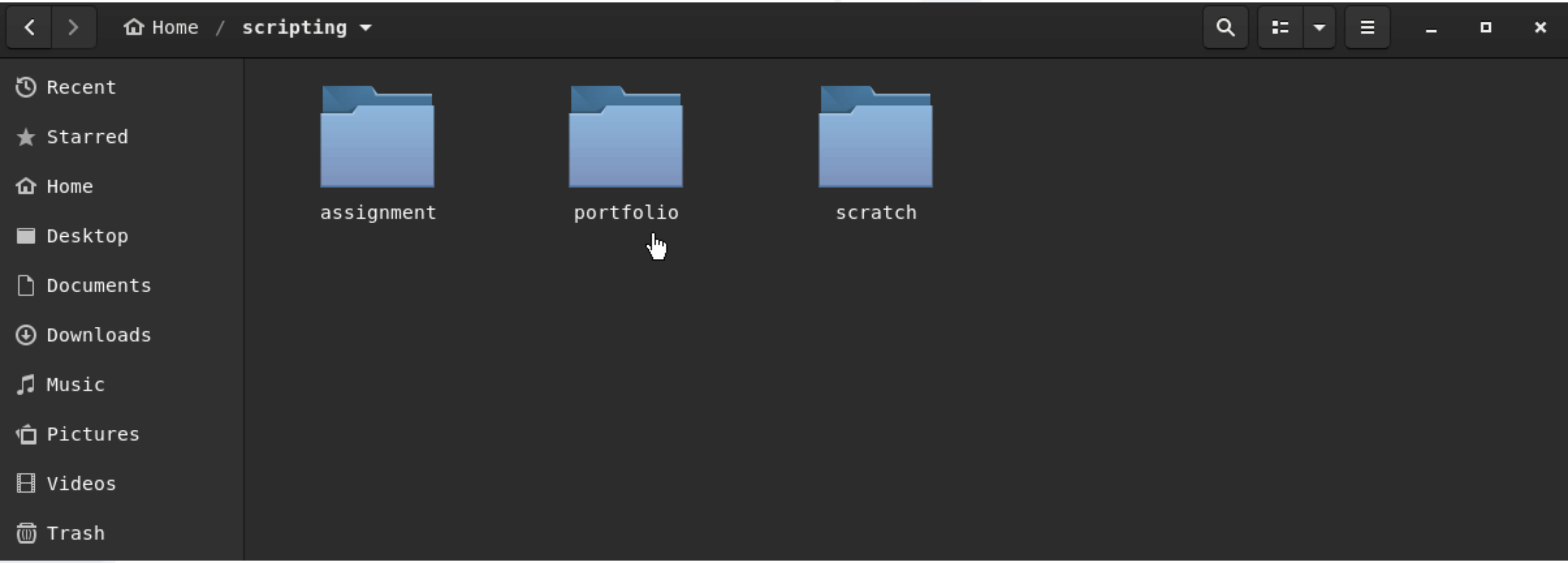
- The most common shell that is found on almost all Unix-Like operating systems is called “bash” or “the Bourne again shell”
  - Compatible with sh commands and scripts
  - Provides a more modern interface and more advanced features

The Bash Shell

- This unit will primarily focus on bash scripting and using the bash shell to automate tasks
- Each keyword used in bash scripting is actually a program
- Even control structures such as “while” or “if” are actually executable programs being run by the bash shell
- The shell is the glue that binds these commands together

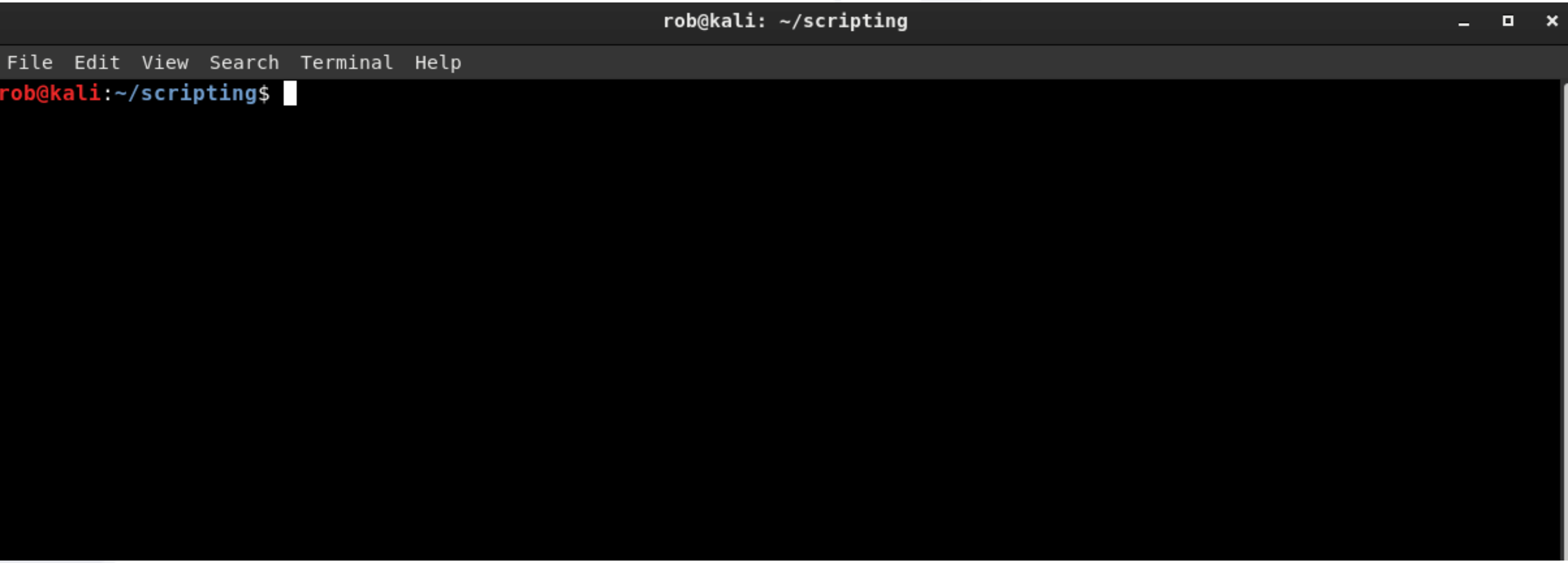
The Bash Shell

- Tasks that can be accomplished by the Graphical User Interface (GUI)



The Bash Shell

- Can also be accomplished using the bash Command Line Interface (CLI)



```
rob@kali:~/scripting$ 
```

Common bash commands

- ls
  - List files in a directory
- cd
  - Change directory
- cat
  - Output the contents of a file or add text to a file
- echo
  - Output/print text to the command line
- mv
  - Move a file to a new location
- cp
  - Copy a file to a new location
- rm
  - Delete a file or files
- mkdir
  - Create a new directory

Text Editors

Text Editors

- In order to create and edit scripts, Text editors are used
- Different programmers have different preferences in which text editor they choose to use.
- There are two main types of text editor
  - Text-based
  - Graphical-based

Text-based Text Editors

- Text-based editors are often used in server environments where there is no graphical user interface available.
- The most common text-based editors are
  - vi or vim
  - nano
  - emacs

Configuring vim

- vim is a very powerful text editor with a steep learning curve
- vim can be configured by editing the $HOME/.vimrc file
- Settings for syntax highlighting, tab sizes, auto-indenting and mouse support can be configured here

Configuring nano

- Nano is easier to learn than vim but lacks some of its more advanced features.

- vim can be configured by editing the $HOME/.nanorc file

- Settings for syntax highlighting, tab sizes, auto-indenting can be configured here

Graphical-based Text Editors

- Graphical-based editors are often used in desktop development environments where there is a graphical user interface available.

- There are many text-based editors. Some of the more common ones include
  - gedit
  - kate
  - Visual Studio Code
  - Atom
  - Sublime

gedit

- gedit is installed by default on linux-based operating systems that use the GNOME desktop environment
- It’s a simple, yet convenient text editor that can be configured with some powerful options

VS Code

- Visual Studio Code is a cross platform text editor made by Microsoft
- Not to be confused with the IDE: “Visual Studio”
- VSCode is designed to be a light-weight cross platform editor for development
- Additional features can be added through custom extensions

Creating and Executing Scripts

Hello World

- Most programming beginners start out by writing a simple program to output the words “Hello World” on the screen

- In bash scripting, this can be accomplished with the following script:

```
#!/bin/bash
echo "Hello World!"
exit 0
```

Hello World

- Lets break this down into the individual parts

- `#!/bin/bash`
  - This is the first line of any script
  - This is call a shebang (hash bang)
  - This is a special comment at the start of the script to tell the system which shell or interpreter to use.
  - We use /bin/bash for bash scripts but may use /bin/php for php scripts or /bin/python for python scripts

Hello World

- Lets break this down into the individual parts

-
  ```
  echo "Hello World!"
  ```
  - The echo command is a built-in shell command and can be used to write a standard output
  - The standard output is called “stdout” and, by default, will print the message to the screen
  - The information to be printed is enclosed in double quotes (More on quotes later in the unit)

Hello World

- Lets break this down into the individual parts

- `exit 0`
  - The exit command is a built-in shell command and can be used to write a standard output
  - The standard output is called "stdout" and is used to leave or exit the current script
  - The exit code is an integer argument (a number) that is used to inform other scripts that the script has successfully completed
  - A number other than 0 can be used to indicate that some type of error has occurred in the script's execution

Executing the script

- In most operating systems, files are not allowed to be directly executed as scripts by default
- Most scripts will need to have the correct permissions set to allow them to be executed
- A script without the execute permissions can be manually executed by invoking the bash command but this is not ideal

```
$ bash hello1.sh
```

Executing the script

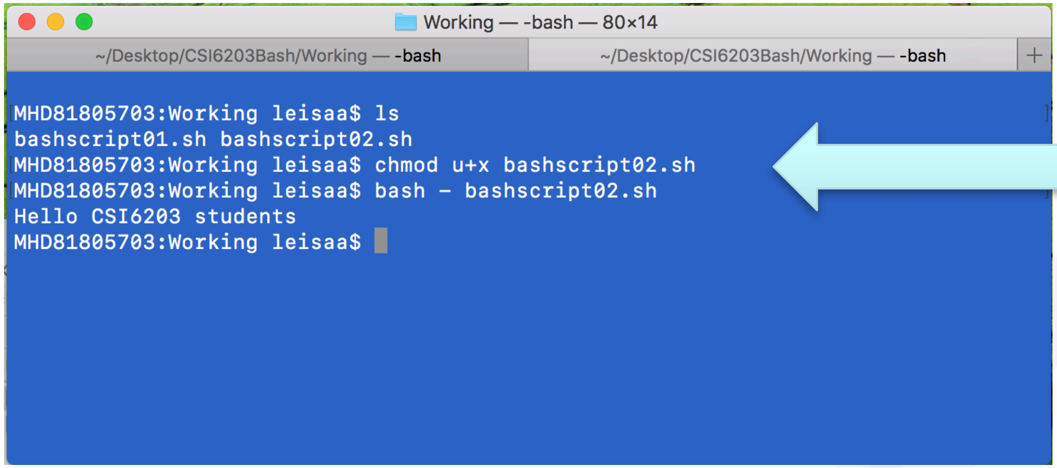
- Instead, permissions can be set using the chmod command

```
$ chmod +x hello1.sh
```

- This only needs to be done once. After the file is marked as executable, the script can be run

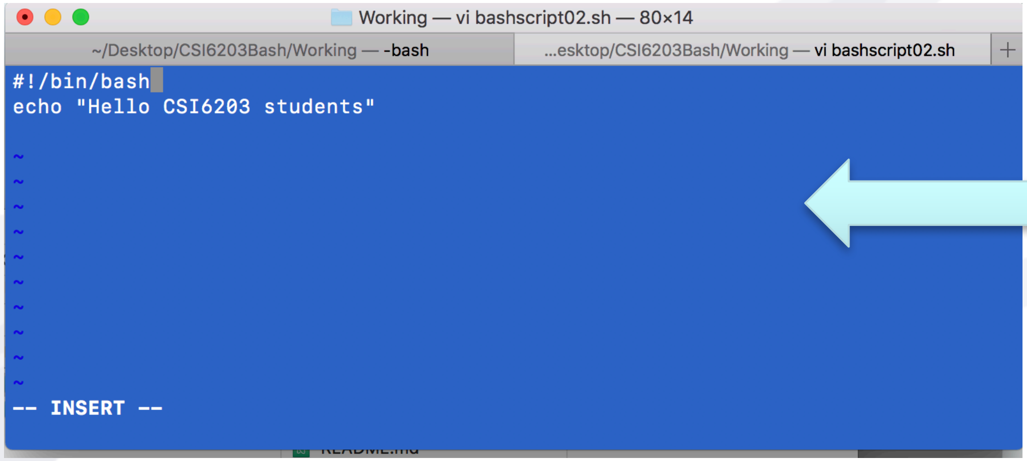
```
$ ./hello1.sh
```

Executing the script


Working — -bash — 80×14
~/Desktop/CSI6203Bash/Working — -bash
~/Desktop/CSI6203Bash/Working — -bash

```
MHD81805703:Working leisaa$ ls
bashscript01.sh bashscript02.sh
MHD81805703:Working leisaa$ chmod u+x bashscript02.sh
MHD81805703:Working leisaa$ bash - bashscript02.sh
Hello CSI6203 students
MHD81805703:Working leisaa$ 
```

executing script


Working — vi bashscript02.sh — 80×14
~/Desktop/CSI6203Bash/Working — -bash
...esktop/CSI6203Bash/Working — vi bashscript02.sh

```
#!/bin/bash
echo "Hello CSI6203 students"

~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

Creating script in VI editor

Variables

Variables

- Information in bash scripts can be stored in variables
- Once the information is stored, it can be recalled later on in the script

```
#!/bin/bash
name="Mokhtar"
age=35
echo "$name is $age years old"
```

```
Mokhtar is 35 years old
```

Arguments

- There are some special variables that are used for receiving input from the command line as they are executed.
- Each argument typed into the command line is referred to by a special numeric variable.

```
#!/bin/bash
echo "Hello $1!"
```

```
$ ./hello.sh Rob
Hello Rob
```

Arguments

- The following variables are available to handle arguments
- These can be used to print custom messages or get the name of the script

Argument Name	Purpose
$0	The name of the script file
$1	The first argument sent to the script
$2	The second argument sent to the script
$#	The number of arguments sent to the script
$*	All the arguments sent to the script

Version Control

Saving scripts

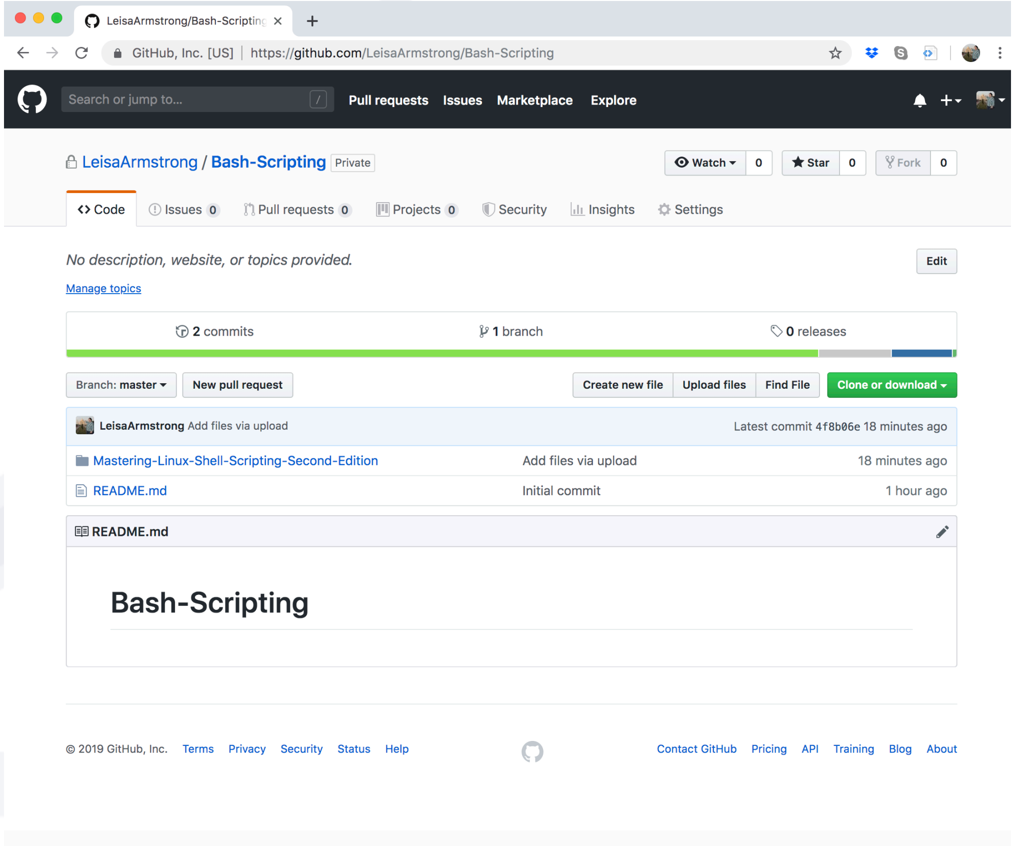
- Script files can be stored in directories just like any other files
- Often, developers will save their scripts in a “version control repository” in order to keep track of any changes that are made to the scripts
- This way, if the script changes or is lost, earlier versions of the script can be recovered

git

- The most common program for version control is called “git”
- git can be used, not only to keep track of changes, but also to allow multiple developers to work on projects
- git keeps track of which changes were made in what versions by what people

github

- github is a company that provides access to online, cloud stored, git repositories.
- This is very convenient for open source developers to be able to share their code with the world and allow many people to work together and collaborate on development projects
- In this unit we will be using git to keep track of our assignment and portfolio work



LeisaArmstrong / Bash-Scripting Private

No description, website, or topics provided.

LeisaArmstrong Add files via upload — Latest commit 4f8b06e 18 minutes ago

Mastering-Linux-Shell-Scripting-Second-Edition	Add files via upload	18 minutes ago
README.md	Initial commit	1 hour ago

README.md

Bash-Scripting

Summary

- Terms to review and know include:
  - Scripting
  - Linux
  - Bash
  - Text Editors
  - Variables
  - Version Control
  - git

References and Further Reading

- Ebrahim, M. and Mallet, A. (2018) Mastering Linux Based Scripting (2nd Ed) Chapter 1, pp 1-34
- http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-5.html
- http://tldp.org/LDP/abs/html/varassignment.html
- http://tldp.org/LDP/abs/html/declareref.html
-