**Shell**

**Prompt**

$ You are in a shell, most likely POSIX(sh) compatible

# root

% You are probably in the C shell

> You are on a continuation line e.g.inside a string

**Set Variable**

p=arguments

gcc -Wall $p.c -o $p

**set enviroment variable**

export CHARSET=UTF-8

// change default editor

export EDITOR=nano

<https://cs-uob.github.io/COMSM0085/exercises/part1/posix1/ssh.html>

Secure Shell

It's used to connect to remote host securely.

Give public key to others, and keep the private key in your own machine.

Do not generate too many key pairs, because you can just use one public key in everywhere.

## Command to log in

ssh -A -J USERNAME@seis.bris.ac.uk USERNAME@rd-mvb-linuxlab.bristol.ac.uk

We can just put the command in '**config' file**

### config

Host seis

HostName seis.bris.ac.uk

User USERNAME

Host lab

HostName rd-mvb-linuxlab.bristol.ac.uk

ProxyJump seis

User USERNAME

Plain text

#### Don't like id\_ed25519?

Host github.com

IdentityFile FILENAME

Plain text

## key

#### Generate key

ssh-keygen -t ed25519

They are stored in ~/.ssh.

#### Secure Copy

scp source destination

[USERNAME@]HOSTNAME:PATH

// paste source to the end of dest, but not cover

cat SOURCE >> DEST

Plain text

#### authorized\_keys

SSH will accept a public key if it is listed in the file authorized\_keys in the user's .ssh folder, the format is **one line per key**. You can put as many keys as you want.

## Debug

-v -vv -vvv after commands

vagrant up

vagrant ssh

//ssh -i KEYFILE vagrant@localhost -p 2222

Plain text

## Shutdown

exit

vagrant halt

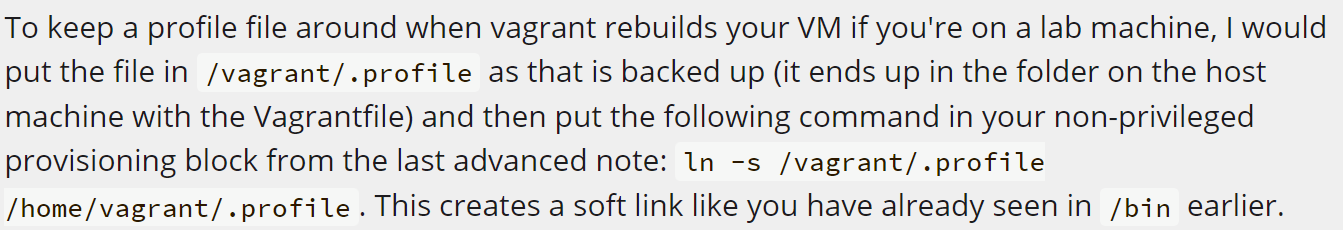
## Lab

Everytime you leave the lab, all the programs will be deleted. So please remember to put commands in Vagrantfile.

Text

Description automatically generated

for git identity



In git1 part1

// ln -s target newName

ln -s /vagrant/.profile /home/vagrant/.profile

Plain text

## Spring

// run on alpine and browse on host

config.vm.network "forwarded\_port", guest: 8080, host: 8080

Plain text

#### on terminal

wget localhost:8080 -q -O /dev/stdout

// double quote every part

program="silly name"

gcc -Wall "$program.c" -o "$program"

## Tips

Sometimes when your on a new system you’ll get a prompt to set your name and email… just follow the instructions provided. All Git commits need a name and an email address attributed to them.

git config --global user.name 'Joseph Hallett'

git config --global user.email 'joseph.hallett@bristol.ac.uk'

Git Ignore

When a filename or a foldername/ or a suffix like .exe is included in .gitignore, it won't be recorded.

Git Status

Up to date: there have been no commits on your local or the remote repository since you last synchronised.

Ahead of remote: you have made commits locally that you have not yet pushed to the remote.

Behind remote: someone else, or you on a different computer, have made commits to the remote that you do not have on this computer yet.

Diverged from remote: both your computer and the remote have had different commits since the last time you synchronised

Conflicts

rebase

Pretend that member two had actually fetched the one commit before starting their own work.

### merge

The second member is about to do a merge, which can either **succeed** (as it should here, because **different people edited different files**) or **fail** with a merge conflict (for example if **different people edited the same** file).

#### Fake Conflicts

git pull >> git merge >> git push

ahead by 1 commits

#### Real Conflicts

git pull >> fix >> git add . >> git commit >> git push

ahead by 2 commits:

First one is the version of member 2, and second one is the fixed version.

## Undo

reset

revert

// revert to the version before target version

git revert B

ABC but A is the same with C

Plain text

checkout

Table

Description automatically generated

Write a caption

## Branch

### create branch

git checkout -b develop

// push the new branch to remote repository

git push --set-upstream origin develop

Plain text

### show branches

git branch

// show all the branches including the remote ones

git branch -a

Plain text

### remote branches

git remote show origin

## C

### Makefile

#### Phony

all : depends on everything you’d like to build

clean : deletes all generated files

install : installs the program

Plain text

## Python

sudo apk add python3 py3-pip

quit: ^D

### package management

#### pip

sudo pip3 install NAME

import PACKAGENAME

#### conda

It can handle packages that pip repository lacks

conda install NAME

## JAVA

#### ad

add to PATH in alpine

export PATH="$PATH:/usr/lib/jvm/java-17-openjdk/bin/"

Plain text

**SQLite**

// Some code

$ sqlite3 database.db

Plain text

**MarioDB**

[[](https://www.tutorialspoint.com/mariadb/index.htm)](https://www.tutorialspoint.com/mariadb/index.htm)

[MariaDB Tutorial](https://www.tutorialspoint.com/mariadb/index.htm)

tutorialspoint

**create database**

// create a database

sudo /etc/init.d/mariadb setup

// run the database server

sudo rc-service mariadb start

// check server state

rc-status

// automate this process

sudo rc-update add mariadb default

Plain text

// start server

$ systemctl start mariadb

$ systemctl enable mariadb

$ mysql

Plain text

// initiate database

mysql -e 'source /vagrant/sample-data.sql'

Plain text

**secure code**

sudo mysql -e 'source /vagrant/secure-setup.sql'

source means "load a file and run it"

// secure-setup script

DELETE FROM mysql.user WHERE User='';

DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1');

DROP DATABASE IF EXISTS test;

DELETE FROM mysql.db WHERE Db='test' OR Db='test\\\_%';

FLUSH PRIVILEGES;

Plain text

**syntax**

// The combination of mul lines can not be the same

UNIQUE(a, b)

Plain text

**show**

// information about tables

DESCRIBE TABLENAME;

SHOW CREATE TABLE TABLENAME;

SHOW TABLES;

SHOW DATABASES;

Normalization

2nf

Each non-key attribute must fully depend on the entire primary key.

3nf

Every non-key attribute should depend on the key, the whole key, nothing but the key.

bcnf

Every attribute should depend on the key, the whole key, nothing but the key.

**SQLite**

// Some code

$ sqlite3 database.db

MarioDB

tutorialspoint

create database

// create a database

sudo /etc/init.d/mariadb setup

// run the database server

sudo rc-service mariadb start

// check server state

rc-status

// automate this process

sudo rc-update add mariadb default

// start server

$ systemctl start mariadb

$ systemctl enable mariadb

$ mysql

// initiate database

mysql -e 'source /vagrant/sample-data.sql'

secure code

sudo mysql -e 'source /vagrant/secure-setup.sql'

source means "load a file and run it"

// secure-setup script

DELETE FROM mysql.user WHERE User='';

DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1');

DROP DATABASE IF EXISTS test;

DELETE FROM mysql.db WHERE Db='test' OR Db='test\\\_%';

FLUSH PRIVILEGES;

Plain text

**syntax**

// The combination of mul lines can not be the same

UNIQUE(a, b)

Plain text

**show**

// information about tables

DESCRIBE TABLENAME;

SHOW CREATE TABLE TABLENAME;

SHOW TABLES;

SHOW DATABASES;

Plain text

**Normalization**

Write a caption

**2nf**

Each non-key attribute must **fully** depend on the **entire primary key**.

**3nf**

Every non-key attribute should depend on the key, the whole key, nothing but the key.

**bcnf**

Every attribute should depend on the key, the whole key, nothing but the key.

// Some code

<dependencies>

<dependency>

<groupId>org.mariadb.jdbc</groupId>

<artifactId>mariadb-java-client</artifactId>

<version>2.7.1</version>

</dependency>

<dependency>

<groupId>net.java.dev.jna</groupId>

<artifactId>jna</artifactId>

<version>5.6.0</version>

</dependency>

<dependency>

<groupId>net.java.dev.jna</groupId>

<artifactId>jna-platform</artifactId>

<version>5.6.0</version>

</dependency>

</dependencies>

[Filesystem Hierarchy Standard](https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html)

file system mannuals

/bin

executable file (include many command like ls/grep)

Have a look with ls -l /bin: the very first character of each line indicates the file type, the main ones being - for normal file, d for directory and l for a so-called soft link.

Text

Description automatically generated with medium confidence

Write a caption

**/sbin**

collection of programs, typically ones that only system administrators will use

**/usr**

system, read-only (not user)

* /bin was only for binaries needed to start the system - or at least the most important binaries that needed to live on the faster of several disk drives, like your shell.
* /usr/bin was where most binaries lived which were available globally, for example across all machines in an organisation.
* /usr/local/bin was for binaries installed by a local administrator, for example for a department within an organisation.

**/usr/lib and /usr/bin**

In any case, /usr and its subfolders are for normally read-only data, such as programs and configuration files but not temporary data or log files. It contains subfolders like /usr/bin or /usr/lib that duplicate folders in the root directory.

Ubuntu's way of cleaning this mess up is to make its /bin just a link to /usr/bin and putting everything in there. On alpine linux, there is still a distinction between the two, but most binaries in both folders are links to /bin/busybox anyway. For example, if you do which ls you find /bin/ls, but which which shows /usr/bin/which, but both of these are in fact just links to /bin/busybox.

**/usr/include**

For the next exercise, please also install the gcc package that gets you a C compiler, as well as musl-dev which contains header files like stdio.h. These, in case you wondered, live in /usr/include.

**/etc**

system-wide configuration files and typically only root (the administrator account) can change things in here

**/lib**

dynamic libraries - windows calls these .dll files, POSIX call these .so

**Alpine**

sudo apk add NAME

**Download Files**

wget

curl

**Extract Tar**

tar zxvf FILENAME

tar -xvf FILENAME

The options here are x=extract a file, v=verify (print the name of every processed file to standard output), f=the filename is in the following argument. You may sometimes see this command without the '-' for these options -- this works because tar supports an older convention where options are not prefixed with a dash, but to be safe you should stick to the modern convention (which it also understands).

To create a tar file yourself, the command would be tar -cvf ARCHIVE.tar FILE1 FILE2... where c=create the archive if it doesn't exist, and assume all arguments not consumed by another flag refer to files to be added. In fact, tar -xvf ARCHIVE.tar FILES... also works and only extracts the named files from the archive.

Exit with

⌃↩

## Root

// switch to root

sudo bash

Plain text

#### add user

sudo adduser NAME

Plain text

#### Switch User

su USERNAME

## Permission

chmod u=rw,g=x,o=rx \*

chmod u+s \*

(setuid)

## Group

sudo usermod -g GROUPNAME USERNAME

Plain text

(DONOT use -a -G, since the groupname will remain the same)

#### And remember to change the file group

chgrp -v GROUPNAME USERNAME

Plain text

## Sudo Users

// check the users' permissions

sudo cat /etc/sudoers

Plain text

// the first one would let everyone in group wheel run

// commands using sudo (this is the default on some other

// linux distributions), whereas the second one would allow

// everyone in the group sudo to do this, but would prompt for their

// own password beforehand.

%wheel ALL=(ALL) NOPASSWD: ALL

%sudo ALL=(ALL) ALL

Plain text

**Package**

sudo apk add nano

apk info NAME

apk info -a NAME

Plain text

* sudo apk update fetches the new package list from the repository.
* sudo apk upgrade upgrades every package that you already have installed to the latest version in your **local package list** (downloaded when you do an apk update).

**/usr**

For the next exercise, please also install the gcc package that gets you a C compiler, as well as musl-dev which contains header files like stdio.h. These, in case you wondered, live in /usr/include.

**Nano**

**syntax highlight**

If you install the nano-syntax package, you get syntax highlighting in nano, but you need to configure this first. The syntax files themselves live in /usr/share/nano, for example c.nanorc for the C language, but you have to include the ones you want in a file ~/.nanorc. For example, if this file contains the line include /usr/share/nano/c.nanorc then nano will do syntax highlighting on C files. You can turn this on and off with Alt+Y.

**redirect**

< and > are for files, command must be at the left of the statement

| is for streams

// output stream

$ COMMAND > FILE overwrites FILE

$ COMMAND >> FILE appends to FILE

Plain text

**error**

the stream number 1 is standard stream, whilst number 2 is standard error

// redirect error into files

$ COMMAND > FILE 2> FILE2

$ COMMAND > FILE 2>&1

// redirect standard stream

ls -l 1 > tmp

Plain text

**files vs streams**

// A program that uses a standard stream can be told to use a file instead by

PROGRAM < FILE (standard input)

PROGRAM > FILE(standard output)

PROGRAM 2> FILE(standard error)

Plain text

// A program that uses a file can be told to use standard input/output instead by

$ pdftotxt NAME.pdf NAME.txt

// output stream

$ pdftotxt NAME.pdf - | grep software

Plain text

**Pager**

ls | less

Plain text

**Grep**

// grep the first number of lines

head -n NUMBER

// skip the last number of lines and print the rest

head -n -NUMBER

// grep the last number of lines

tail -n NUMBER

// start to print from line number

tail -n +NUMBER

// -i case-insensitive, and with -v it only prints lines that do not match

grep [-iv] EXPRESSION

## 1. Checker

Run it on everything you ever write

2. Basics

Shebang

#! at the begging of shellscripts

// #!+path to interpreter+arguments

#!/usr/bin/env bash

or

#!/bin/bash

or

#!/bin/sh

codes

Env

it will look through the PATH and tries to find the program specified and runs it.

And we don't need to write the path to interpreter but just set the PATH.

Set PATH

export PATH="${PATH}:/extra/directory/to/search"

Better, put the line into:

~/.profile

Execute

// Then type these lines

chmod +x my-script.sh

env ./my-script.sh

(or just one line) sh my-script.sh

Exit

$? contains the exit sign

0 success

positive number means failure

[ $? -eq 0 ] && printf "Command succeeded\n"

Plain text

## 3. Syntax

A; B run A then run B

A | B run A and feed its output as the input to B

A && B run A and if successful run B

A || B run A and if not successful run B

Plain text

### Control flow

// if

if test -x myscript.sh; then

./myscript.sh

fi

// loop

for file in \*.py; do

python "${file}"

done

Plain text

### Variable

GREETING="Hello World!"

echo "${GREETING}"

export GREETING (exist in the programs you start as an environment variable)

unset GREETING

#### Standard Variables

${0} Name of the script

${1}, ${2}, ${3}… Arguments passed to your script

${#} The number of arguments passed to your script

${@} and ${\*} All the arguments

Plain text

## 4. Strict Mode

[Bash Strict Mode](http://redsymbol.net/articles/unofficial-bash-strict-mode/)

Write a caption

Man

export MANPATH=/usr/share/man

maven

**Maven Build**

mvn archetype:generate

Plain text

**show folders**

find .

Plain text

**pom.xml**

dependency

plugins

// enable exec:java

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.0.0</version>

<configuration>

<mainClass>org.example.App</mainClass>

</configuration>

</plugin>

Plain text

**compile and run**

// source code

mvn compile

mvn exec:java

// Unit tests

mvn test

Plain text

mvn package creates a jar file of your project in the target/ folder.

mvn clean

I assume that you will be storing your Java projects in git repositories. In this case, you should create a file .gitignore in the same folder as the pom.xml and add the line target/ to it, since you don't want the compiled classes and other temporary files and build reports in the repository.

mvn spring-boot:run

Graphical user interface, text, application, Teams

Description automatically generated