Introduction to COMP519 Labs (Lab Intro 1)
Using the Departmental Linux Systems

1 Introduction

The lab PCs using a Windows operating system are only part of the department's computing facilities. There are also a number of systems using a Linux operating system (currently Rocky Linux 7.9), which are available to all members of the department. The assignments on COMP519 will ask you to produce *web pages* and *web-based applications*. The web server hosting those and providing them to the public is itself a Linux system. You are not able to directly work on the web server, but there are other Linux systems that provide an easy way to set up web pages and web-based applications on the web server via their shared filestore. All these systems together can be considered to be the *production servers* for your applications, that is, they host the applications and make them accessible to the public. Typically, applications would only be placed on production servers after development is fully completed and extensive testing has been done on a separate set of systems, the *development servers*. To simplify matters, we will not make that distinction. We assume that all development work is conducted on the production servers.

This practical is intended to familiarise you with the departmental Linux systems. The tasks described below will guide you through the process of accessing and logging onto a Linux system, using the command line interface and editing text files.

While you work through the tasks below compare your results with those of your fellow students and ask a demonstrator for help and comments if required.

2 Logging in to the Linux systems

The Departmental Linux systems are not physically accessible, but can be accessed over the network. You use the same University (MWS) username and password to log in to both the Windows and Linux systems, but the personal filestore on the Linux systems, called your '*home directory*', is separate from the `M:` drive on the Windows systems.

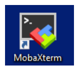
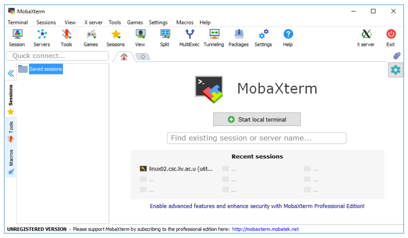


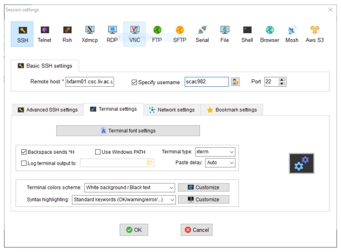
Figure 1: MobaXterm Shortcut

Commence by logging in to the Windows PC in the same way as you did previously. Now double-click on the "`MobaXterm`" shortcut (Figure 1) on the left-hand side of the desktop to open the MobaXterm application (Figure 2a). Click on "`Session`" in the toolbar, this will open a window for session settings. In the toolbar of that window click on "SSH", this will open a new window in which you can enter the connection details for an SSH connection to one of our Linux servers. In the text field to the right of the label "`Remote Host`" enter the name of a Linux server, `lxfarm01.csc.liv.ac.uk` to `lxfarm16.csc.liv.ac.uk`, click on the button to the left of "`Specify username`", then enter your University username into the text field to the right of that label. Click on the tab "`Terminal Settings`" below the text fields (Figure 2b). You will find an option "`Terminal colors scheme`" with a drop-down menu to the right of it. In that menu chose the option "`White background / Black text`" (unless you are really into retro colour schemes, in which case you should start with "`Black background / White text`" and customise that to use green text). Then click on the "`OK`" button at the bottom of the window. A new tab will open in the main window pane of MobaXterm in which you will see a prompt asking for your password (Figure 2c). Enter the password for your University (MWS) account. Note that there will be no visiual indication that you are typing anything. Press `RETURN` once you have typed in


MobaXterm
Start local terminal
Find existing session or server name...
Recent sessions
Enable advanced features and enhance security with MobaXterm Professional Edition!

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

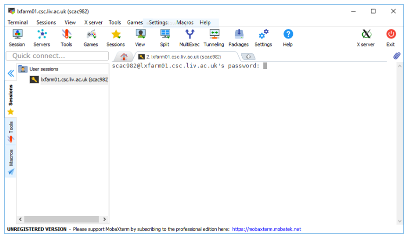
(a) MobaXterm


Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh Aws S3
Basic SSH settings
Advanced SSH settings Terminal settings Network settings Bookmark settings
Terminal font settings
OK Cancel

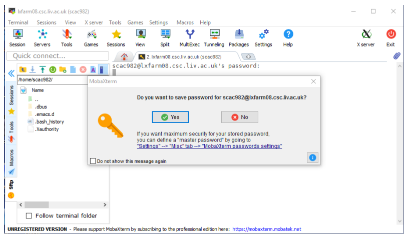
Remote host * lxfarm01.csc.liv.ac.u ☑ Specify username scac982 Port 22

☑ Backspace sends ^H ☐ Use Windows PATH Terminal type: xterm
☐ Log terminal output to: Paste delay: Auto
Terminal colors scheme: White background / Black text Customize
Syntax highlighting: Standard keywords (OK/warning/error/...) Customize

(b) Open an SSH Session


```
scac982@lxfarm01.csc.liv.ac.uk's password:
```

(c) Enter Password


```
scac982@lxfarm08.csc.liv.ac.uk's password:
```

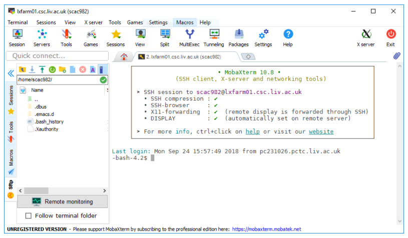
Do you want to save password for scac982@lxfarm08.csc.liv.ac.uk?

Yes No

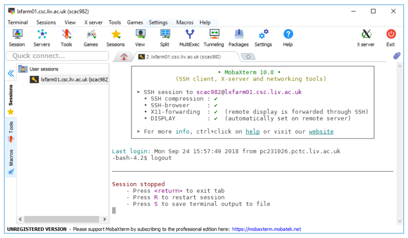
If you want maximum security for your stored password, you can define a "master password" by going to "Settings" --> "Misc" tab --> "MobaXterm passwords settings"

☐ Do not show this message again

(d) Save Password


```
                    • MobaXterm 10.8 •
        (SSH client, X-server and networking tools)

 ➤ SSH session to scac982@lxfarm01.csc.liv.ac.uk
   • SSH compression : ✔
   • SSH-browser     : ✔
   • X11-forwarding  : ✔  (remote display is forwarded through SSH)
   • DISPLAY         : ✔  (automatically set on remote server)

 ➤ For more info, ctrl+click on help or visit our website

Last login: Mon Sep 24 15:57:49 2018 from pc231026.pctc.liv.ac.uk
-bash-4.2$ 
```

(e) SSH Session Opened


```
Last login: Mon Sep 24 15:57:49 2018 from pc231026.pctc.liv.ac.uk
-bash-4.2$ logout

Session stopped
    - Press <return> to exit tab
    - Press R to restart session
    - Press S to save terminal output to file
```

(f) Logging Out

Figure 2: Using MobaXterm for SSH sessions

all characters of your password. You will then be asked whether you want MobaXterm to store your password so that next time you connect to this particular Linux server you do not need to enter your password again (Figure 2d). It is up to you whether you do so, it is the typical trade-off between convenience and security. Independent of the choice you make you will then probably be asked which Duo dual factor authentication option you want to use. Follow the instructions to get past this authentication step. You should now finally see a command prompt in the main window pane of MobaXterm (Figure 2e). Also, note that on the left to the main window pane you have a pane with a file browser showing the files in your home directory on the Linux server.

3 Using the Command Line Interface

As mentioned, in the main window pane of MobaXterm you see a *shell prompt*, typically, `bash-4.2$`. This prompt, which will be represented throughout this practical by ▶, is followed by a square or underline—the *cursor*. Anything you type on the keyboard will appear here. You can use the left and right arrow keys to move the cursor back and forth within the text you have entered, and this can be used to correct typing errors.

The *shell* is a command language interpreter that executes commands read from the standard input device, in this case your keyboard, or from a file. So, anything you type is intended as command to the operating system. When you have finished typing a command, press the `RETURN` key. This tells the shell to run the command you specified, relative to your current *working directory*, and to show any output produced by the command. It will then display another prompt, ready for your next command.

Here is an example that uses the `hostname` command to discover which Linux server you are using:

```
▶ hostname
lxfarm06.csc.liv.ac.uk
▶
```

Throughout the rest of this practical, we will not always show the command prompt in the examples. However, if you type in a command, press `RETURN`, and no new command prompt appears, then either you have not completed the command yet, say, there is an open string that you have not closed yet, or the command is still running, say, you have opened an editor 'in the foreground' and have not closed it yet. There are then three possibilities:

- If the command is not being executed yet, then you can try to properly complete the command and press `RETURN` again.
- If the command is being executed, but it does not terminate, then you can terminate the execution of the command by using the key combination `CTRL-C`. Note that if the command you are executing is an editor or software development environment, the text/program you were working on might be lost.
- If the command is being executed, but it does not terminate, then you can suspend the execution of the command by using the key combination `CTRL-Z`. You should then see a command prompt again. With the command `bg` you can move the just-suspended command to the background where it continues the execute, or, with the command `fg` bring it back to the foreground. The combination `CTRL-Z` followed by the command `bg` is the most reasonable cause of action if the command you are executing is a text editor, software development environment, or similar.

3.1 Basic Commands

The simplest commands are those concerned with viewing and manipulating files and folders. The following series of commands illustrate this for both Linux and Windows. Try working through the Linux examples and make sure you understand what each command does (note: '`l`' is the letter ℓ, not the number '`1`').

Linux	MS Windows	
`cd`	`M:`	Change your *working directory* to your *home directory*
`pwd`	`pwd`	Show the name of the *working directory*
`ls -l`	`dir`	Long listing of the files and folders in this directory
`mkdir COMP519`	`mkdir COMP519`	Create a new folder (directory)
`cd COMP519`	`cd COMP519`	Change the working directory to the named sub-folder
`ls`	`dir /w`	Short listing of files and folders in this directory
`cd ..`	`cd ..`	Change the working directory up a level (.. = "parent directory")
`cp /etc/php.ini t1`	`copy \etc\php.ini t1`	Make a copy of a file
`cat t1`	`type t1`	View the contents of the file `t1`
`more t1`	`more t1`	View the contents one page at a time (press the 'q' key to quit)
`mv t1 php.ini`	`rename t1 php.ini`	Rename a file

Linux	MS Windows	
`cp php.ini COMP519`	`copy php.ini COMP519`	Make a copy of a file in a different folder
`mv php.ini COMP519/t2`	`move php.ini COMP519\t2`	Move a file into another folder (and rename it)
`ls COMP519`	`dir COMP519`	List the contents of a folder
`rmdir COMP519`	`rmdir COMP519`	Try to delete a (non-empty) folder (which fails)
`mv COMP519/* .`	`move COMP519\* .`	Empty the folder (by moving the contents to the current directory)
`rmdir COMP519`	`rmdir COMP519`	Delete the (empty) folder
`rm php.ini t2`	`del php.ini t2`	Delete the test files

Be careful when using `rm`. Linux assumes you know what you are doing. So, if you ask it to delete a file, then it will be deleted. It will *not* simply be moved to the "`Wastebasket`" folder (which is what the file manager does). So it's typically not possible to "undelete" a file that has been deleted by mistake—when it's gone, it's gone. The same holds for Windows and `del`.

Note that Linux uses a filesystem with *case-sensitive* identifiers—the folder `COMP519` is different to one called `comp519`. Be very careful to type the names of files or folders *exactly* as they appear in the file manager, or in the output of `ls` or `dir`. Windows is more forgiving, and will ignore case differences.

Also be aware that by default the Windows File Manager will often hide the *filename extension*, that is, the last three or four characters after the final '.'. When using the command line, you typically need to give the full filename, *including* the filename extension.

3.2 Wildcards

The third-from-last command above (emptying the test directory before deleting it) illustrates a new idea—the use of *wildcards*.

Most of the commands above specify the name of a file or folder to work with. And as the last command shows, it's often possible to manipulate several files at the same time, by listing them on the command line. But if there are large number of files to manipulate, typing all of them out would be both time consuming and subject to errors. So both Linux and Windows provide a wildcard mechanism, to match the names of multiple files (or folders) at once.

Using the command

```
▶ mkdir ~/COMP519; cd ~/COMP519
▶ touch test1 test2 test3.txt test10
```

create a directory and four files in it. Next, try the following sequence of commands. Think about the results you see, and what these patterns might mean.

```
▶ ls test?
▶ ls *
▶ ls *.*
▶ ls *.txt
▶ ls test*
▶ ls *st?
```

3.3 File Downloads

We will later explore how to edit files. For that we will a sample Java file. You can download one using the following command:

```
▶ wget https://student.csc.liv.ac.uk/~uhustadt/COMP519/examples/HelloWorld.java
```

This should create a file `HelloWorld.java` in your `COMP519` directory.

3.4 Redirection

Another useful technique is *redirection*, where the output of a command can be saved to a file.

```
▶ ls test* > out1.txt
▶ cat out1.txt
```

This can be very useful when you come to test your programs. But note that any errors will still be displayed

```
▶ ls yourFile* > out2.txt
ls:  cannot access yourFile*:  No such file or directory
▶ cat out2.txt
```

Redirection works on both Linux and Windows command line terminals.

The main problem with this approach is that it only saves the *output* of the program, and not anything that is typed on the keyboard. Redirecting output to a file also means that you will not see any prompts or instructions that might be displayed by the program. In one sense, this does not really matter—as long as you know what information you need to supply, you can type this "blindly" and the program should run correctly.

But it would be better if you could see the output (including any prompts) and still have everything saved to a file. On Linux, this can be done using the `script` command:

```
▶ script -c "command" logFile
```

(where *`command`* is the command whose output you want to capture and *`logFile`* is the name of a file in which you want to store that output). Try experimenting with this, using some of the commands above, for example, try

```
▶ script -c "ls -l /etc/" out3.txt ▶ cp out3.txt  /www.txt
```

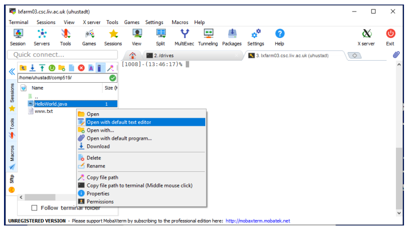
(Although it will only really become useful when you start running programs that expect input from the keyboard). Note that `script` is only available on the Linux systems—there is not an equivalent mechanism under Windows.

3.5 Pipes

A variation of output redirection is the idea of a "pipe" - using the output of one command as the input to another. Try the following in the Linux terminal window

```
▶ ls | sort -r
▶ cat out.txt | tail -5
▶ ls /lib | less
```
(*press the 'q' key to quit*)

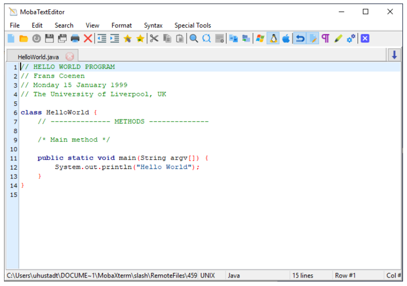
(The symbol '`|`' is to the left of the character 'z' on UK keyboards.) Compare the results of these pipeline commands, with the output generated by the first command in each pipeline on its own. Also, find out what the difference between `less` and `more` is.


lxfarm03.csc.liv.ac.uk (uhustadt)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. /drives
3. lxfarm03.csc.liv.ac.uk (uhustadt)
/home/uhustadt/comp519/
Name Size
HelloWorld.java 1
www.txt
Open
Open with default text editor
Open with...
Open with default program...
Download
Delete
Rename
Copy file path
Copy file path to terminal (Middle mouse click)
Properties
Permissions
Follow terminal folder
Sessions Tools Macros Sftp

```
[1008]-(13:46:17)%
```

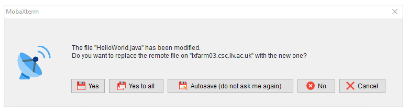
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

(a) Context Menu


MobaTextEditor
File Edit Search View Format Syntax Special Tools
HelloWorld.java
C:\Users\uhustadt\DOCUME~1\MobaXterm\slash\RemoteFiles\459 UNIX Java 15 lines Row #1 Col #

```
// HELLO WORLD PROGRAM
// Frans Coenen
// Monday 15 January 1999
// The University of Liverpool, UK

class HelloWorld {
    // -------------- METHODS --------------

    /* Main method */

    public static void main(String argv[]) {
        System.out.println("Hello World");
    }
}

```

(b) Editor



The file "HelloWorld.java" has been modified.
Do you want to replace the remote file on "lxfarm03.csc.liv.ac.uk" with the new one?

(c) Options for saving files

Figure 3: Using the MobaXterm text editor

Pipes are also available under Windows, though they are much less widely used. The last command would work in much the same way (`dir C:\Windows\System | more`). Windows does not typically include the same range of "filter" commands such as `sort`, `head` and `tail`.

Most (traditional) Linux commands are designed to process the contents of the specified files, or (if no files are listed) to work on "standard input" as part of just such a pipeline of commands.

3.6 Command Line History

When developing a computer program, you will typically find yourself repeating the same sequence of commands again and again—editing the file containig the source code, compiling this file, running the resulting program, editing the file to fix any errors, compiling the corrected file, running the program again, and so on. This means that you will end up typing the same commands over and over again.

Both Linux and Windows command shells include a *command history* mechanism, which remembers the previous commands that you have typed and allows you to recall them and run them again. Try using the up and down arrow keys to step through this list.

3.7 Filename Completion

There is one final function of the shell to mention, namely, *filename completion*.

Quite often you will have several different files in the same folder, with significantly different names. The Linux shell allows you to type the first few characters of a filename (sufficient to uniquely identify that file), and then hit the `TAB` key. This will automatically complete the name of the file, just as if you had typed it at the keyboard. This is extremely useful - particularly if you are using meaningful filenames (which can be relatively long), or if your typing is not particularly accurate!

If the prefix you have supplied is not unique, and there are two files that could possibly match, the shell will complete as much as it can, and leave you to complete it.

4 Editing Text Files using MobaXterm's Editor

There are various ways in which you can edit a file on the departmental Linux systems.

The preferred way is to use the built-in editor of MobaXterm. To use it, in the file browser pane of MobaXterm right-click on a file, say, the file `HelloWorld.java` in your `COMP519` directory.

This opens a context menu and you open the file in MobaXterm's editor by selecting the option "`Open with default text editor`" (Figure 3a). A copy of the file will then be retrieved and stored on your own PC. If you make changes to the file and then try save it, a dialogue window will pop up (Figure 3c) asking you whether the file should be transferred back to the Linux system that you are connected to. In almost all cases, "`Autosave (do not ask me again)`" is the right option to select.

MobaXterm's editor uses tabs to manage the files you have open. Right now there is one tab for the one file `HelloWorld.java` (Figure 3b). If you were to open another file, additional tabs would appear and allow you to easily switch from one file to the next.

For program development it is often helpful to have the lines of the code numbered, as error messages by a compiler or interpreter often indicate the line number at which an error has been found. This feature is enabled by default and you can see the line numbers on the left-hand of the editor window.

MobaXterm's editor uses *syntax highlighting* to distinguish language constructs from user-defined elements of a program.

Spend some time exploring all the features that MobaXterm's editor offers.

5 Other Text Editors

If you can't use MobaXterm and its editor, for instance, because you are using a Linux or MacOS PC, then you basically have three options:

(a) use a text editor or integrated programming environment on your own PC and use the `rsync` command to transfer files between your PC and the departmental Linux systems;

(b) use the remote editing facilities that some text editors, e.g., Notepad++, and integrated programming enviroments, e.g., Visual Studio Code, have;

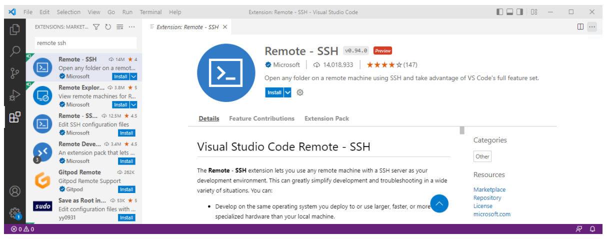
(c) use an editor that is executed on the departmental Linux system but displays its graphical user interface on your PC (this requires a fast internet connection).

In the following we only cover the second possibility with the Visual Studio Code. On the departmental Windows PCs, Visual Studio Code is already installed. For your own device you can download Visual Studio Code from

`https://code.visualstudio.com`

To enable remote editing in Visual Studio Code some extra steps need to be taken.

1. Open Visual Studio Code.
2. In the so-called activity bar on the far-left of the editor, click on the "`Extensions`" button.
3. To the right of the activity bar you should now see a search bar and below that a list of already installed extensions. Enter 'remote ssh' into the search bar (Figure 4a). In the list of search results, select "`Remote - SSH`" and click on "`Install`".
4. After successful installation there is a new green button in the bottom left corner of the editor. If you hover with the cursor over this button it says "`Open a Remote Window`" (Figure 4b). Click on that button.
5. Visual Studio Code will now go through a number of steps to establish a connection with a remote computer. At the top of the editor a dialogue box will appear that asks you to enter various bits of information and to select between various options.


Remote - SSH v0.94.0 Preview

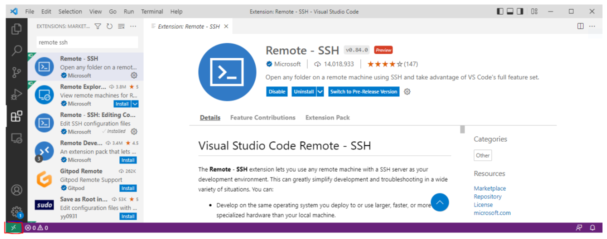
Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.

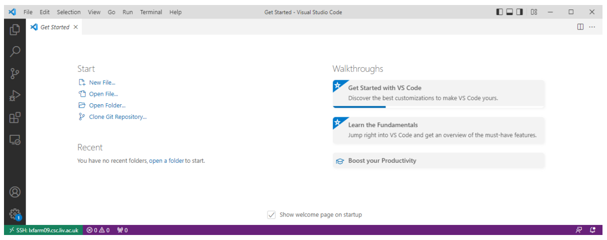
Visual Studio Code Remote - SSH

The **Remote - SSH** extension lets you use any remote machine with a SSH server as your development environment. This can greatly simplify development and troubleshooting in a wide variety of situations. You can:

- Develop on the same operating system you deploy to or use larger, faster, or more specialized hardware than your local machine.

(a) Package Installation


(b) Remote Host Connection

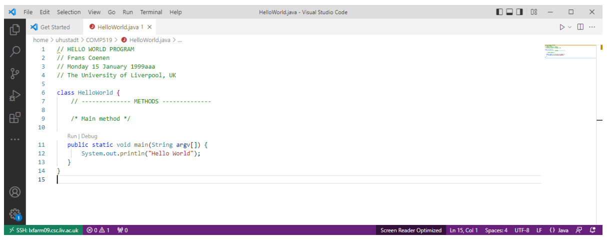


Discover the best customizations to make VS Code yours.

Jump right into VS Code and get an overview of the must-have features.

You have no recent folders, open a folder to start.

(c) Open a Remote File


```
// HELLO WORLD PROGRAM
// Frans Coenen
// Monday 15 January 1999aaa
// The University of Liverpool, UK

class HelloWorld {
    // -------------- METHODS --------------

    /* Main method */

    public static void main(String argv[]) {
        System.out.println("Hello World");
    }
}
```

(d) Editing

Figure 4: Visual Studio Code

6. In the dialogue box, start by selecting "`Connect to Host...`".
7. Next select "`Add New SSH Host...`".
8. Enter `sgxyz@lxfarm09.csc.liv.ac.uk`, where you replace `sgxyz` with your own MWS username.
9. In response to the instruction to "`Select SSH configuration file to update`", select the file `C:\Users\sgxyz\.ssh\config`, where again `sgxyz` is your own MWS username. The editor should then tell you that the configuration file has been successfully modified and it offers you the option to '`Connect`". Select that option.
10. A drop-down menu will appear that asks you to specific the operating system of that PC. Select "`Linux`".
11. A message will appear that tells you that `lxfarm09.csc.liv.ac.uk` has fingerprint "`SHA256...`" and ask you whether you want to continue or now. Select "`Continue`".
12. You will then be asked to enter your password. Enter your MWS password.
13. You should then be connected to the server and Visual Studio Code should look as shown in Figure 4c. In the main pane of the editor you should see a list operations that you can now perform, including "`New File...`", "`Open File...`", "`Open Folder...`" (Figure 4c).
14. Select "`Open File...`". This opens a very simple file selection dialogue. Use it to open the file `HelloWorld.java` in your `COMP519` directory. The file should then appear in the main pane of the editor as shown in Figure 4d.

Visual Studio Code uses tabs to manage several files at the same time. It uses *syntax highlighting* to indicate the structure of your code. The editor should already show line numbers next to your code. Likewise, it should highlight matching brackets: place the cursor on any curly bracket in the code; the background of the bracket should turn grey and so should the background of the corresponding opening or closing bracket matching.

If you make changes to a file, then you can save those changes using "`File→Save`" or via the key combination `CTRL-s`. A *tab* containing a file with unsaved changes will have a dot on the right-hand side of the title bar of the tab.

Add a comment to the file, then save it. Check on the Linux system that the file has indeed changed.

Once you feel that you understand how Visual Studio Code works, close it by clicking on the cross in the top right corner, via "`File`→`Exit`", or via the key combination `ALT-F4`.

6 Logging Out

You end the SSH session with one of the commands `exit` or `logout`. You will then see a message in the main window pane telling you that the session has been stopped (Figure 2f). If you do not see this message, then there are still commands running in the background and the connection has not properly been terminated. Make sure that any application with a graphical user interface, for example, text editor, has been closed, then click on the cross symbol in the top right corner of the tab above the main window pane. Typically, you will be shown a prompt informing you that one or more processes are still running and asking you whether you are sure that you want to close the tab, and thereby close the connection. If you confirm, the tab will be closed and the connection is properly terminated.

Note that the pane to the left of the main window pane changes. It now shows a list of previous sessions that you have established, in particular, you see the names the computers that you connected to and the user name you have used. You can reconnect to a particular computer by double-clicking on the corresponding entry in the list. If you have authorised MobaXterm to save the password that you have used, then the connection is reestablished without prompting you for a password. Give it a try, check that an SSH connection is indeed established, then log out again.