Introduction to Lab Practicals (Lab Intro 3)
Access Control, Synchronisation and Remote Access

1 Introduction

This practical is intended to familiarise you with the file access control mechanisms of Linux file systems. By default, your own files can only be accessed by yourself. It is important that you understand why this is so, how you can check who can access your files or the files of others, how you restrict or open access to files, and how you change what can be done with a file.

In addition, we will consider how you might best synchronise the files on the Linux filestore with the files you have at home on a PC or laptop and how you can remotely access the Linux servers.

This document can be found at

`http://intranet.csc.liv.ac.uk/~ullrich/COMP519/notes/labintro03.pdf`

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

2 File Permissions

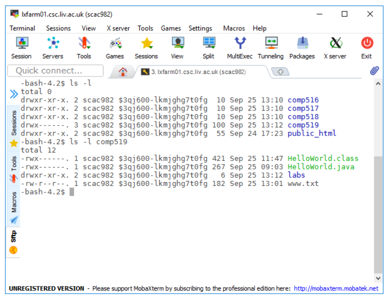
We will use the Department's Linux servers for most of the exercises that follow. Use MobaXterm to connect to one of the departmental Linux servers.

2.1 Overview

You have already seen that using `ls -l` you can get a '*long listing*' of the files in the current directory. If you have completed the exercises of Practical 1, then your home directory should contain a subdirectory called `comp519` and files `www.txt` and `HelloWorld.java` in that subdirectory. If you have used names different from these, then you need to adjust the exercises accordingly. Executing the commands

- ▶ `ls -l`
- ▶ `ls -l comp519`

should result in output resembling that shown in Figure 1 (remember: '`l`' is the letter ℓ, not the number '`1`').



```
-bash-4.2$ ls -l
total 0
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  10 Sep 25 13:10 comp516
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  10 Sep 25 13:10 comp517
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  10 Sep 25 13:10 comp518
drwx------. 3 scac982 $3qj600-lkmjghg7t0fg 100 Sep 25 13:12 comp519
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  55 Sep 24 17:23 public_html
-bash-4.2$ ls -l comp519
total 12
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 421 Sep 25 11:47 HelloWorld.class
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 267 Sep 25 09:03 HelloWorld.java
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-r--r--. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$
```

lxfarm01.csc.liv.ac.uk (scac982)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages X server Exit
Quick connect...
3. lxfarm01.csc.liv.ac.uk (scac982)
Sessions Tools Macros Sftp
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

Figure 1: Permissions

As you can see, the output of `ls -l` consists of several columns with the right-most column obviously containing the names of files and directories. The third column from the left, the one containing `scac982` in Figure 1, indicates the *owner*. For your own files, the corresponding column should show your username. The fourth column from the left, the one containing a string like `$3qj600-lkmjghg7t0fg` in Figure 1, indicates the *group*. Again, the group shown for your files will be different. A *group* is a collection of users/accounts and any user/account belons to one or more groups. The left-most column, a rather cryptic looking string of characters and dashes, shows the *permissions* (alternatively called *access rights*) for each of the files and directories.

All modern operating systems use *access control lists* to control who can do what with a particular file system object. To this end, each file system object is associated with such an access control list that contains *access control entries*, each of which gives an individual user or group the right to perform an operation such as reading, writing, or executing the file system object.

Linux, like any traditional UNIX operating system, recognises three classes of users with respect to operations on file system objects: *owner*, *group*, and *other*. Operations are categorised as *read* (r), *write* (w), and *execute* (x). Finally, the file system categorises file system objects into various kinds of objects, including *files* and *directories*. Having read (r), write (w), and execute (x) permission takes slightly different meaning for files versus directories:

Permission		For a file	For a directory
read	(r)	allowed to view file contents	allowed to view directory contents
write	(w)	allowed to write to file	allowed to remove or add new files to directory
execute	(x)	allowed to execute file	allowed to access files in the directory

Some clarification is in order regarding permissions for directories:

- To remove or add a file to a directory, or to otherwise modify an already existing file in a directory, you actually need both write (r) and execute/access (x) permission for the directory.
- Similarly, while read (r) permission for a directory, without execute (x) permission, does indeed allow you to see what files are in a directory, you will get an error message for each file telling you that you do not have access permission for the file. Furthermore, you will not be able to see any additional information for these files, e.g. their owners or the permissions for these files.

So, what does the information shown in Figure 1, and partially repeated below, tell us about permissions for the files and directories involved?

```
drwxr-xr-x.  comp519
-rw-r--r--.  www.txt
-rwx------.  HelloWorld.java
```

- The first character indicates the *type* of the file: '`d`' stands for 'directory', '`-`' for 'regular file', '`l`' for link, etc.

  So, `comp519` is a directory, while `www.txt` and `HelloWorld.java` are regular files.
- The next block of three characters indicates the permissions that the owner of the files has.

  So, '`rwx`' for `comp519` and `HelloWorld.java` means that the owner has read, write and execute permission. Remember, since `comp519` is a directory while `HelloWorld.java` is a regular file, these permissions have a slightly different meaning for each of these.

  For `www.txt`, '`rw-`' indicates that the user has read and write permission, but not execute permission.
- The next block of three characters indicates the permissions for members of the group.

  For `comp519`, the three characters are '`r-x`', so the group has read and execute permission for this directory but not write permission. For `www.txt`, the three characters are '`r--`', so the group has read permission. Finally, for `HelloWorld.java`, the three characters are '`---`', so the group has no rights with respect to this file.
- The next block of three characters indicates the permissions for other users.

  As it happens, for the three file system objects considered here, as the third block of three characters is identical to the second block of three characters, other users have the same permissions as the group.

2.2 Changing File Permissions: Symbolic Notation

To change the permissions for file system objects you use the `chmod` command. In is simplest form, `chmod` takes two arguments:

1. A description of the permissions that you want to set or a description of how you want to change the existing permissions. This description can be given in symbolic notation or numeric notation.

   (a) In symbolic notation, you need to specify for which user or group of users you want to change the permissions, how you want to change them, and which permissions you want to change:

Which user?		How to change the permission?		Which permission?	
u	user/owner	+	add this permission	r	read
g	group	-	remove this permission	w	write
o	other	=	set exactly this permission	x	execute
a	all (of the above)				

   For example, '`u-x`' indicates that you want to remove the execute permission from the owner, i.e. from yourself, while '`a+w`' means that you want to add write permission for all users, including yourself.

   Regarding the first and third group, you can pick more than one character. For example, '`ug+rw`' means that for both user/owner and group you want to add both read and write permission.

2. A list of file system objects for which you want to change the permissions, with a space separating the files system objects within the list.

Putting both together,

▶ `chmod u-x comp519/HelloWorld.java`

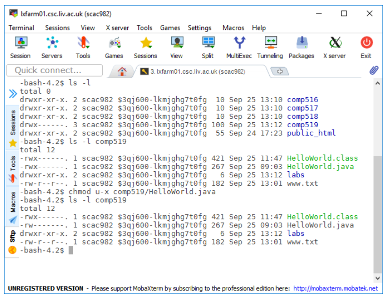
removes execute permission from the owner of the file `comp519/HelloWorld.java`.

Execute the command above and use

▶ `ls -l comp519`

to see what the effect of the command has been. The output from `ls -l comp519` should resemble that shown in Figure 2: The permissions for the owner of the file should have changed from '`rwx`' to '`rw-`'.

In Figure 1 we have seen that the permissions for the file `www.txt` give read permissions to all users (owner, group, and others). Such generous read permissions should be avoided if at all possible as you are required to keep your files secure from unauthorised access[1].


lxfarm01.csc.liv.ac.uk (scac982)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages X server Exit
Quick connect...
3. lxfarm01.csc.liv.ac.uk (scac982)
Sessions Tools Macros Sftp

```
-bash-4.2$ ls -l
total 0
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  10 Sep 25 13:10 comp516
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  10 Sep 25 13:10 comp517
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  10 Sep 25 13:10 comp518
drwx------. 3 scac982 $3qj600-lkmjghg7t0fg 100 Sep 25 13:12 comp519
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg  55 Sep 24 17:23 public_html
-bash-4.2$ ls -l comp519
total 12
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 421 Sep 25 11:47 HelloWorld.class
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 267 Sep 25 09:03 HelloWorld.java
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-r--r--. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$ chmod u-x comp519/HelloWorld.java
-bash-4.2$ ls -l comp519
total 12
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 421 Sep 25 11:47 HelloWorld.class
-rw-------. 1 scac982 $3qj600-lkmjghg7t0fg 267 Sep 25 09:03 HelloWorld.java
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-r--r--. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$
```

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

Figure 2: chmod (1)

[1]For example, if a file that is readable by all users would contain your work for a COMP519 assignment, then all students could potentially see what you have done and one of them could submit a copy as their own work. We would consider that to be collusion involving both parties, i.e. yourself and the other student, not plagiarism.

To restrict the read permissions for `www.txt` use the command

▶ `chmod og-r comp519/www.txt`

and see what the effect is using `ls -l comp519` (see Figure 3).

Let us see what the default permissions are for a newly created file. To create such a file you can use the `touch` command:

▶ `touch comp519/newFile.txt`

will create a new file called `newFile.txt` in the directory `comp519`. Execute this command and use

▶ `ls -l comp519`

to see what the permissions for `newFile.txt` are (see Figure 4).

As you can see, the file is readable and writable by the owner, i.e. yourself, and also readable by group and world. If you were to create a new file under Windows, then by default the execute permission would also be set.

So far we have not seen an example of an executable file. Let us create one.

Using your favourite editor, e.g. gedit, create a new file in your home directory, called `myFirstShellScript`, with the following content (also see Figure 5):

```
#!/bin/sh

echo "Hello World!"
```

Here, the first line indicates which interpreter should be used to execute the rest of the file, namely, the file `/bin/sh`, the system's default shell. That will be the GNU Bourne-Again SHell or bash for short.

Check with `ls -l` what the permissions are for the file `myFirstShellScript` once you have saved it. Not surprisingly, it is readable and writable by the owner, but nothing else.

Try to execute the file by using the command

▶ `./myFirstShellScript`

in the same directory where this file is stored. You should get an error message telling you that you do not have permission to execute the file. This is correct as so far nobody has execute permission for this file.

Let us change that using the command

▶ `chmod u+x myFirstShellScript`

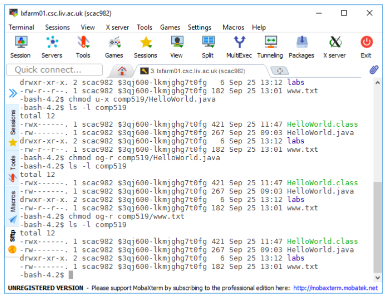
Then try to execute `myFirstShellScript` again. This time you will succeed and the script will produce the output

```
Hello World!
```

See Figure 6.

Now that you know how to change the permissions of a file system object, you can check whether what has been said on page 2 about permissions for directories is true. Do the following:

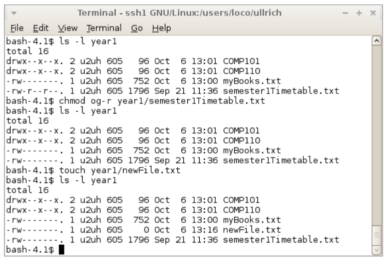
1. Make sure that the working directory is your home directory.


lxfarm01.csc.liv.ac.uk (scac982)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages X server Exit
Quick connect...
3. lxfarm01.csc.liv.ac.uk (scac982)

```
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-r--r--. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$ chmod u-x comp519/HelloWorld.java
-bash-4.2$ ls -l comp519
total 12
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 421 Sep 25 11:47 HelloWorld.class
-rw-------. 1 scac982 $3qj600-lkmjghg7t0fg 267 Sep 25 09:03 HelloWorld.java
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-r--r--. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$ chmod og-r comp519/HelloWorld.java
-bash-4.2$ ls -l comp519
total 12
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 421 Sep 25 11:47 HelloWorld.class
-rw-------. 1 scac982 $3qj600-lkmjghg7t0fg 267 Sep 25 09:03 HelloWorld.java
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-r--r--. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$ chmod og-r comp519/www.txt
-bash-4.2$ ls -l comp519
total 12
-rwx------. 1 scac982 $3qj600-lkmjghg7t0fg 421 Sep 25 11:47 HelloWorld.class
-rw-------. 1 scac982 $3qj600-lkmjghg7t0fg 267 Sep 25 09:03 HelloWorld.java
drwxr-xr-x. 2 scac982 $3qj600-lkmjghg7t0fg   6 Sep 25 13:12 labs
-rw-------. 1 scac982 $3qj600-lkmjghg7t0fg 182 Sep 25 13:01 www.txt
-bash-4.2$
```

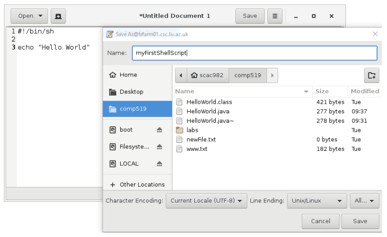
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

Figure 3: chmod (2)


Terminal - ssh1 GNU/Linux:/users/loco/ullrich
File Edit View Terminal Go Help

```
bash-4.1$ ls -l year1
total 16
drwx--x--x. 2 u2uh 605   96 Oct  6 13:01 COMP101
drwx--x--x. 2 u2uh 605   96 Oct  6 13:01 COMP110
-rw-------. 1 u2uh 605  752 Oct  6 13:00 myBooks.txt
-rw-r--r--. 1 u2uh 605 1796 Sep 21 11:36 semester1Timetable.txt
bash-4.1$ chmod og-r year1/semester1Timetable.txt
bash-4.1$ ls -l year1
total 16
drwx--x--x. 2 u2uh 605   96 Oct  6 13:01 COMP101
drwx--x--x. 2 u2uh 605   96 Oct  6 13:01 COMP110
-rw-------. 1 u2uh 605  752 Oct  6 13:00 myBooks.txt
-rw-------. 1 u2uh 605 1796 Sep 21 11:36 semester1Timetable.txt
bash-4.1$ touch year1/newFile.txt
bash-4.1$ ls -l year1
total 16
drwx--x--x. 2 u2uh 605   96 Oct  6 13:01 COMP101
drwx--x--x. 2 u2uh 605   96 Oct  6 13:01 COMP110
-rw-------. 1 u2uh 605  752 Oct  6 13:00 myBooks.txt
-rw-------. 1 u2uh 605    0 Oct  6 13:16 newFile.txt
-rw-------. 1 u2uh 605 1796 Sep 21 11:36 semester1Timetable.txt
bash-4.1$
```

Figure 4: chmod (3)



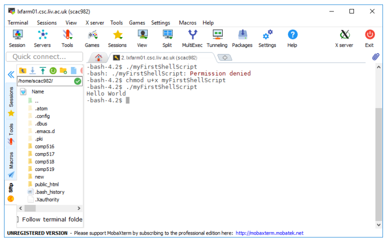
```
#!/bin/sh

echo "Hello World"
```

Name: myFirstShellScript

Name	Size	Modified
HelloWorld.class	421 bytes	Tue
HelloWorld.java	277 bytes	09:37
HelloWorld.java~	278 bytes	09:31
labs		Tue
newFile.txt	0 bytes	Tue
www.txt	182 bytes	Tue

Character Encoding: Current Locale (UTF-8) Line Ending: Unix/Linux All...

Figure 5: Shell script


lxfarm01.csc.liv.ac.uk (scac982)
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. lxfarm01.csc.liv.ac.uk (scac982)
Follow terminal folder

```
-bash-4.2$ ./myFirstShellScript
-bash: ./myFirstShellScript: Permission denied
-bash-4.2$ chmod u+x myFirstShellScript
-bash-4.2$ ./myFirstShellScript
Hello World
-bash-4.2$
```

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

Figure 6: Executing files

2. Change the permissions for directory `comp519` to '`r--------`', i.e. read permission for the owner only, no other permissions for the owner, no permissions for group or other.

   Then use `ls -l comp519` to see whether you can still obtain a long listing of the contents of the `comp519` directory.

3. Now give yourself execute permission for `comp519` in addition to read permission, i.e. set the permissions to '`r-x------`'.

   Use `ls -l comp519` again to see what the long listing of the contents of the `comp519` directory looks like.

4. Next, see whether write permission alone allows you to create a file in the the directory `comp519`. To do so, execute the commands

   ▶ `chmod u=w comp519`
   ▶ `touch comp519/testFile`

   The system should deny you the permission to create `testFile`.

5. Adding execute permission to the directory should solve this problem:

   ▶ `chmod u+x comp519`
   ▶ `touch comp519/testFile`

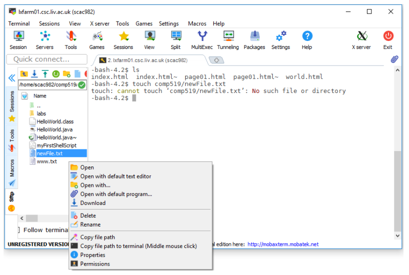
   This time creating the file `testFile` should succeed.

2.3 Changing File Permissions: MobaXterm

If your files happen to be on a system that can be accessed remotely using MobaXterm, as the files on the Linux filestore are, then MobaXterm provides you with an easy-to-use graphical interface to manipulate file permissions.

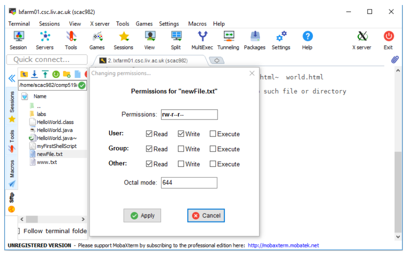
1. Open MobaXterm and connect to the Linux server.

2. In the file browser in the left pane of MobaXterm navigate to the directory `comp519`, right-click on the file `newFile.txt` that you have created in one of the previous exercises. MobaXterm shows you a context menu and among the operations that you can perform is one called 'Permissions' (Figure 7a). Select it.



```
-bash-4.2$ ls
index.html  index.html~  page01.html  page01.html~  world.html
-bash-4.2$ touch comp519/newFile.txt
touch: cannot touch 'comp519/newFile.txt': No such file or directory
-bash-4.2$
```

(a) Contex menu for files



Changing permissions...

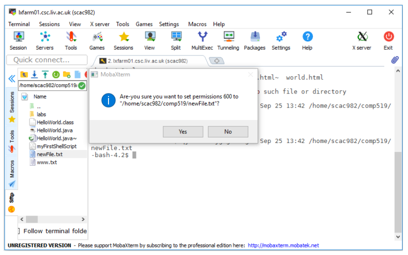
Permissions for "newFile.txt"

Permissions: rw-r--r--

User: [✓ Read] [✓ Write] Execute
Group: [✓ Read] Write Execute
Other: [✓ Read] Write Execute

Octal mode: 644

(b) File permissions menu for files



Are-you sure you want to set permissions 600 to "/home/scac982/comp519/newFile.txt"?

(c) Confirmation of change

Figure 7: Changing File Permissions using MobaXterm

3. You then see a table that shows you what the current file permissions for `newFile.txt` are (Figure 7b). You see that the file is readable by user, group and others (world-readable). By clicking on a particular cell you can add or remove permissions. Remove read permissions from group and others, then click 'Apply'.

4. MobaXterm will ask you for confirmation (Figure 7c). Confirm that you want the changes to be applied.

5. Check in the terminal window of MobaXterm that the file permissions have changed as intended by executing the command

   ```
   ▶ ls -l ~/comp519/newFile.txt
   ```

6. It is worth pointing out that MobaXterm determines the file permissions of each file in a folder when the folder is first opened in the file browser. So, if you change permissions on the command line then MobaXterm might not pick up those changes. Use the 'Refresh folder' function of the file browser to make sure that MobaXterm shows you up-to-date information.

3 Synchronisation

You may have your own PC and/or laptop that you want to use for your studies, in which case you will have to transfer files between the Linux filestore and your PC/laptop. You might use a USB pen drive, but students quite often forget these, or forget to copy all the necessary files onto the pen drive, or the pen drive gets lost or ceases to work. Also, a pen drive only allows you to copy files onto your Windows filestore and you still then need to get it onto your Linux filestore.

An alternative is the use of MobaXterm or `rsync` to transfer files between a PC or laptop at home or the Windows filestore and the Linux filestore.

3.1 MobaXterm

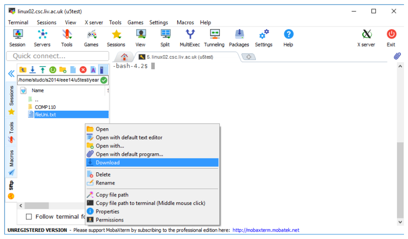
While SFTP is a reasonable way to transfer new directories to and from the Linux filestore, MobaXterm provides an easier way to transfer just a few files.

To illustrate the process, do the following. Using the Windows file explorer, create a new text file called `fileWindows.txt` in the `comp519` directory on the `M:` drive. Alternatively, create such a file on your personal PC/laptop. Also, using a MobaXterm SSH connection to a Linux server create a file `fileLinux.txt` as follows

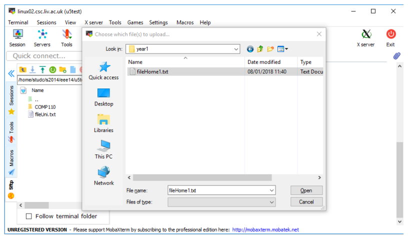
```
▶ ls -1 $HOME > ~/comp519/fileLinux.txt
```

We want to transfer `fileLinux.txt` from the Linux filestore to the directory `comp519/` on the Windows filestore. To do so, in the file browser in the left pane of MobXterm, find the directory `comp519`, double-click on it to open the directory in the file browser. Then right-click on the file `fileLinux.txt`. This opens a menu (Figure 8a) in which you select "Download". Another file browser opens which allows you to select the directory to which you want to download the file to. Use the file browser to select the directory `comp519` on the `M:` drive.

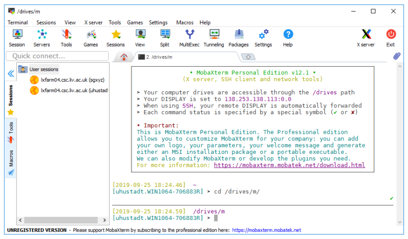
Now the reverse direction. To upload the file `fileWindows.txt` to the directory `~/comp519/` in the Linux filestore, first select this directory in the file browser of MobaXterm. Then click on the upward arrow above the file browser pane. Again, another file browser opens which allows you to select the file or files that you want to upload. Use the file browsers to select the file


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

(a) Download file



(b) Upload file



```
• MobaXterm Personal Edition v12.1 •
(X server, SSH client and network tools)

➤ Your computer drives are accessible through the /drives path
➤ Your DISPLAY is set to 138.253.138.113:0.0
➤ When using SSH, your remote DISPLAY is automatically forwarded
➤ Each command status is specified by a special symbol (✓ or ✗)

• Important:
This is MobaXterm Personal Edition. The Professional edition
allows you to customize MobaXterm for your company: you can add
your own logo, your parameters, your welcome message and generate
either an MSI installation package or a portable executable.
We can also modify MobaXterm or develop the plugins you need.
For more information: https://mobaxterm.mobatek.net/download.html

[2019-09-25 18:24.46]  ~
[uhustadt.WIN1064-706883R] ➤ cd /drives/m/
                                                                 ✓
[2019-09-25 18:24.59]  /drives/m
[uhustadt.WIN1064-706883R] ➤
```

(c) Local terminal

Figure 8: Transferring files using MobaXterm

`fileWindows.txt`, then click on "Open" (Figure 8b). MobaXterm will then transfer the file. Explore the other options that MobaXterm offers you: You can create directories on the remote system, delete files, and create new files.

3.2 rsync

`rsync` is a program that transfers files and directories from one location to another in a way that minimises the amount of data that needs to be transferred. The two locations do not necessarily have to be on the same computer. `rsync` has several advantages over other, simpler, file transfer programs: (i) `rsync` can restrict transfers to files that have been changed instead of blindly transferring all files and (ii) even where a file has to be transferred, `rsync` will by default only transfer parts of the file that have changed, not the complete file. This not only minimises the amount of data that needs to be transferred but also minimises the time the transfer takes.

The protocol that is typically used for the transfer of files between locations on different computers is SSH (Secure Shell), a cryptographic network protocol. SSH uses the client-server model and requires that on the computer that will act as server for the transfer of files is running a SSH daemon that accepts connections from a SSH client. This is the case for all Linux servers in the Department and the server `lxfarm.csc.liv.ac.uk` is specifically set up to be accessible from outside the Department.

MobaXterm does not only allow to establish remote connections using various protocols, but also provides a local terminal that supports a wide range of UNIX commands, including `rsync`. In MobaXterm, simply click on the plus symbol in the tab list or click on "`Session`" and then on "Shell" among the session types offered. The local terminal (Figure 8c) looks very similar to a remote SSH session but uses a different prompt that includes your username and the hostname (though this is customizable). Drives are accessible through the `/drives/` path, so your Windows filestore on the `M:` drive is accessible via `/drives/m`. In all likelihood, your files on the Windows filestore and the Linux filestore are currently in sync, so let us first create a difference.

```
▶ cd /drives/m/
▶ ls -l >> comp519/fileWindows.txt
```

Then, to synchronise the contents of the `comp519` directory on the Windows filestore and the `comp519` directory on the Linux filestore, and all their subdirectories, execute the following command in the MobaXterm local terminal:

```
▶ rsync -auvz -e ssh comp519/ 'sgxyz@lxfarm.csc.liv.ac.uk:comp519/'
sgxyz@lxfarm.csc.liv.ac.uk's password:
```

where you have to replace `sgxyz` with your own username. After you have entered your University

(MWS) password, modified files will be transferred, for example:

```
sending incremental file list
comp519/fileWindows.txt
sent 167 bytes received 25 bytes 20.21 bytes/sec
total size is 338 speedup is 1.70
```

The options `-auvz -e ssh` tell `rsync`:

- to transfer files in archive mode (`-a`), which ensures that symbolic links, devices, attributes, permissions, ownerships, etc are preserved in the transfer,
- to skip files that are newer on the receiver (`-u`),
- to show which files are transferred (`-v`),
- to compress the data that is to be transferred (`-z`) and
- to use ssh as network protocol (`-e ssh`).

Used in this way, `rsync` does not delete files on the receiver, i.e., your Linux filestore, that have been deleted on the Windows filestore. In order to delete those files on the receiver, the `--delete` option can be used:

```
▶ rsync --delete -auvz -e ssh comp519/ 'sgxyz@lxfarm.csc.liv.ac.uk:comp519/'
```

Obviously, this operation can easily lead to the loss of files. To see which files will be deleted on the receiver, without them being deleted right away, one can ask `rsync` to perform a dry run by adding the `-n` option to the command:

```
▶ rsync -n --delete -auvz -e ssh comp519/ 'sgxyz@lxfarm.csc.liv.ac.uk:comp519/'
sending incremental file list
deleting HelloWorld.java
sent 23 bytes received 18445 bytes 7387.20 bytes/sec
total size is 417446325 speedup is 22603.76 (DRY RUN)
```

Finally, if you have changed files on the Linux filestore and want to transfer those files back to the Windows filestore, you simply have to swap destination and source when using `rsync`:

```
▶ rsync -auvz -e ssh 'sgxyz@lxfarm.csc.liv.ac.uk:comp519/ comp519'
```

The manual page for `rsync` provides lots of additional information including a description of all the options of `rsync`.

Note also that if you are using MobaXterm at home, you can interact with the Linux server that you are connected to from a laptop or PC at home in the same way as you can from a Windows system on campus. So, you can remotely edit files or upload files that you have created at home and you can execute commands on the departmental Linux servers.

The best way of working remotely will depend on the speed and reliability of your internet connection. If your connection is fast enough, then you might be able to open an editor with graphical user interface on `lxfarm.csc.liv.ac.uk` and edit files directly on our systems. With a slower connection you might want to use a command line editor instead. Note that you can have more than one connection / terminal open at the same time. So, you can have one terminal with a command line editor and other terminal to execute commands such as compiling and running programs. The third alternative would be to just use the connection to transfer files that you create on your system.

apropos	aspell	bash	cat	cd	chsh	clear	cp	date
diff	diffpp	egrep	enscript	find	history	kill	ln	mv
quota	rename	sort	ssh	tail	tar	tcsh	time	top
uptime	vim	zile	zip					

Table 1: Useful Linux commands

4 Further Study

This concludes our short introduction to Windows and Linux. There is obviously still a lot to learn, even about the commands that you have used in these practicals.

To explore the possibilities of Linux further you can first of all take advantage of the *manual pages* that are available for almost every Linux command. In a terminal, manual pages can be read using the command `man`. In its simplest form `man` takes a single argument, namely, the name of the manual page that you would like to read; the name of the manual page is typically identical to that of the corresponding command or program. So, for example,

```
▶ man chmod
```

will show you the manual page for the command `chmod`. To display this manual page, `man` uses `less -s` by default. So, you can use the key bindings for `less` to navigate through a manual page. Of course, you can use

```
▶ man less
```

to learn what those key bindings are. Table 1 lists a number of other commands that you should explore by reading their manual pages and experimenting with them.

To learn more about the Linux command line and shell scripting you can refer to the following e-book available from the library:

> W. E. Shotts: The Linux command line: A complete introduction.
> No Starch Press, 2012. http://library.liv.ac.uk/record=b2626812~S8
> [last modification 16 December 2012, accessed 02 October 2015]

Longer term you should aim to be able to install and set up an XAMPP stack (Apache HTTP Server, MariaDB or MySQL, PHP, Perl) on a Linux Virtual Machine under Windows 10 as well as directly on a Windows 10 system. VirtualBox, a free and open-source hypervisor for x86 computers, is installed on our Windows 10 PCs and ready-to-use virtual machines can be found at https://virtualboxes.org/. Alternatives are Hyper-V and VMware Workstation Player, each with their advantages and disadvantages. All this will not be subject of any of the modules of your degree programme, but you should try to acquire the necessary knowledge and skills through self-study.