CHAPTER
4

INTRODUCTION TO UNIX SYSTEM

In this chapter you will learn about

- ❖ Some rules on computer usage.
- ❖ How to log in to and out of NUSNET-IV.
- ❖ How to install new applications via Windows Shopping.
- ❖ How to log in to a UNIX machine, change your password and log out.
- ❖ How to use basic UNIX commands to organise and move around your directories, to set access permission of files, to manipulate and browse files, and to print files.
- ❖ How to use the pico editor to create and edit text files.
- ❖ How to customise your UNIX environment.

4.1 Introduction

Being a computing student, most of your academic activities will revolve around the computers. The School of Computing (SoC) boasts a broad range of computing facilities that cater to the diverse needs of the undergraduates and postgraduates, as well as the administrative, research and academic staff. One place to find information on this is to follow the link to 'Computing Facilities' in the School's web site (after which, you should follow up with a personal visit to the various laboratories). The URL is https://www.comp.nus.edu.sg/cf/ (only accessible to SoC staff and students). The rate at which computing facilities are changed and upgraded is usually too fast for a book like this, or even the web pages sometimes, to catch up.

They say the only constant is change. Around here, we have constant changes.

This chapter attempts to give a very quick preview of the computing environment that you will work in from the very first day. It covers selected services that you cannot afford to live without for very long. What you learn here is merely the tip of the tip of the iceberg (no typographical error here). As forewarned in the beginning of this book, most of this iceberg belongs to the kind of knowledge that will not be taught formally, but only awaits your exploration and discovery.

The rest of this chapter will bring you through a tour on each of the following topics: logging onto the system, introduction to the **UNIX** operating system, and introduction to the **pico** editor. As the coverage skims over the very basic, students with prior knowledge may skip certain sections, while students who need more help will have to do further reading on their own.

4.2 Rules On Computer Usage

All computer users have to abide by the rules and regulations governing the use of computer facilities within the university. The detailed regulations came together with the agreement you signed when you got your computer account document, and you should read them over very carefully. Below are just some misadventures that could bring disciplinary actions against you, with penalty ranging from account suspension to barring from examinations to expulsion from university.

- Sharing your password with others, allowing others to use your computer account, or assessing another person's account.
- Playing games during office hours.
- Misbehaviour and eating in programming laboratories.
- Sending mass mails or chain mails, or advertising programs or services locally or overseas.
- Running processes that are considered redundant or destructive.
- Sending or posting any kind of vulgar, distasteful, derogatory, or prejudicial mails or messages.

Every NUS student is given an NUSNET account to allow access to the various resources and services in the NUSNET network, provided by the NUS Computer Centre. Do read up the policies on this website:

http://www.nus.edu.sg/comcen/main/getstart.htm

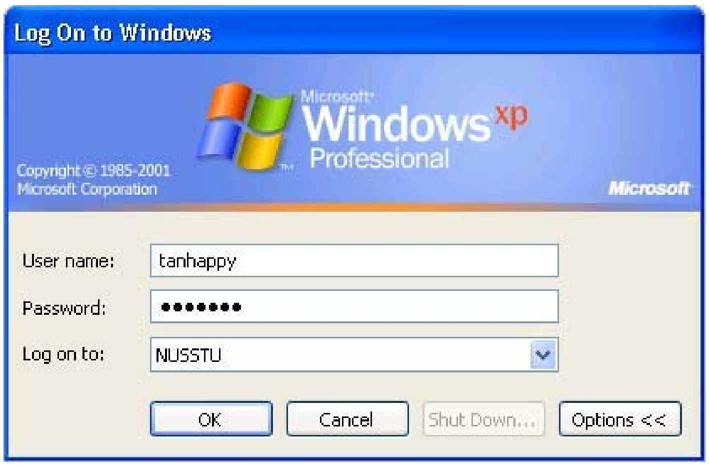
Besides the NUSNET account, every SoC student is also given an SoC UNIX account, provided by SoC. You should also read up the rules, regulations and policies on the School's computing facilities and services on this website:

https://www.comp.nus.edu.sg/cf/rules/

4.3 Logging In To NUSNET-IV

NUSNET-IV (http://www.nusnet.nus.edu.sg) is the NUS campus-wide intranet. Services include Microsoft Exchange mail system, Microsoft Office, and other Windows applications.

1. In the various programming labs where Windows XP is installed, you can make use of the PCs to log into NUSNET-IV.
2. At the screen, you will see the following display.


Log On to Windows
Microsoft Windows xp Professional
Copyright © 1985-2001 Microsoft Corporation
Microsoft
OK
Cancel
Shut Down...
Options <<

User name: tanhappy
Password: •••••••
Log on to: NUSSTU

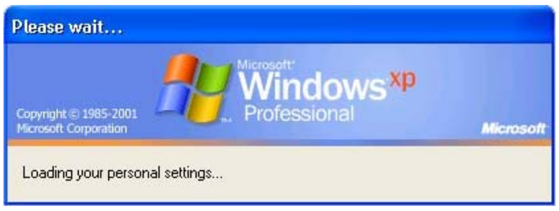
If you do not see the above screen, you can press Ctrl-Alt-Delete for the login screen to appear. Look for the following keys

- Ctrl (at the lower left corner of your keyboard)
- Alt (at both side of your spacebar)
- Delete (one of the 6 keys above the arrow/direction keys)

Press the 3 keys at the same time.

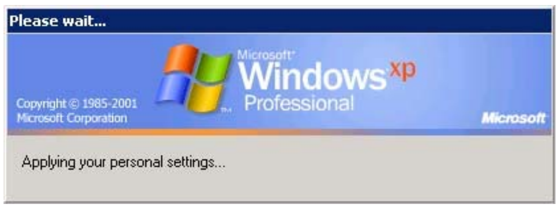
3. In the 'Name' field, type in the *username* (also known as *user-id*) that has been assigned to you at your matriculation. It should be of the form isc*xxxxx*, where *xxxxx* are digits. Example: isc40123.
4. From the 'Log on To' drop-down list, select the domain **NUSSTU**.
5. Type in your *password* in the 'Password' field.
6. Once done, click 'Ok' to log in.
7. Your username and password will be authenticated.

8. If it is successful, the screen will change, and you will see the following:


Please wait...
Microsoft Windows xp Professional
Copyright © 1985-2001 Microsoft Corporation
Microsoft
Loading your personal settings...

It is loading your customized settings from the NUSNET server. Wait patiently.

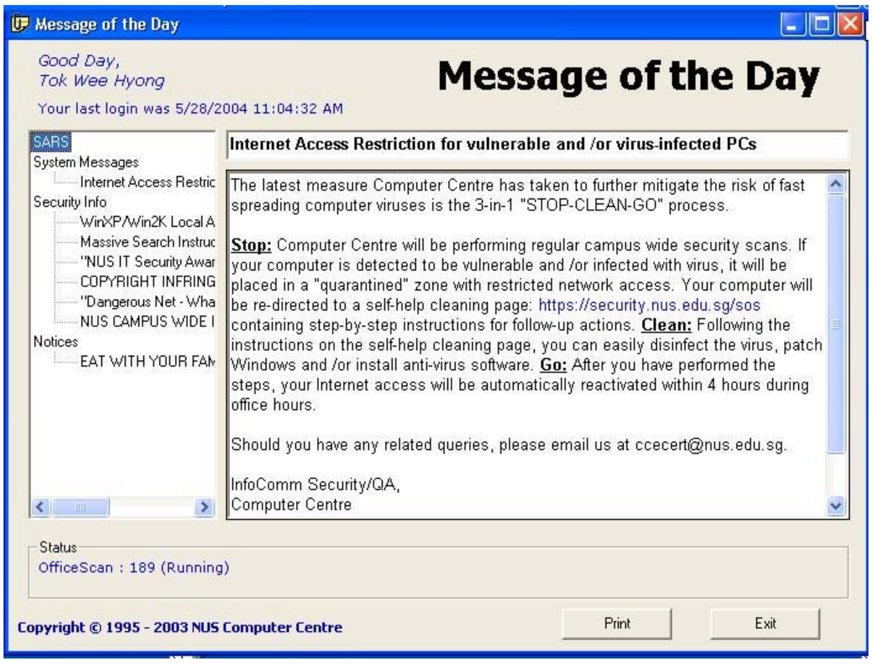
9. After a while, the screen will change to the following:


Please wait...
Microsoft Windows xp Professional
Copyright © 1985-2001 Microsoft Corporation
Microsoft
Applying your personal settings...

Keep waiting.

10. Once the screen clears, you are now logged on to NUSNET-IV.

11. After a while, you will also see the 'Message of the Day' pop-up box appearing. This provides you with the latest information from Computer Centre.



Good Day,
Tok Wee Hyong
Your last login was 5/28/2004 11:04:32 AM

Message of the Day

Internet Access Restriction for vulnerable and /or virus-infected PCs

The latest measure Computer Centre has taken to further mitigate the risk of fast spreading computer viruses is the 3-in-1 "STOP-CLEAN-GO" process.

Stop: Computer Centre will be performing regular campus wide security scans. If your computer is detected to be vulnerable and /or infected with virus, it will be placed in a "quarantined" zone with restricted network access. Your computer will be re-directed to a self-help cleaning page: https://security.nus.edu.sg/sos containing step-by-step instructions for follow-up actions. **Clean:** Following the instructions on the self-help cleaning page, you can easily disinfect the virus, patch Windows and /or install anti-virus software. **Go:** After you have performed the steps, your Internet access will be automatically reactivated within 4 hours during office hours.

Should you have any related queries, please email us at ccecert@nus.edu.sg.

InfoComm Security/QA,
Computer Centre

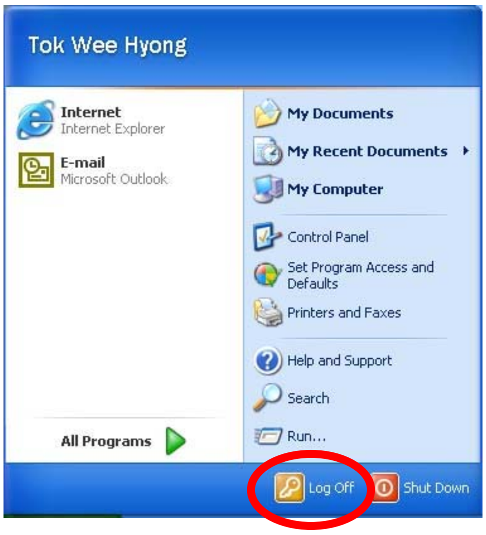
Status
OfficeScan : 189 (Running)

Copyright © 1995 - 2003 NUS Computer Centre

4.4 Logging Out Of NUSNET-IV

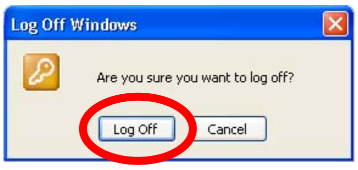
To log out of NUSNET-IV, follow these steps.

1. Click on the 'Start' button at the lower left-hand corner of your screen. You will see the following screen:



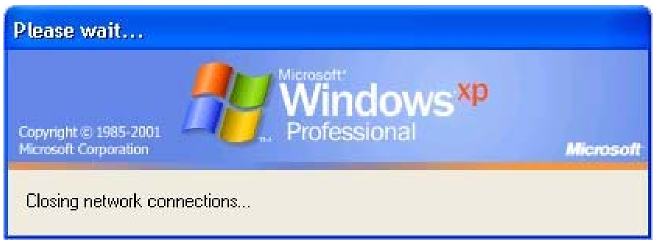
Click on 'Log Off' as shown above.

2. A pop-up window for confirmation will appear. Click 'Log Off'.



Are you sure you want to log off?

3. The following screen will appear.


Please wait...
Microsoft Windows xp Professional
Copyright © 1985-2001 Microsoft Corporation
Microsoft

Closing network connections...

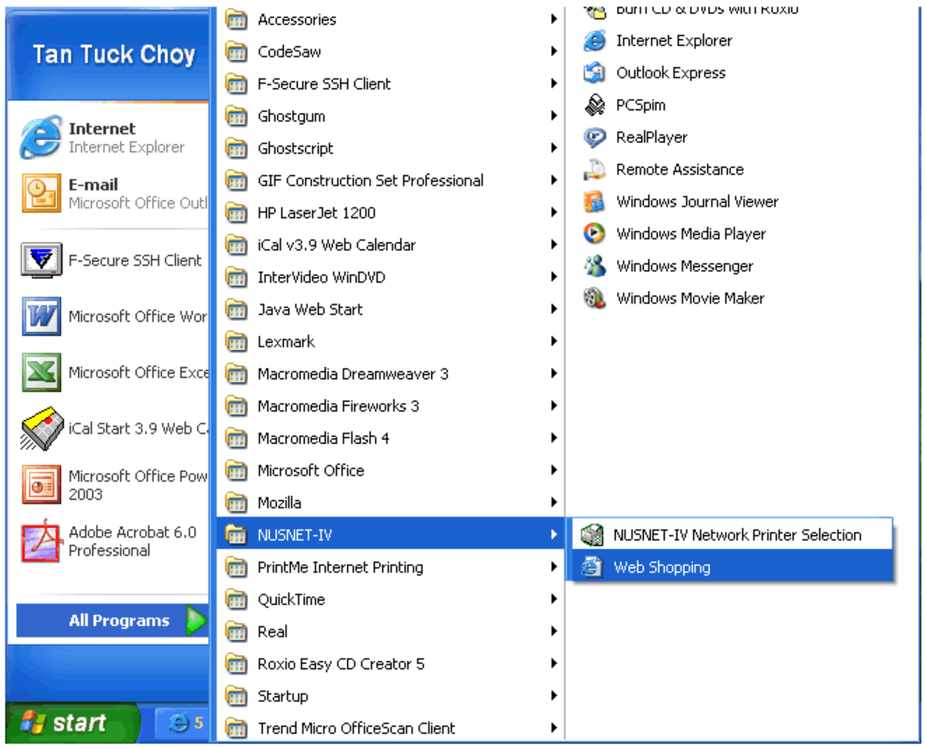
Wait patiently.

4. After a while, you will get back to the login screen.

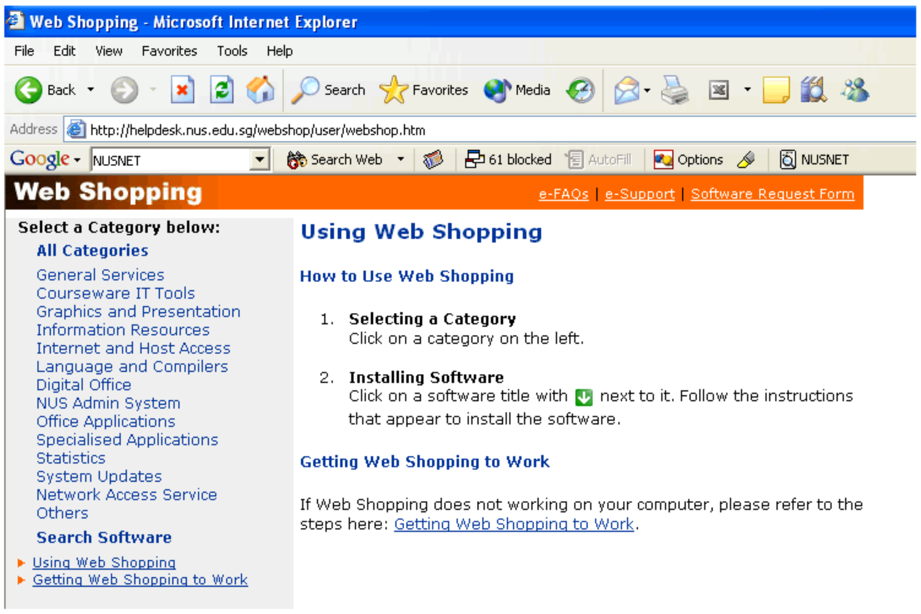
4.5 Installing New Applications Via Web Shopping

In the course of your study, you may need certain software. The university provides the web shopping service to allow you to shop for the software at your finger tip!

1. Either enter the following URL http://helpdesk.nus.edu.sg/webshop/ in your web browser, or click on the 'Start' button at the lower left-hand corner of the screen, followed by selecting 'All Programs' → 'NUSNET-IV' → 'Web Shopping' as shown below.



2. You will see this screen:



http://helpdesk.nus.edu.sg/webshop/user/webshop.htm

Web Shopping e-FAQs | e-Support | Software Request Form

Select a Category below:

All Categories

General Services
Courseware IT Tools
Graphics and Presentation
Information Resources
Internet and Host Access
Language and Compilers
Digital Office
NUS Admin System
Office Applications
Specialised Applications
Statistics
System Updates
Network Access Service
Others

Search Software

- Using Web Shopping
- Getting Web Shopping to Work

Using Web Shopping

How to Use Web Shopping

1. **Selecting a Category**
   Click on a category on the left.
2. **Installing Software**
   Click on a software title with next to it. Follow the instructions that appear to install the software.

Getting Web Shopping to Work

If Web Shopping does not working on your computer, please refer to the steps here: Getting Web Shopping to Work.

Don't worry about paying, they aren't billing you.

Follow the instructions on the web page to install the software you need.

4.6 Logging In To A UNIX Machine

The School of Computing has a wide range of computing equipment to support staff and students: UNIX servers, personal computers, network and teaching laboratories. For more information, please refer to http://www.comp.nus.edu.sg/cf-pub/our_facilities.html.

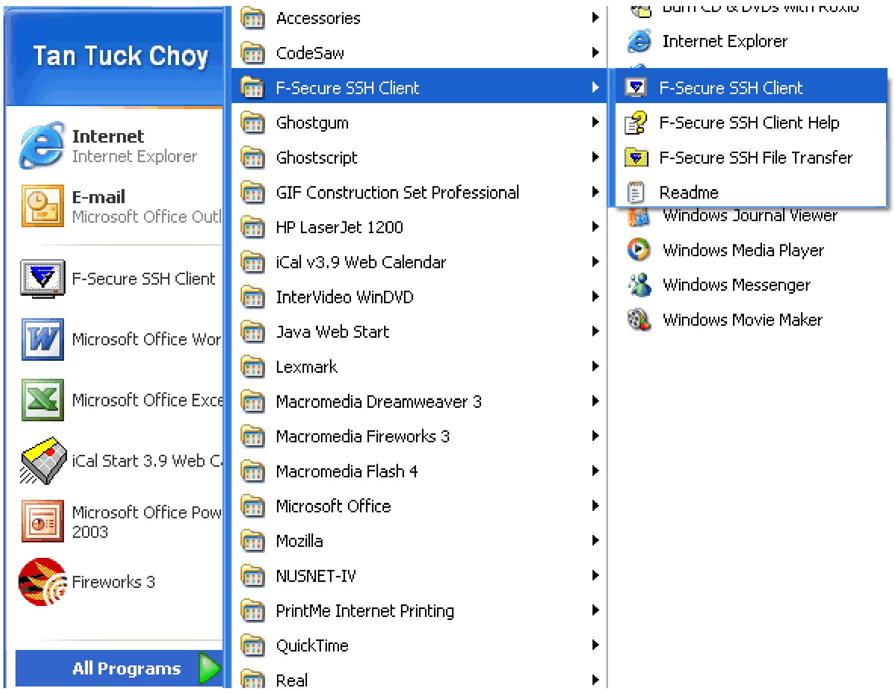
UNIX is a multi-tasking, multi-user operating system. In SoC, four Sunfire 4800 serves as the file-servers and SoC login machines. All UNIX computers and workstations in the School use the same password and login procedure.

You should be familiar with UNIX as the system provides web, print, news and email services. (Recall that every SoC student has an SoC UNIX account besides the NUSNET computer account). You will also be required to work on UNIX for many of your modules.

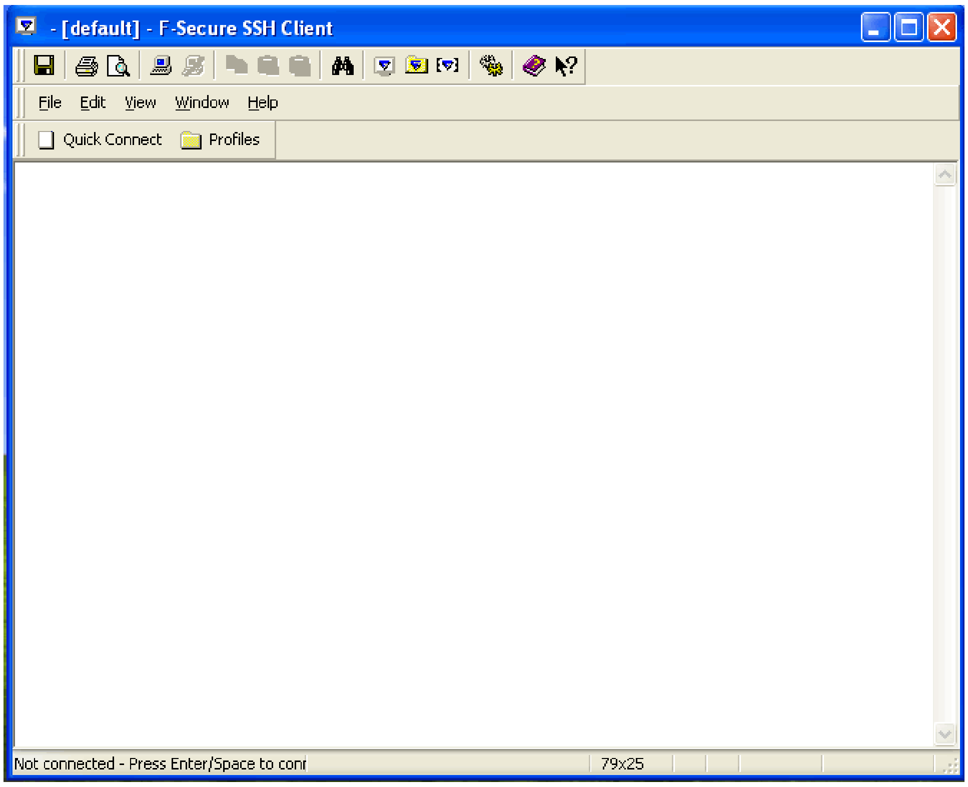
In the following examples, system messages and responses are printed in a constant-width typeface, user responses are shown in **bold** and comments are in *italics*. Optional arguments in command lines are enclosed in brackets [ ]. Control keys are preceded with the caret symbol ^, as in ^c for Ctrl-c.

After you have logged on to NUSNET, and if you need to connect to one of the UNIX servers in the school, do the following:

1. Click on the ‘Start’ button at the lower left-hand corner of the screen, followed by selecting ‘All Programs’ → ‘F-Secure SSH Client’ → ‘F-Secure SSH Client’ as shown below.

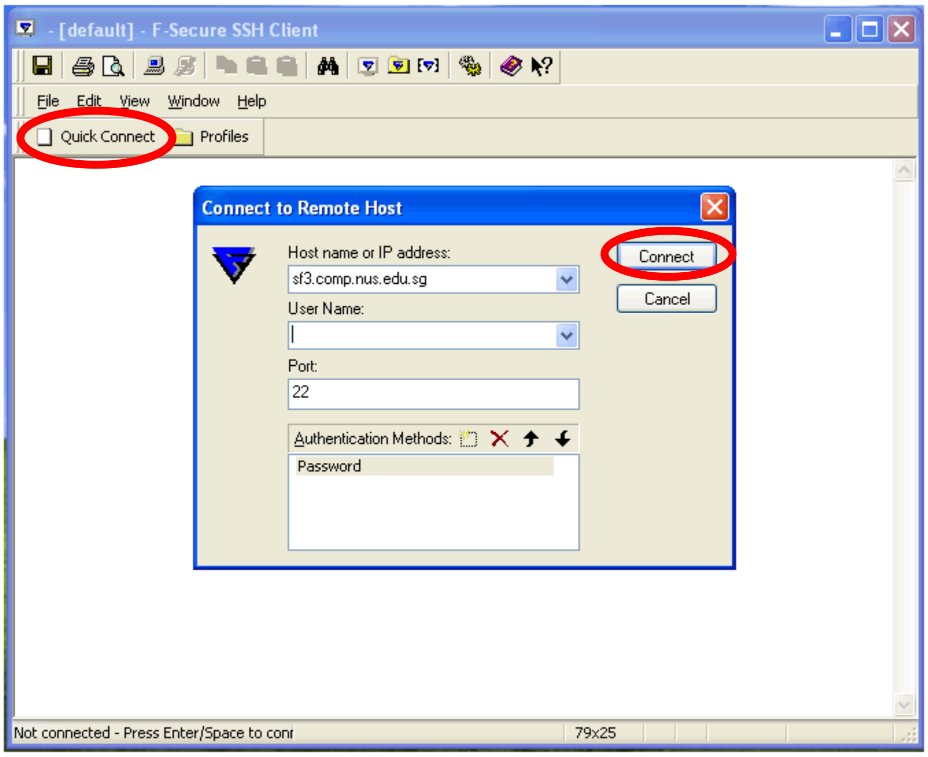


2. You will see this screen:



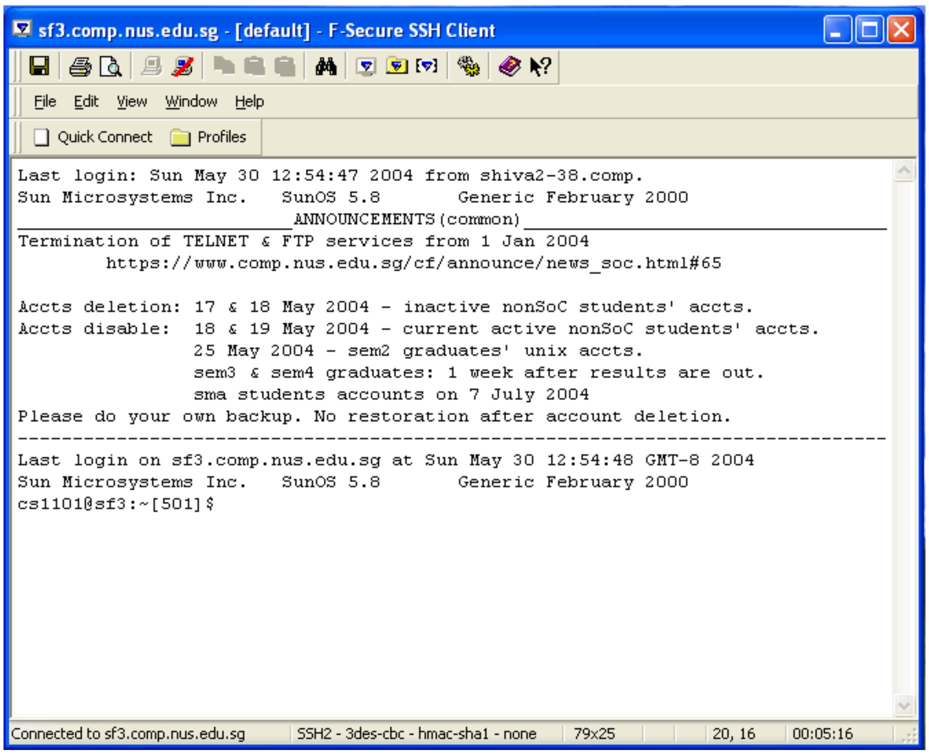
Not connected - Press Enter/Space to conr 79x25

3. Click on ‘Quick Connect’. A pop-up window will appear. Enter **sf3.comp.nus.edu.sg** for ‘Host name or IP address’, your SoC UNIX username (isc*xxxxx*) for ‘User Name’, and **22** for ‘Port’ as shown below. Click on ‘Connect’ in the pop-up window.



4. Another pop-up window will appear asking you for your password. Enter the password given to you and click ‘OK’ in the pop-up window.

5. You will be greeted by a screen below (actual contents will vary), showing that you have logged into sf3 (one of the UNIX systems). Viola!


sf3.comp.nus.edu.sg - [default] - F-Secure SSH Client
File Edit View Window Help
Quick Connect
Profiles
Connected to sf3.comp.nus.edu.sg
SSH2 - 3des-cbc - hmac-sha1 - none
79x25
20, 16
00:05:16

```
Last login: Sun May 30 12:54:47 2004 from shiva2-38.comp.
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
_____________________________ANNOUNCEMENTS(common)_________________________________
Termination of TELNET & FTP services from 1 Jan 2004
        https://www.comp.nus.edu.sg/cf/announce/news_soc.html#65

Accts deletion: 17 & 18 May 2004 - inactive nonSoC students' accts.
Accts disable:  18 & 19 May 2004 - current active nonSoC students' accts.
                25 May 2004 - sem2 graduates' unix accts.
                sem3 & sem4 graduates: 1 week after results are out.
                sma students accounts on 7 July 2004
Please do your own backup. No restoration after account deletion.
--------------------------------------------------------------------------------
Last login on sf3.comp.nus.edu.sg at Sun May 30 12:54:48 GMT-8 2004
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
cs1101@sf3:~[501]$
```

6. If the login is unsuccessful, check that you have entered your username as well as password in the exact way as they appear on your computer account document. For SoC student accounts, the username consists of the first eight letters of your name (if your name exceeds eight letters) in lower-case. Before you type anything, check that the Caps Lock key is off, as UNIX is case-sensitive.

Upon successful login, you will see some greeting messages and the MOTD (messages of the day, posted by the system administration). You are then placed in your *home directory*. You will see this prompt:

```
    garfield@sf3:~[12]$
```

garfield is your username (user-id), **sf3** is the system, **~** denotes your home directory, and [12] is the command serial number. The above is the *primary prompt*. (There is a *secondary prompt*, which is the > symbol by default.) For simplicity, we usually refer to it as the $-prompt. The $-prompt is a signal indicating to you that the system is ready to accept your input.

If you do not see the prompt, it could mean one of the following things:

1. You have just issued a command that requires some time to complete, and the system is currently executing it. Just wait for its completion.
2. You have done something not quite correct. Enter control-c (interrupt) to return to the system prompt.
3. The system is having trouble. Check with your senior or the SoC Technical Helpdesk at S15 ground floor.

4.7 Changing Your Password

Since this is your very first login, you should change your password to one that you can remember better. There are some rules that you need to observe in choosing a valid password. These rules can be read from **man passwd**.

1. It must have at least 6 characters, and only the first 8 characters are significant.
2. It must contain at least 2 alphabetic characters and at least one numeric or special character. Alphabetic characters can be upper-case or lower-case letters.
3. It must be different from your user-id and any reverse or circular shift of your user-id.
4. It must be different from the old password by at least 3 characters.

Apart from the above rules, you are to choose a password that is not easy for others to guess. Hence, your NRIC number, birthday, pet's name, are all bad choices. Refer to https://www.comp.nus.edu.sg/cf/unix/passwd.html for more tips.

The command to change your password is **yppasswd**.

```
garfield@sf3:~[12]$ yppasswd
Getting the user entry for garfield.
Changing password entry for garfield on yp server
Old password:
New password:
Re-enter new password:
```

The system prompts you for the old (current) password, then the new password twice, for verification purpose. Remember your new password. If you forget your password, you would need to approach the SoC Technical Helpdesk on the ground floor of S15 to request for a new password.

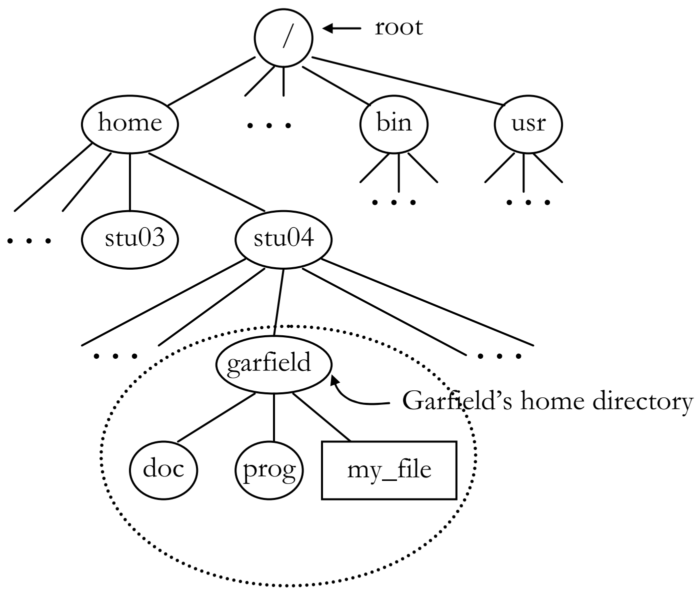
4.8 Logging Out Of A UNIX Machine

1. To log out, at the UNIX prompt, type **exit** or **logout**. Do not type **ex** to log out as 'ex' is an editor! If you do that, you will invoke the editor and encounter the editor's prompt, which is : (colon) . Just enter 'q' to quit from the editor. Then enter **exit** to log out gracefully.
2. You may close the SSH window after you have logged out successfully.

4.9 The UNIX File System

In UNIX, *file* is a generic term that includes the *regular file* (data file which can be in text or binary format), the *directory file* (commonly known as *directory* or *folder* in other operating systems), and *special file*. It is important to understand the file system in UNIX. UNIX organises all files in a hierarchy, as shown below.

Computer scientists are weird. They draw trees with roots on top, and leaves at the bottom.



In the diagram, the circle represents a directory, and the rectangle a regular file. Only a directory may contain directories (which are also called subdirectories; the inverse relation is the parent directory) as well as files within it. A (regular) file cannot store other files or directories. Imagine a directory or folder as a room in the house. The room may contain your files, as well as smaller rooms, which in turn may contain other files and even smaller rooms. Of course, this is just an analogy. A subdirectory is no 'smaller' than its parent directory in the physical size sense, as it may contain more files than its parent. However, a subdirectory is 'below' its parent in the hierarchical structure, and this affects the way we identify and address it.

As UNIX is a multi-user system, no single user owns the entire file system. The 'house' starts at the root, denoted by / (this is the slash, not the backslash \ in DOS), and it contains many system and administrative subdirectories. Your home directory lies somewhere (possibly a few levels) below the root. In the diagram above, Garfield's home directory is at the address /home/stu04/garfield. Such an address is termed a *pathname*. In a pathname, a leading slash is recognised as the root directory, while the other slashes are merely delimiters.

Garfield may create his (or its?) own files and directories in his home directory and directories below it. Files and directories that lie outside his territory (his territory is enclosed within the dotted circle) do not belong to him and hence he has no right of access to them, unless the owner of these files grants him the access.

4.10 Basic UNIX Commands

You are about to learn some commands that allow you to move around in the system and manipulate your files. For the DOS-acquainted, below is a list of some DOS commands and their (near) equivalents in UNIX. Note that unlike DOS, UNIX is case sensitive.

DOS	**UNIX**	**Purpose** (in UNIX context)
dir	ls –l	List files in the current directory.
dir/w	ls	List files in the current directory.
dir/a	ls –a	List all files, including hidden files, in the current directory. Hidden files are files whose names begin with a period (e.g.: '.plan', '.profile').
cd	cd	Change directory. 'cd' with no arguments brings you right to your home directory.
del	rm	Remove file.
rd	rmdir	Remove directory.
md	mkdir	Make new directory.
rename	mv	Rename file.
move	mv	Move file.
copy	cp	Copy file.
type	cat	Catenate file (display file's content).

4.10.1 Command Format

Before we go through the commands, let us look at the typical format of a UNIX command:

<command> <options> <arguments>

<command>	is the name of the command;
<options>	is a list of options; and
<arguments>	is a list of arguments.

There must be a delimiter between the command and its options (if present), and between the options and the arguments (if present). The delimiter is a white space that consists of at least a space character or a tab.

Command Options

Options are normally preceded by a minus (–) character. However, in some cases, the plus (+) character and other characters are used. There must not be any space between the minus character and the option code. For example, **who –u** is valid, but these are all invalid:

```
who–u
who – u
w h o –u
```

Try out **who** and **who –u** now. What do you see? Are the outputs different? Options are used to modify the default behaviour of the basic command, resulting in the command either working in a slightly different flavour, or generating a slightly different output. The basic function of the command remains. The command **who**, with or without option, still displays a list of the users, but shows more information when you specify an option like **–u**.

Some commands, like **pwd**, do not support any option, whereas others, like **ls**, have as many as up to 20 options. Do not worry about having to remember all the options. In actual practice, only a handful among all the options is commonly used. You will know which are the more popular options as you go along.

Specifying Multiple Options

Sometimes you may want to specify more than one option for a command. Most of the times, you can use one of these methods:

1. Put the options separately, delimited by at least a space (or tab). For example: **ls –F –a**
2. Put them together, using only one minus (–) character. For example: **ls –Fa**

However, there are exceptions where only the first method or the second is valid. There are also cases where you need to use a combination of both methods. All these are command-dependent and option-dependent; you should check up the command first.

4.10.2 Self-Help: The `man` Command

The **man** command is what an independent learner would need to find out more about individual UNIX commands. It provides online help on various topics and commands. The usage is:

'man' is short for 'manual (pages)'. So, don't accuse UNIX for being sexist, but you may accuse it for being terse. However, some think that such brevity is rather sexy.

```
man <topic>
```

Therefore, **man who** displays help information on the command **who**, and **man ls** tells you everything about the **ls** command. Try these out as well:

man helpinfo	for FAQs (frequently asked questions)
man rules	for rules and regulations on the School's computing systems and resources
man policies	for policies on the School's computing systems
man printers	for help on printing files and information on the School's printing facilities
man man	for help on how to use the **man** command

The **man** command with the **–k** option performs a keyword search of specified entries. For example, **man –k user** lists all command summaries with the pattern 'user'. Regular text patterns (see **man grep**) are allowed here.

You should cultivate the habit of reading up the man pages. For the rest of this section, we will introduce the commands, and briefly describe their functions. You are to spend some time familiarising yourself with these commands, and try them out on your own.

4.10.3 Organising Your Home Directory

These commands enable you to work around your directories: **pwd**, **ls**, **mkdir**, **rmdir** and **cd**.

Where am I? – the pwd *command*

The **pwd** (print working directory) shows you your current working directory.

```
garfield@sf3:~[13]$ pwd
/home/stu04/garfield
```

Absolute and Relative Pathnames

/home/stu04/garfield is where you currently reside. A pathname is an address of a file or a directory. If it begins with a slash / (the root), it is called an *absolute pathname*, where we trace the path from the root. If it does not begin with a slash, then it is a *relative pathname*, where we trace the path from the current directory.

The tilde (~) symbol represents a user's home directory (hence it is shorthand for /home/stu04/garfield in Garfield's account). The dot (**.**) represents the current directory, while the double-dot (**..**) represents the parent directory. These shorthands appear at the beginning of pathnames.

What does my directory contain? – the ls *command*

The **ls** (list files) command displays the files and subdirectories in your current directory.

```
garfield@sf3:~[xxx]$ ls
c       doc       my_file
```

The output shows that you have three files in your current directory (hidden files are not shown). There is no way you can tell whether the files listed are regular files or directories. You may add the **–F** option to indicate the file types:

```
garfield@sf3:~[xxx]$ ls –F
c/      doc/      my_file
```

Note that a slash is appended to the name of a directory. A regular file has no slash appended to its name, and an executable file (not shown here) has an asterisk (*) suffix.

Alternatively, you could use the **–l** option (or a few other options):

```
garfield@sf3:~[xxx]$ ls –l
drwx------   1 garfield stu04    512 May 10 09:39 c
drwx------   1 garfield stu04    512 Jun 21 19:43 doc
-rw-------   1 garfield stu04    142 May  5 14:20 my_file
```

As you can see, **–l** option reveals more information. The first character indicates the file type ('d' for a directory, '–' for a regular file), followed by 9 characters that form the access code of your file. Access code will be covered with the **chmod** command later.

Try out **ls –a** to list out all the files in the current directory, including hidden files whose names begin with the dot (.). Among the files listed you should see two files: '.' and '..'. As mentioned, the current directory is represented by a single dot (.), and the parent directory represented by double-dot (..).

Making and removing directory – the mkdir *and* rmdir *commands*

If you are an organised person, you would not like to lump your lecture notes, love letters, and song books together and put them into a single cabinet. You would sort and categorise the documents, create folders within the cabinet, and put related documents into a folder, aptly named.

The directories are like compartments. You are free to create subdirectories under your home directory, and more sub-subdirectories under these subdirectories, to form a hierarchy in the same way as the UNIX file system is organised.

Creating a directory is possible with the **mkdir** (make directory) command. Just enter:

```
garfield@sf3:~[xxx]$ mkdir cs1101
```

Many commands work silently. That is, you should not expect pompous messages like "Hi, Garfield! Thank you for using the mkdir command. We have created the directory cs1101 under your order. Enjoy your day!" flashing across the screen announcing the successful execution of your command, neither would you hear thunderous applause coming from the system. No, UNIX is a silent worker that does only the necessary. You will be notified via error messages when there is an error, but if everything goes well as in this case, you will be greeted with the $-prompt, silently and shyly awaiting your next command.

You may do a check to convince yourself that the **mkdir** command did work:

```
garfield@sf3:~[xxx]$ ls -F
c/      cs1101/      doc/      my_file
```

Try this:

```
garfield@sf3:~[xxx]$ mkdir cs1101/prog
```

Make a guess, what does it do? Yes, it creates a directory **prog** under the **cs1101** directory. Hence, **prog** is two levels down from your home directory. This example shows that the argument can be any valid pathname. In this case, the relative pathname **cs1101/prog** specifies the target, which is **prog** under **cs1101** under the current directory, since a relative pathname starts its trace from the current directory. Of course, you could also have used the equivalent absolute pathname **/home/stu04/garfield/cs1101/prog**, but this is a little wordy, or the shorter absolute pathname **~/cs1101/prog**, or the relative pathname **./cs1101/prog**.

So many ways to do the same thing!

To check that **prog** exists in **cs1101**, you may do this:

```
garfield@sf3:~[xxx]$ ls -F cs1101
prog/
```

To remove a directory, use the **rmdir** (remove directory) command.

```
garfield@sf3:~[xxx]$ rmdir cs1101/prog
garfield@sf3:~[xxx]$ rmdir cs1101
```

Note the sequence. You cannot remove a non-empty directory, so **prog** must be removed before **cs1101**.

(There is a way to remove a directory with all its files and sub-directories, but we will keep it simple for now and let you explore this later. Hint: with the **rm** command.)

Moving around – the cd *command*

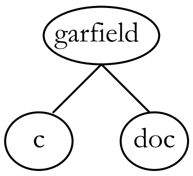
When you have many directories, you would like to move around your file system to do your work. The **cd** (change directory) command does just that.

```
garfield@sf3:~[xxx]$ cd c
garfield@sf3:~/c[xxx]$ ls
example1.c     example2.c     example3.c
```

You have just moved your current directory to the **c** subdirectory, as reflected in your prompt (if your prompt does not show the current directory, you may use the **pwd** command to check your current location). The **ls** command lists all the files in the new current directory. Of course, you would have achieved the same end had you typed **ls c** at your home directory, but changing directory allows you to type shorter pathnames in your commands.

Now that you are in the **c** directory, to move to the **doc** directory under your home, you type:

```
garfield@sf3:~/c[xxx]$ cd ../doc
garfield@sf3:~/doc[xxx]$
```



Typing **cd doc** is wrong, as that would be taken as an attempt to move into a **doc** directory under **c**, the current directory; but there is no **doc** directory under **c**. As **doc** is on the same level as **c**, you need to specify **../doc** as the pathname, in order to locate the target directory correctly. Recall that in a pathname, double-dot (**..**) refers to the parent of the current directory.

To return to your home directory, use **cd** with no argument.

```
garfield@sf3:~/doc[xxx]$ cd
garfield@sf3x:~[xxx]$
```

4.10.4 File Access Permissions

Recall the output of the **ls –l** command:

```
garfield@sf3:~[xxx]$ ls –l
drwx------   1 garfield stu04    512 May 10 09:39 c
drwx------   1 garfield stu04    512 Jun 21 19:43 doc
-rw-------   1 garfield stu04    142 May  5 14:20 my_file
```

For all these files and directories, the owner is garfield, and the group is stu04. All Computing students matriculated in 2004 belong to this same group stu04. We have discussed the meaning of the first character (‘d’ to indicate a directory, and ‘–’ a file). Now, let us study the next 9 characters. These characters constitute the access code (or protection code) of the file/directory.

The access code comprises access information for three categories of users: **u** (the owner of the file, in this case, garfield), **g** (the group members, in this case, group stu04), and **o** (all other users).

$$\texttt{-}\underbrace{\texttt{rw-}}_{u}\underbrace{\texttt{---}}_{g}\underbrace{\texttt{---}}_{o}\quad \texttt{my\_file}$$

For each category of users, the access is encoded in three characters, representing the three types of access privileges:

r	read permitted
w	write permitted
x	executable (for files)/accessible (for directories)

The absence of a type of access is marked by a '–'. Therefore, in the above example, **my_file** is readable and writable, but not executable[1], by garfield, while group members and other users have no access to the file at all.

Files and directories are created with the default permissions (readable to everybody, but writable only by the owner). You may use the **umask** command to change the default setting.

You may also change the access code of existing files and directories. For example, Garfield may want to share the file **example1.c** in his subdirectory **c**, with the rest of his group members, to allow them to read the file. In this case, he would need to add group's read access to **example1.c**, as well as all the directories that lie in the path from his home to the directory that holds the file, in this case, his home directory and the **c** directory. We use the **chmod** (change mode) command. This is one way of doing the job:

```
garfield@sf3:~[xxx]$ chmod g+rx .
garfield@sf3:~[xxx]$ chmod g+rx c
garfield@sf3:~[xxx]$ chmod g+r c/example1.c
```

This form of the command is known as *symbolic mode*. If Garfield wants to remove the read access from group for **example.c**, he could do this:

```
garfield@sf3:~[xxx]$ chmod g–r c/example1.c
```

We could change settings for different users at once. Besides + to add permissions and – to remove, we could also use = to set absolute permissions, as in:

```
garfield@sf3:~[xxx]$ chmod g=rw,o+r c/example2.c
```

The above command grants group members read and write access, and adds read access for other users, to the file **example2.c** in the **c** directory.

Apart from symbolic mode, we could also use the *octal mode* in the **chmod** command. The code is interpreted as 9 bits, where 0 denotes the denial of the corresponding access right, and 1 denotes the granting of the right.

[1] A file can still be executed by the owner even though it does not have the 'x' access on, but that is left to the readers to explore.

The example below shows the use of octal mode:

```
garfield@sf3:~[xxx]$ chmod 754 c
```

754 is an octal number that is equivalent to the binary number 111 101 100 (spaces added for clarity). This bits sequence represents the permission rwxr-xr--.

This is a cursory treatment on the **chmod** command. Read up the man pages for more information.

4.10.5 Manipulating Files

Some commands to create and manipulate files will be covered here: **cp**, **mv**, **rm**, **cat**, **pr**, **pg**, **more**, and **less**.

Copying files – the cp *command*

A useful command is **cp** (copy) which has this format:

cp <source> <target>

where <source> is the pathname of the source file, and <target> the pathname of the destination directory. Sometimes, a new filename may be specified in <target>, otherwise the new file would assume the source filename.

```
garfield@sf3:~[xxx]$ cp my_file my_file_2
garfield@sf3:~[xxx]$ cp my_file doc
garfield@sf3:~[xxx]$ cp my_file doc/another_file
```

The first command makes a copy of the file **my_file**, calls it **my_file_2** and leaves it in the current directory. The second command creates a duplicate of **my_file** and places it in the **doc** directory, under the same filename **my_file**. The third command makes yet another copy of **my_file** and places it in the **doc** directory as well, but this new copy is given the name **another_file**. If the target directory contains a file with the same name prior to the operation, the file will be overwritten by the newcomer.

In UNIX, filenames may begin with a letter (upper-case or lower-case), digit, period (.), or underscore (_). Like UNIX commands, filenames are case sensitive too.

A common usage of the **cp** command is to copy somebody's file into your own directory. Of course, you must be granted read access to that file for this to be permitted. One example is:

```
garfield@sf3:~[xxx]$ cp ~tantc/c/answers .
```

You have learned that ~ is the shorthand for your home directory. Similarly, *~userid* is the shorthand for the home directory of a user with this *userid*. The above example copies the file **answers** in the **~tantc/c** directory into Garfield's current directory. Note that unlike DOS, the target directory must be specified, so it is wrong to omit the dot (.) in the second argument.

Moving and renaming files – the mv *command*

An equally useful command is **mv** (move) which has the same syntax as the **cp** command. This command enables you to move file to another directory, or rename a file, or both.

```
garfield@sf3:~[xxx]$ mv my_file_2 my_file_3
garfield@sf3:~[xxx]$ mv my_file_3 doc
garfield@sf3:~[xxx]$ mv doc/my_file_3 my_file_4
```

Let us go through the commands one by one. The first command renames the file **my_file_2** to **my_file_3**, in the current directory. The second command moves **my_file_3** into the **doc** directory, retaining the filename. The third command transfers the file **my_file_3** which is now in the **doc** directory back to the current directory, and names it **my_file_4**.

Deleting files – the rm *command*

The command **rm** (remove) is used to delete files:

```
garfield@sf3:~[xxx]$ rm my_file_4
garfield@sf3:~[xxx]$ rm doc/my_file
```

These will remove the file **my_file_4** from the current directory, and the file **my_file** from the **doc** directory.

Warning: The **rm** command does not prompt you for confirmation, so be careful when you use it. To include prompting, use **rm –i**. (Since it is a danger to use **rm** if the user is careless, most of the time an alias is set to make **rm** works as **rm –i**. See 'Aliases' on page 64.)

Some commands allow you to specify more than one argument. The above example could be done with a single **rm** command:

```
garfield@sf3:~[xxx]$ rm my_file_4 doc/my_file
```

Be sure how the command works before you specify multiple arguments.

Viewing text files – the cat *command and its relatives*

The command **ls** allows you to view the content of a directory, that is, what files are contained in that directory. How about files? To view the content of a regular file that is in text (ASCII) format, you may use the **cat** command or one of its relatives: **pr**, **pg**, **more**, **less**. To view binary files, you need other methods, which are not covered here.

No, the relatives of 'cat' are not animals. Neither is 'cat' itself.

```
garfield@sf3:~[xxx]$ cat my_file
This is the first line of my_file,
HAND stands for Have A Nice Day...
... and this is the 3rd and final line.
```

As you might have guessed, the **cat** (catenate) command displays the content of the file **my_file** on the screen.

The **pr** (print) command formats the file with page headers and padded trailing lines, usually 66 lines per page, making the output suitable for sending to the printer. The command, however, displays the content on the screen.

The command **cat** is suitable for small files. For longer files, the output generated by the **cat** command would scroll off the screen too quickly for good reading. One way is to use the control-key sequence ^s (or press the ScrollLock key) to hold the output momentarily, then release it with ^q (or press the ScrollLock key again). But this tests the agility of the fingers.

In UNIX, 'less' is more than 'more'.

A better alternative is to use the **pg** (paginate), **more**, or **less** command. These commands display a screenful of the file's content at a time, pausing for you to view the page before you enter a key to proceed to the next screen. Each of these variants works slightly differently, with **less** being the most versatile and **pg** the simplest, but basically you press Enter to proceed to the next page, and 'q' to quit.

You may also use a text editor to view and modify the content of a text file. An editor will be introduced in another section.

4.10.6 Printing Files

Printing of files is an essential task. Here, you will learn about the **lpr**, **lpq** and **lprm** commands. Refer to **man printers** for information on printers, and rules on the use of them. You may also refer to **man printquota** for information on your hardcopy quota. You can use the **pusage** command to check how much of the printing quota you have used up for that month.

To print a file, you use the **lpr** command which has this syntax:

lpr –P<printer-id> <filename>

Each printer has a printer-id which you can gather from the **man printers** pages. An example is psmr.

```
garfield@sf3:~[xxx]$ lpr –Ppsmr my_file
```

The above command sends a print job to the printer **psmr**, to print **my_file**. You may check the print queue for the status of your job using the **lpq** command:

```
garfield@sf3:~[xxx]$ lpq –Ppsmr
Rank   Owner      Job  Files          Total Size
active tantc      822  exercise1      38284 bytes
1st    garfield   823  my_file          142 bytes
```

You may also remove your print job from the print queue. First, you need to check the print job number with the **lpq** command as shown above. Then you issue the **lprm** command:

```
garfield@sf3:~[xxx]$ lprm –Ppsmr 823
```

4.10.7 Useful UNIX Features

The introduction is not complete without mentioning, brief though it might be, about some useful features that can make your working in UNIX more efficient and effective. The following note is taken from the link on facilities in the School's web site.

Meta-characters

When you log on to a UNIX machine, you are in a UNIX shell that accepts, interprets and executes your command. There are a number of variants of the shell; Bourne shell, bash, csh, tcsh are just a few. The default shell in sunfire is bash. You may **man bash** for more information.

The shell treats command line arguments as *regular expressions* and attempts to match them with files in the current directory. Meta-characters are used in regular expressions. Some of these special characters are '?'. '*', and '[ ]'. For instance,

```
$ cat part[4–6]
$ rm part*
$ ls part?
```

In the above examples, **part[4–6]** is expanded into **part4 part5 part6** (if these files exist) by the shell; **part*** is expanded into all the filenames that begin with 'part' (hence, '*' is usually called the wildcard character); and **part?** is expanded into filenames five characters long that begin with 'part' (hence '?' matches any single character).

Devices, files and pipes

Files and devices have identical interfaces in UNIX. As such, commands which read input from the keyboard (standard input) or write output to the terminal screen (standard output) may read from and write/append to files instead.

The shell interprets these special characters '<', '>', and '>>' as *take input from*, *write to* and *append to*, respectively, and performs the necessary input/output redirections. For example,

```
$ date >file1
$ ls >>file1
$ mail tantc@comp.nus.edu.sg <file1
```

In the above examples, the output of **date** is redirected to the file **file1**, instead of appearing on the screen. Hence, you will not see any output on the screen. **file1** is created if it did not exist, or overwritten if it did. In the second command, the output of **ls** is appended to **file1**. Finally the **mail** command takes the file **file1** and sends it to the user **tantc@comp.nus.edu.sg**.

The output redirection symbol (>) may be used to create an empty file. Typing:

```
$ >abc
```

will create a new empty file **abc**.

One wonderful feature about UNIX is the pipe (|). It allows the output of one command to be fed as input for the next command in the pipe, without the need of creating temporary files.

```
$ who | wc –l
$ sort groceryList | head | tail –3
```

The first pipe above returns the number of users currently logged into the system, by running the **who** command, and then passing its output to the **wc –l** command that counts the number of lines. The second pipe sorts the file **groceryList**, then extracts the first ten lines of the sorted list by using the **head** command, and finally displays the last three lines among these ten lines by calling the **tail** command. The result is that the 8th, 9th and 10th lines of the sorted list are displayed.

UNIX is a rich system full of lessons, awaiting your exploration. Do spend some time on experimenting it, and you will be rewarded handsomely.

4.11 A Text Editor: pico

A text editor is a software that allows you to create and modify text files. Unlike word processors like WinWord or WordPerfect, text editors do not have a rich set of formatting features.

There are a number of text editors available in UNIX, ranging from the simple line editors like **ed** and **ex**, to the screen editors like **pico**, **joe**, **vi**, **vim**, and **emacs**. Here, we will cover the **pico** editor, an easy-to-learn and simple editor for the beginners. You are advised to pick up at least one other more advanced editor like **vim** or **joe**, to make your editing sessions more productive as these editors provide more advanced features. Naturally, the learning curves for these advanced editors are much steeper, but you will benefit in the end.

The syntax for invoking the pico editor is as follow:

pico [options] [filename]

Some options are:

+*n*	(*n* is a number) causes pico to start at the *n*th line of the file
–w	disables word wrap (thus allowing editing of long lines)
–z	enables ^z suspension of pico
–v	sets to view mode, disallowing any changes to the file

If a filename is not specified or the file does not exist, then a new file will be created.

Note: Control key is represented by ^ below. For example, ^Y means control-Y.

How to navigate?

- Arrow keys are used to move the cursor.
- ^Y Page up
- ^V Page down

How to save file?

- ^O You will be prompted for a file name. Press Enter to save the current file, or key in a new file name. You will also be prompted to make a save upon exiting pico.

How to get out of pico?

- ^X If you have modified the content of the file, you will be prompted to save the file before exit.

How to refresh screen?

- ^L Sometimes, your screen gets messed up when someone writes to you. In this case, you may refresh the screen to clear off the unwanted messages.

How to check current line number?

- ^C Shows you the line number and which portion of the file (how far from the beginning) you are at.

How to do cut and paste?

- ^K Cuts the entire line the cursor is currently on. A block of text can be cut by using ^K repeatedly on consecutive lines or by using ^6 to mark the block.
- ^U Pastes the text that was cut by the previous ^K.
- ^6 Marks a block of text. Arrow keys are used to mark the text. ^V and ^Y can also be used to make marking faster by marking them page by page.

How to insert the content of another file?

- ^R You will be prompted for the name of the other file. Type the pathname of the file relative to your home directory. Alternatively, ^T will display a list of files/directories for you to choose from. ^C is used to cancel this request.

How to do a spell check?

- ^T You will be prompted for a replacement at every misspelled word detected. Enter the new word and press Enter to make the replacement.

How to quit pico temporarily?

When a friend chats you, you may want to quit **pico** temporarily and get back to where you have left off later. First, you need to enable the suspension feature by including the –z option. At the pico screen, you press ^z (control-z) and you will be brought to the UNIX shell. After you are done, you can type **fg** (foreground) to return to pico.

4.12 Customizing Your Environment

Shell Variables

There are two types of shell variables in UNIX:

- System shell variables. These include the environment variables, positional parameters, and other special parameters.
- User-defined shell variables, created by the users.

You may use the **set** command to view a list of the shell variables. The following are some of the environment variables provided by the system. Note that system shell variables are typically in uppercase.

HOME	pathname of your home directory
PATH	search path for finding and executing commands and shell scripts
TERM	type of terminal
PS1	primary prompt string (default '$')
PS2	secondary prompt string (default '>')
SHELL	type of shell
PWD	pathname of current directory
LOGNAME	your userid

You may use the **echo** command to view the value (which is a string) of a variable. The **echo** command simply echoes its argument on the screen.

```
$ echo How are you?
How are you?
```

We use the *variable substitution* character '$' to represent the value of a variable. For example, **$HOME** represents the value stored in the variable **HOME**, and **$SHELL** represents the value stored in **SHELL**. To see their values, use the **echo** command:

```
$ echo $HOME
/home/stu04/garfield
$ echo $SHELL
/bin/bash
$ echo $TERM
vt100
```

You should not meddle with some of the system variables, but for a few others, you may change their values to customise your environment. For instance, you may change you primary prompt to your favourite string:

```
$ PS1='Hello, dear: '
Hello, dear:
```

Note that there must not be spaces around the = character. From now on (until you log out of UNIX), your primary prompt will become ‘Hello, dear: ’. Isn’t it fun?

Besides system variables, you may also create your own shell variables, known as user-defined variables, as shown below (we have reverted the prompt back to ‘$’):

```
$ myvar='I like ice-cream'
$ echo $myvar
$ I like ice-cream
```

Such user-defined variables are useful for shell script programming, which is beyond the scope of this chapter.

Aliases

Another useful feature you may consider is the creation of *aliases*, which are synonyms of commands.

For example:

```
$ alias tata='exit'
$ alias sw='who | cut –c1–8 | sort'
```

Again, there must be no spaces around the = sign. The first statement creates an alias **tata** for the **exit** command. From now on, you may use **tata** as a substitute for **exit**. The original command **exit** may still be used. Similarly, the second statement creates a short alias for a long pipe, saving you keystrokes.

Recall that the **rm** command does not prompt you for confirmation, but **rm –i** does. Instead of keying in the longer version, you may create a short alias for **rm –i**, or better still, name this alias **rm**, as follows:

```
$ alias rm='rm –i'
```

Now, you would not need to worry about deleting a file carelessly. (But if you are a die-hard careless person, then even this might not help much!) To ‘cancel’ an alias, use the **unalias** command:

```
$ unalias tata
$ unalias rm
```

Creating the* .profile *file

All the changes that you have made to the shell variables, and the aliases that you have created, will be gone once you log out of your current UNIX session. To save you the trouble of typing in these commands all over again the next time you log in, the system provides a **.profile** file in your home directory for you to store these definitions. Note that the filename begins with a dot and so it is a hidden file. **ls** will not be able to list the file, but **ls –a** will. This **.profile** file is like the **autoexec.bat** file in DOS, which is automatically executed each time you log into the system.

Type **ls –a** at your home directory to see if **.profile** exists. If it does not exist, you have to create it. Create or modify the file using an editor. Just type in all the definitions or changes you would like to have.

Note that adding the definitions into the **.profile** does not activate the definitions right away. It will only be activated the next time you log in, since **.profile** is only executed when you log in. If you want to see immediate effects, type the command:

```
$ . .profile
```

This forces an immediate execution of **.profile**. Note the dot at the beginning of the line. The dot (.) command runs the file in the current shell.

4.13 Summary

You get a peek into the School's computing environment in this rather long chapter. After logging on to NUSNET-IV through the Windows XP computer, you log into one of the UNIX servers to perform your chores, like programming for your CS1101/CS1101C module.

Familiarity with the UNIX operating system is a necessity, and the chapter leads you through a tour on some UNIX commands, covering basic but important activities such as managing your directories and files, setting access permission to files, and creating, manipulating, browsing and printing files. It also covers some meta-characters, input/output redirection and pipe.

A simple text editor, pico, is introduced. You are to know the editor well as you need this tool to create and edit files.

Customising your UNIX environment can be done by changing system shell variables, or creating your own aliases, and put these into your **.profile**.

Besides technical information, the chapter also presents information on facilities and policy matters. Some of the man pages where you can gather such information are **helpinfo**, **rules**, **policies**, **printers**, **printquota**, **passwd**, and **password**.

In the next chapter, you will learn about some means of communication.

Note

The chapter was updated in May 2004, reflecting the many changes since it was written in 1999. However, as facilities and services are continually being added and upgraded, there might be new changes by the time you read this. Please refer to the respective websites and other resources for the latest information.

Acknowledgement

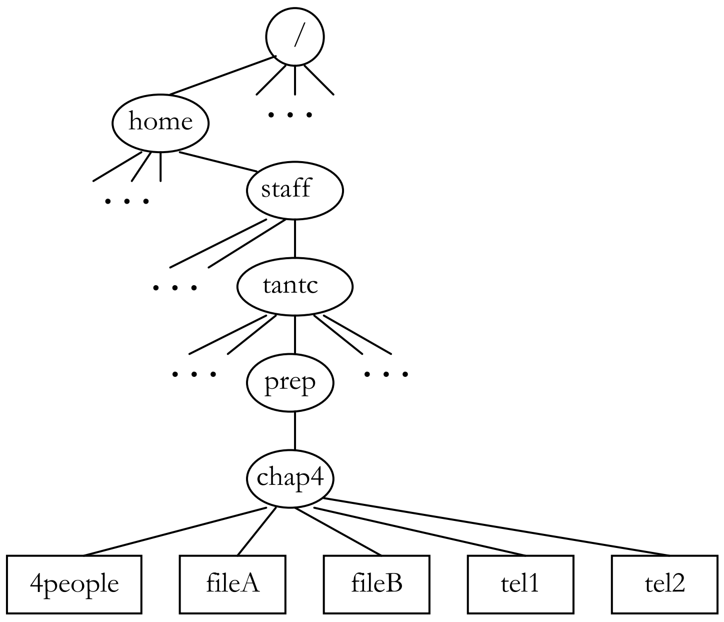
These resources have provided great aid in the writing of this chapter: the "SoC Survival Guide" produced by the Computing Club, the "Survival Guide for CPE Freshies" written by the CPE seniors, and the "UNIX Systems Introductory Guide" in the Facilities link of the School's web site, prepared by Associate Professor Derek Kiong. I would like to express my gratitude to these parties.

I would also like to thank Mr. Tok Wee Hyong for contributing his writing for sections 4.3 to 4.6.

Exercises

You should have read the chapter the tested the various commands before you attempt the exercises here. Bear in mind that given a task, there might be more than one way of achieving it, though some ways are better than others.

1. The diagram below shows the partial file system of the user tantc (hey, that's me!).



Using ~tantc as a shorthand for /home/staff/tantc, how could you change directory into the **chap4** directory? Once you are in the **chap4** directory, how do you list the files in there, and view the content of the file **4people**? (The protection codes of relevant directories and files have been lowered to allow read access from your account.) Return to your own home directory. How do you view the content of the file **4people** from your directory?

2. Create a subdirectory **chap4** under your home directory. How do you copy the files **4people**, **fileA**, **fileB**, **tel1** and **tel2** from tantc's account into your own **chap3** directory?

3. Find out about the **diff** command. Get into your **chap4** directory view the contents of **fileA** and **fileB**. Then try out this command: **diff fileA fileB**.

4. Create a subdirectory **tel_dir** under your own **chap4** directory. How do you transfer the two files **tel1** and **tel2** from your **chap4** directory into the **tel_dir**?

5. Get into the **tel_dir** directory. View the contents of the files **tel1** and **tel2**. Merge the two files into a new file **tel3**, without changing the original files. View the content of **tel3** to check.

6. Go to your **chap4** directory and test these commands out:
   ```
   ls
   ls  tel_dir
   ls  4people
   ls  *
   ```
   Understand how **ls** works with different arguments.

7. Find out about the **echo** command. Try these out at your **chap3** directory:
   ```
   echo  4people
   ls  4people
   ```
   The commands produce the same output. Does this mean that **echo** is equivalent to **ls**?

8. Go into the **tel_dir** directory and view the file **tel1**. Use the command below to sort **tel1**:
   ```
   sort  tel1
   ```
   The file **tel1** is left intact, and the output (sorted content) is shown on the screen. How do you save the output into a file **sorted_tel1**?

9. Read the **man** pages of the **sort** command to find out how you can sort **tel1** by names.

10. Pick two paragraphs from this chapter. Using the **pico** editor, type the two paragraphs of text into a file called **passage**.

11. The **grep** command allows you to search a file for a pattern, for example: **grep 'minds' fileA**. The **find** command assists you in looking for files. These are two useful commands. Read up the man pages on them.

Websites:

http://www.nusnet.nus.edu.sg/
NUSNET

https://www.comp.nus.edu.sg/cf/
Computing facilities in the School of Computing

http://www.nus.edu.sg/comcen/main/getstart.htm
Policies on the use of NUS IT resources.

https://www.comp.nus.edu.sg/cf/rules/
Rules, regulations and policies on computing facilities and services in School of Computing.