Introduction, Operating Systems

COSC301 Laboratory Manual

In this introductory laboratory, we shall know how the labs will build on each other. We shall then practice some basic lab skills. We will practice these skills implicitly in later labs, meaning that the labbook will expect us to know how to do these, so please pay attention and ask questions whenever one is struggling to understand or perform any of the activities in this laboratory. It is *not* recommended to do this lab outside of the prescribed lab environment for this lab.

Please report any problems you have with the lab material; this will help students both in this year and subsequent years.

As an overview for this lab session, we will be practicing the following basic skills: seeking help and working as a supportive community, accessing class resources, changing your password, introduction and correct use of VirtualBox, and practicing some basic terminal skills. Mastery of these essential skills will help you complete the assigned labs with fewer problems, greater speed and gain a more thorough understanding of the lab material and its relation to the lectures.

> **Important**
>
> It is vitally important that you learn to cope intelligently with errors and learn to diagnose problems; being able to give a good diagnostic report is a very desirable outcome of this paper. Working on other labs can be a useful way of coping if you're stuck waiting for help.

1. Overview of the Laboratories

The laboratories in this paper build on some of the previous labs. Therefore, some labs will require you to have completed the work in some of the previous laboratories. This means you not only have to install and configure a service, but also maintain it, which involves some configuration management skills.

Our network will start off very small, and we shall slowly grow it as needed. However, we shan't have any more than about five virtual machines at most in any one lab, due to host resource requirements.

The network we shall build mirrors quite nicely what you might find in a Small Or Home Office (SOHO) network, so you should immediately be able to reproduce and build on what you learn in this course at home. This is an important way of solidifying and expanding your knowledge and skills, which is crucial for developing your experience in preparing for work in the industry.

We start building our network in this lab by importing a ready-made machine. This machine shall have a hostname of "client1", and in VirtualBox, we shall refer to as as "cosc301-client1", in order that all COSC301 related machines are grouped together. In later labs, we will add a second client using a Live CD, then quickly move on to installing our server virtual machine which will eventually become our gateway in our SOHO network to the wider network.

2. Accessing class resources

During much of this paper, you will be developing a network of computers. To accomplish this, you will sometimes require resources such as CD-ROM disk images ("ISO images") or entire machines which may have been prepared ("appliances"). It may also include documentation; whatever is needed. The exception would be the on-line edition of the labbook, which should be available from the course website, where this pdf was found.

Another resource which will be *very* important is where you need to store your virtual machines.

To easily find the resources like ISO images and appliances and to create a storage for your virtual machines, open a Terminal window from the MacOS dock and type the following commands:

```
$ ln -s /home/cshome/coursework/301/resources/ ~/Desktop/resources
$ mkdir ~/Desktop/myvms
$ cd ~/Desktop/myvms
$ pwd
/home/cshome/h/hzy/Desktop/myvms
This is the fully qualified path of myvms.
You may see something slightly different, but remember the output for later use.
```

Now you should have on your Desktop the two folders: `resources` and `myvms`, which you will need to access often while working on the labs. The output of the last command above will tell you the fully qualified path of the folder `myvms`, which you will need shortly.

Note that the contents of the `myvms` ("My Virtual Machines") folder is individual to your VirtualBox. You can read from or write to it like your home directory. However, the `resources` folder will be read-only to you.

3. Seeking help and working as a community

In this paper, you are never really alone when you are working on your lab material; there are always avenues of support readily available. One of the essential skills an administrator needs to be able to do is to know how to tap these support avenues.

But tapping these resources takes time, so you also have to try and solve or at least investigate the problem yourself to some extent, so you can give a good description of the problem. It is this description which is *essential* to getting a prompt and useful response. It is our continued experience that when one starts writing down what steps they have done to try and solve a problem – with a view to submitting a request for help or bug report – that they end up solving the problem themself.

Some support avenues that are available to you are described below. The best option will depend on your circumstances:

Classmates
: Much of the value of teaching this course is in learning from others mistakes, which has enriched our experience as well. There is a lot of truth in the saying "to teach is to learn twice". Therefore, you are allowed to share experiences with each other when trying to

solve problems. Naturally, this does not mean giving each other the answer to a question — that would be plagiarism — but rather helping each other to diagnose problems and evaluate solutions.

Teaching Team
This includes your Demonstrators as well as your Lecturers.

Class Resources
In this lab, you will learn how to access the "resources" folder for this paper, which contains various screen casts, scripts and other files that are useful or necessary for completing the labs quickly.

Class Website
The class website [http://www.cs.otago.ac.nz/cosc301/] (as you have already known) is where you can find important information including: an on-line version of the laboratory manual, which should always be the most up-to-date version; the course schedule with links to the lecture notes; the PDF version of the marksheet; assignment material and even some videos to help you to perform some of the lab activities.

Class E-mail list
Any class correspondence will be sent to the cosc301@cs.otago.ac.nz mailing list. This is where you will find lab notices such as updates and corrections. Please also use this as an avenue for asking questions outside of lab time. *Also, please feel free to post useful replies, so long as they do not give assessment answers directly.*

The Internet
A lot of help can be found on the internet; one of the more likely things you will want to use this for is to look for particular error messages. Note that Usenet News (available via such avenues as Google groups) is also a rich resource. There are many support avenues around which you should learn and share; it is a very useful way of increasing your experience and problem solving ability[1].

At your convenient time, skim-read Eric S. Raymond's "How To Ask Questions The Smart Way [http://catb.org/~esr/faqs/smart-questions.html]" and Simon Tatham's "How to Report Bugs Effectively [http://www.chiark.greenend.org.uk/~sgtatham/bugs.html]". In fact, you should probably consider the first to be required reading; it *will* save you a lot of time and face in your future student and professional career… Even some Post-Graduate students need to read this. In fact, it's so useful, you are required to at least skim-read "How To Ask Questions The Smart Way" for part of the self-assessment of this lab.

4. Correct use of VirtualBox

The "myvms" folder on your Desktop is where you will need to store your virtual machines.

You must first configure VirtualBox appropriately such that the location of the virtual machine files will be stored in the "myvms" folder.

Procedure 1. Initial Configuration of VirtualBox on Mac OS X

Following this procedure, you will configure VirtualBox to save your VM disk images on "myvms", which is regularly backed up by our IT support.

[1] It can also raise your Internet profile, which is useful when someone is looking to hire you.

1. Start VirtualBox, which you will find in the `Applications` folder or the dock on your Mac OS X.

2. From the menu, choose VirtualBox → Preferences….

3. Under the General section, set the Default Machine Folder field to `~/Desktop/myvms` by clicking the drop-down menu on the right of the field and finding the folder from the menu. Here "~" means your home directory such as `/home/cshome/t/tom`.

> ## Note
>
> If you want to type the folder into the field directly, make sure you type the fully qualified path (as shown before) into the field.

4. From the same VirtualBox → Preferences… window, under the Update section, untick Check for Updates. This can prevent VirtualBox from showing update notification.

We shall now use some simple features of VirtualBox, importing a simple virtual *appliance*, which is prepared for you, and practicing some basic skills related to VirtualBox.

> ## Note
>
> If interested, read this note box.
>
> To save you some time, we have created a client for you, and you can import it into your VirtualBox environment. This is basically the concept of a *virtual appliance* whereby an entire operating system and software stack has been configured for a particular task, ready to be imported into a virtualisation environment. You can learn more about Virtual Appliances with VirtualBox by looking at the “Importing and Exporting virtual appliances” episode of the Fat Bloke’s Shorts [http://www.virtualbox.org/wiki/VirtualBoxTV]. Client1 is a default installation of Ubuntu 18.04 LTS “Bionic” desktop amd64 installation, with updates applied and configured to use a particular Ubuntu software repository. Additionally, some software specific to VirtualBox (“Guest Additions”) has been installed to help the guest run faster and with greater capability. We shall discover more about the Guest Additions when we look at the lab concerning System Installation.
>
> We provide some local media containing the large installation files for the appliance. They are made locally available for performance reasons, otherwise it can be too slow.

From within the VirtualBox window (ie. not a window for a virtual machine, but the window that lists the available virtual machines), import Client1 using File → Import Appliance… and Choose the `cosc301-client1-latest-version.ova` file from the `~/Desktop/resources/Appliances` folder. Accept the settings in the following dialog by clicking “Continue”. Importing the appliance will take a while, perhaps 5-10 minutes.

> ## Schedule your time intelligently
>
> Whenever you have a lengthy wait on your hands, work on other things that aren’t blocking on the current task. This gets you thinking about project and time management. For example, while waiting for the virtual appliance to

import, open up a terminal on your Mac OS X (or Linux workstation) and go through the Terminal Agility section of this lab, which doesn't require the use of VirtualBox or Client1.

Once you have imported the virtual machine, Start it. Log in as the user "Miss A. Laneous", who has the username `mal`. The password is `Quack1nce4^`. This user, with the same password, will be used throughout the machines in this paper; but do note that these are separate accounts that share neither password storage nor home directory contents.

Exercise

As an exercise make sure that the Client1 VM is running and that you have successfuly logged into the guest. Note that, this type of *Exercise* boxes is very important for you in the rest of the labbook. They indicate what are expected to work in the labs, which will be expected to work in the assignments of client and server in this paper. Some exercises are for you to have better understanding of the lab work.

Note that, after you log in, the virtual machine may automatically offer to update software packages, which may take a while. You may leave it until it finishes, or cancel it and choose to do so later.

A Password Mnemonic

A duck walks into an elevator, but is too short to reach the buttons, so an attendant says "quack once for up". You may find such mnemonics useful to generate and remember strong passwords and pass-phrases.

5. Using sudo and su for Administration

In subsequent labs, you will need to regularly manage the system which will require administrator (the "root" user) rights. Because this affects system security, we're briefly going to introduce you to it, and you'll practice the skills very often throughout the rest of the paper.

On a standard Unix/Linux host, superuser privileges are generally attained through the use of the substitute user command **su**. This is considered better than logging in as root because it creates an audit-trail, meaning we can look at the login records and determine how root privileges were gained. For example, we might look at the login record and find that a user called "test" logged into the system at 3am in the morning from some remote system in the Russian Federation (all of which are likely very unusual) and got access to the root account using **su**.

On modern Unix-like hosts, you also have the option of using the **sudo** command. Whereas **su** gives you a session (sub-shell) as root, **sudo** gives you the ability to run a single command as root. **sudo -s** can be used to give you a shell also, much like **su**.

Don't do everything as root!

If you do everything as root, you *will* eventually fall victim to yourself. Instead, use **sudo** where you can, and **sudo -s** or **su** only when you need to.

Actually, neither command requires that you end up as root. Indeed, **su** is short for *substitute user*, so you could change to any other user. This is perhaps useful if you need to change

from root to a normal user, or from a normal user to a special account, such as a database administrator.

All Unix-like systems will have **su**, and many will have **sudo**, depending on administrator preference. It is installed by default on Mac OS X and Ubuntu, and is many other Linux systems. One of the things that make both Ubuntu and Mac OS X a bit different is that the root account is by default locked on both systems, and require the use of **sudo** to gain priveleges.

sudo has a different authentication model than **su** or practically any other authentication system you will have used. Instead of having to enter in the root password – assuming that account is not locked – you instead enter *your own* password. This is to authenticate yourself to the **sudo** command, as someone else may have started using your account if you had left your workstation unattended for a while. The system administrator (root) will have added you to a file called `/etc/sudoers` (which is edited using the command **visudo**, but don't try that now). This allows the system administrator to grant particular users the ability to run particular commands as a particular user. This allows the system administrator the ability to give partial root access without disclosing the root password.

Because the user "mal" was the user that was created when the system was installed, that user has already been added to the `sudoers` file.

sudo will remember when you last successfully authenticated. If you use **sudo** shortly thereafter, it will not ask you for your password. By default, this timeout is 15 minutes, and can be overridden in the `sudoers` file.

Try the following exercise to practice getting root access. Do this in a terminal window on Client1; there should be a Terminal icon in the Gnome panel at the left of your virtual machine; if not you can search for Terminal. Before you begin, follow the instructions below.

We're first going to use our normal user account to look at the permissions on a protected file that root can only read. We'll try to read it as a normal user, which will fail with "Permission denied". We'll then elevate our privileges using **sudo** and see that it works.

```
$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1144 2015-01-08 09:57 /etc/shadow
```

We'll see more about filesystem permissions in a later lecture/lab, but for now you'll just need to understand that this file is not readable by a normal user, only the "root" user (and members of the "shadow" group, which we can ignore) can do anything with this file. In case you're wondering, this file stores all the local users passwords in a secure manner, which is why it has restricted access. Let's just verify that a normal user can't access it:

```
$ whoami
mal    The command ran with the rights of the "mal" user.
$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

We first used the **whoami** command to learn the usercode we were running under; actually, what we found out is what user the **whoami** command was running as. The **cat** command outputs a file to the terminal, but it failed because the user "mal" doesn't have access to read `/etc/shadow`. Okay, so how *do* we get access to that file? We use **sudo**, which is as simple as the following:

```
$ sudo whoami
[sudo] password for mal: enter mal's password, not root's!
root
$ sudo cat /etc/shadow
```

```
root:!:14621:0:99999:7:::        You see the commands output
daemon:*:14545:0:99999:7:::      which has many lines
bin:*:14545:0:99999:7:::         that are not important
…                                so we'll use … to hide them.
```

Great! So we verified, using the **whoami** command, that we can change our privileges to the "root" user by **sudo**, and further verified that we could now do something we couldn't do with our regular "mal" user. Running commands as root is very common, so common that there is a common notation you will find in technical documentation[2]. Look closely at the prompt, in the prompts you will see in this labbook, all commands that run as a normal user have a prompt of $, while all the prompts for a root-user command will have #, so in the following transcript, the first line is asking the same as the second line:

```
# whoami
root
$ sudo whoami
root
```

Sometimes, the command you need to run is too awkward to easily run using **sudo**, and you really want to run it from a shell. To do that, use **sudo -s**. Try this now:

```
$ whoami
mal
$ sudo -s      Start a shell with root privileges
root# whoami   Yes, I'm root
root
root# exit     Leave our root shell
exit
$ whoami       Back to where we were in line 1
mal
```

Exit from your root shell by tying **exit** or **^D**. This will exit your inner, root shell, and return you to your previous shell, which was running as "mal".

6. Shutting down

When you are finished, temporarily, with a virtual machine, you have a few options to shut it down. To do this on Mac OS X, do the same as you would do to close a window: click on the red gem (of the red, yellow and green gems in the top left of a Mac OS X window). You can also use **Host+Q**, remembering that on a Mac, the host key is by default the left ⌘. When you do so, a dialog will slide out, allowing you to choose one of the three options:

- Save the machine state: this is like hibernating a laptop, the contents of the system and graphics memory is saved by VirtualBox to non-volatile storage (ie. somewhere on disk). The only thing that cannot be preserved is external state such as network connections, so these will most likely fail when the machine is woken up. However, this should be used most of the time in your day-to-day work.

- Send the shutdown signal: this is equilavent to pressing the power-switch on a modern computer. It will cause a power-management signal to be sent to the operating system, which will commonly do things like a) pop up a menu on-screen to ask the user what it should do; b) sleep, or c) shutdown.

- Power off the machine: *Never* choose this option unless you have to. This is like forcing a machine to turn off, such as by pulling out the power cord or holding in a power button for

[2] They come from the Bourne shell of which your shell is a derivitive. % is also sometimes used, which comes from the C-shell family of command-line shells.

five seconds. The operating system has no chance to clean up, so filesystem corruption can occur. This should be a last resort when the above two options do not work.

> **Exercise**
> As an exercise practise the first option to save the state of your virtual machine; its state should show as “Saved” if successful.

7. Terminal Agility

When you know a few tricks, working inside a terminal can be a very fast way of operating. Alas, even experienced people don’t know a lot of these ways, and end up not being as productive as they could be. So in this section, we aim to increase your agility in the terminal by giving you a few exercises for you to practice in the following labs.

You do not need to memorise this list; but if you never hear of them, you may never think of them. The most important ones are marked with a pencil icon; it would pay to memorise these few to begin with.

Most shells use a library called readline; since this is widely available on different systems, these keystroke sequences will work in many shells, on Linux as well as Mac.

> **Important**
>
> The following list uses the Emacs style of keystoke notation, so C-a means to type **a** while holding down the **Ctrl** key.
>
> M-a means to hold down the **Meta** key, which is not a key you will find on modern keyboards; instead, use the Left Alt (or the Left Option key on a Mac).

Cursor movement

C-a, **C-e**
: ✎ Move to the beginning and end of the line. Type some text in your shell, use the keystrokes to move to the beginning and end of the line.

M-f, **M-b**
: Move forward and back a word at a time, but note that this doesn’t always work; the keystroke might instead invoke menu functions instead. Thankfully, some Linux systems, Ubuntu included, define **Ctrl+LeftArrow** and **Ctrl+RightArrow** for the same task.

Editing commands

Backspace, **C-h**
: ✎ Delete character to the left of the cursor. Knowing about C-h can be useful, because Backspace is not always reliable on some systems.

Del, C-d
: Delete character to the right of the cursor.

C-w
: ✎ Delete the word to the left of the cursor. Type a few words in the shell, then use C-w to erase them, word by word. This is much faster than using backspace. It can be faster to erase and retype a whole word than to use backspace.

C-c
✎ Abort the current command being entered. Begin typing a command, and then type C-c, to see it disappear.

C-l
Clear screen. This commonly also means redraw screen in various full-screen terminal applications, such as editors.

History commands

C-r
✎ Start searching backwords in your history, incrementally.

For example, assume you had in your command history a command that mentioned a file `dhcpd.conf`. You can then type **`C-r`** and then start typing **`dhcpd.conf`**, as you type each character, you will be shown items in your history that match. You can type **`C-r`** again to show matches earlier in your history.

You can use the standard editing keystrokes to edit the command if you like, or you could just type **Enter** to run the command again.

!!
✎ (pronouced “bang-bang”) is a sequence that you will probably use very often, as it expands to the last command line you used. This is useful when you forget to use **sudo**:

```
$ some long admin command
Operation not permitted       Oops, forgot to use sudo
$ sudo !!
sudo some long admin command
Now it works, and we saved a lot of typing or navigating our history.
```

Command Completion

Tab Completion
✎ One incredible time-saving device a lot of students *not* using is Tab completion. This makes it *much* faster to accurately navigate around the filesystem and commands.

To use Tab-completion, you simply type **Tab**, and it will complete as much your current word as it can before it gets ambiguous. If you type **Tab** again, it will present you with a list of choices. If there is a large number of choices, you will be prompted with a `--More--` prompt, and you can press **Space** to view the next page, or **q** to quit. The best way is simply to try it out. Don’t be afraid, you have a lot to gain.

C-x C-e
This one is perhaps less useful for a beginner, but if you remember it, it will make developing longer commands (often the precursor to a script) much easier. This keystroke sequence will open your current command in your preferred editor (which can be set using the `EDITOR` environment variable, which by default on Ubuntu is **nano**).

8. Self-assessment

Exercise
As an exercise do the following tasks.

1. You must consult the manual pages for sudo(8) and sudo_root(8) (only available on Ubuntu systems) to answer these questions.

List briefly the advantages and disadvantages of using **sudo** as set up on Ubuntu, and of leaving the root account disabled. Ensure you understand what is being said in the documentation.

2. List five major points of the “How To Ask Questions the Smart Way” that might be of particular relevance to a system administrator asking for help. (note that most are relevant, and make sure you’ve actually read the document)