Practical 1

Theme

- executing programs on remote computers in a secure manner
  - starting a secure remote Unix shell
  - starting and controlling X11 programs remotely
- deploying a chat system client locally and remotely

Key concepts: remote login, secure shell, remote execution, remote X11 clients, remote deployment

1.1. Start up and essential configuration

a) *Login to Ubuntu.*

You may need to reboot the computer if it is currently running MS Windows.

It is helpful if you have this sheet open on your desktop so that you can cut and paste some commands instead of typing them. Unfortunately, the cut and paste does not work reliably between Acrobat Reader and the terminal. *Evince* is a fast and reliable PDF viewer alternative to the Acrobat Reader.

Hint: *Cut & paste of text in X11* can be done conveniently by selecting the text simply with the mouse and then pasting it using the *middle mouse button* (wheel). The more common commands `Ctrl-C` and `Ctrl-V` also work.

Ubuntu lab cheat-sheet is available on the module website with more tips.

b) *Start a terminal window.*

From the Gnome main panel, choose Applications>Accessories>Terminal.

The terminal starts up running the command-driven interface `bash` (Bourne-again shell), which enables one to launch programs by typing their names followed by arguments separated by spaces. The cheatsheet documents some handy tricks that speed-up one's interaction with `bash` considerably.

c) *Configure your shell.*

One of the most handy features is the ability to recall previous commands using the arrow keys. This can be further enhanced so that one can type the first few characters of a previous command and look up the command using the arrow keys without seeing previous commands that do not start with the typed characters. To enable this enhancement, execute the following command:

```
> gedit .bashrc
```

An editor should open either empty or with a complicated-looking code in it. Scroll to the very end of the file and cut and paste the following two new lines to it:

```
bind '"\e[A"':history-search-backward
bind '"\e[B"':history-search-forward
```

While you are at it you may also add a few handy shortcuts:

```
alias l="ls -lF"
alias c="cd .. && ls -lF"
```

The first shortcut is for producing a detailed listing the contents of the current folder and the second one for switching to the parent folder while listing its contents.

The new features will apply in any newly opened terminal. To enable the new features in the current terminal, execute:

```
> . .bashrc
```

d) *Create a folder for this module.*

You can name anything you like but in the lab sheets it will be assumed that the module folder is `CS3250-DS-1011`.

It is highly recommended that you create the folder within your linux home folder because the home folder shared with Windows (ie your H: drive) does not work reliably under Linux in the labs. To set it up execute the following commands in a terminal:

```
> cd
> ln -s /users/nfs/[first letter of your login name]/[your login name] linuxhome
> mkdir linuxhome/CS3250-DS-1011
> ln -s linuxhome/CS3250-DS-1011 CS3250-DS-1011
> ls CS3250-DS-1011
```

It is important that you execute all the lines and no errors are reported. (Although harmless errors occur if you try to execute some of these commands repeatedly, eg trying to create a folder twice.) If an unexpected error is reported, ask for assistence. If the last command reports no error, you have most probably succeeded.

1.2. Start an X-windows program from the shell

a) *Run the command `gnome-system-monitor` in the shell.*

You can exit the program by pressing `Ctrl-C` in the terminal.

Hint: It saves one time and prevents many mistakes if one makes use of the `Tab` key completion while typing in *bash*. Eg type *gno* and then press `Tab` then type the next character or two and press `Tab` again and so on until you have *gnome-system-monitor*.

b) *Run `gnome-system-monitor` without blocking the shell.*

The easiest way to achieve this is by running the command as before but adding a & at the end of the command: *gnome-system-monitor &*

If you started a program that blocks the shell and do not wish to terminate it and then restart it with &, you can also regain the shell as follows:

– press `Ctrl-Z` to pause the program
– execute the special command *bg* to resume the program at the background

1.3. Use SSH to conveniently control a remote computer

a) *Log in to a server using* `ssh`.

The complete command to type is:

```
ssh duck
```

If you want to connect from outside the campus, run `ssh duck.aston.ac.uk` instead.

b) *Check which computer your shell is running on using the command* `hostname`.

c) *Find out who is logged in on the server at the moment using the command* `w`.

d) *Investigate how busy the computer is using the command* `htop` *and then press* `q`.

e) ***Log out*** *of the remote computer by typing* `exit` *and ensure you are back in the shell on your local computer using the* `hostname` *command.*

1.4. Run graphical X11 programs remotely

a) *Log in to the server again, this time enabling X11 connections*

– Exit from the server's shell started in previous command (`exit`),

– check that you are in the local shell (`hostname`),

– recall the ssh command using the arrow keys and edit the command so that it reads as follows:
`ssh -X duck`

– and finally execute this command.

b) *Start a graphical load monitor on the server.*

Execute `gkrellm &` in the server's shell. This should bring up the same kind of monitor you saw before but this time showing the state of the server, not the local computer.

c) *Inspect your files on the server.*

Execute `nautilus . &` to open a graphical file manager.

1.5. Deploy and test a Java chat client

a) *Download file* `lab1-chat.zip` *from BlackBoard and extract it into your* `CS3250-DS-1011` *folder.*

b) *Start the client from command line.*

Use the `cd` command to make the extracted folder your current folder (aka "go into this folder"):

```
cd
cd CS3250-DS-1011/lab1-chat/src
sh compileClient.sh
sh runClient1.sh
```

c) *Test the client.*

Type some messages into the client and observe your messages displayed on the large screens among messages from other students. (The messages from other students will appear in your terminal too.)

d) *Deploy the chat server on* `duck`

Copy the whole `chat-system` folder to your home folder on `duck` by the following commands:

```
cd
cd CS3250-DS-1011
rsync -a lab1-chat duck:
```

Then use `ssh -X duck` to log into `duck` and then compile and start the client as before.

Note that deploying a client on the same computer where the server is running is not generally a good idea. This exercise is designed to give you an experience of deploying a program on a remote computer without the overhead of introducing a new scenario where such an action would make more sense.