6G7Z1004 Advanced Computer Networks and Operating Systems
(Lab 5: File systems and virtualization)
A part if this lab was originally designed by Emma Norling MMU, UK

In this lab session, we will look at the Linux file system as well as some virtualization aspects.

Task A: Linux file system

Files, directories and links

The mount command under Linux is used for mounting partitions, but if you use it without any arguments, it can also be used to see the partitions that are already mounted. Enter the command

mount

and see if you can work out the details related to your home directory. (Reminder: you can type **pwd** to find the full path name of your home directory.) There will be many lines of output (lots of things are mounted); for each line, the information displayed is:

[path] on [mount] type [type] ([r/w permissions]...)

To learn more about this command type **man mount**

Change your working directory to the top level of your os directory (assuming that you have already created a directory called os), then create a subdirectory called tmp (**mkdir tmp**). Create a file in your os directory called test (use the command **gedit test**), with a single line of text as its contents: "This is the first test". Now create a file in the tmp subdirectory, also called test, with a single line of text as its contents: "This is the second test". Now enter the command

ls -i test tmp/test (type **man ls** to understand the purpose of using the option –i)

The **-i** option to **ls** issues the inode (index node) numbers for each file, as well as their name. Note that although the file names are the same (but not the directories), the inode numbers are different. Now delete tmp/test (**rm tmp/test**) and enter the commands

ln test tmp/test (type **man ln** to learn more about this command)
cat tmp/test
ls -i test tmp/test

Note that not only are the contents of tmp/test the same as test, but the inode number is the same. This is because the **ln** command creates a link to a file, rather than a new file. Delete the original file (**rm test**). What do you expect will happen to tmp/test? Perform a directory listing to see if your prediction is right. If not, discuss with your lab tutor to understand why not.

Now create yet another file called test, in your top-level os directory, with the contents "This is the third test". Enter the command

ln test tmp/test

You should get an error message! Now enter these commands:

ln -s test tmp/test2
ls -i test tmp/*
cat tmp/test2

You should have seen three files listed, each with different inodes. What does the **-s** flag for ln do? Which file is **tmp/test2** linked to? Why?

Now enter the commands

rm tmp/test
ls -i test tmp/*
cat tmp/test2

What has happened to test2? Why is the behaviour so different when the link is symbolic (created with ln -s) rather than hard?

Finally, clean up your workspace. Delete the tmp directory and its contents, and delete the test file in your os directory.

Task B: Virtualization

This tutorial will provide you with skills required to operate virtual machines in a linux environment. You will also perform an experiment to identify the impact of running a host operating system under full system simulation as opposed to virtualization.

Step 1

The files that you are going to need for this task are quite large and you have limited space in your home area. Additionally, you might not realize that your home folder is not located on the machine that you are working on, the files actually reside on central server. Running a virtual machine disk image from a remote drive can severely impact performance. To work around this issue we can create a folder inside the **/tmp** directory to place our working files. Don't save anything that you want to keep in this folder, as there is a high chance it will be wiped when you log out. Inside the /tmp directory create a folder with the name **vm**.

cd /tmp
mkdir vm
cd vm

Next, we will copy the iso files from the server into the local tmp directory. If you are unsure of what an ISO file is read the following webpage http://en.wikipedia.org/wiki/ISO_image

In the /var/tmp directory, you will find a variety of Linux ISOs. Copy the most recent Ubuntu ISO (the one with the largest release number) to the directory that you have created in /tmp.

Step 2

Load the iso file using qemu with the following command.

qemu-system-x86_64 **-cdrom *NAME_OF_ISO_FILE***

(*for more details type* ***man*** **qemu-system-x86_64**)

When you run qemu a new window will appear, this is the graphical output of the virtual machine you have just powered up. When you load the image you will be prompted by the boot-loader in the new window, to interact with the new machine you will need to capture the mouse and keyboard. To do this click your mouse in the virtual machine window. This will more than likely result in the mouse pointer disappearing, to escape the window press the **ctr+alt** key and move the mouse outside of the window. To shut down the virtual machine you can either click on the exit icon on the desktop and select shutdown, this will perform a clean shutdown or we can just click the **x** in the top right hand corner of the qemu window. This is the same as switching the power off to the machine.

Step 3

Next, we are going to start the virtual machine using the same command, but this time we are going to time how long it takes to be presented with the desktop. Using your watch or mobile phone, time the delay between hitting the enter key and the operating system loading up and displaying the web page in the browser.

Step 4

Next, we will load the Ubuntu image using **KVM**. Qemu is a full systems simulator of the x86 hardware, whereas KVM provides full system virtualization. What impact will this have on the bootup time? Perform the same experiment as detailed in step 3 using the following command, and record the results.

kvm -cdrom *NAME_OF_ISO_FILE*

Step 5

We can vary the number of processor cores and memory size that a virtual machine has access to using the **smp** and **m** parameters. Record the time needed to load the VMs using both qemu simulation and kvm virtualisation for 1,2,4 cores and 256, 512 and 1024 mbytes of RAM.

Step 6

Investigate the networking capabilities of qemu. Start up two virtual machines and try to use the ping command to verify connectivity. Try to ping the virtual machines of the person sitting next to you. Why are you unable to ping between virtual machines on separate machines?

Type "whats my ip" into google from both virtual machines or type **ip addr** in the terminal of the virtual machine. What is the IP shown? Why is it the same for both machines?