UWS UNIVERSITY OF THE WEST of SCOTLAND

COMP09024 Unix System Administration

Laboratory 6: Package Management

Package Management

Learning Outcomes

- **Managing system backups using the `tar` command**
- **Installing software from a repository using `apt-get`, and `aptitude`**
- **Compiling and installing software locally (`wget`, `md5sum`, `make`)**

Working with Backups

Creating a Backup on (Virtual) Media

First we will get acquainted with the 'mother' of all backup commands **`tar`** (**t**ape **ar**chive); a general purpose archive utility originally designed for backing up to magnetic tape. Although quite old, **`tar`** works well for normal data backup and it is still one of the main programs used for backing up and restoring data in Unix. Note that in addition to **`tar`** there are also other specialised backup utilities including: **`cpio`** and **`dd`**.

We begin by creating a directory for our backup exercises, in which we will place some ASCII files, namely *world.dat* and *vmstat.dat*.

```
[root@UWS /]# mkdir /home/backuptest
[root@UWS /]# echo 'hello World' > /home/backuptest/world.dat
[root@UWS /]# vmstat 1 100 >> /home/backuptest/vmstat.dat
```

These files are created just for this exercise, but you should know what **`vmstat`** does. In the command line above we tell **`vmstat`** to produce 100 snapshots of the memory status within 1 second intervals, therefore it will require 100 seconds to complete.

For demonstration purposes we will once again mount a virtual floppy. You may wish to reread the instructions for this given in laboratory 2 and download a fresh copy of virtual_floppy.img from the week 2 folder on Moodle. After uploading the virtual_floppy.img file execute:

```
[root@UWS /]# losetup /dev/loop0 /root/virtual_floppy.img

[root@UWS /]# mount /dev/loop0 /media/floppy
```

After this we can backup the test directory **/home/backuptest** on to the floppy as follows:

```
[root@UWS /]# tar -c /home/backuptest > /media/floppy/backuptest.tar
```

Questions

- **Q6.1)** What is the meaning of -**o rw** in the command for mounting the floppy?
- **Q6.2)** What is the meaning of the -**c** qualifier in the tar command?
- **Q6.3)** What is the name of the tar file on the floppy?
- **Q6.4)** Note down what was displayed by the **tar** command.

Now examine the size of the file **backuptest.tar** on the floppy, using the **ls –s** command and compare it with the actual size of the whole directory **/home/backuptest** on the hard-disk by using the **du** (**d**isk **u**sage) command on **/home/backuptest** like:

```
[root@UWS /]# ls -s /media/floppy/backuptest.tar
[root@UWS /]# du -k /home/backuptest
```

The last line of the output should give you a summary of blocks (1024bytes), as used by the **/home/backuptest** directory.

Questions

- **Q6.5)** What is the meaning of the -**s** qualifier in **ls**? What does the -**k** qualifier achieve when used in conjunction with the **du** command?
- **Q6.6)** What is the difference in size between the original directory and the backup on the floppy? Does **tar** perform any compression?

Retrieving Data from a Backup

When retrieving data from a backup it is important not to overwrite the source of the original backup, as might happen if you retrieve data to the directory which was backed up in the first place. As a precaution it's always better to retrieve files to a separate directory or disk partition, check them and finally copy them back to the original location in the directory hierarchy. For this it is always a good idea to set e.g. on the **/tmp** directory before starting a backup:

```
[root@UWS /]# cd /tmp
```

Use the **ls** command to check if there are any files currently in **/tmp** that you wish to preserve. Now use **tar** in extract mode:

```
[root@UWS /]# tar xvf /media/floppy/backuptest.tar
```

Questions

- **Q6.7)** Examine **/tmp/home/backuptest** directory. Is this an exact copy of the original **/home/backuptest** directory?
- **Q6.8)** Explain the meaning of the **xvf** qualifiers. Why is there no hyphen sign?

Compare the original and the backup file using the **cmp** command:

```
[root@UWS /]# cmp /tmp/home/backuptest/world.dat /home/backuptest/world.dat
```

Questions

- **Q6.9)** Why is there no output from the **cmp** command?
- **Q6.10)** Compare the date of creation of both files. Did the backup file inherit the same creation date?
- **Q6.11)** Whilst examining **/tmp** do you see why **tar** removes the leading '**/**' of the path when creating an archive? Please refer to your answer to Q6.4. Consider what would happen if the leading forward-slash was not removed by the tar command and you later attempted to un-tar the tarball (tar archive) in superuser mode.

Creating and Retrieving a Compressed Backup

As you should have discovered, **tar** doesn't ordinarily compress data. Fortunately, a standard Linux distribution includes a bunch of highly efficient compression tools such as **gzip**. One way to use **gzip** is to pipe the output of the **tar** command to **gzip** before storing the result. Such compression of tarballs for distribution is **standard** practice. Try:

```
[root@UWS /]# tar -c /home/backuptest | gzip > /media/floppy/backuptest.tar.gz
```

Note that it is also possible to archive and gzip without using a pipe by specifying the -**z** flag.

```
[root@UWS /]# tar -cz /home/backuptest > /media/floppy/backuptest.tar.gz
```

Question

- **Q6.12)** Check the size of the resulting file **/media/floppy/backuptest.tar.gz.** What is the compression rate now, if you use the whole size of the directory **/home/backuptest** as a reference point?

Retrieving data from the compressed file is a two-step process requiring that we first uncompress then un-tar it. The inverse command for **gzip** is called **gunzip**.

```
[root@UWS /]# cd /media/floppy
```

```
[root@UWS /]# gunzip < /media/floppy/backuptest.tar.gz > /media/floppy/unzipped.tar
```

```
[root@UWS /]# tar xvf /media/floppy/unzipped.tar
```

Please check that a copy of the **/home/backuptest** directory was created with the above commands.

Alternatively, you can use:

```
[root@UWS /]# tar xvfz /media/floppy/backuptest.tar.gz
```

Extracting a single file

It is also possible to extract a single file from an archive: Make sure that you are in the /**tmp** directory then delete the directory **/tmp/home/backuptest** on your system.

```
[root@UWS /]# cd /tmp
[root@UWS /]# rm -r home
```

now let's extract:

```
[root@UWS /]# tar -xp home/backuptest/world.dat < /media/floppy/backuptest.tar
```

Now try:

```
[root@UWS /]# tar -xp /home/backuptest/world.dat < /media/floppy/backuptest.tar
```

Questions

- **Q6.13)** What means the **-p** option ?
- **Q6.14)** Why did the second command using **/home/backuptest/world.dat** as the destination not work?

Here is an exercise that combines backups and shell scripting:

Question

- **Q6.15)** Write a shell script that produces a zipped-tar version of any input directory called '**dir_name**', whereby '**dir_name**' represents the relative directory name, not the absolute one. The output should be send to **/tmp** or a floppy device and should be of the form: **dir_name.tar.gz** .

Installing Software

Installing Packages from a Repository

Before installing any new software we shall review the relevant system configuration files and update/amend our system if necessary. There are several steps that may or may not be necessary depending on your environment.

First of all try the following command, which tells the Debian *Advanced Package Tool* (apt-get) to update its list of software package sources (known as *repositories*).

```
[root@UWS /]# apt-get update
```

```
deb http://deb.debian.org/debian buster main
```

You should see some messages indicating that up to date information is being downloaded.
Now, let's use the package manager to install and remove the **nsnake** game:

```
[root@UWS /]# apt-get install nsnake
```

Once the game is installed, run it in background:

```
[root@UWS /]# /usr/games/nsnake & >/dev/null 2>&1
```

Use the **pidof** command to find out whether or not the **nsnake** game is now running in the background. I

```
[root@UWS /]# pidof nsnake
```

Now kill the the game

Questions

- **Q6.16)** What does the **nsnake** package do?
- **Q6.17)** What does the number shown in the output of the command **pidof** represent? (Knowledge of **pidof** may come in handy in exams and when writing shell-scripts.)

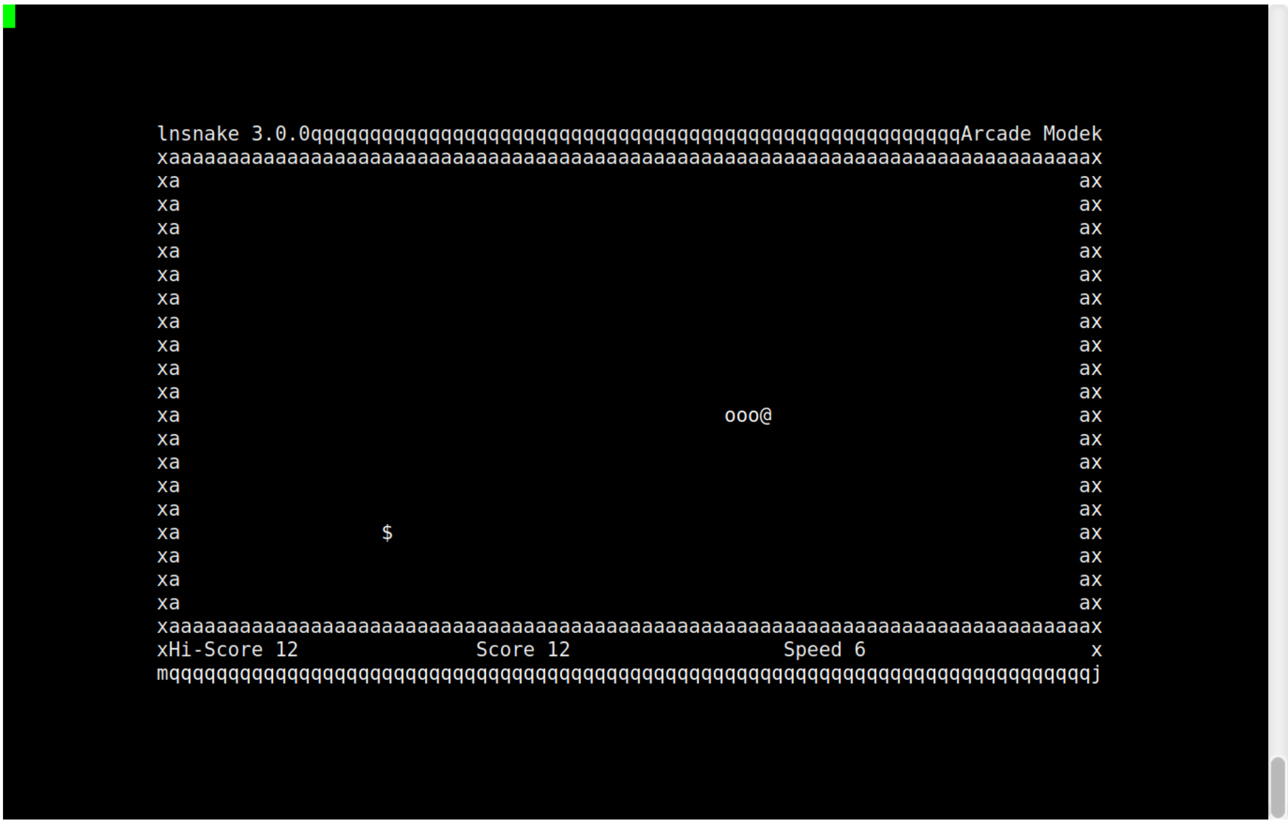
Now kill the **nsnake** background process using the previously obtained PID:

```
[root@UWS /]# kill -9 <PID>
```

Now run the game and just press “Enter” to start playing. Use the arrows to control the snake. The goal is to get as much ‘$’ as possible!

`[root@UWS /]# /usr/games/nsnake`

`You should see something like:`



`Now try too remove the package by typing:`

`[root@UWS /]# apt-get remove nsnake`

<u>**Question**</u>

- **Q6.18)** Verify that **`nsnake`** has been deleted by checking **`/usr/games/nsnake`** . What did you find this time?

There is a higher-level interface to APT called **`aptitude`** that enables you to search for available packages and to find out more information about both installed and available packages.

Note: In this web based virtual machine, the aptitude command is not available. Here is a (non-exhaustive) list of **`aptitude`** commands for you to try:

aptitude update		Equivalent to **apt-get update**
aptitude install	<packagename>	Equivalent to **apt-get install**
aptitude remove	<packagename>	Equivalent to **apt-get remove**
aptitude show	<packagename>	Gives information and installation status of packages (similar to **apt-cache show**).
aptitude search	<pattern>	Searches for matching package names.

Compiling Software from Source

To demonstrate the process of compiling software locally we will create a simple C program to display a message. Create a new file named "hello.c" with the following content:

```
#include <stdio.h>

int main(int argc, const char *argv[])
{
        // printf() displays the string inside quotation
        printf("Hello, World!\n");
        return 0;
}
```

Calculate the md5 checksum of the hello.c

```
[root@UWS /]# md5sum hello.c
```

Questions

- **Q6.19)** Briefly outline the functionality of the **md5sum** command.
- **Q6.20)** Did you get a match with the md5 checksum and if so what does this prove?

Now we need to compile the C source file. Rather than using "gcc" we are using "tcc" a tiny, a small but hyper fast C compiler. You can find more information about "tcc": https://bellard.org/tcc/tcc-doc.html and more information about "gcc": https://gcc.gnu.org/onlinedocs/

Now run the following command to compile the C program you have created:

```
[root@UWS /]# tcc hello.c -o hello
```

And run it:

```
[root@UWS /]# ./hello
```

Questions

- **Q6.21)** What is the difference is size between hello.c and hello?
- **Q6.22)** Can you explain the size differences?
- **Q6.23)** Modify the hello.c to display your name and banner ID instead of “Hello, World!”.
- **Q6.24)** Compile and run the modified program.
- **Q6.25)** Do a screenshot and add it to your notes and lab book.

– END OF LAB –