DAT151-OBLIG2
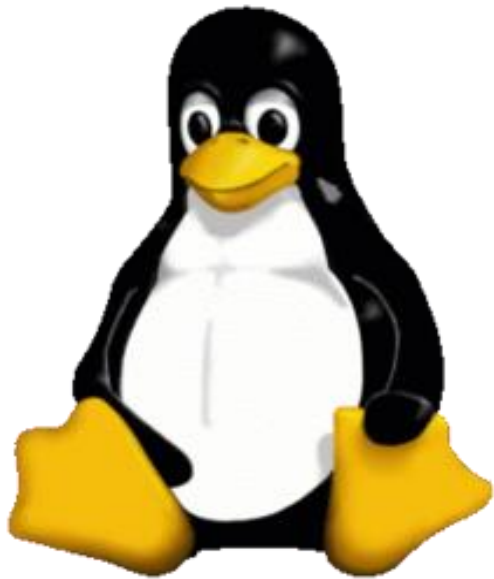
Database and Unix System Management

HVL

Adrian R.J. Mortensen

Content

Part 1 – The filesystem

Task 1: Filesystem basics

1. Windows vs Linux/Unix filesystems

   Describe some of the key differences between the Linux/UNIX filesystem and how files and folders are organied in windows.

   In a windows file system you have the individual disks / partitions given their own drive letter and you have to navigate between these to enter the folders. In linux all drives and partitions can be mounted into the already defined tree-structure without having to deal with drive letters.

2. Mounting

   What is a "mount point"? Create a new folder and use the mount command to let this be the mountpoint for another directory/filesystem. Explain what happens for the command "mount -t ntfs /dev/sda2 /opt"

   A mount point can be imagined as a shortcut to a resource or folder. Say if you mount a new drive to the system the mountpoint would be where you can access this resource/drive.

   Mounting folder:

   ```
   $ mkdir MntPoint
   $ mkdir MntFldr
   $ touch MntFldr/fil1
   $ sudo mount --bind MntFldr/ MntPoint/
   $ ls MntPoint/
   fil1
   ```
   The command would mount a specific type (ntfs) located in sda2 (the resource) to the point /opt.

3. Files

   In most Unix systems, there are seven types of files defined. What are these? Use the file command to display the information of a few files. In which of the even categories do these files belong.

   The seven types of files are:

   - Normal files

   - Directories

   - Hard links

   - Symbolic links

   - Sockets

   - Named pipes

   - Character device

   - Block device

   The files I tried where all normal files or directories.

4. Links

From a user perspective, hard links and symbolic links allow the same file to exist in more than one place. Find an empty directory and create a new file with some text in it. Create a hard link for this file in the same directory using the command ln. Now create a symbolic link for the file using ln -s. Then delete the original file, leaving only the links

```
$ touch fil.txt
$ ls / > fil.txt
$ ln fil.txt hardL.txt
$ ln -s fil.txt softL.txt
$ rm fil.txt
$ ls -al
total 4
drwxrwxr-x. 2 admo admo  40 Jan 31 13:16 .
drwxrwxr-x. 6 admo admo  84 Jan 31 14:11 ..
-rw-rw-r--. 1 admo admo 104 Jan 31 13:12 hardL.txt
lrwxrwxrwx. 1 admo admo   7 Jan 31 13:15 softL.txt -> fil.txt
(Red flashing warning on symbolic link)
$ cat hardL.txt
(outputs all directories in /)
$ cat softL.txt
cat: softL.txt: No such file or directory
```

a) The Hard link worked fine, because it points to the inode of the file. Not to the file that was deleted.

b) The Symbolic link did not work because it is linked to the file therefore it has no link to the inode without the file existing.

Task 2: File attributes and permissions

1. file attributes

In the traditional Linux/UNIX/ filesystem model, every file comes bundeled with a set of 16 bits. What are these bits, and why are they needed? Explain.

The 16 set of bits contain, nine permission bits. Three bits to show the file's "mode". And four bits of file type information. They are needed to determine for instance permissions and what type of file it is.

2. Octal number permissions

The nine permission bits are often represented using octal numbers. Explain what these are, and how the permission bits 110 100 101 can be represented using octals. What permissions does a file with the octal value 745 have?

The numbers represent who has permission to what on a file. The order is shown in the figure below.

1 representing having permission to, and 0 representing not having permission.

Who	Owner			Group			Everyone (others)		
Permission	*Read*	*Write*	*Execute*	*Read*	*Write*	*Execute*	*Read*	*Write*	*Execute*
Binary	1	1	0	1	0	0	1	0	1
Octal	6			4			5		
Octal	7			4			5		
Binary	1	1	1	1	0	0	1	0	1

3. Setting permissions (chmod)

   Chmod is a tool that allows users to change the permission of bits of a file, either by providing octal values, or by a mnemonic syntax where you can combine a set of targets (u,g,o for user, group and other, or a for all three) combined with an operator(+,-,= for add, remove or set) before providing the wanted permissions. For instance to add read and write permissions to the group and all other for a file the command chmod go+rw can be used

a) Create a new file and use chmod with the mnemonic syntax to add read and write permissions to the file owner, read and execute permissions for the group

```
Commands:
$ touch hei.sh
$ nano hei.sh
(Added some runnable code)
$ sudo chmod g=rx hei.sh
$ sudo chmod o-r hei.sh
$ ls -al
total 4
drwxrwxr-x. 2 admo admo 35 Jan 31 13:32 .
drwxrwxr-x. 6 admo admo 84 Jan 31 14:15 ..
-rw-r-x---. 1 admo admo 55 Jan 31 13:26 hei.sh
```

b) Create a new file with the same permissions, but now use chmod with octal numbers to set the permissions.

```
$ touch fil2.sh
$ sudo chmod 650 fil2.sh
$ ls -al
total 4
drwxrwxr-x. 2 admo admo 35 Jan 31 13:32 .
drwxrwxr-x. 6 admo admo 84 Jan 31 14:15 ..
-rw-r-x---. 1 admo admo  0 Jan 31 13:32 fil2.sh
-rw-r-x---. 1 admo admo 55 Jan 31 13:26 hei.sh
```

c) Can you think of any reasons why octal numbers can be preferred over mnemonic syntax

   I find it faster and easier to keep control over.

Part 2 – Software installation and management

Task 3: Package management systems and lower level package management

1. Package management system

What is a package management system, and why do we need one? What is the package fromat used by the package management system on your machine?

The package management system on the centos system by default is RPM (RedHat Package Manager). And the high level package manager is yum. We need a package manager to help keep programs and packages up to date on the system. Aswell as makes in a lot easier to install new programs. This would be a lot harder without a package manager and in bigger businesses impossible.

2.Lower level package management

Package management can often be considered to have two layers, where lower level package management contains tools used to install, uninstall and query packages.

  a) *Use the package management tool available on your system to list all installed packages. How many are installed on your system? Select one package and use the tool again to display all its dependencies.*

```
$ sudo yum list installed
Outputs a long list of packages
$ sudo yum deplist zip.x86_64
```

  b) *Use wget to download the package for the application called openvpn*

```
$ sudo wget https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/Pack-
ages/o/openvpn-2.4.8-1.el8.x86_64.rpm
```

It downloads.

Task 4: High-level package management

1.Installing software

Explain how these systems can locate and install software. What is the high-level package management system on your machine?

The high level package manager on the CentOs machine is yum. High level package managers usually point to a web/ftp server with repositories of software. This is how the package management system find the software.

2.Software repository

What is a software repository? If you are using tum as your high-level package management system, add the epel software repository to your system.

A software repository, stores software packages.

Enabling the epel software repository on CentOs 8:

```
yum install epel-release
```

3.Installing programs with high-level package manager.
Try using yum or apt to install the openvpn package. Does this differ in any way from task 3? If it installs, remove the package afterwards.

```
yum install openvpn
yum remove openvpn
```

(all yum commands were run as sudo)

It differs in the way that you don't need to first download the package, start the installation and then find out if you have the dependencies.
With the high level package manager it finds the package from the repository (in this case "epel") and installs it with the required dependencies.

Therefore the installation went without any problems.