Operating Systems Administration
Laboratory Guide

Alejandro Duran
Mauricio Alvarez
Xavier Martorell
René Serral
SaSa Tomic

Department of Computer Architcture
Barcelona School of Informatics (FIB)
Universitat Politecnica de Catalunya (UPC)

Laboratories

Laboratory 1

Installation of the OS

1.1 Objectives

The goal of this first session is to install a Debian/Linux operating system from scratch on a Intel x86-based computer. The installation will be made on a removable USB disk.

At the end of installation you should be able to use the installed operating system, ie, doing a correct boot and login process.

1.2 Before you start

- Review UNIX basic commands: **cd**, **ls**, ...
- Basic usage of the **vi** editor

1.3 Installation

1.3.1 Access to the CD-ROM

In order to make the installation you need to access the CD-ROM. For doing that you have to mount it correctly. First, we need to create a directory as the mount point

```
# mkdir cdrom
```

Now we can try to mount it with this command:

```
# mount -o ro /dev/cd /cdrom
```

What errors do you get? What can be the reason?

Although we have started the operating system with a CD, it does not recognize the unit that used during the boot process. This is due to the fact that it was the BIOS who has loaded the system image into the memory and then has transfered the control to the kernel. This system does not know from where it has been booted (CD, DVD, network, USB, ...). The operating system that we have running does not have loaded the kernel module required to support the CD fylesystem called: ISO-9660. The Linux kernel module is called: **isofs**.

Load this module using the **modprobe** commmand. You can check the loaded modules with the **lsmod** command.

When you have lodaded the isofs module try again to mount the CD-ROM

1.3.2 Disk configuration: partitions

The next step to be performed is to partition the USB drive. To do so, use the command fdisk on the device /dev/usb. With this command you can:

- Determine which is the geometry of your disk, and its size
- Create the partitions indicated in the next table (select the appropriate size for each one). Remember that partitions 1 to 4 are "primaries", and that if any of them is "extended", the "logical" partitions inside it are numbered starting at 5.

Device	Type	File system	Size	Mount-point	Comments
usb1	Primary	ext3		/	Check the disc every 28 days
usb3	Primary	-swap-			
usb5	Logical	reiserfs		/usr/local	
usb6	Logical	ext3		/home	1 i-node for every 128KB
usb4		free	20GB		Reserved for future use

Table 1.1: Partition table

- Change the type of the swap partition to "Linux Swap", with the 't' command
- Write the contents of the partition table to disk (remember to do so before leaving from the "fdisk" command).

Check that the device files corresponding to the new partitions appear in the /dev directory.
Write here the full names of such new files:

1.3.3 Disk configuration: Creating the file systems

Once you have created the necessary partitions, you must initialize the file system on those partitions that will contain your files, and prepare the swap area for use.

To format the swap area should use the command:

```
# mkswap device
```

Later, you can activate the swap area with:

```
# swapon device
```

To create a Linux file system in the other partitions, we use the command:

```
# mkfs.fstype device
```

for each partition where you initialize a filesystem, where fstype can be: *ext2*, *ext3* or *reiserfs* as the type of filesystem you want to create on the partition.

Depending on the type of file system that you want to use, the options to create the file system can be different. See the different options which can be used on each command help.

Now give format to the partitions you made earlier given the indications you have on the table (put attention to the comments column of USB6).

1.3.4 Mounting the filesystems to install the system

Now we have to mount the filesystems in a temporary directory to be able to install the software. First, we need to load support for the ext3 and ReiserFS filesystems. Load the relevant modules. Please note that mounting an ext3 filesystem without loading its module gives no error but the system will not work appropriately (no journal will be used, and it will behave like an *ext2* filesystem).

Now we will create the mount points for the installation, using a new directory:

```
# mkdir /linux
```

and now we will mount all filesystems inside such a mount point (/linux), creating the appropriate directories inside, at the same time.

```
# mount partition directory
```

The previous table shows in which mount point (directory) each partition has to be mounted (remember, always inside /linux; for instance, / partition into /linux, /home partition into /linux/home, etc.).

We will also mount the directory /dev inside /linux, to reflect the current existing devices. To do that, use the following command:

```
# mkdir /linux/dev
# mount -o bind /dev /linux/dev
```

Use the *mount* command without any parameters to see which filesystems are mounted, and verify that all USB disk partitions are mounted correctly in the appropriate directories, including the /dev directory with the system device files.

1.3.5 Installation of the base system

Once you have prepared the partitions, our next step is to install the base operating system. This process may vary depending on the system. Usually system software is organized into packages, and the software installer decompresses them into the destination directory, and then automatically configures them (maybe with some hints from the user).

In our case, we will install from a prepackaged system image that is in the ASO server (asoserver.pc.ac.upc.edu). You will have to decompress it into the USB disk. To be able to connect to ASO server, we will need to perform a minimum configuration of our network.

Which IP address do you use? (local-ip):

To perform this minimum configuration of the network run the command:

```
# ifconfig eth1000e local-ip netmask 255.255.255.0 # route add default gw 10.10.41.1
```

Then, we need to configure address resolution through DNS. Details about that can be found later in this session, for now we only need to edit the file /etc/resolv.conf with the contents:

```
nameserver 147.83.41.104
```

Afterwards, *cd* into the filesystem that will become the root (/) of your new installation (remember, mounted on /linux):

```
# cd /linux
```

And now get the image, uncompress it, and untar it:

```
# wget -O- ftp://asoserver.pc.ac.upc.edu/packages/aso-install.tar.gz | tar xzf -
```

In this command -O- is the capital 'O' letter, no the '0' number.

Now look at the contents of the /linux directory. You should see that it has been populated with the basic components of your future system.

1.4 Basic system configuration

Before rebooting, you should do some more steps: configure system mountpoints through the /etc/fstab file and install a boot loader.

The configuration files in operating systems based on Unix/Linux are on the default directory /etc and almost always in text format. The Linux environments have many tools for text processing from command line (ie, *grep*, *sed*, *tail*, *cut* ...) and editors (*vi*, *joe*, *emacs*, ...). During the installation process, only the '*vi*' editor will be available in most cases.

1.4.1 Configuring the file systems table (/etc/fstab)

In order for the file systems to be mounted correctly at system power-on, you need to generate a correct /etc/fstab file. Copy the /etc/fstab file from the current installation system to /linux/etc and make the necessary changes:

- Add your swap partition: device none swap defaults 0 0
- Add root partition: device / ext3 defaults 0 1
- Add the rest of filesystems you created previously: device mountpoint fstype defaults 0 2
- Remove the entry for /sys
- Leave the entry /proc without change

Why /proc and /sys do not have any device attached?

1.4.2 Changing root directory

At this point, you can change the root directory of your system, and temporarily use the software that you installed in the system, instead of the one coming from the CD. To change the root of your system, use:

```
# chroot /linux
```

From this point on, you can use the system we have installed, and access for example manual pages with the command *man*.

1.4.3 Configuring the keyboard

In Debian, the keyboard layout is in the file /etc/console/boottime.kmap.gz. The directory /usr/share/keymaps of our future system contains all keyboard layouts available. Copy the keyboard layout you want to use to the default file on boottime.kmap.gz.

1.4.4 Configuring the boot process

Formerly, the operating system was installed on a partition that was marked as bootable in the partition table. The BIOS was searching for it, and then booting the system. This meant that we could have only one system Working on a PC, and that if we wanted to boot from another partition, we should change the partition table and reboot. To address this limitation, second-level boot managers boot (bootstrap loaders) help in booting several operating systems. A boot loader is a set of programs residents in the disk drive, that allow the user to load other operating systems (including from other disk drives). They do the same as BIOS does with them: loading the OS into memory and transferring control. Among the most used we find: *LILO* (Linux Loader), *GRUB* and *NTLDR* (used in systems from Microsoft).

Currently, we have the system installed, but we need to somehow point where our OS is to the BIOS so that next time you boot the computer it is done properly. To this end we will install the *GRUB* boot manager.

To configure correctly the boot of the machine using GRUB, we need to execute the script provided by Linux (remember to keep your root (/) directory set to /linux):

```
# grub-install /dev/sdb
```

This script prepares the /boot directory in order to contain the information needed to boot the machine. The steps it performs (you do not need to repeat them) are:

- Creates the directory /boot/grub.
- Copy the files needed for *GRUB* to /boot. They can be found in /usr/lib/grub/i386-pc/.
- Installs the bootloader in the MBR of the USB disk.

Besides the boot configuration, the system must inform to *GRUB* which kernel must be used to properly boot, this can be accomplished by editing the file /boot/grub/grub.cfg. Give it a look, and closely inspect the following exerpt:

```
menuentry 'Debian GNU/Linux, with Linux 2.6.39-1-686-pae' --class debian \
  --class gnu-linux --class gnu --class os {
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='(hd1,msdos1)'
        search --no-floppy --fs-uuid --set=root e0729d2e-5f2b-4e20-9c41-fcfae136257d
        echo    'Loading Linux 2.6.39-1-686-pae ...'
        linux   /boot/vmlinuz-2.6.39-1-686-pae \
  root=UUID=e0729d2e-5f2b-4e20-9c41-fcfae136257d ro  quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-2.6.39-1-686-pae
}
```

In *GRUB*, the line `set root = '(hd1,msdos1)'` stands for the first partition (msdos1) from the second harddrive (hd1). The instruction `search ...` validates that the partition pointed by the indicator is present and usable. The kernel is specified by the line `linux ...` and the `initrd ...` stands for the particular ramdisk used by the kernel in order to initialize the hardware and load the necessary modules.

As it can be noted in various parts of the configuration there is the partition UUID.

What do you think is the UUID?

What is the meaning of the different parameters passed to the kernel?

Note: it is possible to get the UUID of a particular harddisk partition using the command `blkid`.

A useful way to access to grub configuration is by pressing the **'e'** key when grub is loaded at boot time, this allows us to edit boot options (without saving them) to boot when there is an error in the file grub.cfg.

In order to automatically update the grub.cfg file to contain the UUID of the particular partition you are using it is necessary to run a Debian specific script: **update-grub**.

You can now exit the chroot shell.

Unmount all filesystems and eject the CD. Remove the CD from the CD reader, and reboot using the *shutdown* command.

How have you executed the *shutdown* command?

Could we use other system commands to reboot? Which ones?

1.5 Post-configuration

Boot the newly installed system. Now you have to perform a series of system configuration tasks on this new system. The system has two user accounts: root and aso. Login as aso. Use the password "aso". In general you should use always an unprivileged user to minimize the possibility of damaging your system by mistake. When you need to have user privileges (i.e. become root), use the command *su*:

```
$ su
# privileged-command
# exit
```

or

```
# su -c "privileged-comand"
```

1.5.1 Configuration of the system initialization scripts

Remember that the init program configures the system running the init scripts that are configured for runlevel *S*, and then moves to run the default runlevel scripts.

What is the default runlevel of your system?

Where did you find this information?

The files run for each runlevel are in the /etc/rcX.d where X is the runlevel in question. Note that all files in these directories are links (note also that there are two kinds of links). The scripts are really in the /etc/init.d directory.

When we want to add a script to a runlevel, we must copy the script to /etc/init.d, and then make the appropriate links in the appropriate /etc/rcX.d directories.

Now we want to create a new script to be run only when entering run-level 3.

First create the file aso.sh in the /etc/init.d, containing these commands:

```
#!/bin/sh
do_start() {
        logger -s -t aso "Starting ASO service"
}
do_stop() {
        logger -s -t aso "Stopping ASO service"
}
case "$1" in
        start)  do_start
                ;;
        stop)   do_stop
                ;;
        *)      echo "Usage: $0 start|stop" >&2
                exit 3
                ;;
```

```
esac
```

Using the command **update-rc.d**, you can create the necessary links so that the "service" *aso.sh* starts when entering runlevel 3 and it is stopped when going out of runlevel 3.

Please complete the parameter list that you need to provide to **update-rc.d** to do this:

Now we can check that we have done all the steps correctly. Indicate to **init** that we move on to runlevel 3:

```
# init 3
```

Look at the system logs (/var/log) to see if messages printed by your script are there.

Where do you find the appropriate log file?

Now go back to runlevel 2 and check the messages that come in, indicating that the service has stopped.

1.5.2 Configuring filesystems

In file systems ext2 and ext3 we find certain properties that can be changed after formatting, with the **tune2fs** command. Using this command, change the frequency of checks of the filesystem in the usb1 partition to 28 days.

What other parameters can be adjusted with the tune2fs command?

1.5.3 Configuring system login messages

There are several configuration files that handle messages that appear during the process of login into the system. We want to change some of these messages.

Where are configuration file commonly located?

Before the login prompt "asoclient login:", it appears a message similar to "ASO Linux 1.0 asoclient ttyX". Often, we want to change this message. Now we want to change it for something like this (usually a message indicating that normal users activities can be recorded for security reasons):

```
######################################################################
# This system is for the use of authorized users only.               #
# Individuals using this computer system without authority, or in#
# excess of their authority, are subject to having all of their  #
# activities on this system monitored and recorded by system     #
# personnel.                                                         #
#                                                                    #
# In the course of monitoring individuals improperly using this  #
# system, or in the course of system maintenance, the activities #
# of authorized users may also be monitored.                         #
#                                                                    #
# Anyone using this system expressly consents to such monitoring #
# and is advised that if such monitoring reveals possible        #
# evidence of criminal activity, system personnel may provide the#
# evidence of such monitoring to law enforcement officials.      #
######################################################################
```

Which file should contain this message? (Hint: search for the file with the original contents that you want to replace)

After login, we obtain another message, the motd (or message-of-the-day). Usually, this is used to give last-minute information to the users about the status of the system (for instance, contact information or system news).

Locate such file, and change it to report on how to contact with the system administrators.

Which file have you modified?

1.5.4 Network configuration

The next step in the lab session is to configure the network. This means that once this stage is complete, your system should be capable of communicate with other systems via the IP protocol. First, we will do the network configuration by hand and then we will use DHCP to configure it permanently.

Parameter	configuration value
IP address	
Mask	
Gateway	
DNS Server	

Manual configuration

The manual network setup usually involves three steps:

1. Configuring the network interface using the **ifconfig** command
2. Configuring the routing table using **route** command
3. Setup name resolution in /etc/resolv.conf

Check the *man pages* that correspond to these commands and files.
Which interfaces are configured in your system?
What is the full list of interfaces available? (include the unactive ones)
To configure properly the network, have in mind this data:
The machines you are using have several network interfaces. Configure the inteface corresponding to the Gigabit Ethernet (eth1000e).
Which command do you use to bring the network interface up?
Now you have to add the default gateway to the routing table. Which command do you use?
Finally, create the file /etc/resolv.conf with the DNS information. How can we check that we have correctly configured our network?

Permanent configuration

Now we want the network to be configured properly at boot time and do not have to do it manually each time. First, bring the interface down using:

```
# ifconfig eth# down
```

Use *ifconfig* to verify that the interface no longer comes to the list of active interfaces.
On Debian (and other systems) the network configuration resides in several files in the /etc/network directory.
Which initialization scripts use such files in /etc/network?
In particular, we are interested in the interfaces file that is where you configure different interfaces. Right now there is only configured the loopback interface. Add an entry in the interfaces file to configure your network interface with the parameters you used previously. First add a line to indicate that we want to activate the interface automatically at boot:

```
auto eth1000e
```

After indicating that, we will give all the necessary parameters to configure the interface:

```
iface eth1000e inet static
```

And immediately all the necessary parameters (except the DNS server):

```
address 10.10.41.XX
network 10.10.41.0
netmask 255.255.255.0
gateway 10.10.41.1
```

Now, to check that we have configured it correctly, we could do a reboot. But in fact we do not need that. We can use the commands **ifup eth1000e** and **ifdown eth1000e** to tell the system to reconfigure an interface according to the interfaces file.
Once you have made it work, now we want to obtain the network settings the system automatically using DHCP. Check the manual interfaces file (**man interfaces**) and use DHCP to configure eth1000e.
What changes you made to /etc/network/interfaces?

1.6 Bibliography

More information about installing Linux can be found in [1, 2]. A detailed description of the Linux boot process can be found in [3]

Laboratory 2

Application Management

2.1 Objectives

- Install software ready to use in a specific operating system (binaries)
- Install software starting from source code

2.2 Before you start

it would be good if you can think about these questions, and answer them.

- Which command can you use to connect to a ftp server?
- Which **ftp** command lists the contents of a directory in the server side?
- Which **ftp** command gets a file from the server?
- Which **ftp** command allows to get more than a single file at once?
- How can we list the contents of a tar file?
- What should we do if the file is gzipped?
- How can we extract the contents of a tar file?
- How about a tar.gz file? And a tar.bz2?
- How can we create a link to a file?
- How can we create a softlink to a file?
- What is the use of the PATH environment variable?

2.3 Introduction

The process of installing software on an operating system is essentially the copy of files that make up the application in the appropriate directories and, optionally, the modification of some parameter settings of the application.

2.3.1 Software management systems

Since the introduction of dynamic libraries, before installing a particular software, we have to take care of installing its dependencies. Dependencies consist of other software that has to be installed in advance, because it is used by our new application. To help administration of dependencies, several software management systems have appeared. These systems keep track of the libraries and applications installed, so that they allow easy install, update, and uninstall software.

Specifically, our ASO Linux is based on Debian. Debian organizes its software in a series of compressed files (packages similar to tar.gz files). Each file can include binaries, libraries, configuration files, manual pages, other documentation, and moreover, contain information on dependencies with other packages. Debian packages have a .deb extension.

2.3.2 The UNIX graphical environment: X-Window

The X-window system (or X11, or X) is a protocol to display graphics, providing a standard toolkit for building graphical user interfaces (GUI). X provides the basic frame of reference, but does not define the user interface. This is left to client programs. Further, X makes use of a client/server model that communicates the X server, locally or networked with client programs. The server accepts requests for graphical output (windows) and sends back to user the input received from the peripherals (keyboard, mouse, or other).

The X system has no specification on the specifics of the user interface such as: buttons, menus, etc. Instead, the software applications are responsible for the appearance of their windows. To allow that several diverse applications have a similar looking, they usually rely on the services of window managers, and desktop environments.

There are different implementations of the X-window system for Linux (and other UNIX systems). The most common and the one that we use is called X.org. In addition to the X server, window managers and desktop environments also require the installation of their own packages.

Window manager : is responsible for controlling location and appearance of the windows of graphical applications. There are many window managers with different functionalities. Some of them are: Metacity, Kwin, enlightenment, or Compiz

Display Manager : allows to start a new session in the machine. The display manager screen presents the user with a login and password validation. Therefore it performs functions like the `getty` and `login` programs do on character mode terminals. Some common managers are: `XDM` (the X Window Display Manager), `GDM` (Gnome Display Manager) and `KDM` (KDE Display Manager). The Display Manager is a service that can start and stop as the rest of system services using startup scripts that are in the /etc/init.d dirctory

Desktop environment : provides a unified interface to the user for applications with graphical icons, tool bars, backgrounds, etc. The desktop environment typically consists of a window manager, a screen manager and its own a set of applications and libraries. The most common desktop environments are KDE and GNOME, but there are many more.

Table 2.1 shows a list of several desktop environments with its corresponding window and display manager

Desktop Envoronment	Window Manager	Display Manager	Graphical library
GNOME	Metacity	GDM	GTK+
KDE	Kwin	KDM	QT
XfCE	Xfwm4		GTK+
LXDE	Openbox	LXDM	GTK+

Table 2.1: Desktop Environments and its depending window and display managers

2.4 Installation of binary packages

2.4.1 Manual installation

We want to install the make application into our system. First, we need to get the software to install. The packages you will need are in the ASO ftp server.

What is the ASO ftp server's IP?

To access the FTP server use the following command:

```
$ ftp ip_servidor
```

Access to the server, and go into the packages directory. Download the appropriate package containing the `make` command.

To install a package .deb, use the `dpkg` command (Debian PacKaGe). The following command should work:

```
$ dpkg --install <file.deb>
```

Read the messages that `dpkg` prints during the process and ensure that no problems are reported. You should get used to such kinds of messages.

The `dpkg` command also allows to obtain information about installed packages, and files, and to uninstall packages. Please see the help provided by the `dpkg` command itself and/or its man page, and complete the following table:

Action	Options	Arguments
Install a package	-i or –install	
Uninstall a package		
Purge a package		
List package		
List files in a package		
Find which package a file belongs to		

Table 2.2: package management with `dpkg`

What is the difference between uninstall and purge a package?

Installation of the X-window system

Now, use `apt-get` to install an X server. The package you install is `x-window-system` or `xorg`. Note also that the `apt-get` tool installs all the dependencies needed and asks you the questions necessary to configure the X server.

What command have you used?

In addition to an X server, we need a window manager and a desktop environment. If we do not know of anyone, we can search the database package.

Another interesting tool of APT is `apt-cache`. `apt-cache` searches among the information that the system got from the repositories after being updated. Using `apt-cache` you can find the desktop environments we have available to be installed in the system.

- What command have you used?
- List some of the desktop environments you have found

Now we want to install the `lynx` program (a text-based web browser), and `lftp` (an advanced ftp client). Download their packages from the ASO server, and install them with the `dpkg` command.

Execute the `lynx` and `lftp` commands to ensure that they work correctly. Solve any problems you find.

2.4.2 Installation with a package manager

Package managers are useful to facilitate the installation of large applications (which usually have many dependencies) and also make it easier to keep systems up to date.

Debian has a toolset, called APT (advanced front-end for dpkg) that you can use to search, download and install software and all its dependencies, and keep the system updated in an easy and convenient way. There are also several graphical front-ends (`Synaptic`, `Adept`, ...) that we will not use in this course.

Configuring the software repositories

First we need to correctly configure the APT repositories from where the manager can get the .deb files to install in the system. These repositories can be on remote servers or even on our server (e.g. in a cdrom) and we can have configured as many of them as we want.

APT configuration files are in */etc/apt*. Inside this directory we will create a *sources.list* file, with the following contents:

```
deb ftp://ip_servidor/debian/ unstable main
```

Now, we have to force our system to get the list of packages available in the newly configured repositories, and all the information related.

This is the appropriate command:

```
$ apt-get update
```

The `apt-get` tool is also useful, among others, to install, update, and uninstall packages.

Which command (and parameters) will we use to update all our installed packages to the more recent version available?

Please, update all your packages to their last available version.

For more information on a specific package (its description, its dependencies, ...) you can use:

```
$ apt-cache show package-name
```

Select a window manager and a desktop environment to install, and get them for your system using `apt-get`.

Which commmand have you used?

Sometimes the default package configuration is not doing well, or the configuration files of a given package can get damaged by an error. In these cases you will need to reconfigure the package and generate the configuration files again. The APT system has a dpkg command to do this:

```
$ dpkg-reconfigure package-name.
```

If you have problems with the X system configuration, you can use this command to reconfigure the X server.

Now install the following packages: `gcc` (compiler), `libc6-dev` (development libraries) and `iceweasel` (Firefox web browser).

When you finish, please run the following command:

```
$ apt-get clean
```

• What is this command doing? • What is the difference with the command apt-get autoclean?

2.4.3 Installation of pre-compiled binaries

Sometimes we want to install software that is not (for whatever reason) into a package on our repositories. In this exercise, we are going to install several versions of the Java SDK (JDK).

Download the files corresponding to the java installation from our ASO FTP server. They are in the packages directory. To uncompress files, simply execute the downloaded files.

Initially we want to install version 1.6 (jdk-6-linux-i586.bin) into */opt/java1.6*.

• What commands do you use to uncompress the file? • In which directory did you get the uncompress files?

Look at the contents and locate where the executable `java` (the one executing the java virtual machine) is. Now move this directory to the destination directory */opt/java1.6-*

Verify that it is properly installed:

```
$ /opt/java1.6/bin/java -version
```

Now repeat this step for JDKs 1.4 and 1.5, installing them into */opt/java1.4* and */opt/java1.5* respectively.

Now, if we try to find out which java version is the default:

```
$ java  -version
```

• Which is the error reported? • Why do we get such an error?

The easiest way to solve this problem is to make a softlink from one of directories that are in our PATH into the binary that we want to be accessible.

Create a softlink from */usr/bin/java* to */opt/java1.6/bin/java*.

Which command have you used to create the softlink?

Now, in addition, we want that each version could be directly accessible with the commands javaVersion (e.g., java1.6)

Which commands do you use to achieve this?

2.5 Installation from source code

Sometimes we have to install an application directly from the source code, either because the package does not exist in the repositories, or because we want the installation of the application to suit somehow to our system.

As an example, we are going to install a small restricted shell that will be used in other lab sessions. Download the *asosh-0.1.tar.gz* from the sources directory in the ASO FTP server.

A usual place to put the source code is in */usr/src*. Uncompress the source code with the `tar` command in that directory.

Which command did you use?

Look at the contents of the directory with the source code. Usually you will find a script named "configure", that will let you configure parts of the compilation and installation processes (enable / disable parts of the code, choose the installation directory, ...). The specific information about this script can usually be found in the INSTALL and README files.

By default, asosh will be installed in */usr/local*. Run the configure script properly to install it in */usr/local/asosh*.

Which parameters did you use?

Note that the test for required libraries gives an error because they are not installed.

- Which is the error reported?
- What is the reason for this error?
- How did you solve the problem? (hint: remember that headers are usually in a separate package)

Once the configure is completed succesfully, the next step is to compile the source code (please check that there are no errors reported when compiling):

```
$ make
```

In general, for these two first steps we do not need special permissions, so it is recommended that you do them within an account other than root. Instead, the last step consists of placing the binaries and other files (configuration data, libraries, ...) into the final location where we want them installed. This usually requires root permissions.

```
$ make install
```

Cerify that everything is installed correctly by running the command `asosh`.

During the compilation process several temporary files should have been generated (e.g., object files). So, once the installation is completed, it is a good idea to delete these files. The Makefile usually contains the rules needed to do that easily.

What command do you use to delete temporary files?

Moreover, usually the Makefile also incorporates rules to undo all the steps made in the process of installation.

Which argument can be supplied to do this?

Laboratory 3

Scripts

3.1 Introduction

Usually operating systems administration tasks should be repeated again and again, so the administrator must enter new orders, sometimes changing only one input parameter. Doing these tasks manually, not only implies a considerable investment of time, but exposes the system to errors when repeating a command in a wrong way. The automation of these tasks using scripting languages improves system efficiency as these are done without human intervention; increases reliability because the commands are executed in the same way each time and also ensures consistency in the execution beacuse these tasks can be easily programmed to run periodically.

Although the automation could be made in any programming language, there are some languages, known as scripting languages, which allow to combine easily system commands with the expressions of the scripting language itself. Additionally they facilitate the manipulation of text files, lists, and directories and other useful tasks for system administration. There are many scripting languages available: the associated to the shell (like Bash or C shell) and other features, such as Perl or Python, with more advaned functionality.

3.2 Objective

Learn how to automate common system administration tasks using scripting languages like Bash and Perl.

3.3 Before you start

- Review basic programming with Bash shell-scripts
- Analyzing the basic Perl constructs.

The practice consists of two parts: the first consist of anlyzing a sample scripts for the detection of "unnecessary" users in the system. In the second part consist of making a script for managing disk space.

Scripts can be done in Bash or Perl. In addition to proposed scripts a final list of problems is presennted for practicing outside the laboratory.

Keep in mind all the time properties of a good script [4]:

1. A script should run without errors.
2. It must perform the task for which it is intended.
3. The program logic must be clearly defined.
4. A script should not do unnecessary work.
5. Scripts should be reusable.

3.4 Script to detect invaliad users

You are asked to make a script that determines which users in */etc/passwd* are invalid. A user is invalid if he/she is in the *passwd* file but did not have any presence in the system (ie. it has no files). Also, there are users that have no files, but that are used to run system daemons. Add an option to declare valid users those that have a running process (-p flag).

3.4.1 Description of the desired results

An example of script's output without and with the -p flag is presented below

```
$./BadUsers.pl
daemon
bin
sys
sync
games
lp
mail
news
aduran
alvarez
proxy
backup

$./BadUsers.pl -p
bin
sync
games
lp
news
aduran
alvarez
proxy
backup
```

3.4.2 Perl version of the script

Fill in the empty spaces with the required Perl expressions:

Input Parameters

First of all we need to detect the input options and make some error control. Initally we should accept only one parameter -p

```
1  #!/usr/bin/perl
2
3  sub print_help{
4    print "Usage: $0 [options]\n";
5    print "Possible options:\n";
6    print "-p : validate users with running processes\n";
7  }
8  if ( $#ARGV < 1 ){
9    &print_help;
10   exit(1);
11 }
12 while ( $#ARGV + 1   ){
13   if ($ARGV[0] ____________) {
14     shift;
15     $p=1;
```

```
16   }else{
17     &print_help;
18     exit(1);
19   }
20 }
```

Read the password file

Now we need that the script opens the user database and read stores all the field in a list

```
21 $pass_db_file="/etc/passwd";
22 open (FILE,$pass_db_file) or
23   die "failed to open file $pass_db_file: $!";
24 @password_db= ____________; # Read all the file
25 close FILE;
```

What are the actions performed by the lines 2 and 3 on the previous code?

Analysis of the user database

Now go through the user databse and search system, for each user, the files that they own in the whole system using the *find* command (see *man find*). The users that do not own any file are going to be stored ina hass as invalid users in order to prcess them later.

```
26 foreach $user_line (@password_db) {
27   ___________________ ; # remove the line break
28   @fields = split(':', $user_line);
29   $user_id = $fields[0];
30   $user_home = $fields[5];
31   if (-d $user_home){
32     $command = sprintf("find %s type f user %s | wc -l",
33                        $user_home, $user_id);
34     $find_out=`$command`;
35     ___________________ ; # remove the line break
36   } else {
37     $find_out = 0;
38   }
39
40   if ($find_out == 0){
41     $invalid_users{$user_id} = $user_home;
42   }
43 }
```

- What is the exact effect of the find command in lines 33 and 33?
- What is the difference between the functions *chomp* and *chop* ?
- What is the purpose of the sentence in line 41?

Users with processes in execution

Now add the option (-p) to not include in the list of invalid user those that who have running processes. To find users who have processes running we use the command *ps aux –no-headers* (see *man ps*) and remove them from the hash of invalid users. Users who are listed as disabled will be those that have no file or running process. Use a regular expression as a delimiter of fields that came from the output of out of the *ps* command. Note that you can find one or more spaces and tabs as separators.

```
44 if ($p == 1){
45   @process_list =`ps aux --no-headers`;
46   foreach $process_item (@process_list){
```

```
47      chomp ($process_item);
48      @fields_proc = split (________, $process_item);
49      $user_proc =  $fields_proc[0];
50      delete($invalid_users{$user_proc});
51    }
52  }
53
54  foreach $inv_user (sort(keys %invalid_users)){
55    print $inv_user -> $invalid_users{$inv_user}\n";
56  }
```

- What is the purpose of the delete function in line 50.
- What is the effect of the sort function in in line 54.

3.4.3 BASH version of the script

now you have the BASH version of the script. Fill-in the empty spaces with the appropriate language constructs.

```
1  #!/bin/bash
2  p=0
3
4  function print_help
5  {
6    echo "Usage: $1 [options] "
7    echo "Possible options:"
8    echo "-p validate users with running process"
9  }
10
11 if [ $# -lt 1 ]; then
12   print_help $0
13   exit
14 fi
15
16 while [ $# -gt 0 ]; do
17   case $1 in
18     "-p")
19       p=1
20       shift;;
21     *) echo "Error: not valid option: $1"
22       exit 1;;
23   esac
24 done
25
26 for user in _____________; do
27   home=`cat /etc/passwd | grep "^$user\>" | cut -d: -f6`
28   if [ -d $home ]; then
29     num_fich=`find "$home" -type f -user $user | wc -l`
30   else
31     num_fich=0
32   fi
33
34   if [ $num_fich -eq 0 ]; then
35     if [ $p -eq 1 ]; then
36       user_proc=__________________
37       if [ $user_proc -eq 0 ]; then
38         echo $user
39       fi
```

```
40      else
41        echo "The user $user has no files in $home"
42      fi
43    fi
44  done
```

- What is the purpose of the shift command in line 20?
- What is the mening of the grep command in line 27?

3.4.4 Detection of unactive users

Now extend the previous script to detect inactive users. An inactive user is defined as someone who do not have any running process, that long ago have not login (see commands *finger*, *last* and *lastlogin*), and that long ago have not changed any of their files (see time options of *find*). The period of inactivity should be indicated through a parameter:

```
$./BadUsers.pl -t 2d (indica 2 dies)
alvarez
aduran
xavim
marcg

 $./BadUsers.pl -t 4m (indica 4 mesos)
xavim
marcg
```

Modify the script to include the support for the new option in order to detect inactive users

3.5 Script for disk space management

Make a scripts that computes the disk space used by each user on the whole system. If the amount of disk space exceeds certain space that is passes as a parameter, then wirte a message to the in question to inform him/her to d delete or compress some files. Specifically, the syntax of the program should be the following:

```
$ disk-usage.pl <space_limit>

Per exemple:
$ ./ocupacio.sh 600M
       root        567 MB
       alvarez     128 KB
       aduran      120 MB
       xavim       23 MB
       ( ... )
```

After this extend the script to add an option for groups: *-g*: With this option the script must return the total disk usage for the each one of the users in the specified group, the total disk usage of the whole group, and put a message to users that exceeds the defined limit.

Therefore, the syntax of the final program will be:

```
$ ocupacio.sh [-g grup] max_permes
Per example:
$ ./ocupacio.sh -g users 500K
alvarez            128 KB
xavim              23 MB
( ... )
```

NOTE: The message should be left on the .profile file. The user must be able to identify and delte the messahe withou problem. This means that alongside the message it should get instructions to remove it without trouble.

3.6 Bibliography

More information about Perl programming can be found in [5]. Two guides of BASH programming are available at The Linux Documentation Project (TLDP) [4, 6]

Laboratory 4

User Management

4.1 Objectives

- Create new user accounts and change their properties
- Disable and safely remove user accounts

4.2 Before you start

Answer the following questions:

- In which files are defined the database for users, the table of passwords, and the group database?
- How UID (user identifiers) can be mapped for new users?
- Which commands can be used to change the owners and permissions of a file? And of all the files in a directory?

4.3 Introduction

At the system each user has an account. An account consist of all the files, resources and information pertaining to each user. User accounts allow the system to differentiate each user's data and processes and allow users to protect their information. For the kernel, users are identified by an integer known as user ID (user identifier or UID). Apart, there is a database associating the UID with a textual name: username. This is the username that the user employs to login. The database includes other information concerning the user like the directory path, the user's full name and the command interpreter (shell).

Creating a new user includes assigning a UID and the modification of the database users to assign their own user settings. Apart, you need at least one group associated with the user and finally to copy configuration file and customization to each user's home directory. Optionally the user can be assigned to multiple groups, allowing system administrator to divide users into groups with different permissions and privileges. This way we can maintain better control over what users can do.

4.4 Profile and user environment

After an interactive login, the shell automatically executes one or more predefined files. Each shell uses different files. The BASH shell runs first the general configuration file */etc/profile* and then run the user configuration files *.profile* or *.bash_profile*. The */etc/profile* allows the system administrator define a common environment for all users, particularly by defining the variable *PATH*. On the other hand *.bash_profile* allows each user to define its own environment adapting the PATH variable, prompt, etc.

When you create the home directory of a user you must copy the files in */etc/skel*. The system administrator can put files in */etc/skel* to provide a basic environment for users. For example, the administrator can create a */etc/skel/.bash_profile* with some basic definitions that, later on, the user may change.

To make life easier for users to modify the file *bash_profile* that there are at */etc/skel*. Verify that the PATH of all users contains the */usr/local/bin* directory and modify the PATH to include a bin directory located on each user's home directory *$HOME/bin*.

Also change the prompt of the system in such a way that it includes the username, the current date and finally the character ">" . For example, the prompt for aduran can be be "$aduran(SatApril10) >$".

- What are the changes you applied to the PATH variable?
- In which environment variable the prompt is defined?
- What are the changes that you applied to the prompt variable?

4.5 Creating users by hand

Now create an user account for each member of your laboratory group. Before doing it define the parameters for each account. Those users should be in the `admin` group.

Parameters	User 1	User 2
UID		
Username		
HOME directory		
Shell		
Groups		

Table 4.1: Parameters for new users

Edit the user database in order to add the new users. User the `vipw` command to edit this file.

What is the difference between using the `vipw` command and edit directly the database with a text editor like `vi`? (Suggestion: Open two `vipw` sessions at the same time)

In a similar way, use the `vigr` to edit the group database in order to create and modify groups.

- What groups you have created?
- Which users are part of that groups?

For security reasons it is better to deactivate the user account until all the creating process has finished.

How you can deactivate an user account, in such a way that the user can not make login?

Deactivate the new accounts that you are creating until you finish all the process.

Create the *home* directory for each user. Copy all the files from the */etc/skel* directory, and assign the correct owner, group and permissions to the new *home* directories.

- What commands and parameters you have used to change the owner of the *home* directory
- What commands you have used to change the file permissions?

Now, assign a password to the new users. For security reasons the password is not written directly in the user database- Instead there is another file, called shadow */etc/shadow*, for putting the password and some parameters related to it.

- What are the security risks associated by putting the password in the user database?
- Which command can be used to edit safely the *shadow* file?
- What is the meaning of the password parameters defined in the *shadow* file?
- what command can be used to modify those password parameters?

To edit the user account and other parameters you can use the commands `chfn` and `chsh`. Use these commands to assign appropriate values to the accounts you have created.

4.6 Automatic creation of users

Most Linux distributions include software to automate the task of user account creation and modification. Some of these tools are `useradd` and `adduser`. They allow to create and assign various parameters required for new user accounts.

Use these commands to register the following accounts:

- Professors: aduran, alvarez, xavim
- Students: student
- Other users: four accounts for four different groups of practices. The username of these accounts will be: asoXX. Where XX is a two digit identifier of a group practices.

Choose and justify the most appropriate username for all users. Analyze that some users may require usernames and directories.

The permissions for each of these user groups (teachers, students, administrators, and other groups as necessary) are defined as follows:

- Teachers will have access control at group level to all files of all users defined.
- Students will have access control at group level to all files of all users, except of teachers.
- Administrators have access control at group level only to the files of their own group.
- Other users (asoXX) will have not access control at group level to any file of any group.

Note that the above conditions specify only access levels. So, you don't need to modify the protections of the directories or files.

How you defined these levels of access without changing permissions of home directories?

4.7 Removing and disabling users

In order to remove a user account it is necessary to delete all files in the mailboxes, print jobs, `cron` and `at` jobs and all references to the user. After that you can delete the lines associated with the user and group databases. As a user may have files outside their home directory you need to search the entire directory tree for files belonging to the user and remove them. Before removing the files a good practice is to make a backup of them.

Now we want to delete one user account, but first we need to make backup of all the files and then delete them.

- Which command(s) you can use to make the backup of all the files of a user? (Hint: *xargs* may be useful)
- What's wrong with the files that have spaces in their name? How you can resolve this? (Hint: see options of `xargs` command or the -exec option of `find`)
- Which command you can use for searching all files of a user and delete them?

It's a good security practice to disable the user account before deleting files and other steps of removing the account.

One way to disable an account is to invalidate the password by changing the shell of the user for a simple program. This program just writes a message giving information to the user of the reasons that its user account has been deactivated. This program is called a 'tail script and looks like:

```
#!/usr/bin/tail -n 2
This account has been closed due to a security problem. Please
contact the system administrator.
```

This script can be assigned as a shell to a user employing the command `chsh` and it can be stored in a separate directory like */usr/local/lib/no-login*.

Use the `chsh` command to assign a tail script in order to deactivate the account of one of the users asoXX.

How you can check that the account has been deactivated?

Now create a script that given a username makes a backup pf all the files of the user, remove all the files of that user, and finally changes the shell for a tail script.

What is the content of this script

4.8 Special users

Some Unix commands, like `shutdown` used to turn off the machine, can only be executed by the root user. In many cases, however, it is necessary that other users can also turn off the machine, but without having access to root privileges. To achieve this, you are asked to create an account used to run a special simplified shell that will allow to run `shutdown` and other special commands that require super-user permissions.

Create a new user with `asosh` as username. Set an appropriate password

When someone makes a login for this account, it is going to run the `asosh` shell that you have installed in the laboratory about application management. For security reasons you should make sure when this user make login it does not execute any shell, apart from the restricted shell `asosh`.

- What are the required set of permissions for this application in order to avoid a direct execution by any user?
- What is the appropriate values for the entry in the user database for `asosh`?

4.9 Sudo and control of application execution

Like `shutdown`, there are other management commands that can only be executed by the root user. It is a bad security practice to use the root account to execute these commands. To solve this is problem you can use the `sudo` command. `Sudo` allows a user to execute a command with root permissions. The configuration of what applications can execute a particular user is defined in the */etc/sudoers* (See `man sudoers`. This file can be edited safely using the command `visudo`.

Make the required changes in the *sudoers* configuration in order to allow the member of the admin group can execute all commands as superusers. Additionally make the necessary changes so that users in the teachers group can execute the script to delete users created in a previous section and all the binaries in the */usr/local/teachers/bin*. Verify that it works by running with `vipw` command.

What changes are required in the */etc/sudoers* file to enable the configuration described above?

Finally disable the root account so you can not login as root. The administration commands should be executed only from the users in the admin group using the `sudo` command

Make sure you can execute administrative commands before disabling the root account

What are the steps required to disable the root account?

4.10 Bibliography

More information about user management can be found in [2] and [7].

Laboratory 5

Backups

5.1 Objectives

Learn how to design and implement backups using basic UNIX tools.

5.2 Before you start

For this session you should be able to answer the following questions:

- How you can pack and unpack one or more files using the `tar` command?
- What is a hard link?
- What's the difference between copying two files (`cp file_a file_b`) and creating a hard link (`ln file_a file_b`)?

5.3 Introduction

One of the most important tasks of a system administrator is performing backups that can restore the complete system in an acceptable amount of time when a system failure that entails data loss occurs. These losses may be due to multiple factors such as hardware failures, software malfunction, human action (accidental or premeditated) or natural disasters [2].

Before making backups the system administrator should decide a policy taking into account aspects such as [7]:

- Select the correct type of physical media for backups taking into account the size, cost, speed, availability, usability and reliability.
- Deciding which files need a backup and where are those files. The most important files are configuration files and user files (usually located in */root* and */home*, respectively). Some files that do not require backups are the temporary files (*/tmp* and the system binaries (*/bin, /sbin/*).
- Decide the frequency and scheduling of backups. This depends on the variability of the data. A database may require multiple daily backups, while a web server can require only one daily copy, and other file systems may require only one weekly copy
- Analyze other aspects such as: where to store the copies to, how long the copies should be maintained, and how quickly you need to retrieve each file type.

Using the information above it is possible to decide a backup strategy. This includes deciding the frequency and type of copies. A common strategy is to make full and incremental copies. In this way, on one hand it is possible to reduce the time and size of data transfers, but, on the other hand, it also increases the complexity of the data restoration process.

A typical strategy is to make full copies weekly (also known as level 0) and incremental copies (known as level 1 or greater) daily. If the ratio of file change is very large we can modify the previous weekly model

for a model where each month there is more a copy of level 0, each week a copy of level 1 (incremental weekly) and every day a copy of level 2 (incremental daily).

Finally, you must decide the most suitable tools to implement the backup strategy that has been designed. In this laboratory we will use `tar` and `rsync`. Also, we will use the same disk as the physical environment to make backups. Take into account that in a real environment this is inconvenient because of the high risk that a data loss also affect the backups.

5.4 Partition to store the backups

To save the backup files that we will generate during the laboratory, create a new directory */backup*. Also create a new partition on the free space that you have and, finally, create an ext-3 filesystem on it. Mount this partition in the */backup* directory so that only root has access permissions. Other users should not even be permitted to read the directory, given that the contents of the backups could be confidential.

What commands you have used to create the partition, format the filesystem, mount the partition and change the permissions of the */backup* directory?

To add more protections in this directory, it should be mounted on write mode only when writing the backups and the rest of the time in read-only mode. Normally, it should be necessary to unmount the partition before changing the write mode but you can change the mount options of a partition without unmounting it using the `remount` option.

```
Mount in read-only mode
# mount -o remount,ro /dev/usbX /backup

Mount in read-write mode
#mount -o remount,rw /dev/usbX /backup
```

What do you need to do in order to mount this new partition automatically in read-only mode at boot-time?

5.5 Making backups using `tar`

5.5.1 full backups

Make a complete copy of the */root* directory (check that there are files in this directory) using the *tar* command. Use meaningful names for backup files: include information about the content, date and time that the backup, and if the backup is full or incremental, etc.

- How you can include automatically the date in the backup file name? for example, *backup-etc-level0-200912041030.tar* (Note: Use the *date* command.
- What command you have used for making the full copy of the directory */root*?
- Why is not good in terms of security to compress the backup file?
- If we want to compress the backup file which option you have to add to the `tar` command?

Sometimes when you do the full copy it is necessary to exclude certain files. In order to do so you can construct a file with a list of files that should be excluded from the backup. Create again the full copy, but this time exclude from the copy the files listed in the *excludes* file. (Create this file and put some file names in there).

What command you have added to the `tar` command to exclude files?

In addition to protecting the backup directory is important to have a mechanism that allows to verify that the backup files have not been modified after their creation. For this task it is common to use a digital signature mechanism, such as MD5, which allow to verify the integrity of a file.

Once you've made the copy, use the `md5sum` command (use `man md5sum` to learn how to use this command) to generate a MD5 hash and store the output in a file. Put a *.asc* as a file extension.

How you have used the `md5sum` command to produce the md5 signature?

5.5.2 Incremental backups

To perform incremental copies modify some files in the */root* directory:

-
- Create new files and subdirectories
- Modify the contents of some files
- Use the `touch` command to change the modification date of some files.

For making incremental backups the `tar` command has the `--newer` option which creates a file that only includes the files that have been modified since a certain date. This date can be specified in two ways: first, placing it directly in the command line eg `--newer ="2007-11-28 12:10"`. The second way is to take the date from a file, in that case the date is taken from the last modification data of that file, for example: `--newer=./file`

Now make an incremental backup of the */root* directory with respect to the complete backup that yo made before using the `tar` command. Additionally create a `md5sum` signature and save it to a different file

- What command you used to make the incremental copy?
- What potential problem has the use of the full backup file for obtaining the backup date when doing the incremental copy? (Note: take into account that the backup process can take a long time). How you can solve this problem?

Now, perform a second round of changes to the */root* directory in order to perform a second incremental copy. Again:

-
- Create new files and subdirectories.
- Modify the contents of some files.
- Use the `touch` command to change the modification date of some files.
- Delete some of the files that you generated for the first copy incremental copy.

Make a second incremental copy of the */root* directory (with respect to the first incremental copy) using the `tar` command. Also create a `md5sum` signature of the second incremental copy and put to it an appropriate name.

- What command you have used to create the second incremental copy?
- How can you verify that the content of the backup is the same as the original directory?
- How can you verify, using the `md5sum` command, the integrity of the backup? That means that the file has not been modified since it was created.

5.5.3 Restoring a backup

Rename the */root* directory to */root.old* to simulate the effect of a data loss. Now restore the directory from the backup (that means to restore the three files that were created: the complete copy, and both incremental).

- In what order you should restore the files to the get the desired result?
- What commands you have used?
- What happened to the files that you had deleted before the second incremental copy?
- How you can detect that some files that have been deleted? When those file will be removed from the backups?

Restoration of a fragment

Rename one of the subdirectories of the *root* directory to simulate that it was deleted. Then, restore only that particular subdirectory from the backup.

What commands you have used to retrieve only a portion of the backup?

5.6 Making backups using `rsync`

Until now we have stored the backups in the same machine that contains the data, but what is most common is to have several machines to backup and store the copies on a central machine using the network. To do this we can use the `rsync` command. It allows to copy a directory (or set of files) to another directory via a network connection. `rsync` uses an efficient checksum algorithm to transmit only the differences between the two directories while, at the same time, compressing files for a faster transmission

This tool lets you copy files from or to a directory located in a remote machine, or directories from the same machine. What it does not allow is to copy directories between two remote machines. Moreover `rsync` allows copying links, devices, and preserve permissions, owners and groups. It also supports exclusion lists and remote connection using Secure Shell (SSH) among other possibilities (for more information see `man rsync`).

5.6.1 Making backups over a network

As mentioned earlier, `rsync` to back up a remote machine. This can be done with `rsh`, or by putting `rsync` in server mode, but it may be dangerous because a local machine in the network could be capturing the data of the connection. In order to solve these problems `rsync` allows secure connections using `ssh`.

What are the required steps for activating the `ssh` server?

5.6.2 Making full backups

Create a directory to make the `rsync` backups in the */backup* partition and then execute the following command: (Note: For the following command to work well it is necessary to activate the root account and put it a valid password)

```
# rsync -avz /root -e ssh root@localhost:/backup/rsync-backup/
```

What is the meaning of the options "avz" passed to `rsync`?

Now, create a file in the */root* directory, and try to do the same `rsync` command as before. Then, delete the file and execute again the `rsync` command.

- What happened to the deleted file?
- What option of `rsync` allows to an exact synchronization the two directories?
- How you can make a copy of all the files in the */root* directory except those that have a *.txt* extension?
- What's the difference between making `rsync /source /destination` and `rsync /source /destination/`?

5.6.3 Making reverse incremental backups

As seen in the previous section, every time you make a copy and synchronize, the directory where we you have the mirror is exactly like the source directory. This is a problem, in some situations, because it does not allow control over the changes made to the files. To solve this you can use use the `--backup` and `--backup-dir` options of `rsync`. The backups generated with these options are called inverse because the full copy is latest, as opposed to `tar`.

Here is a simple script to do reverse incremental backups with `rsync`. Fill it with the corresponding data.

```
 #!/bin/bash
SOURCE_DIR=
DEST_DIR=

# Excludes file: list of files to exclude
EXCLUDES=
# the name of the backup machine
BSERVER=
# put a date command for: year month day hour minute second
BACKUP_DATE=

# options for rsync
OPTS="--ignore-errors --delete-excluded --exclude-from=$EXCLUDES \
   --delete --backup --backup-dir=$DEST_DIR/$BACKUP_DATE -av"

# now the actual transfer
rsync $OPTS $SOURCE_DIR root@$BSERVER:$DEST_DIR/complet
```

Now create a file in the source directory and synchronize the backup with the script described above. Then, modify the new file and synchronize again. Finally delete the file and synchronize again.

- What happens to the backup when the new file is modified?
- And, what happens when it is deleted?

5.6.4 Snapshot-Style Backups

One possibility that gives `rsync` is to make incremental backups where, using the properties of hard links, incremental copies appear like full copies [8].

First, we are going to analyze some properties of hard links.

Review of hard links

The file name does not represent the file itself, it is only are hard link to the `inode`. This allows a file (`inode` to have more than one hard link. For example if you have a file called *file_a* you can create a link to it called *file_b*:

```
# ln file_a file_b
```

Using the `stat` command it is possible to know how many hard links a file has:

```
# stat file_a
```

- How you can detect if *file_a* and *file_b* belong to the same `inode`?
- What happens to *file_b* if there are changes in *file_a* content?
- What happens to *file_b* if there are changes in *file_a* permissions?
- What happens to *file_b* if you copy another file, overwriting the file: `cp file_c file_a`?
- And if it is overwritten with the `--remove-destination` option of `cp`?
- And what happens to *file_b* if *file_a* is removed?

The `cp` command has an option (`-l`) to make a copy that is not a new file but a hard link. Another interesting option (`-a`) makes a copy recursively and preserving permissions of access, time and the owners of files.

Snapshot backups with the `cp` and `rsync`

You can combine `rsync` and `cp -al` to create multiple backups that seem full copies of a file system without requiring to spend the entire disk space required for all copies.

In summary it could be:

```
# rm -rf backup.3
# mv backup.2 backup.3
# mv backup.1 backup.2
# cp -al backup.0 backup.1
# rsync -a --delete source_directory / backup.0/
```

If the above commands are executed each day, the directories *backup.0*, *backup.1*, *backup.2* and *backup.3* appear as if they were full copies of the *source_directory* directory today, the day before today, two days before and three days before respectively. But, actually, the extra space will be equal to the size of the *source_directory* directory plus the total size of changes over the past three days. Exactly the same as a complete backup backups with three incremental that we have done before with the `tar` and `rsync` commands. The only problem is that the permissions and owner properties of copies of past days would be the same as the current copy.

There is an option that makes `rsync` to work with hard links directly `--link-dest` thus making unnecessary the `cp` command. Besides, it preserves the permissions and owners of previous copies.

With this option the previous commands will be like this:

```
# rm -rf backup.3
# mv backup.2 backup.3
# mv backup.1 backup.2
# mv backup.0 backup.1
# rsync -a --delete --link-dest=../backup.1 source_directory/ backup.0/
```

Script to make snapshot backups

Use the script `backup-script-snapshot.sh` which is available on the ftp server under the *sources* directory to make backups of the */root* directory. First, modify the variables in the section "File Locations" with the appropriate values to your system. After this, complete the section with the *rsync* command with the appropriate value for the snapshot backups.

How is the `rsync` command in the script to make the snapshot copies?

Now, make changes to files in the source directory (i.e. create a new file, modify the content and date of files, or delete some files) and execute again the script. Do this several times until you have a current copy and three previous copies.

- What happens to the backup after the execution of the script?
- What is the size of the directory /backup.0 and other directories backup directories?

Finally, we are going to do a restore. Rename the directory */root* to simulate a data loss. Perform a restore using this the most recent backup.

- Where is the most recent data?
- Which command can be used to recover the data?

Laboratory 6

Task Scheduling

6.1 Objectives

Be able to schedule tasks at specific and periodic times and dates

6.2 Before you start

For this session you should have the code of the scripts requested in a previous session:

- ocupacio.sh
- badusers.pl

6.3 Task execution at an arbitrary time

What command is used to execute commands at an arbitrary and specific time?

Use this command to schedule the following tasks:

- A deletion of files in the */tmp* directory at 2:00h this evening.
- Make a list of users connected to the server after 10 minutes. This listing has to be stored in a file a the home directory of the root superuser. The filename should reflect the date in which the list has been taken.
- A shutdown of your machine at the end of the class. Send a warning to all the connected users five minutes before the shutdown.

- What commands you have used?
- We want to restrict the use of task scheduling for all the users and allow only the superuser to do do. Which file you have to create. With contents it should have?

Try to use the mentioned command with another user and check that it does not allow him/her to do so.

6.4 Task execution at periodic times

Look at the `man page` of the `cron` and `crontab` and answer the following questions:

- How you can see what are the contents of a user's `cron` schedule?
- How you add new schedule for a periodic task?
- How you can limit who has access to the `cron` service?

6.4.1 Checking the cron schedules

A particular problem that appears when testing `cron` schedules is how to make sure that we have properly specified the frequency without having to wait to the specific time (particularly when is a long time).

- How you can verify rapidly that a `crontab` schedule is correct without waiting?
- What side effects does it have?

6.4.2 Scheduling disk space monitoring

Now we want to monitor the activities of some users (using the script `ocupacio.sh`). The users are listed at a particular file called */etc/ocupacio.users*. The monitoring should be executed every Sunday.

- What entry you have added to the `crontab`?
- Which user is required for executing this entry?

6.4.3 Scheduling user monitoring

We also want to monitor the unneeded users in the system using the script `badusers.pl`. This control will be executed the first day of each month.

- What entry you have added to the `crontab`?
- Which user is required for executing this entry?
- What you have to do for sending by email the list of users produced by the script?

6.4.4 Scheduling Backups

Finally, the user `aduran` wants to do backups of his home directory. Use cron for that purpose taking into account the following conditions:

- A full backup at the first Monday of each month.
- An incremental backup every Wednesday and Saturday
- The backups should be stored at the *home* directory of the user aduran. Obviously don not perform backups of the backups.
- The name of the backup must include the date in which it has been made and the type of backup.
- To reduce the space occupied by the backups, the first day of each month it will be checked if total backup size is greater than 100 MB. If that's the case, you have to delete some backups, as many backups as necessary to be on the space limit. Remove the backups in chronological ascending order (oldest first).
- The user aduran only wants to receive messages if there are errors.

Explain how you have implemented all the above backup scheduling

Laboratory 7

Configuring the DNS server

7.1 Objectives

Being able to configure a DNS server with primary and secondary zones

7.2 Before you start

- What is the configuration file of the DNS server `named`?
- How do you define a primary area?
- What type of DNS record allows you to add a direct translation?
- What type of DNS record allows you to add a reverse translation?

7.3 Using `dig`

The command `dig` is quite useful for solving problems related to DNS. `dig` allow us to make DNS requests from the command line.

```
# dig www.fib.upc.edu
; <<>> DiG 9.3.4 <<>> www.fib.upc.edu
;; global options:   printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53021
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4
;; QUESTION SECTION:
;www.fib.upc.edu.               IN      A
;; ANSWER SECTION:
www.fib.upc.edu.         172800 IN      A       147.83.41.7
...
;; Query time: 28 msec
;; SERVER: 147.83.41.104#53(147.83.41.104)
;; WHEN: Thu May 10 10:59:03 2007
;; MSG SIZE  rcvd: 203
```

In the previous example we made a request for www.fib.upc.edu domain using the configured DNS server. Very often want to make the request directly against another server. This can be done using the @ option.

```
$ dig @server request
```

The-x option can be used to make request for reverse translations:

```
$ dig @server -x ip_address
```

Another option that is very useful is `-t`. With this option is it is possible to make requests for a particular type of DNS records (eg A, NS, MX, ...).

7.4 Configuring the server

7.4.1 Basic Configuration

Download the source code of the `bind` package from the *sources* directory of ASO ftp server. You should unpackage, configure, compile and install this package (*bind-9.4.1.tar.gz*)

Install the DNS server

Create the file */etc/named.conf* and add the following configuration for having a basic DNS server for the localhost domain:

```
options {
     directory "/var/named";
};

zone "localhost" {
        type master;
        file "local.zone";
};

zone "127.in-addr.arpa" {
        type master;
        file "127.zone";
};
```

What is the meaning of each sentence in this file?

The `rndc` tool allows to submit certain operations to control the DNS server (eg. notify it to re-read the configuration file). Before using it you have to configure the communication between `rndc` and the DNS server (`named`). For doing that follow these steps:

Run the command:

```
# Rndc-confgen> / etc / rndc.conf
```

Check the generated file and copy the settings at the end of this file to the *named.conf* file

Configure localhost zones

The file *local.zone* contains the DNS records for the direct mapping of the localhost zone. Create this file with the following content:

```
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
@       IN      A       127.0.0.1
```

The file *127.in-addr.arpa* should contain the mapping for the inverse resolution of the localhost zone. Create it with the following content:

```
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
```

```
                          604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
1.0.0 IN PTR localhost.
```

Start the DNS server

Start the name server using the command:

```
named
```

You can use the -g option for testing purposes. It prints all the output of the server the screen instead of the log files.

To test the server try for example:

```
dig @127.0.0.1 localhost
```

And for the inverse resolution, try:

```
dig @127.0.0.1 -x 127.0.0.1
```

- What IP you have obtained with the direct resolution?
- And what is the answer in the case of the inverse resolution?

DNS forwarders

Due to the faculty firewall your server can not resolve external names directly so you can configure it to access a server that redirects names requests (this type of configuration is known as forwarder).

Edit the *named.conf* and add the following option:

```
forwarders {
   147.83.41.104;
};
```

Once the settings have been changed you can notify the server to re-load the configuration file using the `rndc reload` command.

Now try to resolve an external address as www.fib.upc.edu to see if it is functioning properly.

7.5 Primary DNS server

7.5.1 Changing the default DNS server

First of all, modify your local configuration of the DNS default server in the file *resolv.conf* for using your own DNS server

(Optional) This change only lasts until you reboot because *resolv.conf* will be overwritten by DHCP. To avoid this you can add the following option in */etc/dhclient.conf*:

```
supersede domain-name-servers 127.0.0.1;
```

7.5.2 Configuring zones

We are going to configure the zone "group" for which we will be the primary server. In this zone there are three records:

- serveraso with IP 10.10.41.97.
- self that has your own IP.
- 'we' as an alias of self.

Modify the configuration file properly and create the zone files. Finally, notify the server and verify that you've done to dig properly.

Hint: In case of problems you can use the tools `named-checkconf` and *named-checkzone*. In */var/log/-daemon* you have also some information about the status of the DNS server.

7.6 Secondary DNS server

Now we are going to configure the zone "aso" as a secondary DNS server. In this case is not necessary to create a zone file because it will be automatically transferred by the primary server (in this case the ASO server).

Modify the file *named.conf* and add the description of the "aso" zone as a secondary server.

What changes you have introduced into the file *named.conf*? (Hint: You can use the option `masters` and type `slave`)

Bibliography

[1] M. Kalle and M. Welsh, *Running Linux*, 5th ed. Sebastopol, USA: O'Reilly Media, 2005.

[2] L. Wirzenius, J. Oja, S. Stafford, and A. Weeks, *The Linux System Administrators' Guide, version 0.9.* The Linux Documentation Project. TLDP. [Online]. Available: http://www.tldp.org/LDP/sag/sag.pdf

[3] M. T. Jones, "Inside the linux boot process," IBM Developer Works, 2006. [Online]. Available: http://www-128.ibm.com/developerworks/linux/library/l-linuxboot/?ca=dgr-lnxw06LinuxBoot

[4] M. Garrels, *Bash Guide for Beginners.* The Linux Documentation Project. TLDP. [Online]. Available: http://tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf

[5] R. L. Schwartz, T. Phoenix, and brian d foy, *Learning Perl*, 4th ed. Sebastopol, USA: O'Reilly Media, July 2005.

[6] M. Cooper, *Advanced Bash-Scripting Guide. An in-depth exploration of the art of shell scripting.* The Linux Documentation Project. TLDP. [Online]. Available: http://tldp.org/LDP/abs/abs-guide.pdf

[7] A. Frisch, *Essential System Administration*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2002.

[8] M. Rubel, "Easy automated snapshot-style backups with linux and rsync," 2004. [Online]. Available: http://www.mikerubel.org/computers/rsync_snapshots/