LPIC-1: System Administrator

Linux Professional Insitute

Clive Micallef

LPIC-1 Exam 101

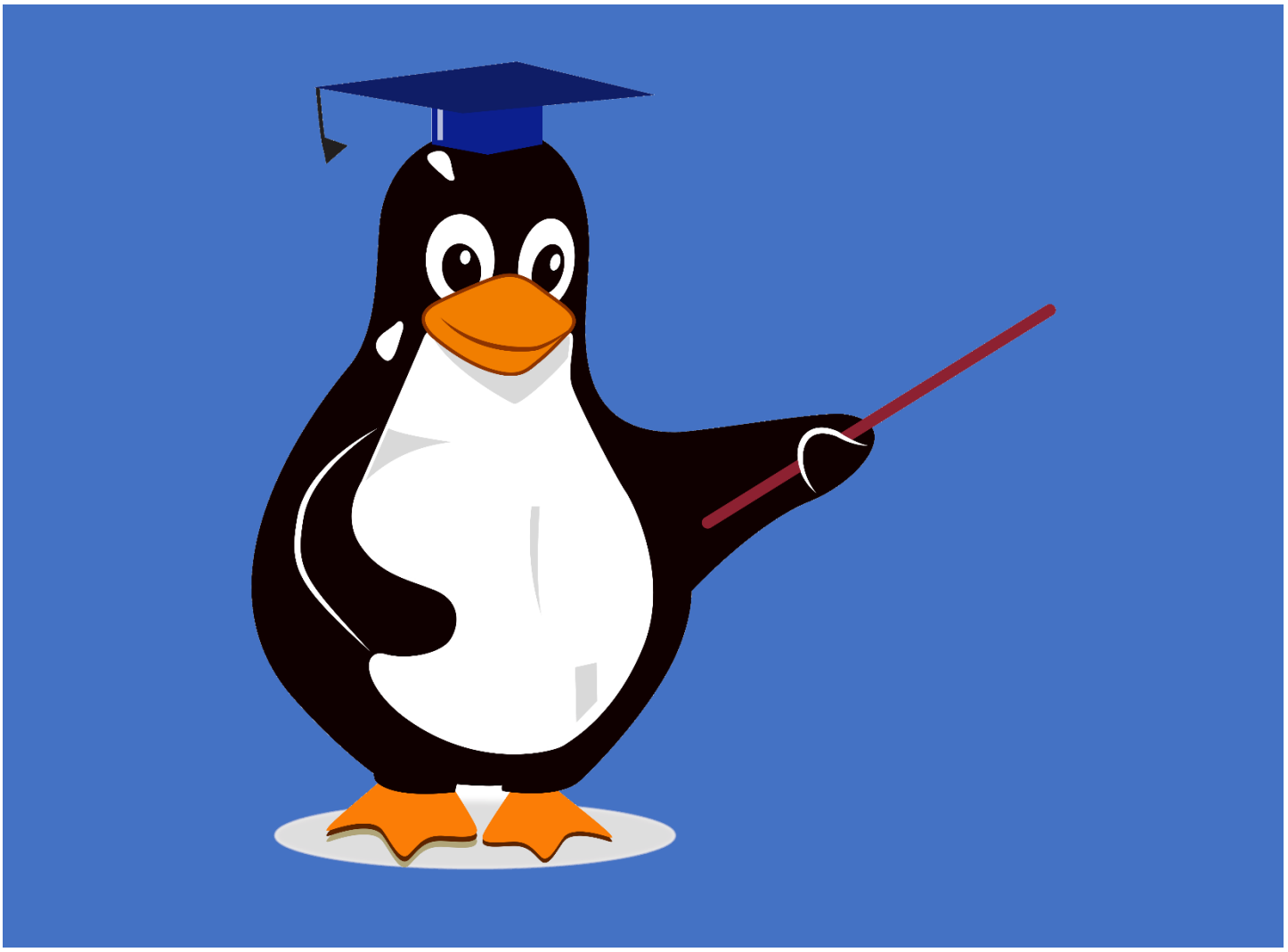


Table of Contents

101.1 – Determine and configure hardware settings – 2

Utilities used to Manage Devices

HAL - Hardware Abstraction Layer

It abstracts your hardware details from you, say and first network care will be eth0. This way Linux will see any hardware as a standard hardware and you will be able to replace the hardware easily.

- **D-Bus** - A daemon. D-Bus stands for "Desktop Bus." Enables processes to communicate with each other and be aware of events occurring on the computer system. (Memory tag: "We all sat on Da-Bus, and had a good conversation as people got on and off.")
- **HAL Daemon (hald)** - HAL stands for "Hardware Access Layer." Provides information to other programs about the hardware available on the running Linux system.
- **hdparm /dev/device_name** - Displays information about a denoted hard drive (device_name).
- **hwinfo** - Displays system hardware information (an overview).
- **Sysfs** - virtual filesystem mounted at /sys, which contains device information. (Memory tag: **Sys** - system device info. fs = file system)
- **udev** - Configured in the /etc/udev files it creates dynamic device files as devices; mounts it and configures it as needed; and alerts other processes.

Files to know:

Note: All can be displayed with the cat command.
/proc/bus/devices - USB devices on computer system
/proc/cpuinfo - system CPU information
/proc/devices - devices installed on computer system
/proc/dma - Direct Memory Addressing (DMA) channels currently used by the Linux system
/proc/ide/*.* - Files with information on the IDE devices on computer system.
/proc/interrupts - IRQ's being used currently by the Linux system (not used if no driver loaded for device).
/proc/iomem - I/O ports assigned on computer system
/proc/modules - Kernel modules used on system
/proc/scsi/*.* - Files with information on the SCSI devices on computer system.
/proc/version - Kernel version
/proc/partitions - Contains major and minor numbers of each partition as well as number of blocks and partition name

- **/dev** directory lists devices as file system
- contains files representing device access points
- devices include, terminal devices (tty), floppy (fd), hard disks (hd, sd), RAM (ram), cdrom (cd)

- Examples: **/dev/sda** designates first hard disk, /dev/sda1 designates first partition on first hard disk
- Hard disks listed in /dev do not necessarily start with sd: they can start with hd, fd, or even some other letter-combination
- **/dev/initrd** is the initial RAM disk loaded by the bootloader. Newer kernels list RAM disks as /dev/ram0, /dev/ram1, ...
- file --special-files /dev/sda shows partitions and startheads and startsectors of the partitions

Coldplug vs Hotplug Device

Hotplug is when you insert hardware into a running computer and **Coldplug** is when you have to turn your computer off to install hardware. USB devices are hot pluggable while PCI cards should be cold-plugged.

PCI Cards
Peripheral Component Interconnect (PCI) - Typically Plug and Play (PnP) devices.
lspci - command that displays info about PCI busses on system & devices connected to the busses.
lspci -k shows the kernel modules associated with each device.

USB Devices
Universal Serial Bus (USB) - protocol and hardware port for moving data to/from devices.
Protocol Standards:
1 - up to 127 devices. up to 120 Mbps data transfer
2 - up to 480 Mbps (4 x Standard 1)
3 - up to 4.8 Gbps (1,000 x Standard 2)
lsusb - Display information about USB devices on computer system.

SCSI Devices
Replaced (though some older systems still have them) IDE drives.
sg_scan - Scans SCSI bus and shows all SCSI devices.
sginfo -l - Shows all SCSI devices

modprobe to add and remove modules from the Linux Kernel

101.2 – Boot the system – 3

Linux Boot Steps

Note: These are the basic steps. However, the exam expects you to know "the flow" of the boot process. The steps can be broken up in various ways. Thus "Step 4" is not always as listed below.

BIOS

1) Power-On Self-Test (POST)
2) Determines the first bootable device found in bootable media list (found in CMOS).
3) Find the Master Boot Record (MBR) in the first sector of the device (called the "boot sector"), and loads the bootloader software into memory (RAM).

Bootloader

4) Creates a virtual temporary "disk" (ramdisk) in memory (RAM). The "disk" is actually a filesystem called "initrd image."
5) Loads the Linux kernel into memory (file name is: /boot/vmlinuz-version.gz, where version is the current version number of the kernel).

Linux Kernel

6) Mounts the filesystem
7) Loads and starts the init daemon (mother of all the other Linux processes) init Daemon
8) Starts up daemons, etc. (Note: This step has changed dramatically recently. As is covered in more detail in objective 101.3)

Bootloader

Bootloader can be **GRUB** (1&2) or **LILO** which are great for disks less than 2TB.
/etc/lilo.conf
/boot/grub/grub.cfg
/boot/grub/menu.lst

LILO

- configuration file: /etc/lilo.conf
Important: After change /etc/lilo.conf you must enter the command lilo at the shell prompt (restarts LILO) so the changes will take effect! If you don't, they won't!

GRUB

GRUB Stages
1 - Stored on MBR. Points simply to stage 1.5 or state 2.
1.5 - Stored after MBR, but before first partition. Contains drivers needed to load Stage 2.
2 - Stored on disk partition. Shows a menu which allows selection of the kernel image to be loaded. (Menu is from configuration file)

GRUB Configuration Files

/boot/grub/grub.conf (Fedora distribution).
/boot/grub/menu.lst (Debian or SUSE distribution)

How to Install Grub on Boot Device of Your Choice

Issue the command: grub-install device
Example: grub-install /dev/sda
Also can be more specific examples:

grub-install --root-directory=/boot hd0 (1st hard drive)
grub-install --root-directory=/boot hd1 (2nd hard drive)

Kernel Modules

General Kernel Module Notes

- Kernel modules are automatically loaded at boot time from the /etc/modprobe.d/dist.conf file.
- (Hardware vendors can add their own .conf files to this directory for module loading).
- The /etc/modprobe.d/blacklist.conf contains a list of modules that are preventing from being
- loaded, even if they are listed in another file in the /etc/modprobe.d directory.
- Kernel components are stored in the /usr/kernel.
- Kernel source code is stored in /usr/src

Kernel Module Commands

lsmod = shows information about what kernel modules are currently loaded on the system. (Basically displays the contents of /proc/modules file.)
insmod= inserts (loads) a single module into the kernel. (needs the full module file name which ends in .mo)
modprobe = loads a single module into the kernel (just needs the module name)
configuration file = **/etc/modprobe.conf**
options:
-C = Change the configuration file
-f = force a module load
-l = list all available modules (similar to lsmod)
-n = no actual load. Performs the necessary checks, etc.
-r = remove a module
-v = verbose

rmmod = removes a kernel module

Kernel

Kernel parameters (sometimes called boot parameters) supply the kernel with information about hardware parameters that it might not determine on its own - say single user mod boot (S)

init

When the kernel finishes loading, it usually starts /sbin/init. This program remains running until the system is shut down. It is always assigned process ID 1.
first process, process in charge, a big family tree of commands:

pstree

init is being replaced in many distros (say ubuntu with upstart) but still is in exam and has its own section.

Dmesg

dmesg command will show the full data from kernel ring buffer up to now. But
cat /var/log/dmesg - will show only the data during the boot

/var/log/messages

After the init process comes up, syslog daemon will log messages. It has timestamps and will persist during restarts.

- Kernel is still logging its own messages in dmesg
- in some systems it might be called /var/log/syslog
- there are many other logs at /var/log

101.3 – Change runlevels / boot targets and shutdown or reboot system – 3

Different Start-ups

- **SysV** (legacy but still popular. Focus on this one!)
- **Upstart** (who uses it? **Ubuntu**)
- **systemd** (who uses it? **Fedora** v15 & up, **OpenSUSE** 12.1 & up, and eventually **Red Hat**)

Runlevels
0 - halt
1or S- Single User Mode
2 - Multi-user, no network
3 - Multi-user, network
4 - User-defined
5 - Multi-user, network, GUI
6 – reboot

Set the default runlevel
SysV:
/etc/inittab file, and line is formatted as follows:
id:#:initdefault:
is replaced by the default runlevel number

Upstart:
Note: same as above (for some distributions moving to Upstart)
/etc/init/rc-sysinit.conf file, and lines is formatted as follows:
env DEFAULT_RUNLEVEL=#
is replaced by the default runlevel number

systemd:
Note: systemd uses the term "target unit" instead of "runlevel"
Default target unit set via a symbolic link command as follows:
ln -sf /lib/systemd/system/runlevel#.target /etc/systemd/system/default.target
Note: the f option on the symbolic link command, forces any current symbolic link to be broken and the new designated symbolic link to be enforced).

Startup shell script locations
SysV:
Three possible locations (depending upon the distribution):
- /etc/rc.d
- /etc/init.d
- /etc/rc.d/init.d

Upstart:
No startup scripts.
Instead it uses startup configuration files, located in /etc/init.
Configuration files determine what runlevels a service or task start on.

systemd:
No startup scripts.
Instead it uses service configuration unit files in /etc/systemd/system or /lib/systemd/system

(which can be overwritten during software upgrades)
Switch System's current runlevels

SysV:

Use either the telinit or init command (there are basically "twins")
init runlevel
example: init 5
Current runlevel:
The runlevel command shows the previous and the current runlevel.
Default runlevel:
Enter the following command to quickly check your default runlevel:
grep :initdefault /etc/inittab

Upstart:
No such thing as runlevels (except for backwards compatibility purposes).
Concerned with system events:
An event is a Linux server occurrence that triggers a needed system state change.

systemd:
Runlevels are not called "runlevels" but instead are called target units.
A target unit is a service (or action) group consisting of a name, type, and configuration file.
A service unit is for managing daemons, while a target unit is simply a group of other units.
Switch a Service's runlevel(s) to Start On

SysV:
chkconfig --level runlevels_to_start_on service_name on
Example: chkconfig --level 35 cups on
(Will start the cups daemon on runlevels 3 and 5)

Upstart:
Edit the /etc/init/service_name.conf file and change the start on and stop on parameters to the desired runlevels (Yes...Upstart does not use runlevels, but it still kind-of does. It is in a heavy state of development (2012-2013). Hopefully, it will settle down soon).

systemd:
Runlevels are not called "runlevels" but instead are called target units.
Edit the /etc/systemd/system/service_name.service file and change the WantedBy parameter to desired .target (e.g. multi-user.target).

Auditing Services
SysV:

All services:

chkconfig --list
service --status-all

A single service:

chkconfig --list service_name
service service_name status
Example: service cups status

Upstart:

All services: initctl list
A single service: initctl status service_name

systemd:

All services: systemctl list-unit-files --type=service | grep -v disabled
A single service: systemctl status service_name.service

Stop/Start/Restart Services

SysV:

service service_name start
service service_name stop
service service_name restart

Upstart:

initctl start service_name
initctl stop service_name
initctl restart service_name

systemd:

systemctl start service_name.service
systemctl stop service_name.service
systemctl restart service_name.service

Shutdown and reboot from command line

shutdown command:

now (shuts down system immediately)
-r (reboots system)
-h (halts system)
-P (powers off system)
-c (cancels shut down)
+number (will shut down the system in number minutes)
hh:mm (shuts down the system at hh:mm)
"message to system users´(sends a shut down warning message to users)

shutdown -r 13:30 "The system will be rebooted at 1:30pm. Please be prepared."
reboot (reboots the system)
poweroff (shuts down system and powers it off)
halt (shuts down system, but does not power it off)

102.1 - Design hard disk layout -2

Primary locations you need to know:

/ root - bottom of the directory tree, the root
/var - variable location, log files and dynamic content (such as web sites)
/home - this is the users home directory where their personal files are stored; each user gets their own home directory /home/bennettnw2 /home/user3
/boot - this is where the kernel is stored and supporting files for the kernel; also the boot files to get the system to boot; often on a diff partition
/opt - third party software vendors use this directory to install their programs; this is also a good candidate to be on its own partition; enterprise uses this directory extensively
/bin – Essential command binaries
/dev – Device files
/etc – Host-specific system configuration
/media – Mount point for removable media
/mnt – Mount point for mounting a filesystem temporarily
/tmp – Temporary files

Partitions:

In Linux world, devices are defined at **/dev/**. First SATA disk is **/dev/sda**, second SATA disk is **/dev/sdb**, ... and first SCSI disk (older systems) is **/dev/hda**.

You have to PARTITION the disks, that is creating smaller parts on a big disk and calling them /dev/sda1 (first partition on first SCSI disk) or /dev/hdb3 (3rd partition on second disk).

```
# fdisk /dev/sda
Welcome to fdisk (util-linux 2.25.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1  *         2048  43094015  43091968  20.6G 83 Linux
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
/dev/sda6       107704320 625141759 517437440 246.8G 83 Linux
```

Swap Space

Swap is **temporary storage** that acts like RAM. When a percentage of RAM is full, the kernel will move less used data to swap.

There are two ways to configure swap.

- **Swap partition**
- **Swap file**.

The swap partition is faster than using a swap file. (similar to a page file in windows) The partition is faster because data is getting written directly to disk rather than data getting written to a file and then to disk as when you are using a swap file.

The general rule of sizing of the swap space used to be 1.5x to 2.0x the size of your available RAM. With RAM being much cheaper nowadays, the rule of thumb is to use at least .50x of your available RAM.

Partitions and Mount Points

/dev/sda is the first hard drive (ssusi, sata, etc.) the "a" means it is the first disk the kernel has found. /dev/sda1 - the 1 means it is the first partition Normally you would have different partitions and those would be represented by: /dev/sda2, /dev/sda3, etc. Each partition has a specific function within the system.

Speaking of partitions and specific functions, each partition would need to be mounted to a specific directory. For example

- /dev/sda1 can be the / (root) partition
- /dev/sda2 can be the /boot partition
- /dev/sda3 can be the /home partition In this way they are isolated from each other

Commands for mounting

- **mount** can be used to mount partitions and also to show all existing mounts without any options
- **lsblk** used to show all block devices on a system and their names -- a block device is basically a hard disk or anything that takes a large amount of data and writes it in block sizes, to a location.
- **fdisk** helpful for partitioning hard drives, for now we will use it to list out partitions on a specific disk; ie: fdisk -l /dev/xvda
- **swapon** --summary will show a summary of the swap usage on a system; the same info can be found in /proc/swaps

Introduction to LVM

Taking a quick look at LVM (Logical Volume Manager) LVM allows the create of "groups" or disks or partitions that can be assembled into a single (or multiple) file systems

- GRUB cannot read LVM metadata, therefore LVM cannot be used as a /boot partition. All other partitions are fine like wine.
- LVM allows for easy and dynamic resizing of volumes
- snapshots can be taken of LVMs

Example Layout of an LVM Group

On the bottom layer you have your PVs (Physical Volumes).

- this is made up of all your block devices on your system Above the PV layer you will have the VG (Volume Group) layer.
- This layer encompasses the PV layer
- A volume group could encompass one disk to At the top, we have LV (Logical Volumes).
- This is where we carve out individual sections of the VG layer for specific mount points
- These logical volumes are akin to partitions
- Then on top of the LV is the file systems mounted to directories

LVM Main Concepts

In many cases you need to resize your partitions or even add new disks and add them to your mount points. LVM is designed for this.

LVM helps you create one partition from different disks and add or remove space to them. The main concepts are:

- **Physical Volume (pv):** a whole drive or a partition. It is better to define partitions and not use whole disks - unpartitioned.
- **Volume Groups (vg):** this is the collection of one or more pvs. OS will see the vg as one big disk. PVs in one vg, can have different sizes or even be on different physical disks.
- **Logical Volumes (lv):** OS will see lvs as partitions. You can format an lv with your OS and use it.

LVM Commands to set up a layout

- **pvs** physical volume scan - checks out all individual physical volumes in an LVM setup
- **vgs** virtual groups scan - scans for the volume groups in the system
- **lvs** logical volume scan - scans for the logical volume

102.2 – Install a Boot Manager – 2

Boot overview

System starts from BIOS and will do a **self-test** called **POST**. Then it will hand over the boot process to the first sector of master boot record (**MBR**).

MBR is on track (Cylinder) 0, side (Head) 0 and Sector 1 of the first disk which defines CHS.

MBR is only 512bytes so we need a smart bootloader to handle larger boot managers and even multiple systems. Some of these boot loaders are **LILO**, **GRUB** and **GRUB2**.

Chain Loading is when a boot loader, loads another boot loader. This is done when a Linux bootloader needs to start a Windows system.

Installing Grub

grub-install [device] is the command used to install GRUB to the specified device. You have to look for the boot mount point with which to install GRUB. **findmnt /boot** will tell you the exact location as a device; it will tell you exact disk and partition like, /dev/vda1. You can even make it easier and use /dev/hd0 or '(hd0)' to refer to the first drive of the system.

If you run grub by itself, it will look for the correct partition to install itself. You can even use a **grub "find"** command where it will let you know where to find stage1 = /grub/stage1.

This search is relative to root being /boot. So, when the answer you receive is /grub/stage1, it really means /boot/grub/stage1

If your system is up and running fine. Running **grub-install** is potentially dangerous. This command is usually run from a liveCD/USB where GRUB is getting installed to a new disk as part of a new system installation. In short, I should not ever really have to run grub-install.

GRUB shell commands:

grub by itself from the bash shell as root will place you into the GRUB shell help will give you more information regarding the list of commands you can use in the grub shell find find /grub/stage1 will tell you (hd0,0) quit will quit the shell for us

Be sure to back up your grub config before tooling around with grub

GRUB2

What are the differences between **MBR** and **GPT**?

- MBR only supported 4 partitions, with 1 being extended to 23 partitions a: - z: for a total of 26 partitions
- MBR only supported partitions sizes up to 2 TB
- GPT (GUID Partition Table) supported 128 partitions
- GPT could support partition sizes up to 1 ZB (zetabyte) or 909.5 million TB

For **GUID** to work we needed a replacement for the traditional BIOS. This is where **UEFI** comes in (Unified Extensible Firmware Interface).

- UEFI was still compatible with legacy BIOS.
- Requires a 64-bit operating system
- Prevents unauthorized operating systems from booting on the system

Booting with **GRUB2** on **GPT with UEFI**

- UEFI was a modern drop in replacement for BIOS; once UEFI gets started it will also look for a MBR in the first 512 bytes of a drive to find a boot loader (boot.img) Stage 1
    - Something to understand about GPT disks:
        - after the MBR, we are going to have a GPT header which lets the system know that this is a GPT style disk.
        - After the header, we will have a partition entry array. A large listing of all the partitions, thier ID, and their locations (similar to fstab?)
        - We will then have the core.img file which is Stage 1.5 just like MBR

- This next step is a big departure from MBR; core.img is looking for /boot/efi partition which can only be either a vfat or FAT32 partition. This partition is known as the ESP (EFI System Partition) UEFI can only read from those types
    - IMPORTANT! the /boot/efi partition is were the boot images are located and booting begins in a UEFI environment
        - this is the hand-off to grub2 /boot/grub2 where Stage 2 begins
        - within /boot/grub2 are two main files
        - grubenv
        - themes

Commands to run to modify our **grub2 setup** and check current config

IMPORTANT! RHEL based = grub2- and Debian based = grub-

- **grub2-editenv** you can use this to edit the environment file located /boot/grub/grubenv
    - typically, you do not want to edit anything under /boot/grub/ like you would with legacy grub
    - so, the best use of grub2-editenv is to extract information from that file with grub2-editenv list
    - grub2-editev list will show us the saved_entry of the default OS and kernel to be booted when grub2 boots the system; view the default boot entry for the grub configuration file
    - So, what file can we modify by hand to edit the boot environment? Glad you asked!

- **/etc/default/grub**
    - this is like modifying grub.conf in legacy GRUB with very similar options like:
        - timeout
        - submenu
        - and also adding options to the command line to modify how GRUB boots
- So, the process is to edit /etc/default/grub and then run grub2-mkconfig to programmatically modify GRUB2

- **grub2-mkconfig** will create or update a /boot/grub2/grub.cfg file based on entries from the /etc/default/grub file
    - NOTE: On Debain systems, the "2" is omitted from the command name
    - it is just grub-mkconfig

- **/etc/grub.d** is another location that you can make changes to so that you can edit the boot environment
    - These are individual configuration files that grub2-mkconfig will read from in order to generate the /boot/grub2/grub.cfg file
    - NOTE: on Debian systems, you would use update-grub to update a GRUB configuration after changes to /etc/default/grub have been made. So said another way, with RHEL you use the same command but with Debian, you would use a different command to update /boot/grub2/grub.cfg if you make changes within /etc/grub.d/
    - These options are what you would edit to make changes to a multiple OS computer

LILO

LILO is the (almost) original LInux LOader. It does not depend on you using a specific filesystem, it can boot Linux kernel images from floppy disks, and it can act as a bootmanager for other operating systems.

LILO works something like this:
• You create the configuration file **/etc/lilo.conf**
• You run **/sbin/lilo**
• **/sbin/lilo** maps out the sectors on the disk that contain the data needed for booting (the kernel, the configuration options)
• **/sbin/lilo** maps out the locations of the disk that contain the data needed for booting (the kernel, the configuration options). This map is stored in /boot/map.
• **/sbin/lilo** installs the boot loader on the disk, and configures it with the location of /boot/map.

One of the important consequences of the way LILO works is that the location of data on the disk may not change after /boot/map has been created. This means that if you modify the configuration in /etc/lilo.conf, install a kernel upgrade, or rename files and directories, you need to run /sbin/lilo again before things work as expected.

Interacting with the Boot Loader

Learning how to work with the GRUB boot menu in both GRUB and GRUB2. Practice booting the system using different kernel boot options and learn how to reinstall GRUB to our disk. Also, a walkthrough of boot using one command at a time.

GRUB Legacy menu

You will have to press any key to get to the menu before the kernel takes over. Once there you will be able to:

- Select which kernel to boot into using your arrow keys
- Press the "a" key to add new kernel arguments to the kernel you wish to boot
    - you can remove the quiet option
    - you can remove the graphical
    - you can add 2 to boot into runlevel 2
- Press the "c" key to get to the grub command line
    - you can read out some of the files in the boot directory
    - you can set new options for your configuration file
    - you can even reinstall GRUB in the event GRUB breaks
        - you could use the install command but that is for GRUB Pros
        - the best way is to use the setup command setup (hd0)

GRUB2 Menu

This will take you directly to the install GRUB2 menu

- There is no option for "a"
- The "e" key to edit; it looks way different as it gives us all the options rather than just the command line
- These are systemd options rather than sysvinit
    - so instead of just appending 2 for runlevel 2; we would need to set up the unit as systemd.unit and the target as rescue.target
    - it would look like systemd.unit=rescue.target
    - use <ctrl+x> to boot with the options we just changed

Booting GRUB Manually

- ls
- ls (hd0,1)/ will show us the boot directory contents
- set root=(hd0,1)
- linux /boot/vmlinz-4.24.9-43-generic root=/dev/vda1
- intrd /boot/initrd.img-4.24.9-43-generic
- ram disks and kernel names are very similar
- use the boot command to boot with all the options we just set

102.3 – Manage Shared Libraries - 2

Shared library

A shared library is a group of files that contains functionality that other applications can use. This is good for developers who only have to focus on their application and can rely on the functionality of shared libraries to interact with the operating system.

- Shared libraries end in a .so file extension which means shared object
- Library files can be found in the following locations on a Linux system:
    - /lib
    - /usr/lib(for 32bit systems) /usr/lib64(for 64bit systems)
        - some 64bit programs can still use the 32bit libraries
    - /usr/local/lib
    - /usr/share
- There are two types of library files: Dynamic (ends in .so) and Static (ends in .a)
    - Statically linked library files are compiled into an application when the application is installed
    - Dynamically linked libraries are called when needed by an application
- **Static** linking is when you add this library to your executable program. In this method your program size is big because it has all the needed libraries. One good advantage is your program can be run without being dependent to other programs / libraries.
- **Dynamic** linking is when you just say in your program "We need this and that library to run this program". This way your program is smaller but you need to install those libraries separately. This makes programs more secure (because libraries can be updated centrally), more advanced (any improvement in a library will improve the whole program) and smaller.

Commands and options when dealing with shared libraries

- **ldd** prints out shared object (library) dependencies
    - ldd /bin/cp shows what library files are linked to the cp application
- **ldconfig** configures dynamic linker run-time binding, create a cache based on tlibrary directories and can show you what is currently cached
    - will create a cached listing of the most used libraries based off of a directory I've provided
    - this command needs to be run as root/sudo
- **/etc/ld.so.conf** configuration file that points to directories and other configuration files that hodl reference to library directory locations
    - so this is the configuration file that links applications and libraries together?

- **LD_LIBRARY_PATH** legacy environment variable that points to a path where library files can be read from
    - you can use this instead of a configuration file to link libraries and applications
    - however, it is best to use a config file as this is old school and outdated but comes in handy if you are just testing something out
    - example of usage `$ export LD_LIBRARY_PATH=/opt/java/jre/lib
    - similar to the PATH variable but this is for libraries

102.4 - Use Debian Package Management – 3

APT

Advance Package Tool used as the default for Debian based Linux distros

- APT is used to install applications and their dependencies
- it is configured to use a network for retrieving data and dependencies
- removes applications and also updates and upgrades packages

How does **APT** work?

- reads the file at /etc/apt/sources.list
    - which contains url listing of software repositories
- once it reads the file and make the calculations for dependencies, it will download all the needed files
- then directs commands to dpkg to get it all done

Commands and options used for apt

- /etc/apt/sources.list
    - configuration file that lists out repository locations for packages
- apt-get upgrade or apt upgrade
- apt-get update or apt update
    - updates the local apt cache so that the upgrade command does not use excessive bandwidth?
- apt-get install or apt install
- apt-get remove will only remove the application but not the dependencies
- apt-get purge will remove the application and the dependencies and config files
- apt-get dist-upgrade will install new packages to get the system up-to-date while the apt upgrade will not install new packages
- apt-get download will download a package file but not install it
- apt-cache search [package] searches through your local apt cache for a package that can be installed
- apt-cache show lists out basic information about a package
- apt-cache showpkg displays more technical information about a package

Using Debian Package (dpkg)

The Debian Package Utility

- The .deb package contains:
    - the application or utility you want to install
    - default configuration files
    - how and where to install the files that come with the package
    - listing of dependencies, the package requires
- Dependencies should already be installed or be installed with the package
    - apt will automatically handle dependencies for you while dpkg will not

Some common commands and flags for use with dpkg

First you will want to download a .deb file using apt-get download [pkg name]

- dpkg --info taking a look at basic information about a package (apt-cache show is similar)
- dpkg --status on a package that has already been installed
- dpkg -l htop list packages that match the string on the search on a system dpkg -l htop
- dpkg -i htop need to be sudo to -i install a package; run --status to verify the package has been installed
- dpkg -L htop will list the files that were installed for htop
- dpkg -c htop will show the contents of the htop .deb package
- dpkg -r htop will remove a package
- dpkg -P htop will purge a package
- dpkg -S htop will search to see if there is anything in the dpkg database that refers to htop
- dpkg-reconfigure IMPORTANT TO KNOW FOR TEST - rerun configuration utility
    - sudo dpkg-reconfigure console-setup

102.5 - Use RPM and YUM package management – 3

RedHat Package Manager (RPM) and Yellowdog Updater Modified (YUM) are fedora /redhat /rhel / centos / .. tools to manage packages. There are also gui tools for installing and updating. As you saw on 102.4, all package managers can do standard functions like installing, updating and removing packages.

YUM adds extra features likes automatic updates, dependency management and works with repositories (collection on packages accessed over network or on a CD).

YUM

The Yellowdog Updater Modified

- Originally used for the Yellowdog Linux Distribution
- Handles RPM package dependencies (keeps you out of dependency hell like apt)
- Installs, upgrades and removes packages
- The yum setup
    - global yum configuration options are set in /etc/yum.conf
    - reads repository information from /etc/yum.repos.d
    - caches latest repository information in /var/cache/yum

Other RPMs (RedHat Package Managers)

- Zypper is used on SUSE Linux Distros
- DNF (Dandified yum) is used on Fedora distros
    - future replacement for yum in RHEL
    - uses same command syntax as yum

Common yum commands:

- **yum update** searches online repositories for updated package compared to what is currently installed on the system; also upgrades packages
    - it will also calculation and process dependencies
- **yum search [string]** searches the yum repositories for the string specified in the search
    - ie: yum search https
    - it will match any package with a reference to the search string
    - yum search uses /etc/yum.repos.d to do it's search
    - It is important to understand the basic contents of a repo file
        - baseurl is the web address that indicates where packages are loaded from
- **yum info [string]** lists information about a specified package
    - yum info httpd
        - this gives us information abut the httpd package like name, architecture, version, repo it comes from and a short description
- **yum list installed** displays list of all installed packages on the system
    - yum list installed | less`
- **yum clean all** cleans up all your cache information and ist local database file
    - use this if your database is old or corrupted
    - the next time I run the yum command it will download fresh meta data from the repositories yum.repos.d directory
- **yum install [package] -y** will assume yes to the install of a package as yum will always want to confirm every package you want to install
- **yum remove [package] -y** will assume yes and remove a package that we have installed
- `**yum autoremove [package]** will remove a package and also its dependencies
- yum whatprovides */httpd will show you what package contains a particular file you are looking for
- **yum reinstall [package]** this will download the package and reinstall for you
    - will need yum-utils installed on your system
    - yumdownloader mc (midnight commander)

The Red Hat Package Manager (rpm)

RPM

- An .rpm package contains
    - the application or utility
    - default config files
    - how and were to install files
    - list of dependencies that the package requires
- The rpm database:
    - located in /var/lib/rpm
    - use the rpm --rebuilddb to repair a corrupted rpm database
- Dependencies need to already be on the system or installed with the package
    - yum handles dependencies automatically but rpm does not

Common rpm Commands

- **rpm -qpi [package]** Queries Package Information
    - the output will be the same as the yum --info command
- **rpm -qpl [package]** lists all the files in a package (query package, list of files)
- **rpm -qa** list all installed packages on the system
- **rpm -i [package]** installs a specified package; often combined with other options to proved more verbose output with progress hashmarks ie: -ivh
- **rpm -U [path-to-package-file]** upgrades an installed package with a newer version
- **rpm -e [package]** this will erase or uninstall the selected package
- **rpm -Va** will verify all installed packages
- **rpm2cpio** converts an .rpm file into a cpio archive file and then you can extract the contents of the rpm package
    - often combined with the cpio command
    - ie: **rpm2cpio [some.rpm] | cpio -idmv**
        - -i means extract
        - -d means extract with the same directory structure
        - -m modifed time the same
        - -v verbose output

NOTE: you can use which to see if a package is installed on your system or not

102.6 – Linux as a virtualization guest – 1

Virtualization and Containers

What is a Virtual Machine**?**

- It is the emulation of a specific computer system type
- They operate based on the architecture and functions of a real computer and ist implementation
    - can involve specialized hardware, software or both
- Virt software will let you set up one operating system within another
    - they will both share the same physical hardware
    - the VM is isolated from the hardware but communicates with the hardware via a hypervisor
- Examples of **hypervisors** are:
    - KVM - kernel virtual machine; built into the linux kernel
    - QEMU - which KVM is built from
    - VMWare - proprietary
    - Xen -
    - VirtualBox - free system

Virtual Machine Basics:

- Two types of virtualization
    - **full virtualization** - does not know it is a virtual machine
    - **para-virtualization** - knows it is using guest drivers
        - para-virtualization typically performs better with guest drivers
- Virtual machine can be cloned or templated to rapidly deeply new systems
    - you will likely need to change the Linux system's Dbus ID
    - dbus-uuidgen will ensure that each running kernel interacts with a system that has a unique ID

Virtual Machines in the Cloud

- Virtual servers can be provisioned from cloud providers
- **cloud-init** is typically used to insure that user data is new and unique
    - creates new SSH keys
    - sets systems default location
    - sets the system's host name
    - sets up mount points

What is a Container?

- A container is an entirely isolated set of packages, libraries, and/or applications that are completely independent from their surroundings
- For getting an application up and running quickly
- Machine Container will share a kernel and file system with the host computer
- Application Container: shares everything but the application files and library files that the application needs
    - able to rapidly deploy web servers when traffic is high
- Examples of **Container Tech**:
    - Docker
    - nspawn (from systemd)
    - LXD
    - OpenShift

What is the difference between the two and why is it important

- **Virtualization**
    - was invented to allow the sharing yet segregation of server instances from each other
    - protects one operating system from another
    - prevention of spare CPU cycles, memory, or disk space go to waste
        - get the best bang for your buck to make use of all the resource
    - emulating virtual hardware through a hyper visor
        - heavy in terms of system requirements
- **Containers**
    - use shared operating systems
    - more efficient in system resource terms
    - more granular management of system resources

103.1 – Work on the command line – 4

We study:

- working with shells and commands to perform basic tasks;
- use and modification of the shell environment, including environment variables;
- work with the history of executed commands;
- work with teams based on the current location.

Terms and utilities:

- bash
- echo
- env
- export
- pwd
- set
- unset
- man
- uname
- history
- .bash_history

The administrator's job with Linux is primarily to operate on the command line. A command line or console is a separate programmable environment with its own settings, capabilities, and tools. There are many different shells, this lesson discusses bash, used in most modern operating systems.

You can see the name of the default shell used by the user at the end of each line of the / ** etc / passwd file. The global command line settings are in the / etc / ** profile file, and the settings for each user are in the files in his home directory.

However, there are several options for the location of the settings, depending on the family of operating systems. In general, the settings are searched in the following order: ~ / .bash_profile, ~ / .bash_login, and ~ / .profile (settings are taken from the first detected file).

The following common commands are used when **working in the console** (there are more of them, but in this topic LPI for some reason focuses on these):

- **cat** - output the contents of the file to the console;
- **cd** - go to the directory;
- **ls** - output directory content;
- **echo** - text output to the console;
- **touch** - update the file editing time or create a new empty file;
- ** **uname** ** - output the name of the OS;

The console, as a separate working environment, contains its own variables: environment variables (global variables used in the OS) and ordinary variables (working within a running console session). The following commands are available for **viewing variables**:

- **env** - output environment variables;
- **export** - turning a variable into an environment variable;
- **unset** - disable variable;

To **create a variable**, use simple syntax variable_name = variable_value. To refer to a variable, the $ sign is specified, for example:

X ** = 12 ** (set the variable X to 12);

echo $ ** X **** ** (display the value of the variable X _); _

To execute a command in the current directory (you can find it with the pwd command), you need to specify the full path to the command (_ for example, / home / semaev / _script), if the current directory is not listed in the values of the PATH environment variable.

The following commands are available for reference:

- **man** - command reference;
- **file** - file reference;
- **whatis** - name help;
- **history** - output command history (the list is stored in .bash_history);

Separately, mention should be made of the exec command, which allows you to execute a command outside the current shell, dropping superuser rights.

103.2 – Process text streams using filters – 2

Objectives

- Send text files and output streams through text utility filters to modify the output using standard UNIX commands found in the GNU textutils package.
- cat
- cut
- expand
- fmt
- head
- od
- join
- nl
- paste
- pr
- sed
- sort
- split
- tail
- tr
- unexpand
- uniq
- wc

Streams

In **UNIX** world a lot of data is in TEXT form. Log files, configurations, user inputs, list of files, ...
. **Filtering** this data means taking an input stream of text and performing some conversion on the text before sending it to an output stream. In this context, a **streams** is nothing more than *"a sequence of bytes that can be read or written using library functions that hide the details of an underlying device from the application"*.

In simple words, a text stream is an input of text from keyboard, a file, a network device, .. and filtering it is automatically changing it.

As you saw in previous section, modern programming environments and shells (including bash) use three standard I/O streams:

- **stdin** is the standard input stream, which provides input to commands.
- **stdout** is the standard output stream, which displays output from commands.
- **stderr** is the standard error stream, which displays error output from commands

Piping (|)

In *normal* cases, you give input from keyboard and output to the monitor. But in real life of a system admin, most inputs come from another commands. If you want to give the output of command1 as the input of command2, you should **PIPE** them as command1 | command2.
this | looks like a pipe!

```
jadi@funlife:~/w/lpic/101$ ls -1 | sort
12
62
amir
jadi
neda
you
jadi@funlife:~/w/lpic/101$ ls -1 | sort -r
you
neda
jadi
amir
62
12
```

UNIX philosophy is building small, strong tools and combine them

Redirection (>)

Another useful way of controlling the streams is >. This help you to redirect your output (mostly to a file).

```
jadi@funlife:~/w/lpic/101$ ls -ltrh
total 0
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 12
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 62
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 neda
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 jadi
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 you
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:34 amir
jadi@funlife:~/w/lpic/101$ ls -ltrh > directory_data
jadi@funlife:~/w/lpic/101$ cat directory_data
total 0
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 12
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 62
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 neda
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 jadi
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 you
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:34 amir
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:37 directory_data
```

Cat

This command simply **outputs its input stream** (or the filename you give it). As you saw in previous section. As most commands, if you do not give an input to it, it will read the data from the keyboard.

```
jadi@funlife:~/w/lpic/101$ cat > mydata
test
this is the second line
bye
jadi@funlife:~/w/lpic/101$ cat mydata
test
this is the second line
bye
```

When inputting data, ctrl+d will end the stream.

it is also possible to add files to each other using cat:

```
jadi@funlife:~/w/lpic/101$ cat mydata directory_data
test
this is the second line
bye
total 0
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 12
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 62
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 neda
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 jadi
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:33 you
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:34 amir
-rw-rw-r-- 1 jadi jadi 0 Jan  4 17:37 directory_data
```

Od

This command ***dump*s files** (shows files in formats other than text). Normal behaviour is OctalDump (base 8):

```
jadi@funlife:~/w/lpic/101$ od mydata
0000000 062564 072163 072012 064550 020163 071551 072040 062550
0000020 071440 061545 067543 062156 066040 067151 005145 074542
0000040 005145
0000042
```

Not good.. lets use two switches:

- -t will tell what format to print (-t a for showing only named characters or -t c for showing escaped chars)
- -A for choosing how to show offsets (-A``` Decimal, Octal, Hex or N````one)

od is very useful to find problems in your text files - say finding out if you are using tabs or correct line endings

Split

Will split files. It is very useful for transferring HUGE files on smaller media (say splitting a 3TB file to 8GB parts and moving them to another machine with a USB Disk).

```
jadi@funlife:~/w/lpic/101$ cat mydata
hello
this is second line
but as you can see we are
still writing
and this is getting longer
.
.
and longer
and longer!
jadi@funlife:~/w/lpic/101$ ls
mydata
jadi@funlife:~/w/lpic/101$ split -l 2 mydata
jadi@funlife:~/w/lpic/101$ ls
mydata    xaa  xab  xac  xad  xae
jadi@funlife:~/w/lpic/101$ cat xab
but as you can see we are
still writing
```

- on normal case, split uses xaa, xab, xac, .. for output files. If can be changed with split -l 2 mydata output which will lead to outputaa, outputab, ..
- the -l 2``` switch told the split to put 2 lines in output files. It is possible to use -b 42to split every 42 bytes or even-n 5```` to force 5 output files.
- if you want numeric output (x00, x01, ..) use -d

need to join these files? cat them with cat x* > originalfile.

wc

wc is ***word count***. It counts the characters, lines and bytes in the input stream.

```
jadi@funlife:~/w/lpic/101$ wc mydata
 9  25 121 mydata
```

It is very normal to count the line numbers with -l switch.

Head & Tail

Shows the *head* (top) of a file or its *tail* (bottom). The default lines to show is 10 but you can specify with -n20 or -20.
tali -f will continue showing the new lines which are being written at the eng of the file. Very useful.

Expand & Unexpand & tr

Expand will **replace the tabs** in a stream **with spaces** (normally 8 but can be defined with -n12 for 12):

```
jadi@funlife:~/w/lpic/101$ cat howcool
jadi	5
sina	6
rubic	2
you	12
jadi@funlife:~/w/lpic/101$ od -tc howcool
0000000  j   a   d   i  \t   5  \n   s   i   n   a  \t   6  \n   r   u
0000020  b   i   c  \t   2  \n   y   o   u  \t   1   2  \n
0000036
jadi@funlife:~/w/lpic/101$ expand howcool | od -tc
0000000  j   a   d   i           5  \n   s   i   n   a
0000020        6  \n   r   u   b   i   c         2  \n   y   o
0000040  u             1   2  \n
0000051
```

Unexpand will do the reverse. The default is converting only the initial blanks but his can be overrided by using -a.

unexpand needs at least two spaces.

The **tr** command ***translates*** A to 1, B to 2 and C to 3 in a stream you have to tr 'ABC' '123'. It is a pure filter so if you need to give it file to work on, you have to use cat:

```
jadi@funlife:~/w/lpic/101$ cat mydata
hello
this is second line
but as you can see we are
still writing
and this is getting longer
.
.
and longer
and longer!
jadi@funlife:~/w/lpic/101$ cat mydata | tr 'and' 'AND'
hello
this is second liNe
but As you cAN see we Are
still writiNg
AND this is gettiNg loNger
.
.
AND loNger
AND loNger!
```

Note: all **a**s are replaced with **A**.

You should know that if you put - instead of a filename, the data will be replaced from the pipe (or keyboard stdin).

```
jadi@funlife:~/w/lpic/101$ wc -l mydata | cat mydata - mydata
hello
this is second line
but as you can see we are
still writing
and this is getting longer
.
.
and longer
and longer!
9 mydata
hello
this is second line
but as you can see we are
still writing
and this is getting longer
.
.
and longer
and longer!
```

pr

This **formats text** for classic *printers*. The default header includes the filename and file creation date and time, along with a page number and two lines of blank footer.

```
pr mydata


2015-01-04 17:58                mydata                Page 1


hello
this is second line
but as you can see we are
still writing
and this is getting longer
.
.
and longer
and longer!
```

It is possible to print in two or more columns and other outdated fun stuff.

When output is created from multiple files or the standard input stream, the current date and time are used instead of the filename and creation date.

nl

Simply **numbers lines**.

```
jadi@funlife:~/w/lpic/101$ nl mydata  | head -3
     1	hello
     2	this is second line
     3	but as you can see we are
```

cat -n will also number lines.

fmt

Will **reformat a text file** within **margins** (say 80 columns width or 60 if you use -w60).

```
jadi@funlife:~/w/lpic/101$ fmt mydata
hello this is second line but as you can see we are still writing and
this is getting longer . . and longer and longer!
```

sort & uniq

Will **sorts** its input(s).

```
jadi@funlife:~/w/lpic/101$ cat uses
you fedora
jadi ubuntu
rubic windows
neda mac
jadi@funlife:~/w/lpic/101$ cat howcool
jadi	5
sina	6
rubic	2
you	12
jadi@funlife:~/w/lpic/101$ sort howcool uses
jadi	5
jadi ubuntu
neda mac
rubic	2
rubic windows
sina	6
you	12
```

if you want to sort NUMERICALLY (so 9 is lower than 19), use -n -r will reverse the search

and the **uniq removes duplicate entries** from its input. Normal behaviour is removing only the duplicated lines but you can change the behaviour for example by giving -f1 to force it to not check fist field.

```
jadi@funlife:~/w/lpic/101$ uniq what_i_have.txt
laptop
socks
tshirt
ball
socks
glasses
jadi@funlife:~/w/lpic/101$ sort what_i_have.txt | uniq
ball
glasses
laptop
socks
tshirt
jadi@funlife:~/w/lpic/101$
```

As you can see, the input HAVE TO BE sorted for uniq to work

uniq has great switches:

```
jadi@funlife:~/w/lpic/101$ cat what_i_have.txt
laptop
socks
tshirt
ball
socks
glasses
jadi@funlife:~/w/lpic/101$ sort what_i_have.txt  | uniq -c  #show count of each item
      1 ball
      1 glasses
      1 laptop
      2 socks
      1 tshirt
jadi@funlife:~/w/lpic/101$ sort what_i_have.txt  | uniq -u #show only non-repeated items
ball
glasses
laptop
tshirt
jadi@funlife:~/w/lpic/101$ sort what_i_have.txt  | uniq -d #show only repeated items
socks
```

how many things I have? wc -l what_i_have.txt :)

Cut

cut command will ***cut* a column** of one file. Good for separating fields:

Let's cut the *first field* of a file.

```
jadi@funlife:~/w/lpic/101$ cut howcool
jadi	5
sina	6
rubic	2
you	12
jadi@funlife:~/w/lpic/101$ cut -f1 howcool
jadi
sina
rubic
you
```

normal delimiter is TAB. use -dx to change it to "x" or use | tr ' ' '\t' | to convert spaces in your stream to TABs.

It is also possible to *cut* fields 1, 2, 3 with -f1-3 or only characters 4,5,7,8 with -c4,5,7,8.
#paste The paste command pastes lines from two or more files side-by-side! You can not do this in a normal text editor.

```
jadi@funlife:~/w/lpic/101$ cat howcool
jadi	5
sina	6
rubic	2
you	12
jadi@funlife:~/w/lpic/101$ cat uses
you fedora
jadi ubuntu
rubic windows
neda mac
jadi@funlife:~/w/lpic/101$ paste howcool uses
jadi	5	you fedora
sina	6	jadi ubuntu
rubic	2	rubic windows
you	12	neda mac
```

join

Our final field-manipulating command is join, which **joins files** based on a matching field. **The files should be sorted on the join field.**

```
jadi@funlife:~/w/lpic/101$ cat howcool
jadi	5
sina	6
rubic	2
```

```
you         12
jadi@funlife:~/w/lpic/101$ cat uses
you fedora
jadi ubuntu
rubic windows
neda mac
jadi@funlife:~/w/lpic/101$ sort howcool > howcool.sorted
jadi@funlife:~/w/lpic/101$ sort uses  > uses.sorted
jadi@funlife:~/w/lpic/101$ join howcool.sorted uses.sorted
jadi 5 ubuntu
rubic 2 windows
you 12 fedora
```

join does not work on numeric fields unless the fields are all the same length. It default delimiter is any white space (TAB, space) and it joins on first field. check man join for more info.

Sed

---

sed is **stream editor**. It is POWERFUL and can do magic! Just like most of the tools we saw, sed can work as a filter or take its input from a file. It uses **regular expressions** and is a great tool for replacing text. If you need to replace A with B only once in each line in a stream you have to say sed 's/A/B/':

```
jadi@funlife:~/w/lpic/101$ cat uses
you fedora
jadi ubuntu
rubic windows
neda mac
jadi@funlife:~/w/lpic/101$ sed 's/ubuntu/debian/' uses
you fedora
jadi debian
rubic windows
neda mac
jadi@funlife:~/w/lpic/101$
```

the pattern for changing EVERY occurrence of A to B in a line is sed 's/A/B/g'.
Remember escape characters? They also work here and this will remove every *new line* from a file and will replace it with a space:

```
jadi@funlife:~/w/lpic/101$ cat mydata
hello
this is second line
but as you can see we are
still writing
jadi@funlife:~/w/lpic/101$ sed 's/ /\t/g' mydata > mydata.tab
jadi@funlife:~/w/lpic/101$ cat mydata.tab
hello
this    is      second  line
but     as      you     can     see     we      are
still   writing
```

103.3 – Perform basic file management – 4

Objectives

- Copy, move and remove files and directories individually.
- Copy multiple files and directories recursively.
- Remove files and directories recursively.
- Use simple and advanced wildcard specifications in commands.
- Using find to locate and act on files based on type, size, or time.
- Usage of tar, cpio and dd.
- cp
- find
- mkdir
- mv
- ls
- rm
- rmdir
- touch
- tar
- cpio
- dd
- file
- gzip
- gunzip
- bzip2
- file globbing

ls

Is used to ***list* directories & files**. It can use absolute and relative paths

```
$ ls -l
total 52
-rw-rw-r-- 1 jadi jadi  146 Jan  5 08:29 alldata
-rw-rw-r-- 1 jadi jadi   30 Jan  5 09:15 howcool.sort
-rw-rw-r-- 1 jadi jadi  204 Jan  5 08:49 mydata
-rw-rw-r-- 1 jadi jadi  121 Jan  4 22:07 mydata.tab
```

First field indicates if this is a file (-) or directory (d).

- -l is for *long* (more info for each file)
- -1 will print one file per line
- -t sorts based on modification date
- -r reverses the search (so -tr is reverse time (newer files at the bottom).

you can always mix switches. A famous one is -ltrh (long+human readable sizes+reverse time).

Copying, Moving & Deleting

cp

This will ***copy* files** from one place / name to another place / name. If the target is a directory, all sources will be copied there.

cp source destination

mv

Will ***move* or *rename* files or directories**. It works like cp command. If you are moving a file on the same file system, the **inode** wont change.
In general:

- If the target is an existing directory, then all sources are copied into the target
- If the target is a directory that does not exist, then the (single) source must also be a directory and a copy of the source directory and its contents is made with the target name as the new name
- If the target is a file, then the (single) source must also be a file and a copy of the source file is made with the target name as the new name, replacing any existing file of the same name.

But use common sense when answering questions or using cp and mv in real life.

rm

Removes (**Deletes**) files.

General notes

Normally, the **cp** command will copy a file over an existing copy, if the existing file is writable. On the other hand, the **mv** will not move or rename a file if the target exists. You can overcome this using the -f switch.

- -f (--force) will cause cp to try overwrite the target.
- -i (--interactive) will ask Y/N question (deleting / overwriting).
- -b (--backup) will make backups of overwritten files
- -p will *preserve* the attributes.

Creating and removing directories

The **mkdir** command **creates directories**.

```
$ mkdir dirA dirB
$ ls -ltrh
-rw-rw-r-- 1 jadi jadi   30 Jan  8 16:45 howcool.sort
-rw-rw-r-- 1 jadi jadi   58 Jan  8 16:45 uses.sort
drwxrwxr-x 2 jadi jadi 4.0K Jan  8 17:11 dirB
drwxrwxr-x 2 jadi jadi 4.0K Jan  8 17:11 dirA
```

- **-p** will create nested directories:

```
$ mkdir newDir/insideNew/lastDir
mkdir: cannot create directory ‘newDir/insideNew/lastDir’: No such file or directory
$ mkdir -p newDir/insideNew/lastDir
$ ls newDir/insideNew/ -ltrh
total 4.0K
drwxrwxr-x 2 jadi jadi 4.0K Jan  8 17:13 lastDir
```

If you need to delete a directory the command is **rmdir** and you can also use the -p for nested removing:

```
$ tree
.
├── dirA
├── dirB
├── howcool.sort
└── uses.sort

2 directories, 2 files
$ rmdir dirA dirB
$ mkdir -p newDir/insideNew/lastDir
$ tree
.
├── howcool.sort
├── newDir
│   └── insideNew
│       └── lastDir
└── uses.sort

3 directories, 2 files
$ rmdir -p newDir/insideNew/lastDir
$ tree
.
├── howcool.sort
└── uses.sort

0 directories, 2 files
```

If you are using **rmdir** to remove a directory, it **MUST BE EMPTY**!

Handling multiple files at once

Most of the times we need to work with more than one file. This is *Linux* and there are ways!

Recursive commands

Recursive means going inside and inside and inside and inside! In many commands -r or -R is dedicated to recursive commands. Say ls. It uses -R :

```
$ ls
howcool.sort  newDir  uses.sort
$ ls -R
.:
howcool.sort  newDir  uses.sort

./newDir:
insideNew  TestFile

./newDir/insideNew:
lastDir

./newDir/insideNew/lastDir:
```

It is more useful when you are copying or deleting. When using cp or rm, -r (or -R or --recursive) will copy/delete all files inside the given source.

```
$ tree mydir
mydir
├── howcool.sort
├── newDir
│   ├── insideNew
│   │   └── lastDir
│   └── TestFile
└── uses.sort

3 directories, 3 files
$ mkdir newCopy
$ cp mydir newCopy
cp: omitting directory 'mydir'
$ cp -r mydir newCopy
$ tree newCopy/
newCopy/
└── mydir
    ├── howcool.sort
    ├── newDir
    │   ├── insideNew
    │   │   └── lastDir
    │   └── TestFile
```

```
└── uses.sort

4 directories, 3 files
```

Same works with **rm**:

```
$ rm newCopy
rm: cannot remove ‘newCopy’: Is a directory
$ rm -r newCopy
```

As you can see we can not rm a file but if using -r (or -R or --recursive) it works because it deletes the dir and whatever inside it.
rm -rf / is EXTREMELY DANGEROUS: force delete whatever in /

Wildcards and globbing

This is a way to say **All files** or **everything which starts with A** or **all files with 3 letter names which end in A or B or C**.

There are main cases:

- * means **any string**
- ? means any single character
- [ABC] matches A, B & C
- [a-k] matches a, b, c, ..., k (both lower-case and capital)
- [0-9a-z] matches all digits and numbers
- [!x] means NOT X.

So... this means that you can use these patterns in your commands to point to these files:

command	meaning
rm *	delete all files
ls A*B	all files starting with A ending with B
cp ???.* /tmp	Copy all files with 3 characters, then a dot then whatever (even nothing) to /tmp
rmdir [a-z]*	remove all directories which start with a letter

Touch

The touch command with no option will update the **modification** date of a file to the current time (will create a file if it is not exists).

```
/touch$ ls -l
total 0
-rw-rw-r-- 1 jadi jadi 0 Jan  8 17:47 myfile
/touch$ touch myfile  #after a minute
/touch$ ls -l
total 0
-rw-rw-r-- 1 jadi jadi 0 Jan  8 17:48 myfile
```

There are also possible:

```
[ian@echidna lpi103-2]$ touch -t 200908121510.59 f3
[ian@echidna lpi103-2]$ touch -d 11am f4
[ian@echidna lpi103-2]$ touch -d "last fortnight" f5
[ian@echidna lpi103-2]$ touch -d "yesterday 6am" f6
[ian@echidna lpi103-2]$ touch -d "2 days ago 12:00" f7
[ian@echidna lpi103-2]$ touch -d "tomorrow 02:00" f8
[ian@echidna lpi103-2]$ touch -d "5 Nov" f9
[ian@echidna lpi103-2]$ ls -lrt f*
-rw-rw-r--. 1 ian ian 0 2009-07-31 18:31 f5
-rw-rw-r--. 1 ian ian 0 2009-08-12 12:00 f7
-rw-rw-r--. 1 ian ian 0 2009-08-12 15:10 f3
-rw-rw-r--. 1 ian ian 0 2009-08-13 06:00 f6
-rw-rw-r--. 1 ian ian 0 2009-08-14 11:00 f4
-rw-rw-r--. 1 ian ian 4 2009-08-14 18:25 f1
-rw-rw-r--. 1 ian ian 0 2009-08-14 18:27 f2
-rw-rw-r--. 1 ian ian 0 2009-08-15 02:00 f8
-rw-rw-r--. 1 ian ian 0 2009-11-05 00:00 f9
```

and the most advanced way is setting time of a file based on another file:

```
[ian@echidna lpi103-2]$ date
Fri Aug 14 18:33:48 EDT 2009
[ian@echidna lpi103-2]$ date -r f1
Fri Aug 14 18:25:50 EDT 2009
[ian@echidna lpi103-2]$ touch -r f1 f1a
[ian@echidna lpi103-2]$ ls -l f1*
-rw-rw-r--. 1 ian ian 4 2009-08-14 18:25 f1
-rw-rw-r--. 1 ian ian 0 2009-08-14 18:25 f1a
```

Finding files

The find command helps us to find files based on MANY criteria. Look at this:

```
$ find . -iname "[a-j]*"
./howcool.sort
./alldata
```

```
./mydir/howcool.sort
./mydir/newDir/insideNew
./howcool
```

- the first parameter says where should be searched (with subdirectories).
- the -name switch indicates the criteria (here iname: searching for files with this name).

a common switch is **-iname** which says "name but case is not important (z is same as Z)". Also -d is commonly used:

```
$ find . -iname "*my*"
./myfiles
./mydata.noenter
./mydata
./mydir
./mydir/hereisMYfile.txt
./touch/myfile
./mydata.tab
$ find . -type f -iname "*my*"
./myfiles
./mydata.noenter
./mydata
./mydir/hereisMYfile.txt
./touch/myfile
./mydata.tab
```

These are the most common file types:

- -type f will search for a regular file
- -type d will search for a directory
- -type l will search for a symbolic link

you can also search for file sizes:

command	meanint
-size 100c	files which are exactly 100 bytes (you can also use **b**
-size +100k	files which are more than 100 kilobytes
-size -20M	files smaller than 20Megabytes
-size +2G	files bigger than 2Gigabytes

So this will find all files ending in *tmp* with size between 1M and 100M in /var/ directory:

```
find /var -iname '*tmp* -size +1M -size -100M
```

you can find all empty files with find . -size 0b or find . -empty

Acting on files

We can act on files with various switches:

switch	meanint
-ls	will run ls -dils on each file
-print	will print the full name of the files on each line

But the best way to run commands on found files is -exec switch. You can point to the file with **'{}'** or **{}** and finish your command with **;**.
This will remove all empty files in this directory and its subdirectories:

```
find . -empty -exec rm '{}' \;
```

or this will rename all htm files to hfml

```
find . -name "*.htm" -exec mv '{}' '{}l' \;
```

At last you have to know the -mtime switch for finding files based on their time.

switch	meanint
-atime -6	file was last accessed less than 6*24 hours ago
-ctime +6	file was changed more than 6*24 hours ago
-mtime -6	file *content* moditication less than time is 6*24 ago
-mmin -90	file's data was last modified less than 90 minutes ago
-amin, -cmin	you guess!

if you add -**daystart** switch to **-mtime** or **-atime** it means that we want to consider days as calendar days, starting at midnight.

Identify a file

That is the file command:

```
$ file mydata.tab
mydata.tab: ASCII text
$ file /bin/bash
/bin/bash: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=cb63ec0718f2022619814c04a5b6cd8a36752a83, stripped
$ file mydata.tab
mydata.tab: ASCII text
$ file /bin/bash
/bin/bash: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=cb63ec0718f2022619814c04a5b6cd8a36752a83, stripped
$ file -i mydir
mydir: inode/directory; charset=binary
```

-i switch prints the complete mime format

Compressing files

Zip

we mostly use **gzip** and **gunzip** in linux. It is very easy:

```
$ ls *  -ltrh
-rw-r--r-- 1 jadi jadi  79K Dec 22 11:52 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt
$ gzip  The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt
$ ls *  -ltrh
-rw-r--r-- 1 jadi jadi  30K Dec 22 11:52 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt.gz
$ gunzip The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt.gz
$ ls *  -ltrh
-rw-r--r-- 1 jadi jadi  79K Dec 22 11:52 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt
```

- gzip preserves time
- gzip creates the new compressed file with the same name but with .gz ending
- gzip removes the original files after creating the compressed file

Bzip2

is another compressing tool. Works just the same but with another compression algorithm.

```
$ ls *  -ltrh
-rw-r--r-- 1 jadi jadi  79K Dec 22 11:52 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt
$ bzip2 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt
$ ls *  -ltrh
-rw-r--r-- 1 jadi jadi  22K Dec 22 11:52 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt.bz2
$ bunzip2 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt.bz2
```

```
$ ls *  -ltrh
-rw-r--r-- 1 jadi jadi  79K Dec 22 11:52 The.Equalizer.2014.1080p.BluRay.x264.anoXmous_eng.srt
```

Archiving files

Sometimes we need to create an archive file from many files for easier moving or backing up. This is done with cpio and tar.

Tar

TapeARchive or tar is the most common archiving tool. In automatically create an archive file from a directory and all its subdirs.

Common switches are

switch	meanint
-cf myarchive.tar	create file named myarchive.tar
-xf myarchive.tar	extract a file called myarchive.tar
-z	compress the archive with gzip after creating it
-b	compress the archive with bzip2 after creating it
-v	verbose! print a lot of data about what you are doing
-r	appeng new files to the currentyp available archive

If you issue absolute paths, tar removes the starting slash (/) for safety reasons when creating an archive. If you want to override, use -p option.
tar can work with tapes and other storages. Thats why we use -f to tell it that we are working with files.

Cpio

Gets a list of files and creates archive (one file) of it which can be opened later.

```
$ ls | cpio -o > allfilesls.cpio
3090354 blocks
```

- -o makes cpio to create an output from its input
- cpio does not goes into the folders. So mostly we use it with find:

```
find . -name "*" | cpio -o > myarchivefind.cpio
```

to decompress it:

```
mkdir extract
mv myarchivefind.cpio extract
cd extract
cpio -id < myarchivefind.cpio
```

- -d will create the folders
- -i is for extract

dd

The dd command **copies data** from one location to another. This data comes from files. You can use it just like copy:

```
$ cat howcool
jadi        5
sina        6
rubic       2
you         12
$ dd if=howcool of=newcool
0+1 records in
0+1 records out
30 bytes (30 B) copied, 0.0227904 s, 1.3 kB/s
$ cat newcool
jadi        5
sina        6
rubic       2
you         12
$
```

- if is In File
- of is Out File

But it is used in many other cases specially writing directly to block devices such as /dev/sdb or changing data to upper/lower case.

This will backup my whole hard to a file:

```
# dd if=/dev/sda of=backup.dd bs=4096
```

or better:

```
# dd if=/dev/sda2 |gzip >backup.dd.gzip
```

Another common usage is creating files of specific size:

```
$ dd if=/dev/zero of=1g.bin bs=1G count=1
```

103.4 – Use streams, pipes and redirects - 4

Objectives

- Redirecting standard input, standard output and standard error.
- Pipe the output of one command to the input of another command.
- Use the output of one command as arguments to another command.
- Send output to both stdout and a file.
- tee
- xargs

We've already talked about basics of piping and redirects in previous sections. Here you will get a deeper understanding of these concepts.

Redirecting standard IO

On a linux system most shells use streams for input and output (a list of characters). . These streams can be from (and toward) various things including keyboard, block devices (hards, usb stick, ..), window of a program, fiels, ...

1. ***stdout*** is the standard output stream, which displays output from commands (file descriptor 1)
2. ***stderr*** is the standard error stream, which displays error output from commands (file descriptor 2)
3. ***stdin*** is the standard input stream, which provides input to commands (file descriptor 0)

What are these **file descriptions**? There are used to control the output. If you need to control where your output goes, you can add n> or n>>.

- **n>** redirects **file description n** to a file or device. If the file already exists it is **overwritten** and if it does not exists, it will be created.
- **n>>** redirects file description n to a file or device. If the file already exists the stream will be appended to the end of it and if it does not exists, it will be created.

if the **n** is not given, the default is *standard output*
The user who runs the command should have write access to the file.

```
$ ls
fiona  habib  mahmoodrm  minoo          mojtaba  sina
$ ls j*
ls: cannot access j*: No such file or directory
$ ls m*
mahmoodrm  minoo  mojtaba
$ ls j* m*
ls: cannot access j*: No such file or directory
mahmoodrm  minoo  mojtaba
$ ls j* m* > output 2> errors
```

```
$ cat output
mahmoodrm
minoo
mojtaba
$ cat errors
ls: cannot access j*: No such file or directory
$
```

Redirecting both stdout and stderr to one location

Sometimes (say during automated tasks) we prefer to send both standard output and standard error to same place, Use **&>** and **&>>** to say *both stderr and stdout*.
It is also possible to use &1 and &2 and &0 to refer to **current place** of stdout, stderr & stdin. In this case ls > file1 2>&1 means *redirect output to file1 and output stderr to same place as stdout (file1)*
Be careful! ls 2>&1 > file1 means *print stderr to current location of stdout (screen) and then change the stdout to file1*

Sending to null

In linux, **/dev/null** is like a trash-can. You can send anything there and it disappears. So it is normal to say:

```
$ ls j* m* > file1
ls: cannot access j*: No such file or directory
$ ls j* m* > file1 2>/dev/null
$ cat file1
mahmoodrm
minoo
mojtaba
```

Redirecting input

The < operand redirects the input.

```
$ cat uses
you fedora
jadi ubuntu
rubic windows
neda mac
narsin arch
$ tr ' ' ',' < uses
you,fedora
jadi,ubuntu
rubic,windows
neda,mac
narsin,arch
```

Here-documents

Many shells, have here-documents (also called here-docs) as a way of input. You use **<<** and a **WORD** and then whatever you input is considered stdin till you give only the WORD in one line.

```
$ tr ' ' '.' << END_OF_DATA
> this is a line
> and then this
>
> we'll still type
> and,
> done!
> END_OF_DATA
this.is.a.line
and.then.this

we'll.still.type
and,
done!
```

Here-Documents are very useful if you are writing scripts and automated tasks.

Pipes

Piping is sending one commands output to another commands input (Piping the stdout to stdin). You use **|** for this task.

As previously seen, many commands use a hyphen - in place of a filename as an argument to indicate when the input should come from stdin rather than a file.

```
$ cat what_i_have.txt
laptop
socks
tshirt
ball
socks
glasses
$ cut -f1 -d' ' what_i_have.txt | sort | uniq -c | sort -nr
    2 socks
    1 tshirt
    1 laptop
    1 glasses
    1 ball
```

If you need to start your pipeline with the contents of a file, start with cat filename | ... or use a < stdin redirect.

Xargs

This command reads input from ***stdin*** and uses them as arguments.

```
$ ls | xargs echo these are files:
these are files: errors f file1 fiona habib mahmoodrm minoo mojtaba output output.txt sina uses
what_i_have.txt
```

if you do not give any command to the xargs , the echo will be the default command (it will show the stdin).

Have in mind that **xargs** breaks input based on blanks and use any part as an argument. You can limit the number of arguments with **--max-args** (same as -n) switch and escape blanks or quote them to prevent them from breaking.

One important switch is **-I.** This is useful if you need to pass stdin arguments in the middle (or even start) of your commands. use the form **xargs -I SOMETHING echo** here is SOMETHING end:

```
$ cat what_i_have.txt
laptop
socks
tshirt
ball
$ cat what_i_have.txt | xargs -I DATA echo I have DATA and I love it.
I have laptop and I love it.
I have socks and I love it.
I have tshirt and I love it.
I have ball and I love it.
```

If you use -L, the input will break by line and not by blanks.

Tee

What if you need to see the output on screen and also save it to a file? Ugly way is redirecting to the file and using tail -f file in another window. Nice way is using tee and giving it one or more filenames for standard output (if you need to save *stderr*, first redirect it to *stdout*). This will write the output to those files and also writes them to the screen:

```
$ ls -1 | tee allfiles myfiles
allfiles
f
fiona
habib
mahmoodrm
minoo
mojtaba
myfiles
sina
```

If you want to prevent overwriting files, use the -a switch to append to files if exists.

103.5 – Create, Monitor and Kill Processes - 4

Objectives

- Run jobs in the foreground and background.
- Signal a program to continue running after logout.
- Monitor active processes.
- Select and sort processes for display.
- Send signals to processes.
- &
- bg
- fg
- jobs
- kill
- nohup
- ps
- top
- free
- uptime
- killall

Foreground and Background jobs

One of the great points of linux on its beginning days, was the ability to run many programs at the same time. This is done with sending programs to the background.

Normally if you run a program on the terminal, it *blocks* your terminal but sending a command to the background will prevent this:

```
xeyes &
```

But what if we started it normally? We can break / cancel it with Ctrl+c or *suspend* it using Ctrl+z.

```
$ xeyes
^Z
[1]+  Stopped                 xeyes
$ jobs
[1]+  Stopped                 xeyes
$ bg
[1]+ xeyes &
$ jobs
[1]+  Running                 xeyes &
$ sleep 1000 &
[2] 7395
$ jobs
[1]-  Running                 xeyes &
[2]+  Running                 sleep 1000 &
```

```
$ fg %2
sleep 1000
^Z
[2]+  Stopped                 sleep 1000
$ jobs
[1]-  Running                 xeyes &
[2]+  Stopped                 sleep 1000
$ bg sle
[2]+ sleep 1000 &
$ jobs
[1]-  Running                 xeyes &
[2]+  Running                 sleep 1000 &
```

-l switch of jobs will also show the process ID

Nohup

The nohup command lets you run your commands even after you logged out and writes its output to **nohup.out**:

```
$ nohup ping 4.2.2.4
nohup: ignoring input and appending output to ‘nohup.out’
^C$ cat nohup.out
PING 4.2.2.4 (4.2.2.4) 56(84) bytes of data.
64 bytes from 4.2.2.4: icmp_seq=1 ttl=51 time=225 ms
64 bytes from 4.2.2.4: icmp_seq=3 ttl=51 time=223 ms

--- 4.2.2.4 ping statistics ---
4 packets transmitted, 2 received, 50% packet loss, time 3010ms
rtt min/avg/max/mdev = 223.584/224.767/225.950/1.183 ms
```

It is common to use 2> to redirect the nohup errors to a file: nohup script.sh > mynohup.out 2>&1 &

Kill

You can control processes by **signals**. Actually pressing Ctrl+c and Ctrl+z is also sending signals. Another way for this is using the kill command:

```
$ jobs
[3]   Running                 xeyes &
[4]   Running                 sleep 1000 &
[5]-  Running                 sleep 2000 &
[6]+  Running                 sleep 3000 &
$ kill %4
$ jobs
[3]   Running                 xeyes &
[4]   Terminated              sleep 1000
[5]-  Running                 sleep 2000 &
[6]+  Running                 sleep 3000 &
$ jobs
[3]   Running                 xeyes &
```

```
[5]-  Running                 sleep 2000 &
[6]+  Running                 sleep 3000 &
```

If is also possible to use PIDs in from of the kill or send other signals:

signal number	signal name	meaning
1	SIGHUP	Informing the process that its controlling terminal (like an ssh connection) is te
15	SIGTERM	normal termination request
9	SIGKILL	forcefully kills the proccess

So you can do a kill -9 8733 to force process ID 8733 to close.
Now you can understand what nohup means: go not answer to the SIGHUP.

killall

Will send the given signal (or 15) to all the processes with the given name:

```
$ jobs
[3]   Running                 xeyes &
[5]-  Running                 sleep 2000 &
[6]+  Running                 sleep 3000 &
$ ps -ef | grep sleep
jadi     7864  7651  0 21:07 pts/1    00:00:00 sleep 2000
jadi     7865  7651  0 21:07 pts/1    00:00:00 sleep 3000
jadi     7977  7651  0 21:14 pts/1    00:00:00 grep sleep
$ killall sleep
[5]-  Terminated              sleep 2000
[6]+  Terminated              sleep 3000
$ jobs
[3]+  Running                 xeyes &
$ ps -ef | grep sleep
jadi     7980  7651  0 21:14 pts/1    00:00:00 grep sleep
```

Monitoring Processes

ps

The ps command shows running processes on your computer.

```
$ sleep 1000 &
[1] 7678
$ sleep 1001 &
[2] 7679
$ xeyes &
[3] 7680
$ ps
  PID TTY          TIME CMD
 7651 pts/1    00:00:00 bash
 7678 pts/1    00:00:00 sleep
 7679 pts/1    00:00:00 sleep
 7680 pts/1    00:00:00 xeyes
 7681 pts/1    00:00:00 ps
```

But using ps aux (= -aux) or ps -ef is also common & shows ALL processes on this system:

```
$ ps -aux | wc -l
293
```

Every process has a ProcessID (PID) and a PPID (Parent Process ID).

finding processes

You've seen that ps -ef shows processes from all users. We can grep on that and see who is running gedit and what is its process ID:

```
$ ps -ef | grep gedit
jadi      6213  4604  9 20:06 ?        00:04:43 gedit
jadi      7725  7651  0 20:55 pts/1    00:00:00 grep gedit
```

but there is also a more direct way:

```
$ ps -C gedit -o user,pid,tty,time,comm
USER       PID TT          TIME COMMAND
jadi      6213 ?        00:04:49 gedit
```

It is also possible to use the --sort switch to sort output based on different fields (+ for ascending & - for descending).

```
$ ps -af --sort +comm,-sid
UID        PID  PPID  C STIME TTY          TIME CMD
root      5486  5478  0 19:59 pts/12   00:00:00 -su
root      4444  1169  0 19:56 tty4     00:00:00 -bash
jadi      6638  5412  0 20:10 pts/0    00:00:04 node /usr/local/bin/sslocal
jadi      7778  7651  0 20:58 pts/1    00:00:00 ps -af --sort +comm,-sid
jadi      7678  7651  0 20:48 pts/1    00:00:00 sleep 1000
jadi      7679  7651  0 20:48 pts/1    00:00:00 sleep 1001
jadi      7775  7651  0 20:58 pts/1    00:00:00 sleep 1000
jadi      7776  7651  0 20:58 pts/1    00:00:00 sleep 1000
```

```
jadi     7777  7651  0 20:58 pts/1    00:00:00 sleep 1000
root     5478  5477  0 19:59 pts/12   00:00:00 su -
root     5477  5008  0 19:59 pts/12   00:00:00 sudo su -
jadi     7680  7651  0 20:48 pts/1    00:00:01 xeyes
```

Top

Processes are changing and sometimes you need to check them live. top command will help you:

```
$top

top - 21:00:44 up  1:16,  5 users,  load average: 1.51, 1.65, 1.78
Tasks: 293 total,   1 running, 292 sleeping,   0 stopped,   0 zombie
%Cpu(s): 19.0 us,  5.0 sy,  0.0 ni, 70.9 id,  5.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   8060264 total,  5359812 used,  2700452 free,   169240 buffers
KiB Swap:  7811068 total,        0 used,  7811068 free.  2250692 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 6570 jadi      20   0 1437752 546064  88312 S  18.4  6.8  12:00.96 firefox
 4870 jadi      20   0 1762516 299120  75664 S  12.2  3.7   7:37.05 compiz
 4492 jadi       9 -11  455152  11516   8940 S   6.1  0.1   1:06.81 pulseaudio
 4532 root      20   0  389028  77116  60192 S   6.1  1.0  12:16.63 Xorg
 4723 jadi      20   0  358936   8288   5512 S   6.1  0.1   9:51.52 ibus-daemon
 5648 jadi      20   0 1641556 203676 102840 S   6.1  2.5   3:20.88 chrome
 7082 jadi      20   0 1210748  73136  42528 S   6.1  0.9   0:36.51 Telegram
 7806 jadi      20   0   33796   3004   2500 R   6.1  0.0   0:00.02 top
    1 root      20   0   29528   4320   2584 S   0.0  0.1   0:01.71 init
```

You can see the processes, system load, uptime, CPU status, memory, ... and do some stuff:

key during top	functionality
h	help
q	quit
M	sort based on memory usage"
c	show full commands
k	kill after asking pid and signal

Free

The free command will show you info about the system memory. The default is *kilobytes* but you can change it with -mfor megabytes, -g for *gigabytes* or even -b for bytes:

```
$ free -m
             total       used       free     shared    buffers     cached
Mem:          7871       5231       2640        332        169       2195
-/+ buffers/cache:       2866       5005
Swap:         7627          0       7627
```

The system should not use swap in long term

Uptime

The uptime command shows the time, how long the system is up, how may users are logged in and the load average of 1, 5 & 15 minutes:

```
$ uptime
 21:18:52 up  1:34,  5 users,  load average: 2.38, 2.64, 2.41
```

103.6 – Modify process execution priorities – 2

Objectives

- Know the default priority of a job that is created.
- Run a program with higher or lower priority than the default..
- Change the priority of a running process.
- nice
- ps
- renice
- top

On a Linux system, we are running a lot of processes and programs on a few CPUs. So, you need a way to tell your OS to give more priority to some tasks or give less resources to some others. In last section you saw the top command to check the CPU usage of each process:

```
$ top

top - 08:44:51 up 13:00,  5 users,  load average: 0.57, 1.50, 1.50
Tasks: 290 total,   2 running, 288 sleeping,   0 stopped,   0 zombie
%Cpu(s): 38.4 us,  9.4 sy,  0.0 ni, 49.3 id,  2.8 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   8060264 total,  7858348 used,   201916 free,   360144 buffers
KiB Swap:  7811068 total,       0 used,  7811068 free.  2842344 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
13605 jadi      25   5 1473652 530700  91128 R  54.5  6.6   3:25.50 firefox
11157 root      20   0  572004 112652  94484 S   6.1  1.4   3:26.18 Xorg
12265 jadi      20   0 1210484  75848  42264 S   6.1  0.9   0:32.06 Telegram
12671 jadi      20   0 1800508 274564  80300 S   6.1  3.4   1:27.35 compiz
15035 jadi      20   0  768688  54920  34228 S   6.1  0.7   0:00.93 /usr/bin/termin
15066 jadi      20   0   33796   3076   2448 R   6.1  0.0   0:00.02 top
    1 root      20   0   29528   4320   2584 S   0.0  0.1   0:02.27 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.27 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H    ```

    Chkecn th
```

There is **NI** column, it shows how **nice** the process is. The nicer the process, the less CPU it asks. Nice can be from -20 to 19 (a process with nice = **-20** is ANGRY and asking for a lot of CPU while a process with nice = **19** is SUPER NICE and lets **other** processes use most of the CPU).

If you do not use nice command, processes will have nice level of 0. This can be checked with nice command:

```
$ nice
0
```

It is also possible to tell ps command to write the nice parameter of processes:

```
$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 15044 15035  0  80   0 -  7453 wait   pts/29   00:00:00 bash
0 S  1000 15052 15044  0  60 -20 -  3976 hrtime pts/29   00:00:00 sleep
0 R  1000 15080 15044  0  80   0 -  4680 -      pts/29   00:00:00 ps
```

Setting priorities when running commands

If you need to change the niceness level of a program you can running it with **nice** command and -n switch (for nice):

```
$ nice -n -20 echo "I am running!"
nice: cannot set niceness: Permission denied
I am running!
$ sudo nice -n -20 echo "I am running!"
I am running!
$ sudo nice -n 19 echo "I am running!"
I am running!
```

Please note to two points:

1. Give high priorities (less than 0) needs root access
2. If you are not root and asking for nice level lower than 0 you'll get an error message but the process will run with normal nice level (0).

If you run a command with nice without any parameters, the nice value will be 10:

```
$ nice xeyes &
[1] 15217
$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 15044 15035  0  80   0 -  7455 wait   pts/29   00:00:00 bash
0 S  1000 15217 15044  0  90  10 - 12522 poll_s pts/29   00:00:00 xeyes
0 R  1000 15218 15044  0  80   0 -  4680 -      pts/29   00:00:00 ps
```

Changing priorities

The **renice** command can change the *niceness* of running processes:

```
$ ps -ef | grep firefox
jadi     13605 11226 30 08:28 ?        00:10:13 /usr/lib/firefox/firefox
jadi     15192 15044  0 09:01 pts/29   00:00:00 grep firefox
$ sudo renice -n -10 13605
13605 (process ID) old priority 5, new priority -10
```

103.7 – Search text files using regular expressions - 3

Objectives

- Create simple regular expressions containing several notational elements.
- Use regular expression tools to perform searches through a filesystem or file content.
- grep
- egrep
- fgrep
- sed
- regex

Regex

Regular expression, Regex, regexp is a pattern to describe what we want to *match* from a text. Here will discuss the form of regex which is used with the **grep** (generalised regular expression processor) command.

There is two kind of regex in GNU grep: **Basic** an **Extended**.

Basic blocks

Adding two expressions

If you need to add (concat) two expressions, just write them after each other.

Regex	Will match
a	ali, mina, hamid, jadi
na	nasim, mina, nananana batman, mona

Repeating

- The ***** means repeating the previous character for 0 or more
- The **+** means repeating the previous character for 1 or more
- the **?** means zero or one repeats
- {n,m} The item is matched at least n times, but not more than m times

Regex	Will match	Note
a*b	ab, aaab, aaaaab, aaabthis	
a*b	b, mobser	Because there is a b here with zero **a** before it
a+b	ab, aab, aaabenz	won't match **sober** or **b** because there needs to be at least one **a**
a?b	ab, a**ab**, b, batman (zero a then b), ...	.

Alternation (|)

If you say a\|b it will match a or b.

Character Classes

The dot (**.**) means any character. So **..** will match anything with at least two character in it. You can also create your own classes with [abc] which will match a or b or c and [a-z] which match a to z.

You can also refer to digits with \d and

Ranges

There are easy ways to commonly used classes. Named classes open with [: and close with :]

Range	Meaning
[:alnum:]	Alphanumeric characters
[:blank:]	Space and tab characters
[:digit:]	The digits 0 through 9 (equivalent to 0-9)
[:upper:] and [:lower:]	Upper and lower case letters, respectively.

Range	Meaning
^ (negation)	As the first character after [in a character class negates the sense of the remaining characters

A common form is .* which matches any character (zero or any length).

Matching *specific locations*

- The caret **^** means beginning of the string
- The dollar **$** means the end of the string

Samples

- ^a.* Matches anything that starts with a
- ^a.*b$ Matches anything that starts with a and ends with b
- ^a.*\d+.*b$ Matches anything starting with a, have some digits in the middle and end with b
- ^(l|b)oo Matches anything starts with l or b and then have oo
- [f-h]|[A-K]$ The last character should be f to h (capital or small)

grep

The grep command can search inside the files.

```
$ grep p friends
payam
pedram
```

There are the most important switches:

switch	meaning
-c	just show the count
-v	reverse the search
-n	show line numbers
-l	show only file names
-i	case insensitive

```
$ grep p *
friends:payam
friends:pedram
what_I_have.txt:laptop          2
what_I_have.txt:pillow          5
what_I_have.txt:apple           2
$ grep p * -n
friends:12:payam
friends:15:pedram
what_I_have.txt:2:laptop        2
what_I_have.txt:3:pillow        5
what_I_have.txt:4:apple         2
$ grep p * -l
friends
what_I_have.txt
$ grep p * -c
friends:2
what_I_have.txt:3
$
```

If is very common to combine grep and find: find . -type f -print0 | xargs -0 grep -c a | grep -v ali # find all files with **a** in them but not **ali**`

Extended grep

Extended grep is a GNU extension. It does not need the escaping and much easier. It can be used with -E option or **egrep**command which equals to grep -E.

Fixed grep

If you need to search the exact string (and not interpret it as a regex), use grep -F or fgrep so the fgrep this$ wont go for the end of the line and will find *this$that* too.

Sed

In previous lessons we saw simple sed usage. Here I have great news for you: **sed understands regex**! If is good to use -r switch to tell sed that we are using them.

```
$ sed -r "s/^(a|b)/STARTS WITH A OR B/" friends
STARTS WITH A OR Bmir
mina
jadi
STARTS WITH A OR Bita
STARTS WITH A OR Bli
hassan
```

Main switches:

switch	meaning
-r	use advanced regex
-n	suppress output, you can use p at the end of your regex (/something/p) to print the output

```
$ sed -rn "/^(a|b)/p" friends
amir
bita
ali
```

103.8 – Perform basic file editing operations using vi - 3

Objectives

- Navigate a document using vi.
- Use basic vi modes.
- Insert, edit, delete, copy and find text.
- vi
- /, ?
- h,j,k,l
- i, o, a
- c, d, p, y, dd, yy
- ZZ, :w!, :q!, :e!

Introduction

vi is a great tool! The best editor ever (some say after Emacs) and it is installed on all linux systems. Some say it is difficult to use and *abnormal* and some say it is the most natural editor there can be. Lets see.
vi can be used with simplest keyboards and over network on ssh terminals

In many systems, vi command is a link / alias to vim and there are many versions of vi. Check with --version switch:

```
$ vi --version
VIM - Vi IMproved 7.4 (2013 Aug 10, compiled Oct 20 2014 16:08:47)
Included patches: 1-273
Modified by pkg-vim-maintainers@lists.alioth.debian.org
Compiled by buildd@
Small version without GUI.  Features included (+) or not (-):
+acl            -farsi          -mouse_sgr      -tag_old_static
-arabic         -file_in_path   -mouse_sysmouse  -tag_any_white
-autocmd        -find_in_path   -mouse_urxvt     -tcl
-balloon_eval   -float          -mouse_xterm     +terminfo
...
```

To edit a file with vi, just give the file name to it:

```
$ vi file.txt
```

vi moded

vi has 2 different modes:

- **Command mode** is where you go around the file, search, delete text, copy paste, replace, ... and give other commands to the vi. Some commands start with a : and some are only a keypress.
- **Insert mode** is where what you type, goes into the file at the cursors position.

Moving the cursor

If you need to move around, use these keys:

key	function
h	One character to the left (only current line)
j	One line down
k	One line up
l	One character to the right (only current line)
w	Next word on the current line
e	Next end of word on the current line
b	Previous beginning of the word on the current line
Ctrl-f	Scroll forward one page
Ctrl-b	Scroll backward one page

Jumping

key	function
G	With no number, will jump to the end & 10G will jump to line 10
H	5H will go to the 5th line from the top of the screen
L	3L will move the cursor to the 3rd line to the last line of the screen

Editing text

These command during the *command mode* will help you enter, edit, replace, .. test:

key	function
i	Enter the insert mode
a	Enter the insert mode after the current position of the cursor
r	replace only one character
o	open a new line below the cursor and go to the insert mode
O	open a new line above the cursor and go to the insert mode
c	clear to a location and go to the insert mode the replace till there and then normal insert (cw will overwrite the current word)
d	delete. you can mix with w (dw) to delete a word. Same as cw but dw does not to to the insert mode
dd	Delete the current line
x	Delete character at the position of the cursor
p	Paste the last deleted text after the cursor
P	Paste the last deleted text before the cursor
xp	swaps the character at the cursor position with the one on its right

Searching

key	function
/	Search forward (/happiness will find the next happiness)
?	Search backward
n	repeat previous search. You can also use / and ? without any parameters)

Search wraps around to the top once the bottom of file is reached

Exiting

It is always funny when you see someone entering to the vi and now knowing how to exit! Learn these and prevent the laughter:

key	function
:q!	Quit editing without saving = runaway after any mistake
:w!	Write the file (whether modified or not). Attempt to overwrite existing files or read-only or other un
:w myfile.txt	Write to a new name
ZZ	Exit and save the file if modified
:e!	Reload the file from disk
:!	Run a shell command

Entering colon (:) during *command mode* will move the cursor to the bottom of the screen and vi will wait for your commands. Press ESC to return back to the normal command mode.

Help

You can always ask for help with :help or :help subject. This way vi will open a help text which you can use / search just like any other text. Close it with :q command.

104.1 – Create partitions and filesystems – 2

Objective

- Use various mkfs commands to set up partitions and create various filesystems such as:
- ext2/ext3/ext4
- xfs
- reiserfs v3
- vfat
- fdisk
- mkfs
- mkswap

Blocked devices

Is a technical term for any storage device which can be formatted to fixed sized blocks and blocks should be able to be accessed individually. That is Hard disks, USB Memories, CDs, ..

In long ls format, the first **b** indicates Block Device:

```
$ ls -l /dev/loop1  /dev/sd[a-z]
brw-rw---- 1 root disk 7, 1 Jan  8 10:46 /dev/loop1
brw-rw---- 1 root disk 8, 0 Jan  8 10:46 /dev/sda
```

Some block devices mostly used as one single filesystem (like CDs & Floppies) and some are divided into **Partitions** (Hard disks).

fdisk

fdisk is the main command for viewing / changing and creating partitions. -l switch is for show:

```
root@funlife:~# fdisk -l /dev/sda

Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1  *         2048  43094015  43091968  20.6G 83 Linux
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
```

- The **Boot** flag shows which partition starts the boot on DOS PCs and has no importance on LILO & GRUB
- Start and End shows the where this partition is located on the disk
- Size is size!
- ID indicated the partition format (82 is swap, 83 is linux data, ..)

It is also possible to run fdisk in interactive mode. m will show you the help:

```
root@funlife:~# fdisk /dev/sda

Welcome to fdisk (util-linux 2.25.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): m

Help:

 DOS (MBR)
  a   toggle a bootable flag
  b   edit nested BSD disklabel
  c   toggle the dos compatibility flag

 Generic
  d   delete a partition
  l   list known partition types
  n   add a new partition
  p   print the partition table
  t   change a partition type
  v   verify the partition table

 Misc
  m   print this menu
  u   change display/entry units
  x   extra functionality (experts only)

 Save & Exit
  w   write table to disk and exit
  q   quit without saving changes

 Create a new label
  g   create a new empty GPT partition table
  G   create a new empty SGI (IRIX) partition table
  o   create a new empty DOS partition table
  s   create a new empty Sun partition table
```

p displays the **current partitions**:

```
Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1  *         2048  43094015  43091968  20.6G 83 Linux
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
/dev/sda6       107704320 625141759 517437440 246.8G 83 Linux
```

You may remember the disk layouts from other chapters. fdisk can create them. Lets first delete my first partition:

```
Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1  *         2048  43094015  43091968  20.6G 83 Linux
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
/dev/sda6       107704320 625141759 517437440 246.8G 83 Linux


Command (m for help): d
Partition number (1-3,5,6, default 6): 1

Partition 1 has been deleted.
```

I'm brave! Now lets create a smaller one there:

```
Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
```

```
Select (default p): p
Partition number (1,4, default 1): 1
First sector (2048-625142447, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-43094015, default 43094015): +15G

Created a new partition 1 of type 'Linux' and of size 15 GiB.

Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1            2048  31459327  31457280    15G 83 Linux
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
/dev/sda6       107704320 625141759 517437440 246.8G 83 Linux
```

This new partitioned is not formatted but still marked 83 for later use. If I needed to use this partition as **swap** I had to set its ID to 82:

```
Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1            2048  31459327  31457280    15G 83 Linux
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
/dev/sda6       107704320 625141759 517437440 246.8G 83 Linux



Command (m for help): t
Partition number (1-3,5,6, default 6): 1
Hex code (type L to list all codes): L

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
```

```
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS fs
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         ee  GPT
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix
Hex code (type L to list all codes): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'.

Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000beca1

Device     Boot     Start       End   Sectors   Size Id Type
/dev/sda1            2048  31459327  31457280    15G 82 Linux swap / Solaris
/dev/sda2        43094016  92078390  48984375  23.4G 83 Linux
/dev/sda3        92080126 625141759 533061634 254.2G  5 Extended
/dev/sda5        92080128 107702271  15622144   7.5G 82 Linux swap / Solaris
/dev/sda6       107704320 625141759 517437440 246.8G 83 Linux
```

b is code for FAT32 (windows 95).

But all we done was in memory! We need to write it to the partition table. **'v'** will **verify** the setup:

```
Command (m for help): v
Remaining 11639159 unallocated 512-byte sectors.
```

It tells me that I have unallocated space! I'm waisting my hard but I'm fine with it. So lets save it with w command (for **write**):

```
Command (m for help): w
```

```
The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Device or resource busy

The kernel still uses the old table. The new table will be used at the next reboot or after you run
partprobe(8) or kpartx(8).
```

Formatting the partition

Linux can handle more than 100 kind of partitions but most commons are:

Format	Description
ext2	second extended filesystem was developed to address shortcomings in the Minix filesystem used in early versions of Linux. It has been used extensively on Linux for many years. There is no journaling in ext2, and it has largely been replaced by ext3 and more recently ext4.
ext3	ext2 + journaling, total storage can be 1EXAByte and each file can be 16TB, ...
ReiserFS	ReiserFS is a B-tree-based filesystem, great for large numbers of small files, journaling, no longer in active development & does not support SELinux, replaced with Reiser4.
XFS	journaling, caches to RAM, great for uninterruptible power supplies
swap	Swap space must be formatted for use as swap space, but it is not generally considered a filesystem.
vfat	FAT32, no journaling, good for data exchange with windows, does not understand permissions and symbolic links
ext4	newer than ext3

You can format your partitions with **mkfs** command (and **mkswap** for swap). This is a front end to commands like mkfs.ext3 for ext3, mkfs.ext4 for ext4 and mkfs.reiserfs for ReiserFS. full list of installed on your system is here:

```
root@funlife:~# ls /sbin/mk*
/sbin/mkdosfs  /sbin/mkfs     /sbin/mkfs.cramfs  /sbin/mkfs.ext3  /sbin/mkfs.ext4dev  /sbin/mkfs.minix
/sbin/mkfs.ntfs  /sbin/mkhomedir_helper  /sbin/mkswap
/sbin/mke2fs   /sbin/mkfs.bfs  /sbin/mkfs.ext2    /sbin/mkfs.ext4  /sbin/mkfs.fat     /sbin/mkfs.msdos
/sbin/mkfs.vfat  /sbin/mkntfs
```

The main switch is -type (or -t) to specify the format:

```
root@funlife:~# mkfs -t ext3 /dev/sda1
mke2fs 1.42.10 (18-May-2014)
/dev/sda1 contains a ext4 file system
        last mounted on /mnt on Mon Dec 22 12:04:22 2014
Proceed anyway? (y,n) y
Creating filesystem with 5386496 4k blocks and 1349040 inodes
Filesystem UUID: 0b5aad86-b507-41b9-a0ff-cf899cb92785
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

root@funlife:~#
```

This will have a same effect: mkfs.ext3 /dev/sda1

If you need to assign a lable to the partition, you have to use the -L lable_name option. Please note that in renect system, people use UUIDs instead of labels. UUID of a disk can be viewed with:

```
$ blkid /dev/sda1
/dev/sda1: UUID="59d8cbdb-0e78-4605-8aaf-cf02fcb85d2e" SEC_TYPE="ext2" TYPE="ext3"
```

GPT

Some systems use GUID Partition Table (GPT) instead of older MBR. In this case you have to use the **gdisk** tool which has more capabilities than fdisk.

```
root@funlife:~# gdisk /dev/sda
GPT fdisk (gdisk) version 0.8.8

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present


***************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format
in memory. THIS OPERATION IS POTENTIALLY DESTRUCTIVE! Exit by
typing 'q' if you don't want to convert your MBR partitions
to GPT format!
***************************************************************
```

Creating different partitions

Partition	Format type	Sample Command	Notes
/dev/sda3	ext4	mkfs -t ext4 -L data/dev/sda3	Named it *dafa*. Or use the mkfs.ext4command
/dev/sdb2	xfs	mkfs -t xfs -i size=512 /dev/sdb2	telling it to have larger inodes (normal is 256)
/dev/sda8	ReiserFS	mkfs -t reiserfs /dev/sda8	Or you can use mkreiserfs command.
/dev/sdc	FAT32	mkfs -t vfat /dev/sdc	Or you can use mkfs.vfat command
/dev/sda2	swap	mkswap /dev/sda2	will be used as swap space

104.2 – Maintain the Integrity of Filesystems – 2

Objectives

- Verify the integrity of filesystems.
- Monitor free space and inodes.
- Repair simple filesystem problems.
- du
- df
- fsck
- e2fsck
- mke2fs
- debugfs
- dumpe2fs
- tune2fs
- xfs tools

fsck

If anything bad happens for your filesystem, you will have a corrupted file system. The command to fix this is fsck. Technically this command is a front end for many commands:

```
jadi@funlife:~$ ls /sbin/*fsck*
/sbin/dosfsck      /sbin/fsck.ext2    /sbin/fsck.fat    /sbin/fsck.vfat
/sbin/e2fsck       /sbin/fsck.ext3    /sbin/fsck.minix
/sbin/fsck         /sbin/fsck.ext4    /sbin/fsck.msdos
/sbin/fsck.cramfs  /sbin/fsck.ext4dev  /sbin/fsck.nfs
```

Some of these are just hardlinks to e2fsck command

A common switch during boot is -A which tells fsck to check all file systems in **/etc/fstab** ordered by *passno* in that file which is 6th field (File systems with *passno* of 0, wont be checked during the boot.

```
root@funlife:~# fsck /dev/sdb
fsck from util-linux 2.25.1
e2fsck 1.42.10 (18-May-2014)
/dev/sdb is in use.
e2fsck: Cannot continue, aborting.


root@funlife:~# umount /dev/sdb
umount: /dev/sdb: not mounted
root@funlife:~# umount /dev/sdb1
root@funlife:~# fsck /dev/sdb
fsck from util-linux 2.25.1
e2fsck 1.42.10 (18-May-2014)
ext2fs_open2: Bad magic number in super-block
fsck.ext2: Superblock invalid, trying backup blocks...
fsck.ext2: Bad magic number in super-block while trying to open /dev/sdb
```

```
The superblock could not be read or does not describe a valid ext2/ext3/ext4
filesystem.  If the device is valid and it really contains an ext2/ext3/ext4
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblock:
   e2fsck -b 8193 <device>
 or
   e2fsck -b 32768 <device>
```

You can also check filesystems with UUID (find them with **blkid** command or with labels):

```
root@funlife:~# fsck /dev/sdb
fsck from util-linux 2.25.1
e2fsck 1.42.10 (18-May-2014)
ext2fs_open2: Bad magic number in super-block
fsck.ext2: Superblock invalid, trying backup blocks...
fsck.ext2: Bad magic number in super-block while trying to open /dev/sdb

The superblock could not be read or does not describe a valid ext2/ext3/ext4
filesystem.  If the device is valid and it really contains an ext2/ext3/ext4
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblock:
   e2fsck -b 8193 <device>
 or
   e2fsck -b 32768 <device>

root@funlife:~# blkid
/dev/sda1: LABEL="movies"
/dev/sdb1: UUID="BA82-BECD" TYPE="vfat" PARTUUID="381add66-01"
root@funlife:~# fsck LABEL=movies
fsck from util-linux 2.25.1
root@funlife:~# fsck UUID="BA82-BECD"
fsck from util-linux 2.25.1
fsck.fat 3.0.26 (2014-03-07)
/dev/sdb1: 14 files, 1972/945094 clusters
```

You can use -N switch to see what command/test is going to be executed:

```
root@funlife:~# fsck -N UUID="BA82-BECD"
fsck from util-linux 2.25.1
[/sbin/fsck.vfat (1) -- /dev/sdb1] fsck.vfat /dev/sdb1
```

If you want to check a XFS filesystem, you have to use xfs_check command

tune2fs

This is a command to **tune *ext*** file systems. It can show information and set many options. The -l option lists the current configs:

```
jadi@funlife:~$ sudo tune2fs -l /dev/sda2
tune2fs 1.42.10 (18-May-2014)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          1651a94e-0b4e-47fb-aca0-f77e05714617
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent
flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              1531904
Block count:              6123046
Reserved block count:     306152
Free blocks:              2302702
Free inodes:              1073461
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      1022
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Mon Dec  1 10:21:42 2014
Last mount time:          Sat Jan 31 17:21:51 2015
Last write time:          Sat Jan 31 17:21:51 2015
Mount count:              32
Maximum mount count:      -1
Last checked:             Mon Dec  1 10:21:42 2014
Check interval:           0 (<none>)
Lifetime writes:          103 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
First orphan inode:       786620
```

```
Default directory hash:   half_md4
Directory Hash Seed:      16c38a41-e709-4e04-b1c2-8a79d71ea7e8
Journal backup:           inode blocks
```

xfs_info

This is same as the tune2fs but for **xfs file systems**.
xfs_info should be used on mounted file systems

du & df

In many cases you want to find out about the **free space** of a **disk** or find how much space a directory is using. This space can be used by the blocks of files or inodes.

inodes contain the information about files. Information like the owner, when the last time it is used or edited, its size, if its a directory or not and peoples access rights on if. The inode number is unique within a particular filesystem and is also called files serial number.

df

The DiskFree command is used to find out about the **free** and **used space** of **file systems**.

```
jadi@funlife:~$ df -TH
Filesystem      Type     Size  Used Avail Use% Mounted on
/dev/sda2       ext4      23G   15G  7.7G  65% /
none            tmpfs    4.0K     0  4.0K   0% /sys/fs/cgroup
udev            devtmpfs  3.9G  4.0K  3.9G   1% /dev
tmpfs           tmpfs    788M  1.4M  786M   1% /run
none            tmpfs    5.0M  4.0K  5.0M   1% /run/lock
none            tmpfs    3.9G   19M  3.9G   1% /run/shm
none            tmpfs    100M   28K  100M   1% /run/user
/dev/mapper/chome ext4     243G  229G   14G  95% /home/jadi
/dev/sdb1       vfat     3.7G  7.8M  3.6G   1% /media/jadi/BA82-BECD
```

Here the `-T` switch make df to show the file system types and `-H` make numbers human readable (in powers of 1000). Please note that `-h` is also human readable but in powres of 1024 (e.g. shows 1k for 1000 bytes.

If you need the inode data, use the -i switch:

```
jadi@funlife:~$ df -i
Filesystem        Inodes  IUsed    IFree IUse% Mounted on
/dev/sda2        1531904 458616  1073288   30% /
none             1007533      4  1007529    1% /sys/fs/cgroup
udev             1003703    542  1003161    1% /dev
tmpfs            1007533    644  1006889    1% /run
none             1007533      3  1007530    1% /run/lock
none             1007533    162  1007371    1% /run/shm
```

```
none            1007533     33  1007500    1% /run/user
/dev/mapper/chome 16171008 269293 15901715    2% /home/jadi
/dev/sdb1             0      0       0     - /media/jadi/BA82-BECD
```

vfat file format has no inodes; there is no owner or access rights on vfat filesystems.

du

The **DiskUsage** command give information about the used space of **directories and files**. The common switches are:

switch	usage
-h	print sizes in powers of 1024 (e.g., 1023M)
-H	print sizes in powers of 1000 (e.g., 1.1G)
-c	show the grand total
--max-depth 2	shows only 2 directories furthur
-s	Only shows the summary and not all the directories one by one

```
jadi@funlife:~/w/lpic$ du
16    ./101
701456    ./done
701464    ./Logo/chert
704588    ./Logo
12    ./data
12    ./100
9432884    .
jadi@funlife:~/w/lpic$ du -c
16    ./101
701456    ./done
701464    ./Logo/chert
704588    ./Logo
12    ./data
12    ./100
9432884    .
9432884    total
jadi@funlife:~/w/lpic$ du -hs
9.0G    .
```

Repairing

We used the **fcsk** for showing file system information but if is designed to ***fix* file systems** too. If the boot time check find a problems, you will be put into a command line to fix the problems.
On non-journaling file systems (ext2) the fsck will show you many questions about each block and you have to say y if you want it to fix them. On journaling file systems (ext3&4, xfs, ..) the fsck has much less tasks to perform.
For **xfs file systems**, we have **xfs_check** command
An important switch is -n which causes these commands **not to fix** anything and just show what was going to be done.

debugfs

This is an interactive tool for **debug an ext filesystem**. It opens the filesystem in read-only mode unless we tell it not to (with -w option). It can un-delete files and directories..

```
root@funlife:~# debugfs /dev/sda2
debugfs 1.42.10 (18-May-2014)
debugfs:  cd /etc/       <-- cd
debugfs:  pwd          <-- show were am I
[pwd]   INODE: 524289  PATH: /etc
[root]  INODE:      2  PATH: /
debugfs:  stat passwd       <-- show data on one file
Inode: 527187   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 1875144872    Version: 0x00000000:00000001
User:     0   Group:     0   Size: 2145
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x548d4241:a7b196fc -- Sun Dec 14 11:24:41 2014
 atime: 0x54cc635b:6acfc148 -- Sat Jan 31 08:38:43 2015
 mtime: 0x548d4241:a01076f8 -- Sun Dec 14 11:24:41 2014
crtime: 0x548d4241:9f1c52f8 -- Sun Dec 14 11:24:41 2014
Size of extra inode fields: 28
EXTENTS:
(0):2188172
debugfs:  ncheck 527187       <-- node check an inode
Inode    Pathname
527187    /etc/passwd
debugfs:  q          <-- quit
```

Superblock

Unix systems use superblocks to save *filesystem metadata*. Most of the times this block is located at the beginning of the file system and replicated on other locations too. The -**n** of **mke2fs** displays superblock locations

```
# mke2fs -n /dev/sda7
mke2fs 1.41.9 (22-Aug-2009)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
7159808 inodes, 28637862 blocks
1431893 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
874 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872
```

Other tools

For the LPIC exam, it is good to know about these commands.

filesystem	command	usage
ext	tune2fs	Show or set ext2 and ext3 parameters or even set the journaling options
ext	dumpe2fs	Prints the super block and block group descriptor information for an ext2 or ext3 filesystem.
ext	debugfs	Is an interactive file system debugger. Use it to examine or change the state of an ext2 or ext3file system.
reiserfs	reiserfstune	show and set parameters
reiserfs	debugreiserfs	Prints the super block and block group descriptor information for an ext2 or ext3 filesystem.
XFS	xfs_info	display information
XFS	xfs_growfs	expand file system
XFS	xfs_admin	change parameters on XFS file systems
XFS	xfs_repair	repair the problems
XFS	xfs_db	checks and debugs the filesystem

104.3 – Control mounting and unmounting of filesystems – 3

Objectives

Configure the mounting of a filesystem. Tasks include manually mounting and unmounting filesystems, configuring filesystem mounting on bootup, and configuring user-mountable removable filesystems.

- Mount and unmount filesystems manually
- Configure filesystem mounting on bootup
- Configure user-mountable, removable filesystems

Mounting and Unmounting

- Describe the linux filesystem concept. A huge tree.
- There are other kinds of mountings: tmpfs, NFS, ..
- It is better to mount on empty directories

Basic commands

```
cat /etc/fstab
mount /dev/sda1 /media
umount /media
```

Some switches

```
mount -t ext4 /dev/sda1 /media
mount -o remount,ro /dev/sda1
```

*Get **info** on **UUID** and Label and Format*

```
blkid /dev/sda2
```

Bootup

/etc/fstab

- **file system:** Label, UUID, device
- **mount point:** swap or none for swap
- **type:** can be auto
- **options:** defaults, rw / ro, noauto, user, exec / noexec, noatime
- **dump:** do dump command backup this? mostly 0
- **pass:** Non-zero values of pass specify the order of checking filesystems at boot time (seen in Integrity of file systems)

note:

- User-mounted filesystems default to noexec unless exec is specified after user.
- noatime will disable recording of access times. Not using access times may improve performance.

swap

```
swapon
swapoff
swapon -s
```

104.5 – Manage file permissions and ownership – 3

Objectives

- Manage access permissions on regular and special files as well as directories.
- Use access modes such as suid, sgid and the sticky bit to maintain security.
- Know how to change the file creation mask.
- Use the group field to grant file access to group members.
- chmod
- umask
- chown
- chgrp

Users and Groups

A linux system can have many users and many groups. You can login with one user and use **su** command to change to another group. Each user belongs to **one primary group** and can be a member of other groups too.

There commands like **whoami**, **groups** and **id** to determine who you are.

```
- $ whoami
jadi
- $ groups
jadi adm cdrom sudo dip plugdev netdev lpadmin sambashare debian-tor
- $ id
uid=1000(jadi) gid=1000(jadi)
groups=1000(jadi),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),102(netdev),108(lpadmin),124(samb
ashare),125(debian-tor)
- $ su root -
Password:
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
- # id
uid=0(root) gid=0(root) groups=0(root)
- # exit
exit
- $ whoami
jadi
```

id shows both user and group information

There info are stored in **/etc/passwd** and **/etc/group**

```
$ cat /etc/group | grep adm
adm:x:4:syslog,jadi
lpadmin:x:108:jadi
```

File ownership & permissions

Files also belong to one user and one group.

```
$ ls -l /sbin/fdisk ~/w/lpic/notes.txt
-rw-rw-r-- 1 jadi users    576 Dec  7 22:30 /home/jadi/w/lpic/notes.txt
-rwxr-xr-x 1 root root 267176 Oct 15 18:58 /sbin/fdisk
```

As you can see, the notes.txt belongs to jadi and a group called *users*.

In many distros, when you create a user, system creates a group with same name and assign that users files to that group

Another part of the ls -l command shows the permissions on that file. Linux system users a 3 layer permission: permissions for the owner, for the group member and for *others*.

Each layer also has 3 different parts:

- **R**ead
- **w**rite (including deletion and edit)
- **e**xecute (reading directory content).

These are shown at the first column of ls - command as -rw-rw-r--. The character meanings are as follow:

bit	meaning
1	What this entry is. Dash (-) is for ordinary files, 'l' is for links & 'd' is for directory
2,3,4	read, write and execute access for the owner
5,6,7	read, write and execute access for the group members
8,9,10	read, write and execute access for other users
11	Indicated if any other access methos (such as SELinux) applies to this file - not part of the 101 exam

As you can see in our example, characters 2 to 10 show the accesses. A - there means "no access on this part" and read, write and execute are shown by r, w & x.
In the following line:

```
$ ls -l /sbin/fdisk
-rwxr-xr-x 1 root root 267176 Oct 15 18:58 /sbin/fdisk
```

We can see that the fdisk can be read, written and and be executed by its owner (root), only be read and executed by whoever is part of group root and be read and executed by all other users.

Let's look at another example:

```
$ ls -l /home/
total 12
drwxr-xr-x 160 jadi jadi 12288 Feb  7 11:44 jadi
```

The first character is a d so this is a directory! The owner (jadi) has read, write and execute access but other members of the **group** jadi and **others** only have read and execute access on this directory (execute means that they can see the files inside it).

Chanhging permissions

It is possible to change the permissions on files & directories using the **chmod** command. There are to ways to tell this command what you want to do:

1. using octal codes
2. using short codes

When using octal codes, you have to to create an octal string to tell chmod what you want to do. This way, 0 means no access, means execute, 2 means write and 4 means read. So, if you want to give read+execute, you have to give 4+1 which is 5. This table shows every possible combination:

Symbolic	Octal
rwx	7
rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--x	1
---	0

If you want to give **rwx** to owner, **rx** to group and only **x** to others, you have to use **751**:

```
$ ls -ltrh myfile
-rw-rw-r-- 1 jadi jadi 0 Feb  8 21:01 myfile
$ chmod 751 myfile
$ ls -ltrh myfile
-rwxr-x--x 1 jadi jadi 0 Feb  8 21:01 myfile
```

But there is also an *easier* method. You can use **+x** to **give execute permission**, **+r** to **give read permission** and **+w** to **give read permission**. Removing these permissions will be like -r.

```
$ ls -ltrh myfile
-rwxr-x--x 1 jadi jadi 0 Feb  8 21:01 myfile
$ chmod u-x myfile
$ ls -ltrh myfile
-rw-r-x--x 1 jadi jadi 0 Feb  8 21:01 myfile
$ chmod +x myfile
$ chmod uo+xr myfile
$ ls -ltrh myfile
-rwxr-xr-x 1 jadi jadi 0 Feb  8 21:01 myfile
```

you can tell chmod whos permission should be granted or removed by doing things like u+r (give read to user), og-w (remove write for other and group).

One very common switch on **chmod** is **-R** for recursive chmoding on files. This will give read permission of all files inside /tmp/ to any user:

```
# chmod -R o+r /tmp
```

Access modes

So, you have access only to your files. But how you should change your password? or use programs which needs access to system files? You should be able to access /etc/passwd or /etc/shadow to change your password but you should not be able to access other people files!

Normally when you run a program, it runs with *your* access levels but linux has two special bits on each file; **suid** (set user id) and **guid** (set group id). If these are set on a file, that file be will be executed with the access of the **owner** of the file and not the user who is running it.

```
$ ls -ltrh /usr/bin/passwd
-rwsr-xr-x 1 root root 50K Jul 18  2014 /usr/bin/passwd
```

Did you note the s in the place of *executable bit* for the user and for the group? That means when any user runs this program, it will be run be the access of the owner of the file (which is root) instead of that users id.

It is possible to set / unser the suid and sgid using chmod and +s or -s instead of x.
The last special option is chmod is the **sticky bit** which lets only the owner of the file to delete it, even if other users have write (delete) access on that directory. This is good for places like /tmp.
Sticky bit is identified by t and will be shown on the last bit of a directory:

```
$ ls -dl /tmp
drwxrwxrwt 13 root root 77824 Feb  8 21:27 /tmp
```

As you can see the sticky bit is set and although all users have write access in this directory, they wont be able to delete each others files.

Lets review how you can set these access modes:

access mode	octal	symbolic
suid	4000	u+s
guid	2000	g+s
sticky	1000	t

guid on a directory will force any new file in that directory to have the guid of the directory itself.

umask

But what will be the access of the new files? What happens when you touch a new file? This is set with **umask**. This command tells the system what permissions **should not be given to** new files:

```
$ umask
0002
```

Which removes write (2) permissions from files.

If we need to change **umask**, it can be done with the same command:

```
$ umask
0002
$ touch newfile
$ ls -ltrh newfile
-rw-rw-r-- 1 jadi jadi 0 Feb  8 21:38 newfile
$ mkdir newdir
$ ls -ltrhd newdir
drwxrwxr-x 2 jadi jadi 4.0K Feb  8 21:38 newdir
$ umask u=rw,g=,o=
$ touch newerfile
$ ls -l newerfile
-rw------- 1 jadi jadi 0 Feb  8 21:41 newerfile
$ umask
0177
```

Note how we use u=rw,g=,o= to tell umask or chomd what we exactly need.

Changing owner and groups

If you need to change the **ownership** or **group** belonging of a file or directory, use the **chown** command:

```
$ ls -ltrh newfile
-rw-rw-r-- 1 jadi jadi 0 Feb  8 21:38 newfile
$ chown root:root newfile
chown: changing ownership of ‘newfile’: Operation not permitted
$ sudo chown root:root newfile
[sudo] password for jadi:
$ ls -ltrh newfile
-rw-rw-r-- 1 root root 0 Feb  8 21:38 newfile
```

A common switch is **-R** to do the **chown** recursively and the general style is chown newuser:newgroup file.

There is also a command specially for changing the group:

```
$ sudo chgrp postgres newfile
$ ls -ltrh newfile
-rw-rw-r-- 1 root postgres 0 Feb  8 21:38 newfile
```

If a user is member of different groups, she can change her **default group** using the **newggrp** command:

```
$ touch newfile
$ ls -ltrh newfile
-rw------- 1 jadi jadi 0 Feb  8 21:53 newfile
$ groups
jadi adm cdrom sudo dip plugdev netdev lpadmin sambashare debian-tor
$ newgrp adm
$ touch newerfile
$ ls -ltrh new*
-rw------- 1 jadi jadi 0 Feb  8 21:53 newfile
-rw------- 1 jadi adm  0 Feb  8 21:54 newerfile
```

104.6 – Create and change hard and symbolic links – 2

Objectives

- Create links.
- Identify hard and/or softlinks.
- Copying versus linking files.
- Use links to support system administration tasks.
- ln
- unlink

On a storage device, a file or directory is contained in a collection of blocks. Information about a file is contained in an inode, which records information such as the owner, when the file was last accessed, how large it is, whether it is a directory or not, and who can read from or write to it.

$ **ls -i $ ls -R**

A link is simply an additional directory entry for a file or directory

$ **ls -il $ vi link2file.txt #will edit both $ mv myfile.txt**

A **link** is simply an additional directory entry for a file or directory, allowing two or more names for the same thing.

- A **hard link** is a directory entry that points to an inode
- A **soft link** or symbolic link is a directory entry that points to an inode that provides the name of another directory entry.

The exact mechanism for storing the second name may depend on both the file system and the length of the name. Symbolic links are also called symlinks.

Soft links, or **symlinks**, merely point to another file or directory by name rather than by inode. Soft links can cross file system boundaries.

$ cp myfile.txt newfile.txt $ vi myfile.txt $ cat myfile.txt $ cat softlink.txt $ cat hardlink.txt $ cat newfile.txt $ ls -il

You can create **hard links** only for files and not for directories. The exception is the special directory entries in a directory for the directory itself and for its parent (. and ..)

$ **ls -ltrhi /etc/grub2.cfg**

You will get an error if you attempt to create hard links that cross file systems or that are for directories.

$ **ln mydir link2dir # error! $ ln -s mydir link2dir**

If you are using relative names, you will usually want the current working directory to be the directory where you are creating the link; otherwise, the link you create will be relative to another point in the file system.

$ **ln -s myfile.txt mydir/ #broken link**

$ **cd mydir $ ln -s ../myfile.txt .**

we can find symbolic links with ls -i or even find:

$ **find . -type l**

and they are useful!

$ which java # linking to specific versions $ ls -ltrhi /etc/grub2.cfg # easier admin tasks $ ls -l /usr/lib64/ # keeping libraries clean

104.7 – Find system files and place files in the correct location – 2

Objectives

- Understand the correct locations of files under the FHS.
- Find files and commands on a Linux system.
- Know the location and purpose of important file and directories as defined in the FHS.
- find
- locate
- updatedb
- whereis
- which
- type
- /etc/updatedb.conf

FHS

Filesystem Hierarchy Standard (FHS) is a document describing the Linux / Unix file hierarchy. It is very useful to know these because it lets you easily find what you are looking for:

directory	usage
bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration
lib	Essential shared libraries and kernel modules
media	Mount point for removable media
mnt	Mount point for mounting a filesystem temporarily
opt	Add-on application software packages
sbin	Essential system binaries

directory	usage
srv	Data for services provided by this system
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data
home	User home directories (optional)
lib	Alternate format essential shared libraries (optional)
root	Home directory for the root user (optional)

The **/usr** filesystem is the second major section of the filesystem, containing **shareable**, **read**-**only data**. It can be shared between systems, although present practice does not often do this.

The **/var** filesystem contains **variable data files**, including spool directories and files, administrative and logging data, and transient and temporary files. Some portions of /var are not shareable between different systems, but others, such as /var/mail, /var/cache/man, /var/cache/fonts, and /var/spool/news, may be shared.

Path

A general linux install has a lot of files; 741341 files in my case. So how does it find out where to look when you type a command? This is done by a variable called **PATH**:

```
$ echo $PATH
/home/jadi/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
;/home/jadi/bin/
```

And for root user:

```
# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

As you can see, this is the list of directories separated with a colon. Obviously you can change your path with **export PATH=$PATH:/usr/new/dir** or put this in **.bashrc** to make it permanent.

which, type and whereis

The **which** command shows the first appearance of the command given in path:

```
$ which mkfd
$ which mkfs
/sbin/mkfs
```

use the -a switch to show all appearance in the path and not only the first one.

But what happens if you which for ?

```
$ which for
$ type for
for is a shell keyword
```

As you can see, which did not found anything for for and we used type.

```
$ type type
type is a shell builtin
$ type for
for is a shell keyword
$ type mkfs
mkfs is /sbin/mkfs
$ type mkfd
bash: type: mkfd: not found
```

The type command is more general that which and also understand and shows the *bash keywords*.
Another useful command in this category is whereis. Unlike which, whereis shows man pages and source codes of programs alongside their binary location.

```
$ whereis mkfs
mkfs: /sbin/mkfs.bfs /sbin/mkfs.ext3 /sbin/mkfs.ext4 /sbin/mkfs.vfat /sbin/mkfs.cramfs
/sbin/mkfs.minix /sbin/mkfs.ext2 /sbin/mkfs.msdos /sbin/mkfs.fat /sbin/mkfs.ntfs /sbin/mkfs.ext4dev
/sbin/mkfs /usr/share/man/man8/mkfs.8.gz
$ whereis ping
ping: /bin/ping /usr/share/man/man8/ping.8.gz
$ whereis chert
chert:
$
```

Find

We have already seen this command in detail but lets see a couple of new switches.

- The -user and -group specifies a specific user & group
- The -maxdepth tells the find how deep it should go into the directories.

```
$ find /tmp/ -maxdepth 1 -user jadi | head
$ find /tmp/ -maxdepth 1 -user jadi | head
/tmp/asheghloo.png
/tmp/tmpAN6Drb
/tmp/wrapper-24115-2-out
/tmp/sni-qt_goldendict_20048-sRlmvN
/tmp/asheghloo.gif
/tmp/zim-jadi
/tmp/3la.txt
/tmp/unity_support_test.0
/tmp/batman.jpg
```

Or even find the files not belonging to any user / group with -nouser and -nogroup.
Like other *tests*, you can add a ! just before any phrase to negate it. So this will find files **not belonging** to jadi: find . ! -user jadi

Locate & Updatedb

Find and know that it is slow, It searches the file system on each run but lets see the fastest command:

```
$ locate happy
/home/jadi/.Spark/xtra/emoticons/Default.adiumemoticonset/happy.png
/home/jadi/.Spark/xtra/emoticons/sparkEmoticonSet/happy.png
/home/jadi/Downloads/jadi-net_radio-geek_040_antihappy.mp3
/usr/share/emoticons/kde4/unhappy.png
/usr/share/pixmaps/fvwm/mini.happy.xpm
/usr/share/pixmaps/pidgin/emotes/default/happy.png
/usr/share/pixmaps/pidgin/emotes/small/happy.png
/usr/src/linux-headers-3.13.0-40-generic/include/config/happymeal.h
/usr/src/linux-headers-3.16.0-25-generic/include/config/happymeal.h
/usr/src/linux-headers-3.16.0-28-generic/include/config/happymeal.h
/usr/src/linux-headers-3.16.0-29-generic/include/config/happymeal.h
```

And it is fast:

```
$ time locate kernel | wc -l
11235

real    0m0.341s
user    0m0.322s
sys     0m0.015s
```

This is fast because its data comes from a **database** created with **updatedb** command which is usually run on a daily basis with a cron job. Its configuration file is /etc/updatedb.conf or /etc/sysconfig/locate:

```
$ cat /etc/updatedb.conf
PRUNE_BIND_MOUNTS="yes"
# PRUNENAMES=".git .bzr .hg .svn"
PRUNEPATHS="/tmp /var/spool /media /home/.ecryptfs"
PRUNEFS="NFS nfs nfs4 rpc_pipefs afs binfmt_misc proc smbfs autofs iso9660 ncpfs coda devpts ftpfs devfs mfs shfs sysfs cifs lustre tmpfs usbfs udf fuse.glusterfs fuse.sshfs curlftpfs ecryptfs fusesmb devtmpfs"
```

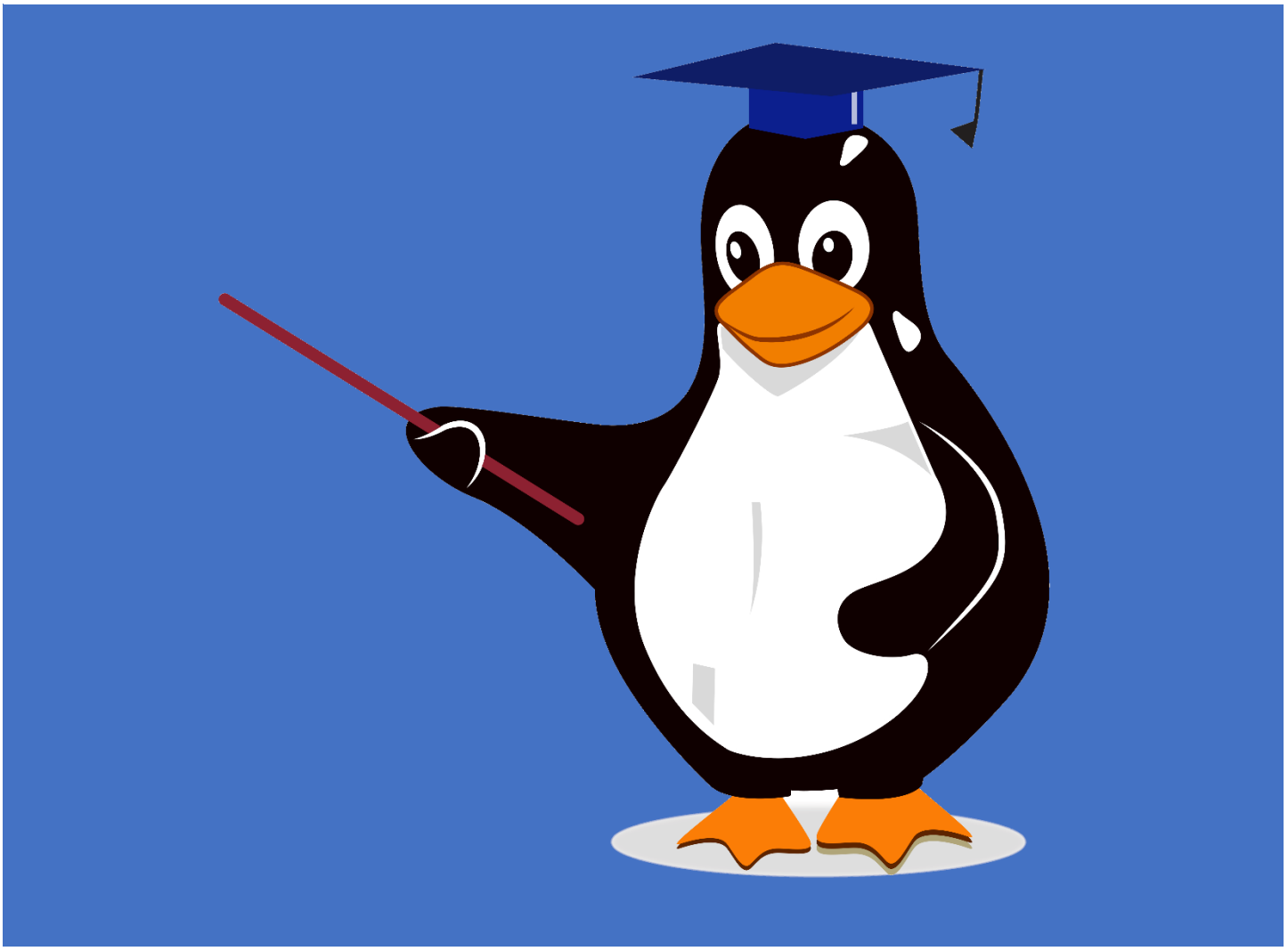
Please note that you can update the db by running **updatedb** as root and get some info about it by -S switch of locatecommand:

```
$ locate -S
Database /var/lib/mlocate/mlocate.db:
        73,602 directories
        711,894 files
        46,160,154 bytes in file names
        18,912,999 bytes used to store database
```

And... the LPIC1 exam 101 is DONE! Congrats.

http://j.mp/jadilpic1

LPIC-1 Exam 102



105.1 - Customize and use the shell environment *Weight - 4*

Objectives

- Set environment variables (e.g. PATH) at login or when spawning a new shell
- Write Bash functions for frequently used sequences of commands
- Maintain skeleton directories for new user accounts Set command search path with the proper directory

Terms

- .
- source
- /etc/bash.bashrc
- /etc/profile
- env
- export
- set
- unset
- ~/.bash_profile
- ~/.bash_login
- ~/.profile
- ~/.bashrc
- ~/.bash_logout
- function
- alias
- lists

Environment variables

We should set them when we login or change shell. Oh! And what happen when we are not logged in? ;)

login vs non-login shell

Sometimes you *login* into the shell, say after a ssh but sometimes you just *open* a shell; say in GUI.

login shell

This happens when you give your user and pass to enter a shell. Many steps are involved to setup your variables and settings. These are the steps:

1- /etc/profile is run

2- A line in /etc/profile runs whatever is in /etc/profile.d/*

Now the global profile is loaded and system will go for user specific profiles:

3- /home/USERNAME/.bash_profile

4- /home/USERNAME/.bash_login

5- /home/USERNAME/.profile

Note that only one of the 3, 4 & 5 will be run. The system will go for .bash_profile and IF IT IS NOT THERE will try for .bash_login and IF IT IS NOT THERE will try to run .profile. If any of these exists, the system wont look any furthur. So if you have only 4 & 5, only the 4 will be run.

At the end, the system loads:

6- /home/USERNAME/.bashrc

which is users information (like aliases).

Note: /etc/profile also loads /etc/bashrc or /etc/bash.bashrc

Interactive (non-login) shell

If you run a shell in an **interactive mode** (non-login) shell say from a GUI, only two things will be run:

1. /etc/bash.bashrc (or in some systems /etc/bashrc)
2. /home/USERNAME/.bashrc

Adding global configs for login shell

You can add your global new config files int /etc/profile.d/ (with .sh at the end). It is cleaner and better than editing the /etc/profile because an update can overwrite your changes if you do so.

Adding global configs for interactive/non-login shell

You can use /etc/bash.bashrc file (some systems /etc/bashrc). This is good for *aliases* and other global configs.

user specific configs

Most of the time PATH and env vars go into the in ~/.bash_profile and aliases go into the ~/.bashrc. Have a look at them!

Aliases

Most of the time they are defined in ~/.bashrc and look like this:

```
alias ll='ls -alF'

alias la='ls -A'

alias l='ls -CF'

It is kind of a shortcut.
```

/etc/skel

This directory contains files which will be used as a **starting template** for each new user.

.bash_logout

Runs when you logout from a login shell. In many distros it only clears the screen so the next person will not be able to watch what you were doing before you logout.

. (and source)

Yes, only a dot! This is a shortcut for the bash source command. You can find it in files like /etc/profile. It runs the executable in front of it as part of the current environment.

Note: If you just execute a file (without source or dot) bash creates a child, runs the executable there and then closes it.

Functions

Like "real" programming languages, Bash has functions, though in a somewhat limited implementation. A function is a subroutine, a code block that implements a set of operations, a "black box" that performs a specified task. Wherever there is repetitive code, when a task repeats with only slight variations in procedure, then consider using a function.

```
funnyls () {
        ls -ltrh
        echo "This is a funny ls"
}
```

Set

Set allows you to change the values of shell options and set the positional parameters, or to display the names and values of shell variables.

Using set we can configure how bash works. These are some samples:

switch	result
-b	Cause the status of terminated background jobs to be reported immediately, rather than before printing the next primary prompt.
-e	return in case a pipline, command, ... return non-zero
-n	Read commands but do not execute them; this may be used to check a script for syntax errors. This option is ignored by interactive shells.
-t	Exit after reading and executing one command.
-C	Prevent output redirection using '>', '>&', and '<>' from overwriting existing files.

Export

Set an environment variable. Mark each name to be passed to child processes in the environment.

```
$ export name=jadi
$ echo $name
jadi
```

Note: when exporting, the variable will exists in child processes (commands you run from the current shell). If you only say `name=jadi` and run a new command in your shell, the name **wont be** jadi in **that** shell.

```
$ name=jadi
$ echo $name
jadi
$ bash
$ echo $name
```

```
$ exit
exit
$ echo $name
jadi
$ export name=jadi
$ bash
$ echo $name
jadi
```

Unset

This command *has nothing to do* with set command! This can unset the defined variables or functions.

```
$ name=jadi

$ echo $name

jadi

$ unset name

$ echo $name
```

You can also unset functions.

env

can set, remove or display variables or even run a command in a modified environment.

```
Syntax
   env [OPTION]... [NAME=VALUE]... [COMMAND [ARGS]...]

Options

 -u NAME
 --unset=NAME
     Remove variable NAME from the environment, if it was in the
     environment.

 -
 -i
 --ignore-environment
     Start with an empty environment, ignoring the inherited
     environment.
```

lists

Bash even has arrays! We will see them later when scripting but you can do things like:

```
$ list=(salam donya man injam)
$ echo ${list[1]}
donya
$ list=("salam donya" "man injam")
$ echo ${list[1]}
man injam
```

105.2 - Customize or write simple scripts - 4

Objectives

- Use standard sh syntax (loops, tests)
- Use command substitution
- Test return values for success or failure or other information provided by a command
- Perform conditional mailing to the superuser
- Correctly select the script interpreter through the shebang (#!) line
- Manage the location, ownership, execution and suid-rights of scripts

Terms and Utilities

- for
- while
- test
- if
- read
- seq
- exec

Shell Scripts

Are a way of automating tasks.

Shebang

If a line starts with **#!** it is called **shebang** and tells the shell what ***interpreter*** to use for running this script.

Note: Normally a `#` at the beginning of a script is for showing *comments*. Do not confuse it with *Shebang* (#!)

In many cases we run shells with `#!/bin/bash` or `#!/bin/sh`

We can use the command we already know in our shell scripts. A sample is:

```
#!/bin/bash


echo

echo "We are learning! Wowww..."

echo
```

Variables

Already seen in some parts. You can define variables like this **VARNAME=VALUE**. A sample:

```
#!/bin/bash

NAME=Jadi

echo

echo "$NAME is learning! Wowww..."

echo
```

Note: you can also do NAME="Jadi"

Command substitution

Sometimes you need to have a variable with the result of something to a variable. In this case you can use $() construct:

```
FILES=$(ls -1)
```

Executing scripts

If the file is executable, we can run them using the ./script_name.sh if we are in the same directory, or give the complete path or include their directory ine $PATH variable. As you can see they are just normal programs.

Another way is giving our scripts name as a parameter to the `bash` or `sh` commands.

Note: you know that for making a file executable we can do `chmod 755 filename` or `chmod +x fiename`.

Conditions

Up to now, we were just running commands one by one. That is not very *programmatic*. If we are going to have some *logic* in our programs, we need *conditions* and *loops*. First we will cover conditions, using the `if` command. Its usage is like this:

```
if [condition]

then

  do something

  do another thing

else

  do new things

  even funnier things

fi
```

Note: else part is optional, `if`, `then`, `fi` is enough.

Conditions can be TRUE or FALSE. A very simple conditions is `if [ "Linux" = "Linux" ]`. Silly? I know but we will change it soon but for now, learn the syntax! Specially the *spaces* and = for checking if two strings are equal.

```
#!/bin/bash

kernel=$(uname -s)

if [ $kernel = "Linux" ]

then

   echo YES. You are using a Linux

else

   echo "Not a linux :("

fi
```

Note spaces and using double quotes (") on second echo because it has `(`character which will be interpreted by bash if we do not enclose the string in a double quote.

The actual checking of the condition is done by `test` command which is writer as `[ some test]`. There are the other options:

conditions	what is means
"a" = "b"	if two strings are equal (here it will return False)
"a" != "b"	string a *is not equal* to string b
4 -lt 40	if 4 is *lower than* 40 (True)
5 -gt 15	if 5 is *greater than* 15 (False)
5 -ge 5	if 5 is *greater or equal* 5
5 -le 3	if 5 is *lower or equal* to 3
9 -ne 2	9 is *not equal* with 2 (True)
-f FILENAME	if file FILENAME exists
-s FILENAME	if file exists and its size is more than 0
-x FILENAME	file exists and is executable

Read

Using `read` we can read the user input. Look at this:

```
1 #!/bin/sh
 2
 3 echo "what is your name?"
 4 read NAME
 5
 6 echo "Hello $NAME"
 7
 8 if [ $NAME = "Jadi" ]
 9 then
10      echo "Oh I know you!"
11  else
12      echo "I wish I knew you"
13 fi
14
15 echo "Bye"
```

For loop

Generally, loops are used to run a specific set of commands more than once. The syntax is like this:

```
for VAR in SOME_LIST;
do
 some stuff with $VAR
 some other stuff
done
```

Note the `in`, `;`, `do` and `done`.

On each loop, the VAR will be equal to one of the SOME_LIST elements. SOME_LIST can be numbers, name of files, words, ...

```
for NUM in 1 2 3 4 5 6;
do
   echo $NUM
done
```

But what if you needed to go 1 to 42? We have the `seq` command which can be used like `seq 1 42` or a shorthand like `{1..42}`.

Good part is we can use non-numbers too!

```
for FILE in $(ls);
do
   echo $FILE
   wc $FILE
done
```

While loop

This is another kind of loop but loops *while* a conditions is TRUE. This is the syntax:

```
while [condition]
do
   do something
   do anohter thing
done
```

Note: If your condition will remains true all the time, the while loop will run *forever*. This is called an *infinite loop*

This is sample:

```
VAR=52

while [ $VAR -gt 42 ]
do
   echo VAR is $VAR and it is still greater than 42
   let VAR=VAR-1
done
```

we will have an infinite loop if we use `let VAR=VAR+1`. Ctrl+C will help us to *break* the loop.

Note the `let` usage! If you just just say `VAR=1` and then `VAR=$VAR+1`, then VAR will be equal to `1+1` as an string!.

Mailing the root user

For sending mail, you need to install **mailutils**. Then the `mail` command will send emails. You can send the mail to the root user by issuing this command:

```
jadi@funlife:~$ mail root
Cc:
Subject: Hi there root
hello there. This is my mail
```

And root will get this email. She can read it using `mail` command.

If you need to send emails in a script, just do:

```
$ echo "Body!" | mail -s "Subject" root
```

This can be easily embeded as poart of your scripts!

105.3 - SQL data management - *2*

Objectives

- Use of basic SQL commands
- Perform basic data manipulation

Terms and Utilities

- Insert
- Update
- Select
- Delete
- From
- where
- group by
- order by
- join

Databases

This module is about SQL language and MySQL is one of the many SQL databases. For this lesson, a database consists of some **table**s and each table has some **row**s and **filed**s. Lets have a look. In this lesson we are not going to *create* or *design* databases. You only need to have a general understanding of databases (SQL databases) and know some command to use (read query or update or add to them). The database I'm going to use in this lesson is called lpic and has two tables contact and info.

mysql command line

You only need to know that mysql is a command line program to interactively connect to a mysql-server. I use it like this:

```
$ mysql -u root -p
```

Which means I'm going to use user root and will provide a password. It was also possible to say:

```
$ mysql -u root -p mypass lpic
```

To provide the pass on command line (not a good idea for security reasons!) and tell mysql program to connect to lpic database when it starts.

Using a database

When you connect to a database, you have to use the use command to select which database you are going to issue commands on. Normally a database server (say mysql) can have 100s of different databases in it, each for one user or program.

```
jadi@funlife:~$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 17

Server version: 5.6.25-0ubuntu0.15.04.1 (Ubuntu)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;

+--------------------+

| Database           |

+--------------------+

| information_schema |

| bad                |

| good               |

| lpic               |

| mysql              |

| performance_schema |
```

```
| ugly                |

+--------------------+

7 rows in set (0.00 sec)



mysql> USE LPIC;

Reading table information for completion of table and column names You
can turn off this feature to get a quicker startup with -A


Database changed

mysql> SHOW TABLES;

+----------------     +

| Tables_in_lpic |

+----------------     +

| info             |

| phonebook         |

+----------------     +

2 rows in set (0.00 sec)
```

As you can see mysql is friendly and shows lovely tables! I've told her to use lpic and then show tables and now I know that I have two tables: info & phonebook.

Note: it is common to type MYSQL commands in CAPITAL LETTERS and names and values and .. in lower case.

SELECT

SELECT is obvious! It selects from a table. When we are not sure what field we are looking for, we can select * to get all fields.

```
mysql> SELECT * FROM phonebook;

+--------  +------------------     +----------------   +

| name  | email                 | phone         |

+--------  +------------------     +----------------   +

| jadi    | jadi@jadi.net  | +9890something |

| nasrin | nasrin@lpic.test | +9898989898         |

| sina | far@from.here | +687randomnum |         |
haale | | 0935secret | +--------+-------------------+----
------------+ 4 rows in set (0.00 sec)
```

Or ask for a specific field:

```
mysql> SELECT name FROM phonebook;
+--------+
| name   |
+--------+
| jadi   |
| nasrin |
| sina   |
| haale  |
+--------+
3 rows in set (0.00 sec)
```

WHERE

You can **add *conditions*** to your SQL queries using **WHERE**. Let's have a look at the other table we have:

```
mysql> SELECT * FROM info;

+--------+--------+--------+-------+

| name  | height | weight | mood |

          +----
+--------  ----    +--------    +-------    +

| jadi    |       180 |     74 | happy |

| sina    |       175 |      81 | happy |

| nasrin |        174 |   68 | happy |

| mina    |       171 |   59 | sad       |

          +----
+--------  ----    +--------    +-------    +

4 rows in set (0.00 sec)
```

What if we only wanted to see our *happy* friends?

```
mysql> SELECT * FROM info WHERE mood = 'happy';

+--------+--------+--------+-------+

| name  | height | weight | mood |

          +----
+--------  ----    +--------    +-------    +

| jadi    |       180 |     74 | happy |

| sina    |       175 |      81 | happy |

| nasrin |        174 |    68 | happy |

          +----
+--------  ----    +--------    +-------    +

3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM info WHERE mood = 'happy' AND weight >= 80;

          +-
          ---
          ---
+-------- -      +--------    +-------    +
| name        | height | weight | mood |
          +-
          ---
          ---
+-------- -      +--------    +-------    +
| sina    |      175 |    81 | happy |
          +-
          ---
          ---
+-------- -      +--------    +-------    +

3 rows in set (0.00 sec)
```

Or if I only needed the **name**:

```
mysql> SELECT name FROM info WHERE mood = 'happy' AND weight >= 80;

+------+

| name |

+------+

| sina |

+------+
```

ORDER BY

This is used if you want to **sort** the data based on one field. Here I'm checking my phone book based on people's names:

```
mysql> SELECT * FROM phonebook ORDER BY name;

+--------  +-----------------       +----------------+

| name  | email                 | phone          |

+--------  +-----------------       +----------------+

| haale |                       | 0935secret   |

| jadi    | jadi@jadi.net  | +9890something |

| nasrin | nasrin@lpic.test | +9898989898 | | sina |
far@from.here | +687randomnum | +--------+--------
----------+----------------+ 4 rows in set (0.00 sec)
```

This order can be done on any field, including numbers:

```
mysql> SELECT * FROM info ORDER BY height;

+--------+--------+--------+-------+

| name  | height | weight | mood |

          +----
+--------  ----    +--------    +-------    +

| mina    |      171 |   59 | sad       |

| nasrin |       174 |   68 | happy |

| sina    |      175 |    81 | happy |

| jadi    |      180 |      74 | happy |

          +----
+--------  ----    +--------    +-------    +

4 rows in set (0.00 sec)
```

GROUP BY

This will ***group*** the output. Unfortunately, this is not very clear.

```
mysql> SELECT * FROM info GROUP BY mood; +---
---+--------+--------+-------+

| name | height | weight | mood |

+ ------+--------   +--------    +-------    +

| jadi |        180 |    74 | happy |

| mina |        171 |      59 | sad  |

+ ------+--------   +--------    +-------    +

2 rows in set (0.00 sec)
```

We've seen that we have only two moods in our table: sad & happy. When SELECTing all fields (that is *) from this table GROUP BY mood, SQL will check all the moods, shows us only ONE from each. This can be used like the **uniq** command you learned from LPIC101:

```
mysql> SELECT mood FROM info GROUP BY mood;

+-------    +

| mood |

+-------    +

| happy |

| sad      |

+-------    +

2 rows in set (0.00 sec)
```

Which gives you all available moods in the table. In real life this is not very useful and most of the times it is combined with count. Have a look:

```
mysql> SELECT count(mood), mood FROM info GROUP BY mood;

+--------------+-------+

| count(mood) | mood |

+  ------------ +-------   +

|           3| happy |

|           1| sad      |

+  ------------ +-------   +

2 rows in set (0.00 sec)
```

Which counts home many rows have that specific mood. So I have 3 happy friends and one sad friend.

Note: count is not part of LPIC 105.3

INSERT

Another clear command. It **adds a new row** to a table. Say I want to add some data to phonebook:

```
mysql> INSERT INTO phonebook (name, phone, email) VALUES ('ghasem',
'+982112345678', '');

Query OK, 1 row affected (0.01 sec)


mysql> SELECT * FROM phonebook;

+--------  +------------------    +----------------  +

| name  | email                | phone          |

+--------  +------------------    +----------------  +

| jadi    | jadi@jadi.net  | +9890something |

| nasrin | nasrin@lpic.test | +9898989898          |

| sina  | far@from.here        | +687randomnum |

| haale |                  | 0935secret       |

| ghasem |                | +982112345678 |

+--------+------------------+----------------+

5 rows in set (0.00 sec)
```

DELETE

This will DELETE from a table. But be careful of what you delete... WHERE is your fiend here:

```
mysql> DELETE FROM phonebook WHERE name = 'ghasem';


mysql> SELECT * FROM phonebook;

+--------  +-----------------     +---------------   +

| name  | email                 | phone          |

+--------  +-----------------     +---------------   +

| jadi    | jadi@jadi.net  | +9890something |

| nasrin | nasrin@lpic.test | +9898989898         |

| sina    | far@from.here  | +687randomnum |

| haale |                       | 0935secret    |

+--------  +-----------------        +----------------+

4 rows in set (0.00 sec)
```

UPDATE

It updates (changes) row and again WHERE is your friend:

```
mysql> SELECT * FROM phonebook;

+--------  +------------------     +----------------   +

| name  | email                  | phone          |

+--------  +------------------     +----------------   +

| jadi    | jadi@jadi.net  | +9890something |

| nasrin | nasrin@lpic.test | +9898989898 | | sina |
far@from.here | +687randomnum | | haale | |
0935secret | +--------+------------------+----------------+ 4
rows in set (0.00 sec)




mysql> UPDATE phonebook SET email='haale@lpic.fake' WHERE name = 'haale'; Query
OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0


mysql> SELECT * FROM phonebook;
                              +--
                              ----
                              ----
                              ----
+--------    +------------------  --                +

| name  | email                  | phone          |

                              +--
                              ----
                              ----
                              ----
+--------    +------------------  --                +

| jadi       | jadi@jadi.net         | +9890something |

| nasrin | nasrin@lpic.test | +9898989898              |

| sina       | far@from.here       | +687randomnum |

| haale | haale@lpic.fake | 0935secret               |
```

```
                                +--
                                ----
                                ----
                                ----
+--------     +------------------  --                  +
```

4 rows in set (0.00 sec)

JOIN

Join will **join/mix two tables**.

```
mysql> SELECT * FROM phonebook JOIN info;
+--------+------------------+---------------+--------+--------+------+---------+
| name   | email            | phone         | name   | height | weight | mood |
+--------+------------------+---------------+--------+--------+------+---------+
| jadi   | jadi@jadi.net    | +9890something | jadi   |    180 |     74 | happy |
| nasrin | nasrin@lpic.test | +9898989898    | jadi   |    180 |     74 | happy |
| sina   | far@from.here    | +687randomnum | jadi   |    180 |     74 | happy |
| haale  | haale@lpic.fake  | 0935secret     | jadi   |    180 |     74 | happy |
| jadi   | jadi@jadi.net    | +9890something | sina   |    175 |     81 | happy |
| nasrin | nasrin@lpic.test | +9898989898    | sina   |    175 |     81 | happy |
| sina   | far@from.here    | +687randomnum | sina   |    175 |     81 | happy |
| haale  | haale@lpic.fake  | 0935secret     | sina   |    175 |     81 | happy |
| jadi   | jadi@jadi.net    | +9890something | nasrin |    174 |     68 | happy |
| nasrin | nasrin@lpic.test | +9898989898    | nasrin |    174 |     68 | happy |
| sina   | far@from.here    | +687randomnum | nasrin |    174 |     68 | happy |
| haale  | haale@lpic.fake  | 0935secret     | nasrin |    174 |     68 | happy |
| jadi   | jadi@jadi.net    | +9890something | mina   |    171 |     59 | sad   |
| nasrin | nasrin@lpic.test | +9898989898    | mina   |    171 |     59 | sad   |
| sina   | far@from.here    | +687randomnum | mina   |    171 |     59 | sad   |
| haale  | haale@lpic.fake  | 0935secret     | mina   |    171 |     59 | sad   |
+--------+------------------+---------------+--------+--------+------+---------+
16 rows in set (0.00 sec)
```

Every single row from first table (phonebook) is copied in front of the second table (info). Not very useful *yet*. It becomes useful when you give a *common field* or tell it to JOIN tables based on a criteria; using WHERE. Here is the magic:

```
mysql> SELECT * FROM phonebook JOIN info ON phonebook.name = info.name;

+--------    +------------------  +----------------+--------+--------         +--------- +-------   +
| name  | email                | phone       | name         | height | weight | mood |
+--------    +------------------  +----------------+--------+--------         +--------- +-------   +

| jadi       | jadi@jadi.net          | +9890something | jadi    |  180 |  74 | happy |
| sina       | far@from.here          | +687randomnum | sina |    175 |       81 | happy |
| nasrin | nasrin@lpic.test | +9898989898             | nasrin |     174 |      68 | happy |
                                +--
                                ----
                                ----
                                ----
+--------    +------------------  --                 +--------- +--------- +--------   +-------    +
3 rows in set (0.00 sec)
```

Great! Now I have my friends list, their moods and their phone numbers! Say I'm bored and I need to phone a cool friend:

```
mysql> SELECT phonebook.name, phone, mood FROM phonebook JOIN info ON
phonebook.name = info.name WHERE mood = 'happy';

+--------  +---------------   +-------  +
| name  | phone              | mood |
+--------  +---------------   +-------  +
| jadi   | +9890something | happy |
| sina   | +687randomnum | happy |
| nasrin | +9898989898       | happy |
+--------  +---------------   +-------  +
3 rows in set (0.00 sec)
```

Note: both tables have a field called name so I needed to use phonebook.name to tell SQL which name I want to show.

```
mysql> SELECT phonebook.name, phone, mood FROM phonebook JOIN info ON
phonebook.name = info.name AND height < 175;

+--------   +-------------+   -------+
| name  | phone              | mood |
+--------   +-------------+   -------+
| nasrin | +9898989898 | happy |
+--------   +-------------+   -------+
1 row in set (0.00 sec)


mysql> SELECT * FROM info WHERE mood = 'sad';

+------+--------+--------+------+

| name | height | weight | mood |

+  -----+-------   +--------   +------  +
| mina |        171 |     59 | sad |
+  -----+-------   +--------   +------  +

1 row in set (0.01 sec)
```

```
mysql> UPDATE info SET mood = 'happy' WHERE name = 'mina';
Query OK, 0 rows affected (0.02 sec) Rows matched: 1 Changed: 0
Warnings: 0



mysql> SELECT * FROM info WHERE mood = 'sad';
Empty set (0.00 sec)



mysql> SELECT * FROM info;

+--------+--------+--------+-------+

| name  | height | weight | mood |

          +----
+--------  ----    +--------    +-------    +

| jadi    |      180 |    74 | happy |

| sina    |      175 |     81 | happy |

| nasrin |       174 |    68 | happy |

| mina    |      171 |    59 | happy |

          +----
+--------  ----    +--------    +-------    +
```

Quit

```
mysql> quit

Bye

jadi@funlife:~$
```

106.1 - Install and configure X11 - *2*

Objectives

- Verify that the video card and monitor are supported by an X server
- Awareness of the X font server
- Basic understanding and knowledge of the X Window configuration file

Terms and Utilities:

- /etc/X11/xorg.conf
- xhost
- DISPLAY
- xwininfo
- xdpyinfo
- X

History

This lesson is useless in modern life! Very strange but practically nothing in this lesson is used in real life because **xorg.conf**, **xhost**, ... is not used in any modern linux system anymore. Maybe they are here so you won't be shocked if you see an older linux.

X

The X Window System is a network transparent window system which runs on a wide range of computing and graphics machines; including practically ALL Linux systems with graphical interfaces. It is also called X11 because of its version, X window system, X server.

/etc/X11/xorg.conf

This is file X used to use for its configuration. In most cases this is automatically generated and works. Newer systems do not have this file so let’s have a look at a xorg.conf I found on the Internet.

```
Section "Files"
        FontPath "/usr/share/X11/fonts/misc"
        FontPath "/usr/share/X11/fonts/100dpi/:unscaled"
        FontPath "/usr/share/X11/fonts/75dpi/:unscaled"
        FontPath "/usr/share/X11/fonts/Type1"
        FontPath "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
EndSection
```

This part is about **Fonts**. When X-Server is running it needs these files. FontPaths tell X11 where fonts are. It also can refer to an IP running a font-server which is not common these days. Font servers used to be responsible of rendering fonts to be shown on clients.

```
Section "Module"
        Load    "bitmap"
        Load    "ddc"
        Load    "dri"
        Load    "extmod"
        Load    "freetype"
        Load    "glx"
        Load    "int10"
        Load    "type1"
        Load    "vbe"
        Load    "dbe"
EndSection
```

These are modules. We are asking X server to load so called modules.

Next we have to define our **InputDevices**:

```
Section "InputDevice"
        Identifier "Generic Keyboard"
        Driver          "kbd"
        Option          "CoreKeyboard"
        Option          "XkbRules"      "xorg"
        Option          "XkbModel"      "pc105"
        Option          "XkbLayout"     "us"
EndSection

Section "InputDevice"
        Identifier "Configured Mouse"
        Driver          "mouse"
        Option          "CorePointer"
        Option          "Device"        "/dev/input/mice"
```

```
        Option          "Protocol"              "ImPS/2"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
EndSection

Section "InputDevice"
    Identifier      "Synaptics Touchpad"
    Driver          "synaptics"
    Option          "SendCoreEvents"        "true"
    Option          "Device"                "/dev/psaux"
    Option          "Protocol"              "auto-dev"
    Option                  "RightEdge"             "5000"
EndSection
```

As you can see each device has an **Identifier**, **Driver** and **some options**. We just defined a mouse, a keyboard and a touchpad and gave them some names. Section "Device"

```
        Identifier "ATI Technologies, Inc. Radeon Mobility 7500 (M7 LW)"
        Driver      "radeon"
        BusID       "PCI:1:0:0"
        Option      "DynamicClocks"  "on"
        Option      "CRT2HSync"      "30-80"
        Option      "CRT2VRefresh"   "59-75"
        Option      "MetaModes"      "1024x768 800x600 640x480
1024x768+1280x1024"
EndSection
```

A graphic card is defined above. Again it has its identifies (name), its drivers and some options (like support resolutions, refresh rates, ...). This device needs a screen and a monitor:

Note: The vesa points to a low resolution, always working driver. It is used for troubleshooting.

```
Section "Monitor"
        Identifier "Generic Monitor"
        Option     "DPMS"
EndSection

Section "Screen"
        Identifier "Screen0"
        Device     "Screen0 ATI Technologies, Inc. Radeon Mobility 7500 (M7 LW)"
        Monitor    "Generic Monitor"
        DefaultDepth    24
        SubSection "Display"
```

```
        Depth           1
        Modes           "1024x768"
    EndSubSection
    SubSection "Display"
        Depth           4
        Modes           "1024x768"
    EndSubSection
    SubSection "Display"
        Depth           8
        Modes           "1024x768"
    EndSubSection
    SubSection "Display"
        Depth           15
        Modes           "1024x768"
    EndSubSection
    SubSection "Display"
        Depth           16
        Modes           "1024x768"
    EndSubSection
    SubSection "Display"
        Depth           24
        Modes           "1024x768"
    EndSubSection

EndSection
```

Note how the screen uses the defined monitor (using its identifier "Generic Monitor") and defined graphic card. Also note the different colour modes (say 24bit 1024x768).

At the end we have to glue all of the above in one place as **ServerLayout**:

```
Section "ServerLayout"
        Identifier "DefaultLayout"
        Screen      "Default Screen"
        InputDevice    "Generic Keyboard"
        InputDevice    "Configured Mouse"
        InputDevice    "Synaptics Touchpad"
EndSection
```

We have a layout with a screen and 3 input devices :)

Note: Do not panic. It is enough for you to understand the Section and EndSection and a general understanding of the xorg.conf

xwininfo

The **xwininfo** command is a window information utility for X. Run it and it waits for you to click on any window and gives you some information about that *window* like its size, position, color depth, ...

```
$ xwininfo

xwininfo: Please select the window about which you
          would like information by clicking the mouse in
          that window.


xwininfo: Window id: 0x5400004 "jadi@funlife: ~/w/lpic/lpic1book"

  Absolute upper-left X: 629
  Absolute upper-left Y: 245
  Relative upper-left X: 10
  Relative upper-left Y: 36
  Width: 655
  Height: 426
  Depth: 32
  Visual: 0x71
  Visual Class: TrueColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x5400003 (not installed)
  Bit Gravity State: NorthWestGravity
```

Window Gravity State: NorthWestGravity

Backing Store State: NotUseful

Save Under State: no

Map State: IsViewable

Override Redirect State: no

Corners: +629+245 -82+245 -82-97 +629-97

-geometry 80x24-72-87

xdpyinfo

This give you information about the running X session. Things like screens, color depth, version, name, ...

```
name of display:    :0
version number:    11.0
vendor string:    The X.Org Foundation
vendor release number:    11701000
X.Org version: 1.17.1
maximum request size: 16777212 bytes
motion buffer size: 256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    7
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 4, bits_per_pixel 8, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
    depth 15, bits_per_pixel 16, scanline_pad 32
    depth 16, bits_per_pixel 16, scanline_pad 32
    depth 24, bits_per_pixel 32, scanline_pad 32
    depth 32, bits_per_pixel 32, scanline_pad 32
keycode range:    minimum 8, maximum 255
focus: window 0x5400005, revert to Parent
number of extensions:    28
```

```
    BIG-REQUESTS

    Composite

    DAMAGE

...

...
```

xhost

This command used to **control the access** to the X server. If you are on a X server and run **xhost** it tells you the access status.

```
$ xhost
```

```
access control enabled, only authorized clients can connect
SI:localuser:jadi
```

As you can see only authorized clients can connect. To open it for all:

```
jadi@funlife:~$ xhost +
```

```
access control disabled, clients can connect from any host
```

And for closing it again:

```
jadi@funlife:~$ xhost -
```

```
access control enabled, only authorized clients can connect
```

Or open it for only one specific IP:

```
jadi@funlife:~$ xhost +192.168.42.85

192.168.42.85 being added to access control list
jadi@funlife:~$ xhost

access control enabled, only authorized clients can connect

INET:192.168.42.85              (no nameserver response within 5 seconds)

SI:localuser:jadi
```

DISPLAY

This variable tell graphical program where to show their graphical output (where to draw their inputs). In normal cases this is set on my own machine:

```
$ echo $DISPLAY

:0
```

But if another X is listening to all IPs (after xhost +) or listening to my machine (after xhost 192.168.42.85) I can change the DISPLAY environment and connect my graphical output to that machine. In this case if I run a graphical program, its output (windows) will be shown on another machine:

```
$ export DISPLAY=192.168.42.85:0

$ xeyes # the eyes will be shown on 192.168.42.85 machine
```

Note: This wont work if you test it on a modern machine. Most X11s do not listen on any port these days.

106.2 - Setup a display manager - *1*

Objectives

- Basic configuration of LightDM
- Turn the display manager on or off
- Change the display manager greeting
- Awareness of XDM, KDM and GDM

Terms and Utilities

- lightdm
- /etc/lightdm/

Display Manager

A **Display Manager** is a graphical interface which lets you login into your system when you turn your computer on. There are many different display managers (say XDM, SDDM, KDM, GDM, ...) but their general functionality is same: show a login form and let the user the enter (or choose) its name, password and the Desktop she needs to use. Also many of the DMs let the user to choose Accessibility Tools (covered in 106.3), connect to the network, change the keyboard layout or change the system volume.

lightdm

Many of the distros use **LightDM** as their display/login manager. It shows the default user (last logged in user) and asks for password. If you have more than one desktop installed (say XFCE, KDE and Gnome) it also lets you choose the one you need.

lightdm can accept *themes* and calls them *greeters*.

/etc/lightdm

All of the lightdm configs are in /etc/lightdm.

```
$ ls -ltrh /etc/lightdm/

total 24K

-rw-r--r-- 1 root root  40 Sep 23 12:56 keys.conf

-rw-r--r-- 1 root root 801 Sep 27 13:03 lightdm-webkit2-greeter.conf -rw-r-
-r-- 1 root root 452 Sep 27 13:08 users.conf -rwxr-xr-x 1 root root 1.5K Sep
27 13:08 Xsession

-rw-r--r-- 1 root root 6.5K Sep 27 13:08 lightdm.conf
```

Some distributions like Ubuntu are using a lightdm.conf.d directory instead of a straight forward lightdm.conf and put their configs there.

```
[SeatDefaults]

...

user-session=gnome

#autologin-user=jadi

#allow-user-switching=true

allow-guest=true

greeter-session=lightdm-webkit2-greeter

...
```

The **greeter-session** tells which greeter (theme) should be used. You can install more greeters using your package manager. Another important config is **user-session** which tells the lightdm what desktop is the default one.

Controlling DMs

The **lightdm** works as a service. You can start, stop & restart it or even use systemctl disable lightdm to disable it on next boots.

You already know how to reboot your computer in text mode from previous lessons (using grub, kernel parameters during the boot or using init command).

106.3 - Accessibility - *1*

Objectives

- Basic knowledge of keyboard accessibility settings (AccessX)
- Basic knowledge of visual settings and themes
- Basic knowledge of assistive technology (ATs)

Terms and Utilities

- Sticky/Repeat Keys
- Slow/Bounce/Toggle Keys
- Mouse Keys
- High Contrast/Large Print Desktop Themes
- Screen Reader
- Braille Display
- Screen Magnifier
- On-Screen Keyboard
- Gestures (used at login, for example GDM)
- Orca
- GOK
- emacspeak

Linux is for everyone

Some people have physical complications. Some cannot see well, some cannot see at all and some cannot user their finger as I can do. Linux have 3 answers:

1- **AccessX** helps people with physical problems to use keyboard/mouse

2- **Visual Settings** help people with vision problems by magnifying the screen and things like that

3- **Assistive Technologies** are things like text-to-speech (tts) and reads the screen for people with visual problem

These options are available in display managers (login screen) and in major desktops (like gnome, kde, xfce, ...). Its logo is a human stretching its hands a legs.

In **Gnome** the config is located at Settings ~ Universal Access. The configurations are categorized and are as follows:

High Contrast
Zoom

Large text

Screen Reader

Screen Keyboard (show a keyboard on screen)
Visual Alerts (instead Beeps, flash the screen)

Sticky Keys (Press shift, then press a -> capital A)

Slow Keys (do not repeat keys after pressing a key for few seconds)

Bounce Keys (if you hit a key twice fast, it won't accept the second one)
Mouse Keys (Arrow keys on number path will work as a mouse)

Simulate Secondary Click (by holding down the click)
Hover click (click by waiting on a button)

TTS

Applications like **Orca** or **Emacspeak** can read the dialog boxes to you so you can decide what the answer only by hearing.

107.1 - Manage user and group accounts and related system files - *5*

Objectives

- Add, modify and remove users and groups.
- Manage user/group info in password/group databases.
- Create and manage special purpose and limited accounts.

Terms and Utilities

- /etc/passwd
- /etc/shadow
- /etc/group
- /etc/skel/
- chage
- getent
- groupadd
- groupdel
- groupmod
- passwd
- useradd
- userdel
- usermod

Changing password

Each user can change her password using the **passwd** command:

$ **passwd**

Changing password for jadi.

(current) UNIX password:

New password:

Retype new password:

passwd: password updated successfully

If the password is too short or too similar to the previous one or even a dictionary word, the passwd command may refuse to change it. Also note that the commands asks for the *current password* first to make sure that someone is not using your computer to change your password.

The **root** user can change **any users'** password to anything (weak passwords) without providing their current password:

```
# passwd jadi
New password:
```

BAD PASSWORD: it does not contain enough DIFFERENT characters

BAD PASSWORD: is too simple
Retype new password:

passwd: password updated successfully
Users and groups

Linux is a multi-user system so you should be able to manage these users. You should be able to **add**, **remove** and **modify** users.

Linux also has the concept of **groups**. You can define groups, give privileges to them and make users members of these groups. For example there can be a "printer" group who has access to printings and you can add user "jadi" to this group.

Each user can be a member of many different groups
Each file belongs to one user and one group

Managing Users

Adding users

Adding a user is done using the **useradd** command. These are the main switches:

switch	**meaning**
-d	home directory (-d /home/user)
-m	create home directory
-s	specify shell
-G	add to additional groups
-c	comment. most of the time, users actual name. Use quotes if comments has spaces or special characters in them

On some systems **useradd** creates the home directory and on some, you have to specify the -m switch yourself. It is good to use it all the time.

When a new user directory is being created, the system will copy the contents of **/etc/skel** to their home dir. **/etc/skel** is used as a template for the home of users.

Modifying users

It supports most of the **useradd** switches. For example you can change *jadi*'s login shell by issuing **usermod -s /bin/csh jadi**. But there are 3 more switches:

switch	**meaning**
-L	lock this account
-U	Unlock the account
-aG	add to more groups (say usermod -aG wheel jadi)

Note: If you do usermod -G wheel,users jadi, jadi will be ONLY the member of these two groups. That is why we use -aG newgoup to ADD a new group to what jadi is a member of. - G is like saying "jadis groups are ..." and -aG is like "add this group to whatever groups jadi is a member of".

Deleting users

If you want to remove a user, use **userdel** as easy as:

userdel jadi

If you add the **-r** switch, the home directory and mail spool will be erased too!

Managing Groups

It is kind of same as users, you can do **groupadd**, **groupdel** and **groupmod**. Each group as an **id** and a **name**.

```
#  groupadd -g 1200 newgroup
```

Adds a group called *newgroup* with id 1200. If needed, the root user can change a groups ID (to 2000) by issuing groupmod -g 2000 newgroup or deleting the group by groupdel newgroup.

Note: If root deletes a group with members, people wont be deleted! They will just wont be the members of that group anymore.

Important files

/etc/passwd

This is the file which contains all the user names and their shells, etc, ..

tail /etc/passwd

scard:x:491:489:Smart Card Reader:/var/run/pcscd:/usr/sbin/nologin
sshd:x:493:491:SSH daemon:/var/lib/sshd:/bin/false statd:x:488:65534:NFS
statd daemon:/var/lib/nfs:/sbin/nologin tftp:x:496:493:TFTP
account:/srv/tftpboot:/bin/false lightdm:x:10:14:Light Display
Manager:/var/lib/lightdm:/bin/false wwwrun:x:30:8:WWW daemon
apache:/var/lib/wwwrun:/bin/false
jadi:x:1000:100:jadi:/home/jadi:/bin/bash

svn:x:485:482:user for Apache Subversion svnserve:/srv/svn:/sbin/nologin
privoxy:x:484:480:Daemon user for privoxy:/var/lib/privoxy:/bin/false

As you can see the format is:

username:password:userid:primary group id:Name and comments:home dir:shell

In old days the password or the hashed password was actually shown in this file but nowadays that is moved to the **/etc/shadow** file.

Note: **/etc/passwd** should be readable to all users so it is not a good place for password! These days if there is a x instead of password, it means *go look at the /etc/shadow* file.

Note how *special users* like lightdm are having /bin/false as their shell; this prevents them from logging into the system for real.

/etc/shadow

This file contains password (hashed passwords) of the users.

See how the **/etc/passwd** is readable for all but **/etc/shadow** is only readable for root and members of the shadow group:

```
# ls -ltrh /etc/passwd /etc/shadow
-rw-r--r-- 1 root root  1.9K Oct 28 15:47 /etc/passwd
-rw-r----- 1 root shadow 851 Oct 29 19:06 /etc/shadow
```

But what is in it?

```
# tail /etc/shadow
scard:!:16369:::::::
sshd:!:16369:::::::
statd:!:16369:::::::
tftp:!:16369:::::::
uucp:*:16369:::::::
lightdm:*:16369:::::::
jadi:$6$enk5I3bv$uSQrRpen7m9xDapYLgwgh3P/71OLZUgj31n8AwzgIM2lA5Hc/BmRVAMC0eswdBGkseuXSvmaz0lsYFtduvuqUo:16737:0:99999:7:::
svn:!:16736::::::
privoxy:!:16736::::::
```

Note: ! means **no password**

Wow! Jadi has an encrypted password there. Some numbers are following that encrypted password too: **16737:0:99999:7:::**. What do the mean? The following table tells you.

filed	meaning
16737	When was the last time this password changes
0	User wont be able to change the password 0 days after each change
99999	After this many days, the user HAVE to change his password
7	...and the user will be informed 7 days before the expiration to change his password

Note: there numbers are "days after 1st of January 1970" or the Epoch time in days. For example 16737 means 16373 days after 1st Jan 1970. Strange but practical!

But we do not need to change these strange number manually. If needed, we can use the **chage** tool to change these numbers. If you issue the **chage** jadi the system will prompt you for all the parameters one by one. Also it is possible to use switches to change specific parameters on command line.

switch	meaning
-l	list information
-E	Set the expiration date. Date can be a number, in YYYY-MM-DD format or -1 which will mean *never*

 # chage -l jadi

Last password change	: Oct 29, 2015
Password expires	: never
Password inactive	: never
Account expires	: never
Minimum number of days between password change	: 0
Maximum number of days between password change	: 99999
Number of days of warning before password expires	: 7

/etc/group

This file contains the groups and their IDs.

 # tail /etc/group

avahi:x:486:

kdm:!:485:

mysql:x:484:

winbind:x:483:

at:x:25:

svn:x:482:

vboxusers:x:481:

input:x:1000:jadi

privoxy:x:480:

Note: See that x there? Theoretically groups can have passwords but it is never used in any distro! The file is **/etc/gshadow**

Checking user info

Previously you saw the **chage -l jadi** but there are more commands for checking user status.

One is id:

```
    #    id jadi
uid=1000(jadi) gid=100(users) groups=1000(input),100(users)
```

Another solution is **getent** (for **get entry**). It can query important *databases* for specific

entries. These databases include /etc/passwd, /etc/hosts, /etc/shadow, /etc/group, ...

funlife:~ # **getent group tor**

tor:x:479:

funlife:~ # **getent passwd jadi**

jadi:x:1000:100:jadi:/home/jadi:/bin/bash

funlife:~ # **getent shadow jadi**

jadi:6enk5I3bv$uSQrRpen7m9xDapYLgwgh3P/71OLZUgj31n8AwzgIM2lA5Hc/BmRVAMC0
eswdBGkseuXSvmaz0lsYFtduvuqUo:16737:0:99999:7:::

107.2 - Automate system administration tasks by scheduling jobs - *4*

Objectives

- Manage cron and at jobs.
- Configure user access to cron and at services.
- Configure anacron.

Terms and Utilities

- /etc/cron.{d,daily,hourly,monthly,weekly}/
- /etc/at.deny
- /etc/at.allow
- /etc/crontab
- /etc/cron.allow
- /etc/cron.deny
- /var/spool/cron/
- crontab
- at
- atq
- atrm
- anacron
- /etc/anacrontab

Crontab format

Crontab files are responsible to run some commands on **specific intervals**. Each line has 5 fileds to specify the run time and whatever after it is considered the command to be run.

A B C D E command and arguments

filed	Meaning	values
A	minute	0-59
B	hour	0-23
C	day of month	1-31
D	month	1-12 (or names, see below)
E	day of week	0-7 (0 or 7 is Sunday, or use names)

Each time field can be a ***** to indicate *ANY*. Also if you have **@reboot** or **@daily** instead of time fields, the command will be run once after the reboot or daily. Let's see some examples:

```
5 0 * * *       $HOME/bin/daily.job >> $HOME/tmp/out 2>&1

    #    run at 2:15pm on the first of every month -- output mailed to paul
         15 14 1 * * $HOME/bin/monthly

    #    run at 10 pm on weekdays, annoy Joe

0 22 * * 1-5     mail -s "It's 10pm" joe%Joe,%%Where are your kids?%

23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ..., everyday"

5 4 * * sun      echo "run at 5 after 4 every sunday"
```

*/5****	echo "each 5 mintues"
42 8,18 * * 1-5	echo "8:42 and 18:42 and only on weekdays (monday till friday)"
@reboot	echo "runs after the reboot"

Note: be careful about using a * on the first filed. That will run your cron on every minute!

Note: Use with care. Something like 42 8 1 1 0 runs ONLY IF 1st Of Jan is a Monday! When a cron runs, the output will be emailed to the owner of the cron.

User specific crons

Cron is a linux service. To see your crons you can use **crontab -l** (list) and for editing it you can use **crontab -e** (edit) which will open the cron files with a special editor and will load your inserted crons (if they are correct) after wards.

The files will be saved at **/var/spool/cron/tabs/** or `**/var/spool/crontabs**':

```
    #    cat /var/spool/cron/tabs/jadi
    #    DO NOT EDIT THIS FILE - edit the master and reinstall.
    #    (/tmp/crontab.khObLu installed on Thu Oct 29 22:04:43 2015)
    #    (Cronie version 4.2)
* * * * * date >> /tmp/date.cron.txt
```

You should never edit these files directly; Use crontab -e instead.

At

We say that **crontab** runs commands on specific intervals but what will happen if you needed to run a command only once? Here at is your friend. $ **at now + 1 min**

```
warning: commands will be executed using /bin/sh
at> touch /tmp/at
at> <EOT>
job 3 at Thu Oct 29 22:12:00 2015
```

Note: As always, at the end of input we press Ctrl+D

If you want to check what is in the queue you can use **atq** and then try **atrm** 4 to delete job number 4;

```
$ at teatime
```

```
warning: commands will be executed using /bin/sh at>
echo "tea time is 4pm" at> <EOT>



job 4 at Fri Oct 30 16:00:00 2015

jadi@funlife:~$ at tomorrow

warning: commands will be executed using /bin/sh at>
echo "tomorrow this time" at> <EOT>


job 5 at Fri Oct 30 22:15:00 2015

jadi@funlife:~$ at 17:30

warning: commands will be executed using /bin/sh at>
echo "this is the first 17:30" at> <EOT>


job 6 at Fri Oct 30 17:30:00 2015
```

```
jadi@funlife:~$ atq

5       Fri Oct 30 22:15:00 2015 a jadi

4       Fri Oct 30 16:00:00 2015 a jadi


  6   Fri Oct 30 17:30:00 2015 a jadi
      jadi@funlife:~$ atrm 4
      jadi@funlife:~$ atq

  5   Fri Oct 30 22:15:00 2015 a jadi

  6   Fri Oct 30 17:30:00 2015 a jadi
```

System wide cron

There is file called **/etc/crontab**. This looks like a normal user file opened with **crontab -e** but has one extra filed:

```
A       B       C       D       E       USER    command and arguments
```

This file should be edited with an editor directly and we can mention which user runs these commands.

```
  #   cat /etc/crontab SHELL=/bin/sh
      PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/
      bin MAILTO=root


  #
  #   check scripts in cron.hourly, cron.daily, cron.weekly, and cron.monthly

  #
```

-*/15 * * * * root test -x /usr/lib/cron/run-crons && /usr/lib/cron/run-crons >/dev/null 2>&1

Note: Have a look at first two line. It configures the shell which will run the commands and the PATH variable plus who will get the output emails.

As you can see this default crontab runs other crons! They are hourly, daily, weekly and monthly crons.

System hourly, daliy, weekly, monthly, .. crons

We have some system level crontab files in **/etc/cron.d/** too. In other words, whatever file which is copied there, will be treated just like /etc/crontab file (a system wide cron file). This make systems much cleaner and lets programs to copy one file there instead of editing the /etc/crontab.

```
$ sudo tree /etc/cron*
root's password:
/etc/cron.d
└── mdadm
/etc/cron.daily
├── google-chrome
├── mdadm
├── mlocate.cron
├── packagekit-background.cron
├── suse-clean_catman
├── suse.de-backup-rc.config
├── suse.de-backup-rpmdb
├── suse.de-check-battery
├── suse.de-cron-local
├── suse.de-snapper
└── suse-do_mandb
/etc/cron.deny [error opening dir]
/etc/cron.hourly
└── suse.de-snapper
```

```
/etc/cron.monthly

/etc/crontab [error opening dir]

/etc/cron.weekly
```

Let’s have a look at one of the **cron.d** files:

```
$ cat /etc/cron.d/mdadm

#

    #    cron.d/mdadm - regular redundancy checks

    #

    #    Start checking each month early in the morning.

    #    Continue each day until all done

PATH=/sbin:/usr/sbin:/bin:/usr/bin

0 1 * * 0 root source /etc/sysconfig/mdadm; [ -n "$MDADM_CHECK_DURATION" -a -x
/usr/share/mdadm/mdcheck -a $(date +\%d) -le 7 ] && /usr/share/mdadm/mdcheck --
duration "$MDADM_CHECK_DURATION"

    0    1 * * 1-6 root source /etc/sysconfig/mdadm; [ -n "$MDADM_CHECK_DURATION" -a -x
         /usr/share/mdadm/mdcheck ] && /usr/share/mdadm/mdcheck --continue --duration
         "$MDADM_CHECK_DURATION"
```

But /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly is **TOTALLY DIFFERENT**. In these directories are actual executable scripts and files. The cron will run these files one a hour, one a day, once a week or once a month based on their directory names.

anacron

The difference between **cron** and **anacron**, is this:

If the system is down when the cron should run a task, that cron job won’t run till the next occurrence! But anacron creates the timestamp each time a **daily**, **weekly** or **monthly** job runs. If the system **boots up** and find outs that one of the anacron jobs are missed, it will run it during the boot!

As you can see **anacron** is useful for important tasks. If you need to take a backup once a week it is better to use anacron instead of cron; or feeding your dog once a day using cron may lead to it staying hungry for a day if the system is down when he should be fed.

Note: anacron checks the timestamps at BOOT TIME and do not handle hourly crons.

Controlling access using files

You have already seen files at **/var/spool/cron/tabs/USERNAME**. There are also 4 more files to control who can and cannot use cron and at. The files are:

```
/etc/cron.allow
/etc/cron.deny

/etc/at.allow
/etc/at.deny
```

In most systems none of these files exist but if you create them, they will become active as follow:

file extension	functionality
.allow	ONLY users mentioned in this file are allowed to run this service. All other users will be denied
.deny	Everybody can use the service except the users mentioned in this file

107.3 Localization and internationalization - *3*

Objectives

- Configure locale settings and environment variables.
- Configure timezone settings and environment variables.

Terms and Utilities

- /etc/timezone
- /etc/localtime
- /usr/share/zoneinfo/
- LC_*
- LC_ALL
- LANG
- TZ
- /usr/bin/locale
- tzselect
- timedatectl
- date
- iconv
- UTF-8
- ISO-8859
- ASCII
- Unicode

Time zone

On linux systems you can use **date** and **cal** commands to check the **date** and the **calendar**.

It is possible to print a custom date using + formatter:

```
[jadi@funlife ~]$ date +'%Y%m%d-%M'

20160103-39

[jadi@funlife ~]$ date +'%Y%m%d-%H%M'

20160103-2239
```

Time zones determines what your time difference is comparing with a reference time zone. This way you can talk about times regardless from your location. In another words, I can tell you "start the change request at 02:30 UTC" we both know when the change will be started in our own time zone (mine is 02:30 minus 3:30).

You can configure your time zone while installing the system or by using a GUI in the system settings. It is even possible to set the time zone by clicking or right-clicking on the date and time on your panel. But as always there is a command line way. The old one used to be tzconfig but it is not used anymore.

Different distros do have their own configuration commands, a general one is:

```
tzselect
```

Please identify a location so that time zone rules can be set correctly.

Please select a continent, ocean, "coord", or "TZ".

1) Africa
2) Americas
3) Antarctica
4) Arctic Ocean
5) Asia
6) Atlantic Ocean
7) Australia
8) Europe
9) Indian Ocean 10) Pacific Ocean

11) coord - I want to use geographical coordinates.

12) TZ - I want to specify the time zone using the Posix TZ format.

This process will suggest you to set a variable called **TZ** as follow to set *your own* time zone, but not the systems:

```
TZ='Asia/Tehran'; export TZ
```

Configuring time zone

There is a directory at **/usr/share/zoneinfo/** containing all the time zone info. These are binary files. If you need to change your systems time zone you need to change 2 files: **cat /etc/timezone**

```
Asia/Tehran
```

And there is a short link at this place:

```
# ls -ltrh /etc/localtime
-rw-r--r-- 1 root root 1.7K Jan 2 18:10 /etc/localtime
```

This file should be replaced by the correct file from /usr/share/zoneinfo/. It is nicer to make a symbolic link rather than copying the actual file. This will prevent the conflicts during next upgrades.

Configuring Languages

You can check the status of current selected system language by issuing **locale**:

```
$locale
LANG=en_US.UTF-8
LANGUAGE=
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
```

These are all environment variables telling system what languages to use. Here I'm using LANG=en_US.UTF-8 which means I'm using English with US variant and UTF-8 encoding.

UTF-8 and other encodings will be discussed a bit later in this chapter

Other variables tell the system how to show different things based on localization systems. For example, if we change the LC_TIME to "en_GB.UTF-8" the time will be printed in Great Britain format from that moment on.

Another important settings is LC_ALL. It can be used to change **ALL** settings. If you do a export LC_ALL=fa_IR.UTF-8, all the settings will be set to that one, with no exception. It is always possible to unset LC_ALL.

LANG=C

Another important point to know is the LANG=C settings. This indicates two things:

1. All language settings will be default (en.US)
2. Binary sort order

It is also possible to do a LC_ALL=C.

Changing or adding locales

This is **not a part of LPIC exam** but it is good to know that on a debian based machine, you can change, add or set your default *locales* using dpkg-reconfigure locales.

Character Encoding

ACSII

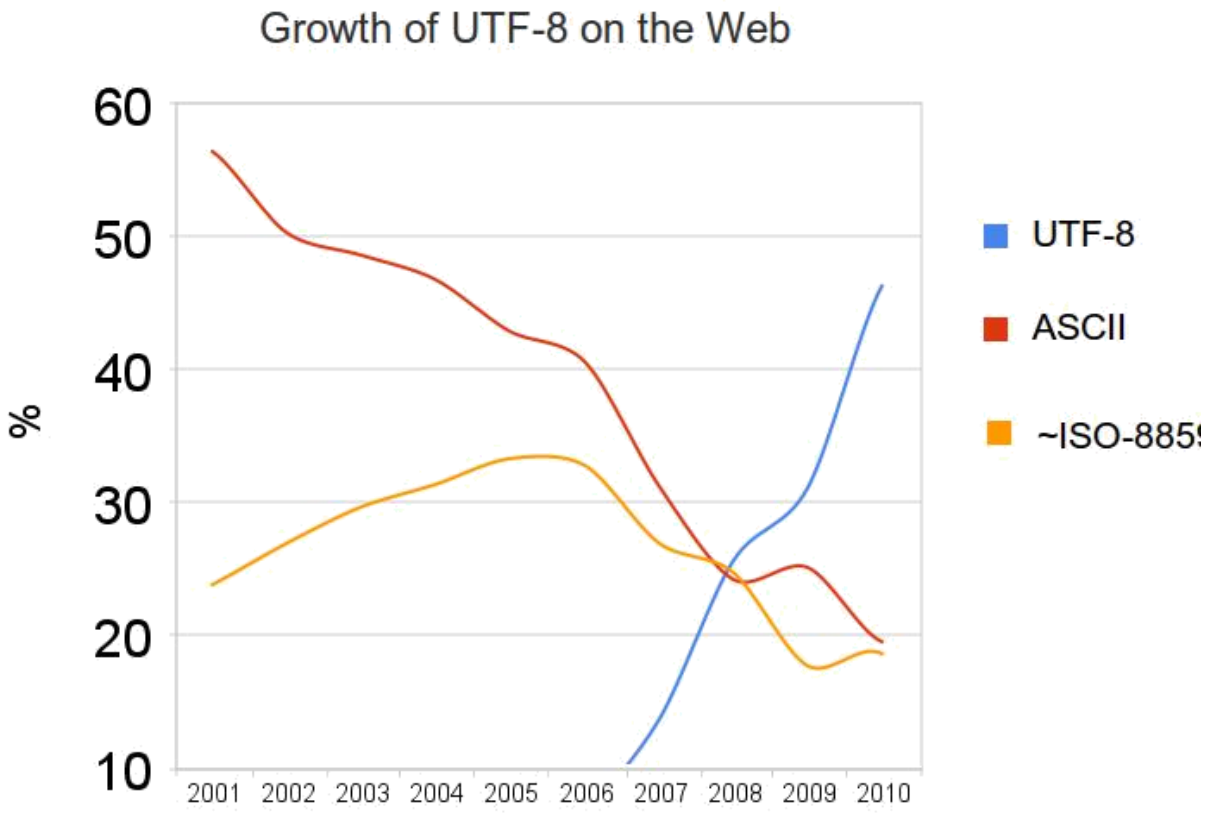
Computers used to work with 7bit characters encoding. That would give us 128 characters which was enough for numbers, punctuation and digits!

ISO-8859

It had more characters and a lots of sets for Thai, Arabic and other languages but still had ASCII character sets.

UTF-8

The Unicode Transformation Format is the newest encoding method. It is a real universal encoding with characters not only for all written languages but also for fun characters like ¾, ♠, π and ⚥. It is backward compatible with the ASCII and uses 8 bit code units (**not 8 bit coding!**). In most cases it is a good idea to use UTF-8 and be sure that your system will work in practically all cases.


Growth of UTF-8 on the Web
60
50
40
30
20
10
%
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
UTF-8
ASCII
~ISO-885

Above table shows how UTF8 is the leader compared with ASCII and ISO-8859.

iconv

If you needed to convert coding to each other, the command is **iconv**. The -l switch will show you all the available codings:

```
iconv -f WINDOWS-1258 -t UTF-8 /tmp/myfile.txt
```

Note: -f is for "from" and -t is for "to". Easy to remember

108.1 - Maintain system time - *3*

Objectives

- Set the system date and time.
- Set the hardware clock to the correct time in UTC.
- Configure the correct timezone.
- Basic NTP configuration.
- Knowledge of using the pool.ntp.org service.
- Awareness of the ntpq command.

Terms and Utilities

- /usr/share/zoneinfo/
- /etc/timezone
- /etc/localtime
- /etc/ntp.conf
- date
- hwclock
- ntpd
- ntpdate
- pool.ntp.org

How a computer keeps its time

There is a clock in your computer; a hardware clock on your motherboard! It has its own battery and keeps the time even when the computer is off. When the system boots, the OS reads this **hardware time** and it sets its **system time** to the hardware clock and uses it from there on.

Hardware clock can be the **localtime** (your computers timezone) or **UTC time** (standard time).
You can check this **by /etc/adjtime**:

```
$ cat /etc/adjtime

0.000000 1451741899 0.000000

1451741899

UTC
```

As you can see in my computer the time is set on UTC so the computers add my timezone difference each time it boots up to the hardware clock. The hwclock can be used to show the time based on the hwtime. See how it works based on the hardware time even after we DO CHANGE the system time:

```
root@funlife:~# date

Mon Jan 4 22:01:18 IRST 2016

    #   date -s "Jan 4 22:22:22
        2016" Mon Jan 4 22:22:22
        IRST                2016
        root@funlife:~# date

Mon Jan 4 22:02:18 IRST 2016
root@funlife:~# hwclock

Mon 04 Jan 2016 10:02:21 PM IRST .108596 seconds
```

Even when the hardware clock is set on UTC, hwclock date shows the date in the localtime (time after adding the timezone to the UTC time)

Older OSs used to set the hardware clock on localtime zone instead of timezone. This can be achived by:

```
    #   hwclock --localtime --set --date="01/05/2015
        22:04:00" If you want to fix it, just issue:

    #   hwclock -u -w
```

In this command -u tell the hardware clock that this is a UTC time and -w tells "sync with systemtime".

NTP

Network Time Protocol is my favourite protocol. It is one of the coolest protocols ever if you dive into its details. But unfortunately for LPIC1 you do not need to go into NTP depths. This protocol uses **NTP servers** to find out the accurate time shown by best atomic clocks on this planet. One of the most famous severs used by ntp people is **pool.ntp.org**. If you check that website you will see that it is a **pool** of ntp servers and by giving your NTP server the pool.ntp.org, it will be redirected to one of the many ntp servers available on that pool.

ntpdate

The most straight forward command to set the **systemclock** is **ntpdate** and used like this:

```
# ntpdate pool.ntp.org
4 Jan 22:15:02 ntpdate[18708]: adjust time server 194.225.150.25 offset -0.006527 sec
```

Then, we need to set the **hwclock** to the just corrected system time by **sudo hwclock -w**.

ntpd

Instead of manually setting the time each time, you can use a linux service called **ntp** to keep your time using some time servers (the most famous one is pool.ntp.org). Install the ntp and start the server:

```
# apt install ntp
# systemctl start ntp
```

Fun fact? You cannot use both! Look at this:

```
root@funlife:~# ntpdate pool.ntp.org
 4 Jan 22:14:25 ntpdate[18670]: the NTP socket is in use, exiting
```

As you can see, now the **ntp** is using the NTP port and **ntpdate** has problems starting up.

Main configuration file of ntp is located at **/etc/ntp.conf**:

```
# cat /etc/ntp.conf

# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help


driftfile /var/lib/ntp/ntp.drift



# Enable this if you want statistics to be
logged. #statsdir /var/log/ntpstats/


statistics loopstats peerstats clockstats filegen
loopstats file loopstats type day enable filegen
peerstats file peerstats type day enable filegen
clockstats file clockstats type day enable
```

```
    #   You do need to talk to an NTP server or two (or
        three). #server ntp.your-provider.example

    #   pool.ntp.org maps to about 1000 low-stratum NTP servers. Your server will
    #   pick a different set every time it starts up. Please consider joining the
    #   pool: <http://www.pool.ntp.org/join.html>
pool 0.debian.pool.ntp.org iburst

pool 1.debian.pool.ntp.org iburst

pool 2.debian.pool.ntp.org iburst

pool 3.debian.pool.ntp.org iburst


    #   Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html for
    #   details. The web page <http://support.ntp.org/bin/view/Support/AccessRestrictions>
    #   might also be helpful.
#
    #   Note that "restrict" applies to both servers and clients, so a configuration
    #   that might be intended to block requests from certain clients could also end
    #   up blocking replies from your own upstream servers.

    #   By default, exchange time with everybody, but don't allow configuration.
        restrict -4 default kod notrap nomodify nopeer noquery limited
restrict -6 default kod notrap nomodify nopeer noquery limited

    #   Local users may interrogate the ntp server more closely.
restrict 127.0.0.1

restrict ::1

    #   Needed for adding pool entries
        restrict source notrap nomodify
        noquery

    #   Clients from this (example!) subnet have unlimited access, but only if
```

```
    #    cryptographically authenticated.

#restrict 192.168.123.0 mask 255.255.255.0 notrust


    #    If you want to provide time to your local subnet, change the next line.

    #    (Again, the address is an example only.)

#broadcast 192.168.123.255


    #    If you want to listen to time broadcasts on your local subnet, de-comment the

    #    next lines. Please do this only if you trust everybody on the network!
         #disable auth

#broadcastclient
```

If needed, you can change the ntp servers to the ntp servers you want to use.

Review the configuration and you will see cool things like giving the ntp service to other computers although you do not need it for passing LPIC.

ntpq

The **ntpq** queries the ntp service. One famous switch is **-p** (for Print) that shows the pool we are using to sync the clock:

```
ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 0.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
 1.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
 2.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
 3.debian.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.000
+46.209.14.1     192.168.5.2      4 u    7   64    1   58.300  -15.546  14.519
-ntp.tums.ac.ir 195.161.115.4     4 u    4  64    1   30.636   2.485   4.025
*194.225.150.25 194.190.168.1     2 u    5  64    1   31.478  -3.870  95.635
+5.160.24.41     192.168.5.2      4 u    3   64    1   90.000  -28.328  21.643
```

In this output:

* means that the ntp is using this server as the main reference, **+** means that this is a good server,
- shows an out of range server which will be neglected.

108.2 - System logging - *4*

Objectives

- Configuration of the syslog daemon.
- Understanding of standard facilities, priorities and actions.
- Configuration of logrotate.
- Awareness of rsyslog and syslog-ng.

Terms and Utilities

- syslog.conf
- syslogd
- klogd
- /var/log/
- logger
- logrotate
- /etc/logrotate.conf
- /etc/logrotate.d/
- journalctl
- /etc/systemd/journald.conf
- /var/log/journal/

History

The Linux logging system is under heavy changes. We will cover the **syslog** but most system have replaced it with **rsyslog** and **systemd journal**s. The strange thing is the fact that **systemd journal** uses a binary file format which is not that common in Linux world.

The logging in linux is orginized based on three concepts: **facilities**, **priorities** and **actions**. When a program generated a log, it tags or labels that log with a facility (like mail) which says what this log is and a priority (like alert). Each tag can have values like the following list:

Facilities: auth, user, kern, cron, daemon, mail, user, local1, local2, ...

Priorities: emerg/panic, alert, crit, err/error, warn/warning, notice, info, debug

As you can guess, the **facilities** work like categories and priorities indicate how important this log is - or in more technical language indicated logs level.

On the **action** part we can have things like these:

action	sample	meaning
filename	/usr/log/logins.log	will write the log to this file
username	jadi	will notify that person on the screen
@ip	@192.168.1.100	will send this log to this log server and that log server will decide what its configs

So a line like this will show the kernel panics to a remote log server and also will log everything on every level to a log file:

```
kern.panic      @192.168.1.100

*.*             /var/log/messages
```

If you log some specific priority, all the **more important** things will be logged too! So if you write **cron.notice /var/log/cron/log**, you are logging the emerg/panic, alert, critical, error, warning and notice logs of the cron category too.

If you need to log ONLY one specific level, add an equal sign (=) before the priority like this local3.=alert /var/log/user.alert.log.

It is important to know that the binary which logs the **kern* category is a standalone daemon. This daemon is called klogd and uses same configuration files. Why? so even after everything is crashed, klogd can log the kernel crashes ;)

Syslog and Rsyslog

Most modern system use **rsyslog** instead of **syslog**. Their functionality is mostly same and here we will only cover the rsyslog.

The main configuration file in **rsyslog**, as you should be able to guess is **/etc/syslog.conf**. It loads some modules on the top and then have lines like this:

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
```

'auth,authpriv.*' means both auth and authpriv properties

Note that sometimes on the action we have a -. This means the log will go the memory cache to prevent disk from spinning all the time.

Again there is a /etc/rsyslog.d/ and it is better for different software and admins to add their specific configs there, instead of editing the main configuration file.

Creating rsyslog listener

If you need to start a rsylog listener and catch other systems log messages, it is enough to add an -r switch to rsyslog options. Just edit the /etc/default/rsyslog and change options from "" to "-r".

cat /etc/default/rsyslog

```
    #   Options for rsyslogd
    #   -x disables DNS lookups for remote messages
    #   See rsyslogd(8) for more details
RSYSLOGD_OPTIONS="-r"
```

And restart the daemon:

```
    #   systemctl restart rsyslog
```

Journalctl

The newer distributions are switching to **systemd** and are using **systemd journal** for their logging. As I mentioned earlier the systemd keeps its logs as binary files and the user should use the **journalctl** to access them.

```
# journalctl
-- Logs begin at Sun 2016-01-03 10:35:53 IRST, end at Tue 2016-01-05 22:34:06 IRST. -- Jan 03 10:35:53 funlife systemd-journald[184]: Runtime journal (/run/log/journal/) is currently using 8.0M.
                                Maximum allowed usage is set to 238.1M.
                                Leaving at least 357.2M free (of currently available 2.3G of space).
                                Enforced usage limit is thus 238.1M, of which 230.1M are still
available.
Jan 03 10:35:53 funlife kernel: Initializing cgroup subsys cpuset
Jan 03 10:35:53 funlife kernel: Initializing cgroup subsys cpu
Jan 03 10:35:53 funlife kernel: Initializing cgroup subsys cpuacct
Jan 03 10:35:53 funlife kernel: Linux version 4.3.0-1-amd64 (debian-kernel@lists.debian.org) (gcc version 5.3.1 20151207 (Debian 5.3.1-3) ) #1 SMP Debian 4.3.3-2 (2015-
Jan 03 10:35:53 funlife kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.3.0-1-amd64 root=UUID=e6b1e8e9-dee3-45cd-b147-44801cc680f0 ro quiet Jan 03 10:35:53 funlife kernel: Disabled fast string operations
```

At the moment, most new systems use systemd and journalctl but also have rsyslog installed and are logging information in both systems.

The config file of journalctl is located at /etc/systemd/journald.conf.

Logger

It is possible to use the logger command to generate some logs:

```
logger local1.info jadi was here
```

And this will appear at /var/log/syslog.

Logrotate

Now we are generating a lot of logs. What should we do with them? How they should be archived? The **logrotate** utility assists us in this area. Its main config file is /etc/logrotate.conf and as any modern program, other config files can go into /etc/logrotate.d/.

```
    #    cat /etc/logrotate.conf
    #    see "man logrotate" for details
    #    rotate log files weekly
weekly


    #    keep 4 weeks worth of
         backlogs rotate 4


    #    create new (empty) log files after rotating old
         ones create


    #    uncomment this if you want your log files
         compressed #compress


    #    packages drop log rotation information into this
         directory include /etc/logrotate.d


    #    no packages own wtmp, or btmp -- we'll rotate them
         here /var/log/wtmp {
   missingok
   monthly

   create 0664 root utmp
   rotate 1

}
```

```
/var/log/btmp {

   missingok

   monthly
```

```
    create 0660 root utmp

    rotate 1

}


# system-specific logs may be configured here
```

It is very easy to see how it works. Let’s check one of them:

```
# cat /etc/logrotate.d/nginx
/var/log/nginx/*.log {
        weekly
        missingok
        rotate 52
        compress
        delaycompress
        notifempty

        create 0640 www-data adm
        sharedscripts
        prerotate
                if [ -d /etc/logrotate.d/httpd-prerotate ]; then \ run-parts
                        /etc/logrotate.d/httpd-prerotate; \
                fi \
        endscript
        postrotate
                invoke-rc.d nginx rotate >/dev/null 2>&1
        endscript
}
```

These are the meaning of some of these parameters:

parameter	meaning
weekly	rotate logs weekly
missingok	it is fine if there is no log for this week
rotate 52	keep the latest 52 logs and delete the older ones
compress	compress the logs
create 0640 www-data adm	create the files with this access and owners
pre & post rotate	run these scripts or commands before and after the rotation

This configuration will create a zipped file for each week, keeping only 52 of them instead of a huge log file for this program.

108.3 Mail Transfer Agent (MTA) basics - *3*

Objectives

- Create e-mail aliases.
- Configure e-mail forwarding.
- Knowledge of commonly available MTA programs (postfix, sendmail, qmail, exim) (no configuration)

Terms and Utilities

- ~/.forward
- sendmail emulation layer commands
- newaliases
- mail
- mailq
- postfix
- sendmail
- exim
- qmail

MTAs

Mai Transfer Agents or MTAs are programs which **handle emails** in your operating system.

There are lot of MTAs available and each distro or sysadmin uses the one she likes more.

Sendmail

One of the oldest options available. It is big and difficult to configure and keep safe and secure so very few systems use it as default MTA.

Qmail

qmail is an attempt to provide an ultra-secure MTA while keeping the MTA compatible with sendmail ideas. It is modular and claims to be free of any security bug. It also claim to be the 2nd popular mail agent on the Internet.

qmail is not a GPL software. It is Public Domain.

Exim

It aims to be a general and flexible mailer with extensive facilities for checking incoming e-mail. It is feature rich with ACLs, authentication, ...

Postfix

This is a new alternative to sendmail and uses easy to understand configuration files. It supports multiple domains, encryption, etc. Postfix is what you may find on most distros as default.

Sendmail emulation layer

As I already said, sendmail is the oldest MTA which is still active. Other MTAs respect his age and provide a *sendmail emulation layer* to keep themselves backward compatible with it. In other words you can type sendmail or mailq on your command line regardless of what MTA you've installed.

Aliases

There are some mail aliases on the system. Defined in /etc/aliases.

```
$ cat /etc/aliases

# /etc/aliases

mailer-daemon: postmaster

postmaster: root

nobody: root

hostmaster: root

usenet: root      # <--- I'm using this sample

news: root

webmaster: root

      www:          root

ftp: root

abuse: root

noc: root
```

```
security: root

root: jadi
```

This tells the system if there is a message for 'usenet' it will sent to the root user. Note that in the last line, jadi is reading the root emails. This line lets me read emails sent to root without needing to login with root.

when this file is update, the newaliases should be run!

```
root@funlife:~# newaliases

root@funlife:~#
```

Sending mail

It is possible to send an email from the command line using the mail command:

```
[jadi@funlife ~]$ mail news

Subject: Email to news user

hahah.. we know where this will go.

this will go to root and then to jadi!


Hi Jadi!

Cc:
[jadi@funlife ~]$ mail
Mail version 8.1.2 01/15/2001. Type ? for help.
"/var/mail/jadi": 12 messages 12 new
>N 1 root@funlife          Sat Jan 02 08:50      39/1373 apt-listchanges: news for f
 N 2 root@funlife          Sat Jan 02 09:01 165/7438 apt-listchanges: news for f
 N 3 jadi@funlife          Sat Jan 02 19:58        18/640 *** SECURITY information fo
 N 4 jadi@funlife          Sat Jan 02 20:04        18/631 *** SECURITY information fo
 N 5 jadi@funlife          Sun Jan 03 10:15      18/664  *** SECURITY information fo
 N 6 root@funlife          Mon Jan 04 12:42       27/941     Cron <jadi@funlife> /home/j
 N 7 root@funlife          Mon Jan 04 17:11       26/845  apt-listchanges: news for f
 N 8 root@funlife          Tue Jan 05 18:42      27/945     Cron <jadi@funlife> /home/j
 N 9 root@funlife          Wed Jan 06 09:17       46/1788 apt-listchanges: news for f
 N 10 root@funlife          Thu Jan 07 12:42      27/945     Cron <jadi@funlife> /home/j
 N 11 root@funlife          Thu Jan 07 18:42      27/943     Cron <jadi@funlife> /home/j
 N 12 jadi@funlife          Thu Jan 7 19:53       17/478  Email to news user

    &   12
        Mess
        age
        12:

From jadi@funlife Thu Jan 7 19:53:08 2016 X-
Original-To: news
```

To: news@funlife

Subject: Email to news user

Date: Thu, 7 Jan 2016 19:53:08 +0330 (IRST)

From: jadi@funlife (jadi)

hahah.. we know where this will go.

this will go to root and then to jadi!

Hi Jadi!

& d

& q

Held 11 messages in /var/mail/jadi

Local forwards

We saw that it is possible to **forward emails** using the /etc/aliases. That file is not writable by normal users so what a normal user like *jadi* should do?

Each user can create a .forward file in her own directory and all mail targeted to that user will be forwarded to that address.

You can even put a complete email address like jadijadi@gmail.com in your .forward file. mail command is not part of LPIC102 but it is good if you play and learn it to some extent. It also can send email from within the scripts like 'echo -e "email content" | mail -s "email subject" "example@example.com"'

Mailq

This command **lists the mail queue**. Each entry shows the queue file ID, message size, arrival time, sender, and the recipients that still need to be delivered. If mail could not be delivered upon the last attempt, the reason for failure is shown. The sysadmin can use this command to check the status of emails still in the queues.

```
$ mailq

-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-------

AA52C228E6B        468 Thu Jan 7 19:59:41 jadi@funlife

(connect to alt2.gmail-smtp-in.l.google.com[2404:6800:4003:c01::1a]:25: Network is unreachable)

                                   jadijadi@gmail.com


-- 0 Kbytes in 1 Request.
```

108.4 - Manage printers and printing - *2*

Objectives

- Basic CUPS configuration (for local and remote printers).
- Manage user print queues.
- Troubleshoot general printing problems.
- Add and remove jobs from configured printer queues.

Terms and Utilities

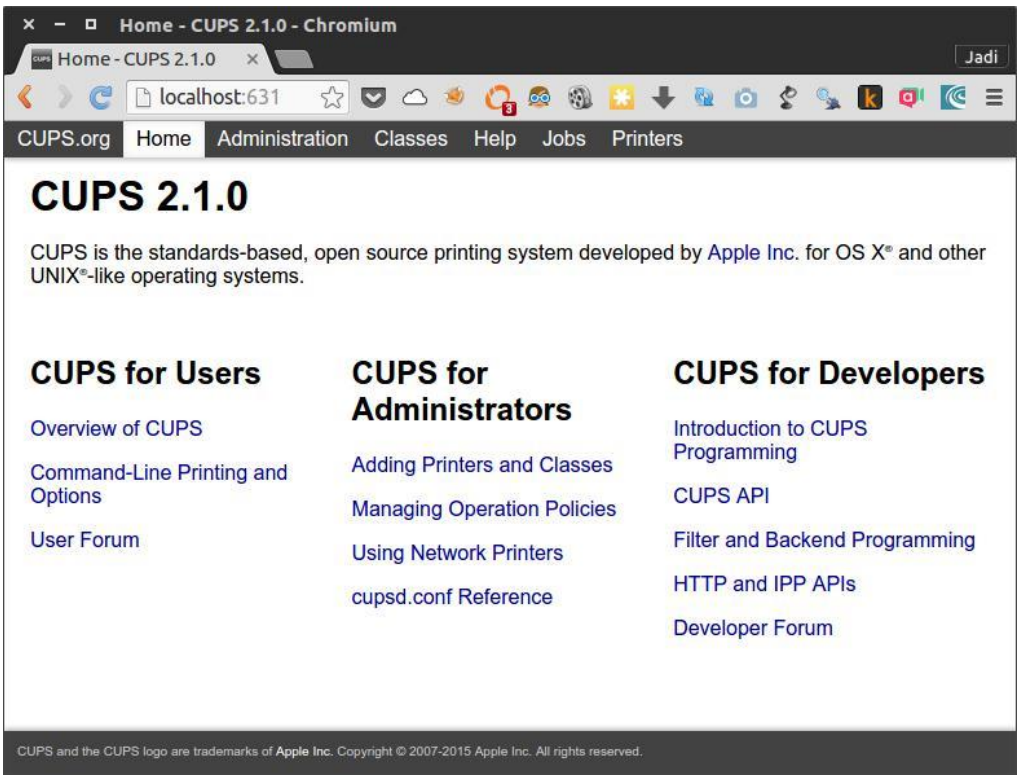
- CUPS configuration files, tools and utilities
- /etc/cups/
- lpd legacy interface (lpr, lprm, lpq)

CUPS

Most Linux distros use **CUPS** for printing. CUPS stands for **Common Unix Printing System**. There are different interfaces for CUPS link command line tools, web based interface and GUIs. CUPS is designed to simplify the printing on various printers from different manufactures.

CUPS web interface

The general way to access the CUPS configuration and info page is going to the servers IP on port **631** from a browser. That will be **localhost:631** or **127.0.0.1:631** from your browser.



Important parts on this webpage are:

|**Jobs tab**|to check the jobs the CUPS is handling|

|**Administration**|For adding printers, managing jobs and configuring the CUPS server|
|**Printers**|Show the printers|

As soon as you push the Add Printer button, you will need to give CUPS admin user password

Suggested Activity: Visit your CUPS web interface and add a printer

Good news is that the CUPS has most of the common printer drivers installed. You just need to choose the printer from the dropdown menu.

Configuration files

As any other linux program, CUPS saves its configuration at /etc directory.

```
# ls /etc/cups

cups-browsed.conf interfaces raw.types subscriptions.conf
cupsd.conf ppd snmp.conf subscriptions.conf.O cups-files.conf
raw.convs ssl
```

One important file is **cupsd.conf**. Have a look at it; it is very easy to understand. For example the Listen localhost:631 line tells the CUPS to listen on localhost port 631.

All the printer data is saved at **/etc/cups/printers.conf**. The web interface or any other GUI is actually editing this file.

```
# Printer configuration file for CUPS v2.1.0

# Written by cupsd

# DO NOT EDIT THIS FILE WHEN CUPSD IS
  RUNNING <DefaultPrinter Apple-Dot-Matrix>

UUID urn:uuid:0f6c2f2b-6338-388a-76de-09f2ef1994d5 Info
Apple Dot Matrix

Location Fake Location

MakeModel Apple Dot Matrix Foomatic/appledmp (recommended)
DeviceURI ipp://fakeprinter/
```

```
State Idle


StateTime 1453402271
ConfigTime 1453402271
Type 8433668 Accepting
Yes

Shared Yes JobSheets
none none
QuotaPeriod 0
PageLimit 0

KLimit 0 OpPolicy
default ErrorPolicy
retry-job
</DefaultPrinter>
```

Legacy tools

Just like the MTA programs, CUPS support all the legacy command line programs too.

command	usage
lpr	print a file
lpq	show print queue/jobs
lprm	rm/remove a file from priner queue
lpc	printer control / troubleshooting program

lpq

The **q** is for **queue** therefor lpq shows the printer queue and is used when you want to see the jobs. If you use the -a switch, the lpq will show the jobs of **all** printers. Alternatively you can use the -P switch to show the jobs of a specific printer. So the following command will show the jobs of a printer called Apple-Dot-Matrix:

```
# lpq -PApple-Dot-Matrix
Apple-Dot-Matrix is ready and printing
Rank    Owner  Job     File(s)                 Total Size
active unknown 1       unknown                 7168 bytes
1st     unknown 2      unknown                 2048 bytes
```

It is strange but there should not be ANY space between -P and the printers name

lpr

This command is used to send a job to a printer. Again the printer is specified by -P.

```
$ lpr -PApple-Dot-Matrix for_print.txt
 lpq
Apple-Dot-Matrix is ready and printing
Rank       Owner     Job  File(s)                      Total Size
active jadi         1      Untitled Document 1              7168 bytes
1st      jadi       2      Untitled1                    2048 bytes
2nd      jadi       3     for_print.txt                  1024 bytes
```

If no printer is specified, the default printer will be used

lprm

The *rm* is for *remove* so the lprm will **remove jobs** from the queue. You need to provide the **Job ID** to this command.

```
$ lpq
Apple-Dot-Matrix is ready and printing
Rank    Owner Job  File(s)                  Total Size
active jadi    1   Untitled Document 1      7168 bytes
1st    jadi    2   Untitled1                2048 bytes
2nd    jadi    3   for_print.txt            1024 bytes
jadi@funlife:/tmp$ lprm 2
jadi@funlife:/tmp$ lpq
Apple-Dot-Matrix is ready and printing
Rank    Owner Job  File(s)                  Total Size
active jadi    1   Untitled Document 1      7168 bytes
1st    jadi    3   for_print.txt            1024 bytes
```

Only root can remove other peoples print jobs

If you need to remove ALL the jobs of a specific printer, you can go with -Pprinter_name -. Yes! that is only one dash (-) after the printer name; that's why this is called a legacy command.

the lprm - will remove all the print jobs

lpc

Here, the **c** is for **control**. lpc lets you check the status (via lpc status) and troubleshoot your printers.

```
$ lpc status
Apple-Dot-Matrix:
   printer is on device 'ipp' speed -1
   queuing is enabled
```

```
printing is enabled

   2    entries
        daemon
        present
```

Here,

queuing is enabled tell us that the queue can accept new print jobs. If the queue is disabled, you can not even send new jobs to the printer.

printing is enabled means that the printer is actually can print on the paper. This will be on the disable state if the printer is out of ink or paper or experiencing a paper jam.

If you are having problems with your printer or need to prevent it from accepting new jobs or let it accept jobs but not print, these four commands will let you achieve your needs:

command	usage
cupsaccept	tells the printer queue to accept new jobs
cupsreject	tells the printer to reject any new job
cupsenable	enables the actual/physical printing of the jobs
cupsdisable	disables the physical printing of the jobs

In all cases you have to provide the printer name of the printer. it is also possible to provide a reason using -r switch.

```
$ cupsdisable Apple-Dot-Matrix -r "need more paper"

$ lpc status

Apple-Dot-Matrix:

   printer is on device 'ipp' speed -1

   queuing is enabled

   printing is disabled

     2    entries
          daemon
          present
```

109.1 - Fundamentals of internet protocols - *4*

Objectives

- Demonstrate an understanding of network masks and CIDR notation.
- Knowledge of the differences between private and public "dotted quad" IP addresses.
- Knowledge about common TCP and UDP ports and services (20, 21, 22, 23, 25, 53, 80, 110, 123, 139, 143, 161, 162, 389, 443, 465, 514, 636, 993, 995).
- Knowledge about the differences and major features of UDP, TCP and ICMP.
- Knowledge of the major differences between IPv4 and IPv6.
- Knowledge of the basic features of IPv6.

Terms and Utilities

- /etc/services
- IPv4, IPv6
- Subnetting
- TCP, UDP, ICMP

IPv4

Internet Protocol or **IP** is an address that points to a destination on the Internet. This destination can be a single computer, a server, a switch or even a network (behind that IP). An IP version four is in the form of A.B.C.D where A, B, C & D are between 0-255. The following are all valid IP addresses:

'''' 1.2.3.4 1.1.1.1 100.0.0.100 192.168.1.4 ''''

each part of an IP address is called an Octet since it is constructed of an 8 bit number (0..255)

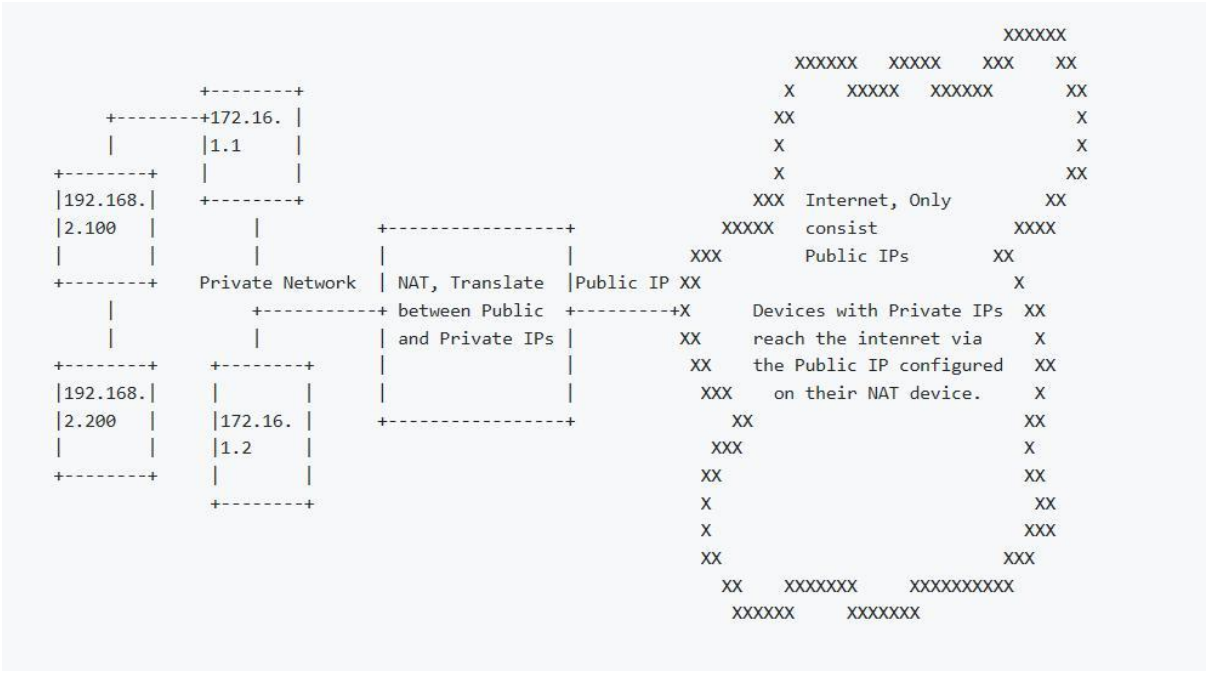
How many IP addresses are available with IPv4? 256*256*256*256 = 4294967296. That is around 4.3 billion - only! Even from this 4.3B, some are not usable (as you will see later) & the *usable* range is around 3.7B.

Private IPs

Any IP address in the form of following addresses are called a *private IP address*:

range	Number of IPs in this range
192.168.0-255.0-255	65K IPs
172.16-31.0-255.0-255	1M IPs
10.0-255.0-255.0-255	16M IPs

Anyone can use any of these IPs on her devices as long as it is not connected directly to the internet. Have a look at this example:



In the middle you can see a **Network Address Translation** or **NAT device**. It is connected to the **Internet** using only one public IP (It might be 5.27.1.30). On the left side of the **NAT box**, there are **4 computers** with Private IPs. Each time any of these devices targets an address on the Internet (a public IP), the NAT device will use his own public IP to request that IP and when he gets the data, will provide it to the *Natted* device - the device behind the NAT. This way we can create huge networks behind a NAT device. In this scenario, all the devices can reach the Internet but no-one from the Internet can reach any of them, directly.

Subnetting

OK! We have around 4B addresses in IPv4. But what happens if I send a packet to another IP? Say I try to print a document on the *local* printer as you learned in previous section? Who should listen for this packet? Does the whole Internet pass it hand to hand? Obviously not! Only my *local* network will wait for such a packet and only my printer (with its local IP) will receive it. When configuring a network, we define **subnet**s. For example, my home network is configured on 192.168.1.0-255. This way my Wi-Fi router is 192.168.1.1, my laptop will be 192.168.1.10, my printer 192.168.1.100, my desktop 192.168.1.2 and my tv 192.168.1.200. When my phone connects to my Wi-Fi it might be 192.168.1.33. Any time any packets tries to reach any IP in my **subnet** (that is 192.168.1.0-255), my router does so. If any devices tries to reach anything out of my subnet (say a public IP), my router will use its NAT functionality to send the packets over the Internet.

This is done using a netmask. But before going there, let me do a fast review on binary representation of number, and specially IP addresses.

Binary representation of IPs

You are probably familiar with binary base - it is the basis of anything related to computers. If not I highly recommend you to do a google search and read if from another source since this book assumes you already know it. In short when writing numbers on binary format (base 2), the only used digits are 0 and 1. An easy way for conversion is using the following table to convert 0 and 1s to our *normal* decimal format.

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	equals 0 in decimal
1	0	0	0	0	0	0	0	equals 128 in decimal
1	0	0	0	0	0	0	1	equals 129 in decimal
0	0	1	0	0	0	0	0	equals 32 in decimal
0	0	0	0	0	1	1	0	equals 6 in decimal
1	0	1	1	0	0	1	1	equals 179 in decimal
1	1	1	1	1	1	1	1	equals 255 in decimal

So 11000000.10101000.00000001.00001111 equals to 192.168.1.15.

Netmask mask

We already saw that subnets are used to create small / local networks. When computers are in one subnet, they can see each other directly without any routing. For any address out of my subnet, a router should handle the packets and send them to their destination. This is done using subnet masks. Let's try an example.

Say we are going to create a subnet like 192.168.1.0-255. Here the 192.168.1 part is same among all the devices and the last *octet* changes. A subnet mask is another string like A.B.C.D which tells the computer what part of the IP in this subnet will remain same among all devices and what parts can be changed. Here, the subnet mask for 192.168.1.0-255 is 255.255.255.0. As you can see any "fixed" octet will have 255 as its subnet mask and the changing (from 0 to 255) octets subnet mask is 0. Can you guess the subnet mask of 10.10.0-255.0-255? That is 255.255.0.0.

fixed parts are 255 and when an octet can be anything from 0 to 255, its subnet mask is 0.

Remember our discussion about binary numbers? We can represent the IPs in binaries:

	network (fixed part)	**subnet (who will**
decimal	192.168.1.	0-255
binary	11000000.10101000.00000001.	00000000-11111111
subnet mask (binary)	11111111.11111111.11111111.	00000000
subnet mask (decimal)	255.255.255.	0

When talking bout subnet masks, we will put a 1 when that digit can not be changed and a 0 when that digit can be changed.

CIDR

Classless Inter-Domain Routing or **CIDR** is another way of talking about subnet masks. Telling someone that "my network is 192.168.1.0 and my subnet mask is 255.255.255.0" is difficult so some people prefer to say "my network is 192.168.1.0/24". This is a shortcut! 24 is the number of 1s in your subnet which is 11111111.11111111.11111111.00000000 in binary and has twenty four 1s. At the beginning this might look difficult but in practice it is more functional to use "my network is 172.16.1.1/16" instead of "my network is 172.16.1.1 with netmask 255.255.0.0".

Here are some famous samples:

decimal netmask	binary netmask
255.255.255.0	11111111.11111111.11111111.00000000
255.255.0.0	11111111.11111111.00000000.00000000
255.0.0.0	11111111.00000000.00000000.00000000
255.255.255.240	11111111.11111111.11111111.11110000
255.255.255.248	11111111.11111111.11111111.11111000

As you can see on the last two examples, the subnet mask (or **netmask**) can start anywhere in any octet.

In short CIDR is just "number of 1s in a netmask".

For a better understanding of subnetting, have a look at Cisco document

IPv6

We saw that there are only around 4B IPv4s available. Just consider that we are around 7B people on the planet earth so there is not even enough IP for every person on IPv4 range. Add to this the latest demands from all the mobile phones, cars, fridges, clocks, TVs, camera, ...! They all want to be on the Internet; this is called Internet Of Things. What should be done? We saw one solution called NAT but the permanent solution is a new version of the IP; IP version 6. In version 6 IPs are not limited to 4 octets anymore.

A sample IPv6 address looks like 2001:0db8:0a0b:12f0:0000:0000:0000:0001 and it can be shortened to 2001:db8:a0b:12f0::1. Here we have decimal numbers from **0000** to **FFFF** and **8** fields which are separated by **:**. This way we will have around 3.4*(10^38) IPs which is enough for whatever we can imagine at the moment. Just imagine that it can allocate 2^52 addresses for every observable star in the known universe.

Although at the moment there is shortage on IPv4, IPv6 is not adopted much yet and most of the Internet is still working on IPv4.

Communication Protocols

When computers communicate, they use Protocols. This protocols are designed to let computers speak in different ways for fulfilling different needs. Here we will have a look at three of the most popular protocols on the Internet: **TCP** / **UDP** / **ICMP**.

TCP

Transmission Control Protocol or **TCP** is designed to make sure that both parties are *speaking* with each other without losing any information. In this protocol the receiver will get anything the sender sends, exactly as it is sent. When you are downloading a program from the Internet, TCP is the best option because you need the file exactly as it is stored on the server. This can be a communication like this:

A- Are you ready?

B- Yes

A- Please share with me the file XYZ

B- Ok. Are you ready?

A- Yes. Start sending

B- OK. This is the part 1 of XYZ

A- OK I got it.

B- Good. Now is it correct?

A- Yes. Please send the second part

B- OK. Are you ready for the second part?

As you can see, TCP needs a lot of communications and sometimes it is not even suitable. When you are listening to music or having a video chat, it is better if the computer just skips some packages in case of problems and continues from the new one it gets.

UDP

You are video-chatting with a friend and network fluctuates. What is a better choice? A) retransmitting the missing packets and/or re-establishing the connection and continue the whole conversation with a 2s delay or B) just show the newer packets we got and continue the live video-conference and just forget about that 2 second fluctuation (missed data)? If your choice is B, it is better if you use UDP (**User Datagram Protocol**) for your chat program. UDP is less reliable: the sender sends packets without communicating much and hearing back from the receiver and receiver listens for packets without negotiating the exact details with the sender. It is **much faster than TCP** but you cannot be sure that 100% of packets will be received by the B party.

ICMP

Internet Control Messaging Protocol or **ICMP** is a specific purpose protocol used to check the connectivity of the servers by the ping command. The first computer just tells "are you there?" and the second will answer "yes I'm here". Have a look at this practical example: [jadi@funlife ~]$ **ping google.com**

```
PING google.com (173.194.32.135) 56(84) bytes of data.

64 bytes from 173.194.32.135: icmp_seq=1 ttl=48 time=239 ms

64 bytes from 173.194.32.135: icmp_seq=2 ttl=48 time=236 ms ^C

--- google.com ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms rtt
min/avg/max/mdev = 236.707/238.174/239.642/1.546 ms
```

As you can see, we are pinging (send ICMP packets) to a server and it answers are back (replying to our ICMP packets).

ICMP is used for network troubleshooting and DOES NOT transfers user data

Port

Any computer has an address but there are many programs running on that computer. By using an IP address, you can tell the Internet the destination of your packets but how can you decide which program on that computer should answer to your packet? We know that a computer with the address of 5.1.23.1 has a webserver and an ftp server on it but how can we reach it and tell it "i want the index.html of your webserver" or "deliver me the XYZ file from your FTP server"? This is done using PORTs.

Ports are numbers where a program LISTENS to. For example, port 80 is reserved for webservers so if I connect to 5.1.23.1:80 I'm sure that I'm talking with the webserver. In the same way, when the file transport protocol (FTP) starts, it starts listening on port 20 & 21 and if I use a FTP client to reach that computer, I will automatically connect to port 21 which is reserved for FTP.

Different ports can use different transmission protocols (UDP or TCP). The default port of some protocols are as follow. These are very important and most admins know them.

port	usage
20, 21	FTP (one data, one control)
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
110	POP3

123	NTP
139	NetBIOS
143	IMAP

port	usage
161, 162	SNMP
389	LDAP
443	HTTPS
465	SMTPS
636	LDAPS
993	IMAPS
995	POP3S

Note that in this table all ports above 400 ends with S, which stands for **S**ecure You can find all of the above ports and many many others in /etc/services

109.2 - Basic network configuration - *4*

Objectives

- Manually and automatically configure network interfaces.
- Basic TCP/IP host configuration.
- Setting a default route.

Terms and Utilities

- /etc/hostname
- /etc/hosts
- /etc/nsswitch.conf
- ifconfig
- ifup
- ifdown
- ip
- route
- ping

ifconfig, up and down

The **ifconfig** is the main command for configuring the network adapters manually. Running it with no arguments, will show all the network adapters and their configurations. $ **ifconfig**

```
enp0s25: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

        ether f0:de:f1:62:c5:73 txqueuelen 1000 (Ethernet)

    RX packets 0 bytes 0 (0.0 B)

    RX errors 0 dropped 0 overruns 0 frame 0

    TX packets 0 bytes 0 (0.0 B)

    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 20 memory 0xd1500000-d1520000



lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet
    127.0.0.1 netmask 255.0.0.0

    inet6 ::1 prefixlen 128 scopeid 0x10<host>
```

```
        loop txqueuelen 1 (Local Loopback)

        RX packets 560719 bytes 339937974 (324.1 MiB)

        RX errors 0 dropped 0 overruns 0 frame 0

        TX packets 560719 bytes 339937974 (324.1 MiB)

        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

lo is a virtual network adapter and is called ***loopback***. It is accessible only from the computer itself. It is used when programs want to speak with the computer they are running on it.

Ethernet networks are called **ethx** or things like **enp0s25**.

It is possible to use **ifconfig** to change the network configurations, but you should have root access:

```
$ sudo ifconfig enp0s25 192.168.42.42

password for jadi:

$ ifconfig enp0s25

enp0s25: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

        inet 192.168.42.42 netmask 255.255.255.0 broadcast 192.168.42.255 ether
        f0:de:f1:62:c5:73 txqueuelen 1000 (Ethernet) RX packets 0 bytes 0 (0.0 B)

        RX errors 0 dropped 0 overruns 0 frame 0

        TX packets 0 bytes 0 (0.0 B)

        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 20 memory 0xd1500000-d1520000


$
```

In the same way, you can change the netmask of an interface with ifconfig eth0 netmask 255.255.0.0 or do both in one step:

```
    #    ifconfig eth0 192.168.42.42 netmask 255.255.255.0
```

It is also possible to turn the interfaces ***up*** and ***down*** using a predefined configuration by:

```
$ sudo ifconfig enp0s25 down

[sudo] password for jadi:

$ ifconfig

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet
        127.0.0.1 netmask 255.0.0.0

        inet6 ::1 prefixlen 128 scopeid 0x10<host>

        loop txqueuelen 1 (Local Loopback)

        RX packets 562273 bytes 340257228 (324.4 MiB)

        RX errors 0 dropped 0 overruns 0 frame 0

        TX packets 562273 bytes 340257228 (324.4 MiB)

        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0


wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet
        192.168.1.35 netmask 255.255.255.0 broadcast 192.168.1.255 inet6
        fe80::8ea9:82ff:fe7b:8906 prefixlen 64 scopeid 0x20<link> ether
        8c:a9:82:7b:89:06 txqueuelen 1000 (Ethernet)

        RX packets 2330388 bytes 2634026235 (2.4 GiB)

        RX errors 0 dropped 0 overruns 0 frame 0

        TX packets 2027352 bytes 511549072 (487.8 MiB)

        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

As you can see _down_ing the interface removed it from the list of active interfaces, using switch -a will tell the ifconfig to show ALL interfaces, even if they are down.

In many systems there are **ifup** and **ifdown** commands directly to up and down interfaces easily. They work just like **ifup eth0**.

Network Gateways

A computer normally can see all of the computers in its own subnet / netmask. But what happens when you send a packet to a computer *outside* of your own network? In this case your computer delivers that packet to an address called **network gateway**. The **gateway** device can **route** the packets between different networks. It has more than 1 interface and is connected to different networks so working like a post office, it can hand over your packets to another network and after several handovers, and your packet will reach its destination.

In your network configurations, there is a **default gateway**. That is the address which is used as a **gateway** when your computer tries to reach a computer outside its network.

Network configuration files

Redhat based systems

Unfortunately Debian based and Redhat based systems use different locations for their nework configuration files. On Redhat, CentOS, Fedora, ... the fields are located at **/etc/sysconfig/network-scripts/.** A sample is as below:

```
$ cat /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

ONBOOT=yes

TYPE=Ethernet

IPADDR=192.168.1.10

NETMASK=255.255.255.0

DNS1=4.2.2.4
```

On these systems, the default gateway is configured via the below file:

```
cat /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=lpictest

GATEWAY=192.168.1.1
```

Debian based systems

On Debian based systems (including Ubuntu) the main configuration file for network interfaces is **/etc/network/interfaces**. This one file has the configuration for all of the interfaces. Have a look:

```
auto lo

iface lo inte loopback


auto eth0

#ifconfig eth0 inet dhcp

iface eth0 inet static

address 192.168.1.10

netmask 255.255.255.0

gateway 192.168.1.1

dns-nameservers 4.2.2.4
```

ifdown and ifup will use these config files

DNS config file

As you saw, we were able to set the DNS configuration in network interface config files. But this is not the only way. There is another file which contains this data: **/etc/resolv.conf**. $ **cat /etc/resolv.conf**

```
# Generated by
resolvconf
nameserver
192.168.1.1
```

If you want to change your DNS on the fly, you can edit this file but it will be lost after reboot or **ifdown** and **ifup**.

Hostname

There is another text file which shows or sets the hostname. That is **/etc/hostname**. $ **cat /etc/hostname**

```
funlife
```

Hosts

The **/etc/hosts** file contains server names and their IPs. It is just like what DNS does but has a higher priority than DNS. If you add something like

```
4.2.2.4 funnyip
```

There and ping funnyip your computer will start pinging 4.2.2.4 without querying any DNS server.

There is an entry on /etc/hosts for your machine. If you are changing the /etc/hostname it is important to add that name to your /etc/hosts line containing 127.0.0.1 too.

Route

The route command can show or change the routing system. As you saw on **default gateway** section, routing is responsible to send your packets to their correct destination. For checking the current route you can issue

```
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    600    0        0 wlp3s0
192.168.1.0     *               255.255.255.0   U     600    0        0 wlp3s0
```

And for temporary **adding a default route**, you can do:

```
route add default gw 192.168.1.1
```

IP

The **ip** command is the new tool for configuring the networking interfaces. You can do many things using it. the **addr** show will show you the current interfaces and their configurations: $ **ip addr show**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet
127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
   valid_lft forever preferred_lft forever
2: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
    default qlen 1000
link/ether 8c:a9:82:7b:89:06 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.35/24 brd 192.168.1.255 scope global dynamic wlp3s0
   valid_lft 254572sec preferred_lft 254572sec
inet6 fe80::8ea9:82ff:fe7b:8906/64 scope link
   valid_lft forever preferred_lft forever
3: enp0s25: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOWN group default
    qlen 1000
```

```
link/ether f0:de:f1:62:c5:73 brd ff:ff:ff:ff:ff:ff


inet 192.168.42.42/24 brd 192.168.42.255 scope global enp0s25
   valid_lft forever preferred_lft forever
```

Ping

Ping is the most straight forward network troubleshooting command. You can check your connection with any server using it. Let's see if my computer sees 4.2.2.4: $ ping 4.2.2.4

```
PING 4.2.2.4 (4.2.2.4) 56(84) bytes of data.

64 bytes from 4.2.2.4: icmp_seq=1 ttl=52 time=103 ms

64 bytes from 4.2.2.4: icmp_seq=2 ttl=52 time=101 ms

64 bytes from 4.2.2.4: icmp_seq=3 ttl=52 time=103 ms

--- 4.2.2.4 ping statistics ---

6 packets transmitted, 6 received, 0% packet loss, time 5007ms rtt
min/avg/max/mdev = 101.465/103.608/108.219/2.263 ms
```

Nsswitch

The **/etc/nsswitch.conf** file is used to configure which services are to be used to determine information such as **hostnames**, **password files**, and **group files**. Mine is

```
# cat /etc/nsswitch.conf
# Begin /etc/nsswitch.conf

passwd: files

group: files

shadow: files

publickey: files

hosts: files dns myhostname

networks: files

protocols: files
```

```
services: files

ethers: files

rpc: files


netgroup: files
```

End /etc/nsswitch.conf

So if someone wants to check a password, the system will try the password *file* on the system. Or if they want to check an ip address of a hostname, my config says hosts: files dns myhostname so the computer first tries the files (/etc/hosts) and then goes for DNS. If I reverse these and change the line to

```
hosts:      dns files
```

Any resolve request will be sent to a DNS server first and the /etc/hosts will be used *only* if the DNS servers answers "I don’t know!"

109.3 - Basic network troubleshooting - *4*

Objectives

- Manually and automatically configure network interfaces and routing tables to include adding, starting, stopping, restarting, deleting or reconfiguring network interfaces.
- Change, view, or configure the routing table and correct an improperly set default route manually.
- Debug problems associated with the network configuration.

Terms and Utilities

- ifconfig
- ip
- ifup
- ifdown
- route
- host
- hostname
- dig
- netstat
- ping
- ping6
- traceroute
- traceroute6
- tracepath
- tracepath6
- netcat

Troubleshooting network problems

When a network related problem is reported to you, you have to take a lot of steps to find out where the root cause of the problem is. For example if the report says "I cannot open webpages", you have to start from checking if the network interface has an ip, if it is up, it the routing is OK and if the DNS works fine and if everything is OK, you have to try reaching a server on the Internet via **ping** command and if you see any problems, you have to use **traceroute** to see where your traffic is going wrong. In this lesson, we will review these basic steps.

ifconfig & ip

As you already know, **ifconfig** and **ip** commands can be used to check the IP address of your interfaces. If a network card is going to work, it needs a correct IP address and netmask. Let me check my own computer to see it has an IP address: [jadi@funlife ~]$ **ip addr show**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default

    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet
    127.0.0.1/8 scope host lo

       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host

       valid_lft forever preferred_lft forever

2: wlp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN
group default qlen 1000

    link/ether f0:de:f1:62:c5:73 brd ff:ff:ff:ff:ff:ff

3: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default
qlen 1000

    link/ether 8c:a9:82:7b:89:06 brd ff:ff:ff:ff:ff:ff

    inet 192.168.1.35/24 brd 192.168.1.255 scope global dynamic wlp3s0
       valid_lft 254836sec preferred_lft 254836sec

    inet6 fe80::8ea9:82ff:fe7b:8906/64 scope link
       valid_lft forever preferred_lft forever
```

```
[jadi@funlife ~]$ ifconfig

enp0s25  Link encap:Ethernet HWaddr f0:de:f1:62:c5:73

         inet addr:192.168.1.35 Bcast:192.168.1.255 Mask:255.255.255.0 inet6
         addr: fe80::8ea9:82ff:fe7b:8906/64 Scope:Link

         UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX
         packets:231586 errors:0 dropped:0 overruns:0 frame:0 TX
         packets:200220 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000

         RX bytes:198053888 (198.0 MB) TX bytes:51583154 (51.5 MB)
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:221752 errors:0 dropped:0 overruns:0 frame:0
          TX packets:221752 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:150859909 (150.8 MB) TX bytes:150859909 (150.8 MB)
```

Both commands show that my IP address is OK. This is assigned to my be the WiFi modem and *192.168.1.35* as IP and *255.255.255.0* looks reasonable.

Ping

This is the most common tool when troubleshooting a network problem. Let's see if I can see my network gateway first.

```
[jadi@funlife ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask       Flags Metric Ref  Use Iface
default         192.168.1.1     0.0.0.0       UG    600    0      0 wlp3s0
link-local      *               255.255.0.0   U     1000   0      0 wlp3s0
192.168.1.0     *               255.255.255.0 U     600    0      0 wlp3s0

[jadi@funlife ~]$ ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

64 bytes from 192.168.1.1: icmp_seq=1 ttl=254 time=3.02 ms

64 bytes from 192.168.1.1: icmp_seq=2 ttl=254 time=3.08 ms ^C

--- 192.168.1.1 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms rtt
min/avg/max/mdev = 3.023/3.052/3.082/0.062 ms
```

I can ping my gateway but is it possible to reach a server on the Internet using its IP address. Let's ping 4.2.2.4; it is very well known server on the Internet and many people use it to check their connectivity.

```
[jadi@funlife ~]$ ping 4.2.2.4

PING 4.2.2.4 (4.2.2.4) 56(84) bytes of data.

64 bytes from 4.2.2.4: icmp_seq=1 ttl=50 time=108 ms

64 bytes from 4.2.2.4: icmp_seq=2 ttl=50 time=111 ms

64 bytes from 4.2.2.4: icmp_seq=3 ttl=50 time=113 ms ^C

--- 4.2.2.4 ping statistics ---
```

```
3 packets transmitted, 3 received, 0% packet loss, time 2003ms rtt
min/avg/max/mdev = 108.160/111.233/113.717/2.338 ms This is
working fine. Bud can I ping google.com too?



[jadi@funlife ~]$ ping google.com

ping: unknown host google.com
```

Aha! We found the problem. In this case I have a correct IP address, I can ping 4.2.2.4 but I cannot ping google.com with the error message "unknown host". This means my computer cannot translate google.com to its IP address; this is a DNS issue:

```
[jadi@funlife ~]$ cat /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
```

No wonder we had problems with surfing the WWW. There is no active DNS in my computer so no one will be able to reach sites by their domain names! This should be fixed by adding a DNS to my configuration files and then use ifdown eth0 and then ifup eth0 to load it.

Routing problems

In some situations you cannot reach the Internet (say 4.2.2.4 or 8.8.8.8) but you can ping your gateway. Have a look:

```
[jadi@funlife ~]$ ping 8.8.8.8

connect: Network is unreachable

[jadi@funlife ~]$ ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

64 bytes from 192.168.1.1: icmp_seq=1 ttl=254 time=3.03 ms

64 bytes from 192.168.1.1: icmp_seq=2 ttl=254 time=3.31 ms ^C

--- 192.168.1.1 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms rtt
min/avg/max/mdev = 3.039/3.174/3.310/0.146 ms
```

What is going on here? Let's see what clues I have:

1) I can reach the gateway

2) When asking for the Internet, my computer does not know what to do. In this case the **default gateway** is missing: the computer does not know what to do if a packet is outside its network mask.

You know that we can set the default gateway using the /etc/network/interfaces config file but there is also a route command to show and change the routing configurations on the fly. Routes added or changed via route command will be lost after a reboot! Permanent configurations should come from configuration files.

Let’s check our current routing state using route command as root.

```
[jadi@funlife ~]$ sudo route

[sudo] password for jadi:

Kernel IP routing table

Destination     Gateway         Genmask       Flags Metric Ref  Use Iface

link-local   *                  255.255.0.0  U             1000  0      0 wlp3s0

192.168.1.0     *                 255.255.255.0           U6000       0 wlp3s0
```

It is stated that "there is no gateway for the network 192.168.1.0 netmask 255.255.255.0". This means that if you ping a server in this network (say 192.168.1.100), it does not needs any routing. But what happens if you ping 8.8.8.8? Nothing stated in our *routing table* above! No wonder that we are getting Network is unreachable when ping 4.2.2.4. Let’s add a default route:

```
[jadi@funlife ~]$ sudo route add default gw 192.168.1.1

[jadi@funlife ~]$ ping 4.2.2.4

PING 4.2.2.4 (4.2.2.4) 56(84) bytes of data.

64 bytes from 4.2.2.4: icmp_seq=1 ttl=50 time=109 ms

64 bytes from 4.2.2.4: icmp_seq=2 ttl=50 time=111 ms

64 bytes from 4.2.2.4: icmp_seq=3 ttl=49 time=199 ms

--- 4.2.2.4 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2002ms rtt
min/avg/max/mdev = 109.493/140.344/199.612/41.922 ms


[jadi@funlife ~]$ sudo route

Kernel IP routing table

Destination     Gateway         Genmask       Flags Metric Ref  Use Iface

default         192.168.1.1         0.0.0.0         UG     0      0      0 wlp3s0

link-local      *                 255.255.0.0  U             1000  0      0 wlp3s0

192.168.1.0     *                 255.255.255.0    U     600     0      0 wlp3s0
```

We were able to ping the Internet after adding a default gateway using route command. It is important to know that we can define even more routes and control exactly where our packets should go based on their destinations.

Netstat

This command can show many different information about our network. The two main functionalities are showing the **routing table** and checking the **listening ports**. Let's check our routing table using it.

```
[jadi@funlife ~]$ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window irtt Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 wlp3s0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 wlp3s0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 wlp3s0
```

It is also very useful to the see what ports are in listening state in our server:

```
[jadi@funlife ~]$ netstat -na | grep 80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
```

The **-na** switch will show all the open ports and here I'm checking if port 80 (web) is active on my server. It is and listening to any connection from 0.0.0.0 which means "anywhere": Any one can connect to my computers web server and request pages.

In these switches, -n stands for *numeric*, -a stands for *all ports* and -r stands for *routes*.

Traceroute

This is a more advanced troubleshooting tool. It is like pinging all the servers between you and your destination one by one and see where our packets go wrong. Let's check how I reach google.com.

```
[jadi@funlife ~]$ traceroute google.com

traceroute to google.com (173.194.32.160), 30 hops max, 60 byte packets

 1  192.168.1.1 (192.168.1.1) 3.090 ms 3.113 ms 3.115 ms

 2  85-15-16-103.shatel.ir (85.15.16.103) 29.904 ms 34.851 ms 34.885 ms

 3  85-15-0-193.shatel.ir (85.15.0.193) 34.880 ms 39.454 ms 39.459 ms

 4  85-15-0-1.shatel.ir (85.15.0.1) 39.451 ms 39.444 ms 43.865 ms

 5  172.18.196.17 (172.18.196.17) 43.887 ms 46.506 ms 51.990 ms

 6  172.18.196.22 (172.18.196.22) 51.943 ms 29.418 ms 29.430 ms

 7  10.10.53.229 (10.10.53.229) 32.421 ms 32.344 ms 34.705 ms

 8  10.10.53.222 (10.10.53.222) 39.814 ms 39.773 ms 39.808 ms

 9  xe-0-3-3-xcr2.fri.cw.net (62.208.252.109) 126.654 ms 131.411 ms 131.363 ms 10
ae29-xcr1.fri.cw.net (195.2.24.221) 126.570 ms 129.053 ms 129.073 ms

11 195.2.19.6 (195.2.19.6) 173.273 ms 173.284 ms 178.795 ms

12 216.239.56.106 (216.239.56.106) 143.786 ms 145.279 ms 216.239.57.143

(216.239.57.143) 115.669 ms

13 209.85.143.25 (209.85.143.25) 111.427 ms 113.970 ms 113.989 ms

14 74.125.37.88 (74.125.37.88) 127.402 ms 125.199 ms 74.125.37.92 (74.125.37.92)

122.175 ms

15 209.85.247.72 (209.85.247.72) 137.551 ms 137.564 ms 140.535 ms

16 72.14.236.249 (72.14.236.249) 161.583 ms 170.447 ms 170.407 ms

17 173.194.32.160 (173.194.32.160) 170.375 ms 72.14.235.231 (72.14.235.231) 159.352 ms
161.666 ms
```

On the first step (1) I reach my own router. On the 2nd step, I'm at my ISP which is shutel. I'm still on shatel on 3rd and 4th steps and so on. After a long path, on the 17th step, I reach my destination. This is used to troubleshoot routing problems.

In some routers, the ping trafic is blocked and you will see * * * at some steps because those servers are blocking ICMP traffic.

There is another command called **tracepath** which is very similar to traceroute. For the LPIC1 level, they are essentially the same!

IPv6

If you are on an IPv6 network you have to use **traceroute6** and **ping6** instead of traceroute and ping.

Dig

The **dig** command is a **DNS lookup tool**. If you are having problem with a domain name, you can check how it is being resolved to IPs; and by whom. [jadi@funlife ~]$ **dig google.com**

```
; <<>> DiG 9.9.5-11ubuntu1.3-Ubuntu <<>> google.com ;;
global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50032
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.            IN    A

;; ANSWER SECTION:
google.com.        293    IN    A    216.58.214.46

;; Query time: 120 msec
;; SERVER: 4.2.2.4#53(4.2.2.4)
;; WHEN: Fri Apr 01 22:00:05 IRDT 2016
;; MSG SIZE rcvd: 44
```

You can see that SERVER 4.2.2.4 is resolving google.com to 216.58.214.46.

Netcat

The **nc** (or **netcat**) utility is used for just about anything under the sun involving TCP, UDP, or UNIX-domain sockets. It can **open TCP connections**, **send UDP packets**, **listen on arbitrary TCP** and **UDP** ports, do **port scanning**, and deal with **both IPv4** and **IPv6**.

Unlike telnet, nc scripts nicely, and separates error messages onto standard error instead of sending them to standard output, as telnet does with some. This is a very capable command and it is enough for you to be familiar with its general concept.

109.4 Configure client-side DNS - *2*

Objectives

- Query remote DNS servers.
- Configure local name resolution and use remote DNS servers.
- Modify the order in which name resolution is done.

Terms and Utilities

- /etc/hosts
- /etc/resolv.conf
- /etc/nsswitch.conf
- host
- dig
- getent

DNS

We already know a lot about **Domain Name Server** - A service who translates domain

names (like yahoo.com) to IP addresses (like 206.190.36.45). A DNS server is used when you try to ping a server using its name. You have seen the config files for DNS and should know that the actual DNS server which is being used by the computer can be checked / changed (temporarily) from **/etc/resolv.conf**:

```
$ cat /etc/resolv.conf

# Generated by NetworkManager
nameserver 192.168.1.1
nameserver 4.2.2.4
```

```
$ ping x.org

PING x.org (131.252.210.176) 56(84) bytes of data.

64 bytes from annarchy.freedesktop.org (131.252.210.176): icmp_seq=1 ttl=45 time=338 ms
```

```
64 bytes from annarchy.freedesktop.org (131.252.210.176): icmp_seq=2 ttl=45 time=333

ms

^C

--- x.org ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1000ms rtt
min/avg/max/mdev = 333.088/335.612/338.136/2.524 ms
```

Dig

The dig tool is specifically build to **query DNS**. If you want to find out where x.org points to,

you can do:

```
$ dig x.org

; <<>> DiG 9.10.3-P4-RedHat-9.10.3-12.P4.fc23 <<>> x.org ;;
global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7483
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;x.org.                         IN      A

;; ANSWER SECTION:
x.org.                  1625    IN      A       131.252.210.176

;; Query time: 35 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Sun Apr 17 12:45:02 IRDT 2016
;; MSG SIZE rcvd: 50
```

As you can see, dig did a **ip lookup** for x.org and told me that the IP is 131.252.210.176. The 1625 is called the TTL or *Time To Live* and show how many seconds before this answer expires. This command also tells us which server is used to find the answer (last 4 lines) and when and how long it took.

There is also a way to tell dig command what server it should use as the DNS:

```
$ dig @8.8.8.8 google.com

; <<>> DiG 9.10.3-P4-RedHat-9.10.3-12.P4.fc23 <<>> @8.8.8.8 google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24313
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             112     IN      A       173.194.32.133
google.com.             112     IN      A       173.194.32.136
google.com.             112     IN      A       173.194.32.132
google.com.             112     IN      A       173.194.32.129
google.com.             112     IN      A       173.194.32.137
google.com.             112     IN      A       173.194.32.130
google.com.             112     IN      A       173.194.32.134
google.com.             112     IN      A       173.194.32.135
google.com.             112     IN      A       173.194.32.128
google.com.             112     IN      A       173.194.32.131
google.com.             112     IN      A       173.194.32.142
```

;; Query time: 238 msec

;; SERVER: 8.8.8.8#53(8.8.8.8)

;; WHEN: Sun Apr 17 13:03:38 IRDT 2016

;; MSG SIZE rcvd: 215

Here I have asked dig to use 8.8.8.8 as its DNS and query google.com. You can see that I've got more than 1 answer (actually much more than 1 answer). My computer can randomly contact any of those IPs to reach the google.com. In other words, google.com is using more than 1 server/IP and 8.8.8.8 provides all of them when queried for that domain.

/etc/hosts

This is a file containing IP addresses and their domain names - statically saved! Lets have a look:

```
$ head /etc/hosts

127.0.0.1 funlife localhost.localdomain                localhost clickadu.com

::1           funlife localhost6.localdomain6          localhost6


10.159.32.155 nsnproxy

172.16.12.134 linuxclass wonderland

193.40.12.135 salma

87.106.233.90 gratis.vps

192.168.59.231 mass1
```

This file can be changed by root and will map some domain names (localhost, mass1, gratis.vps, ...) to some IP addresses. If I ping *mass1* on this computer.. lets see:

```
$ dig mass1


; <<>> DiG 9.10.3-P4-RedHat-9.10.3-12.P4.fc23 <<>> mass1 ;;
global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 39464

;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1


;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 512

;; QUESTION SECTION:

;mass1.                                 IN          A
```

```
;; AUTHORITY SECTION:

. 600 IN SOA a.root-servers.net. nstld.verisign-grs.com. 2016041700 1800 900 604800 86400



;; Query time: 516 msec

;; SERVER: 192.168.1.1#53(192.168.1.1)

;; WHEN: Sun Apr 17 13:15:07 IRDT 2016

;; MSG SIZE rcvd: 109
```

$ **ping mass1**

```
PING mass1 (192.168.59.231) 56(84) bytes of data.

From 85-15-16-103.shatel.ir (85.15.16.103) icmp_seq=1 Packet filtered From
85-15-16-103.shatel.ir (85.15.16.103) icmp_seq=2 Packet filtered
```

My computer pings 192.168.59.231 when I go for mass1 even when the DNS cannot finds this name because that is defined in **/etc/hosts**.

Nsswitch

The **/etc/nsswitch.conf** file tells the system about the priority of lookups, password checks. $ **cat /etc/nsswitch.conf**

```
#

# /etc/nsswitch.conf

#
# An example Name Service Switch config file. This file should be

# sorted with the most-used services at the beginning.

#

# The entry '[NOTFOUND=return]' means that the search for an

# entry should stop if the search in the previous entry turned

# up nothing. Note that if the search failed due to some other reason

# (like no NIS server responding) then the search continues with the

# next entry.

#

# Valid entries include:

#
  #         nisplus                 Use NIS+ (NIS version 3)

  #         nis                     Use NIS (NIS version 2), also called YP

  #         dns                     Use DNS (Domain Name Service)

  #         files                   Use the local files

  #         db                      Use the local database (.db) files

  #         compat                  Use NIS on compat mode

  #         hesiod                  Use Hesiod for user lookups

  #         [NOTFOUND=return]       Stop searching if not found so far

  #


# To use db, put the "db" in front of "files" for entries you want to be

# looked up first in the databases

#

# Example:
```

```
#passwd:     db files nisplus nis

#shadow:     db files nisplus nis

#group:      db files nisplus nis


passwd:      files sss

shadow:      files sss

group:       files sss


#hosts:      db files nisplus nis dns

hosts:       files mdns4_minimal [NOTFOUND=return] dns myhostname mymachines


# Example - obey only what nisplus tells us...
#services:  nisplus [NOTFOUND=return] files

#networks:  nisplus [NOTFOUND=return] files
```

```
#protocols: nisplus [NOTFOUND=return] files

#rpc:        nisplus [NOTFOUND=return] files

#ethers:      nisplus [NOTFOUND=return] files

#netmasks:  nisplus [NOTFOUND=return] files


bootparams: nisplus [NOTFOUND=return] files


ethers:      files

netmasks:  files

networks:  files

protocols: files

rpc:        files

services:  files sss


netgroup:  files sss


publickey: nisplus


automount: files sss

aliases:     files nisplus
```

On the DNS line I have hosts: files mdns4_minimal [NOTFOUND=return] dns myhostname mymachines. This means when the system wants to find the IP address of a name, it first go for the files (/etc/hosts) and then for m**dns**4_minimal and then **dns** and so on. In this case if I add the facebook.com to my **/etc/hosts** like this:

```
127.0.0.1 facebook.com
```

And then point my browser to facebook.com, my computer will try to open a webserver on 127.0.0.1 instead of the real IP of Facebook.

Getent

The **getent** command is a utility to get entries from Name Service Switch libraries (read /etc/nsswitch.conf). If you want to check what the config of your hosts is, you can do as follow.

```
$ getent hosts

127.0.0.1       funlife localhost.localdomain localhost clickadu.com

127.0.0.1       funlife localhost6.localdomain6 localhost6

10.159.32.155  nsnproxy

172.16.12.134  linuxclass wonderland

193.40.12.135  salma

87.106.233.90  gratisvps

192.168.59.231 mass1

192.168.59.232 mass2

192.168.59.233 mass3

192.168.59.234 mass4

192.168.59.235 mass5

192.168.59.236 mass6

192.168.59.237 mass7

192.168.59.238 mass8

192.168.59.239 mass9

127.0.0.1       frctlstartupfailure localtodoer localdeliv

127.0.0.1       frctlmeth

```

110.1 - Perform security administration tasks - *3*

Objectives

- Audit a system to find files with the suid/sgid bit set.
- Set or change user passwords and password aging information.
- Being able to use nmap and netstat to discover open ports on a system.
- Set up limits on user logins, processes and memory usage.
- Determine which users have logged in to the system or are currently logged in.
- Basic sudo configuration and usage.

Terms and Utilities

- find
- passwd
- fuser
- lsof
- nmap
- chage
- netstat
- sudo
- /etc/sudoers
- su
- usermod
- ulimit
- who, w, last

suid and guid

When the **suid** bit is set on an executable file, whoever runs the file, is running the file with the access of the owner of the file. Have a look at the ping command: jadi@funlife ~$ **type ping**

ping is hashed (/bin/ping)

jadi@funlife ~$ **ls -ltrh /bin/ping**

-rwsr-xr-x 1 root root 44K May 8 2014 /bin/ping

The s on the access rights part, tells us that whoever runs the ping command, the ping command will be run with root access. This is needed by root and is OK on my distro.

What happens if someone changes the suid of the vi command? let's see who own vi:

```
jadi@funlife ~$ type vi

vi is /usr/bin/vi

jadi@funlife ~$ ls -ltrh /usr/bin/vi

lrwxrwxrwx 1 root root 20 Jun 1 12:52 /usr/bin/vi -> /etc/alternatives/vi
```

At the moment, the vi is owned by root, so if the suid bit is set, vi will always be run as root!

In that case anybody will be able to edit any file! You can see why it is important to check for suid files on your system:

```
$sudo find / -perm -u+s
```

Same applies for guid. If the guid is set, the file will be run with access of its group.

Open Ports

netstat, fuser and lsof

On module 109.1 we talked about ports, ports are like wholes in our systems used by servers to listen to the outside world. If I'm running a web server on my computer I should have a port open so people can ask that server "please show me your index.html". Many malwares open ports to let the attacker to communicate with them. It is important to check your computer for open ports time to time. The main command for this task is **netstat** using the **-na** or **-ap** or **-tuna** switch.. I'm sure tuna is easy to remember if you have the tuna fish in mind!

```
jadi@funlife ~$ netstat -tuna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:9050          0.0.0.0:*               LISTEN
tcp       25      0 192.168.59.9:49934      192.168.59.192:139      CLOSE_WAIT
tcp        0      0 127.0.0.1:60228         127.0.0.1:1080          ESTABLISHED
tcp        0      0 192.168.1.35:55324      159.203.148.169:8385    ESTABLISHED
tcp        0      0 127.0.0.1:59590         127.0.0.1:1080          ESTABLISHED
tcp        0      0 127.0.0.1:60212         127.0.0.1:1080          ESTABLISHED
tcp        0      0 192.168.1.35:54220      159.203.148.169:8385    ESTABLISHED
tcp        0      0 127.0.0.1:57186         127.0.0.1:1080          ESTABLISHED
tcp        0      0 192.168.1.35:49574      173.194.122.231:443     ESTABLISHED
tcp        0      0 127.0.0.1:59002         127.0.0.1:1080          ESTABLISHED
udp        0      0 0.0.0.0:54502           0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
```

All the LISTEN ports are servers; they are LISTENING for new incoming connections. The ESTABLISHED connections are the active connections between your computer and another computer. In these tables 0.0.0.0 dictates *any address* or *any interface*.

Another useful tool here is **lsof** and **fuser**. The former is already discussed in previous sections.

lsof shows the open files on the system and having in mind that *everything in Linux is a file or a process* you can conclude that this command should be able to display open connections too; and you are right:

```
# lsof -i
COMMAND      PID       USER    FD  TYPE DEVICE SIZE/OFF NODE NAME
privoxy      806   privoxy     4u IPv4       16130    0t0 TCP funlife:8118 (LISTEN)
cups-brow    903      root     8u IPv4       17477     0t0 UDP *:ipp
mysqld       971     mysql   19u IPv4       20875     0t0 TCP funlife:mysql (LISTEN)
tor         1038 debian-tor      6u IPv4       19155    0t0 TCP funlife:9050 (LISTEN)
dnsmasq      1260      nobody 11u IPv4 1910037         0t0 UDP *:18666
adb          1278     jadi          5u IPv4 579541    0t0 TCP funlife:5037 (LISTEN)
chromium- 2891          jadi  88u IPv4 813611          0t0 TCP 192.168.1.35:45702->do-
13.lastpass.com:https (ESTABLISHED)
chromium- 2891          jadi 126u IPv4 1907389             0t0 TCP 192.168.1.35:50642->ntt-
2.lastpass.com:https (ESTABLISHED)
chromium- 2891          jadi 133u IPv4 1909733             0t0 TCP 192.168.1.35:50644->ntt-
2.lastpass.com:https (ESTABLISHED)
chromium- 2891          jadi 268u IPv4 785289              0t0 TCP 192.168.1.35:60736->lf-in-
f188.1e100.net:5228 (ESTABLISHED)
python       4925     jadi    4u IPv4 658287          0t0 TCP funlife:8000 (LISTEN)
Telegram     4943     jadi    39u IPv4 773463         0t0 TCP 192.168.1.35:44732-
>149.154.175.50:https (ESTABLISHED)
dhclient     9984     root    6u IPv4 787885          0t0 UDP *:bootpc
nginx       11095      root    6u IPv4       17998    0t0 TCP *:http (LISTEN)
nginx       11099     www-data  7u IPv6  17999         0t0 TCP *:http (LISTEN)
chrome      14264     jadi 114u IPv4 788089           0t0 UDP *:mdns
chrome      14264     jadi 126u IPv4 1872872             0t0 TCP funlife:60370->funlife:socks
(ESTABLISHED)
chrome      14264     jadi 138u IPv4 1908382             0t0 TCP funlife:60408->funlife:socks
```

```
(ESTABLISHED)
```

Wow! this command shows the command, PID, user running it and source and destination IP and tells of if this is a LISTENING or STABLISHED connection.

If you want to check which process is using port 80, you can grep the output of any above commands or simply use the fuser command to find all the PIDs related to that specific port:

```
root@funlife:/bin# fuser 80/tcp

80/tcp:               11095 11096 11097 11098 11099
```

nmap

nmap is the toolbox of hackers! You can nmap a server to find out about a lot of data about that server:

```
# nmap localhost


Starting Nmap 7.01 ( https://nmap.org ) at 2016-06-04 21:32 IRDT
Nmap scan report for localhost (127.0.0.1) Host is up (0.0000070s
latency).

rDNS record for 127.0.0.1: funlife

Not shown: 995 closed ports

PORT      STATE SERVICE

80/tcp  open http

1080/tcp open socks

3306/tcp open mysql

8000/tcp open http-alt

9050/tcp open tor-socks


Nmap done: 1 IP address (1 host up) scanned in 1.66 seconds
root@funlife:~#
```

In the most basic form, **nmap** checks all the open ports from 1 to 1000 and prints the results. There are a lot of switches to find other information about the hosts and they are used by every single hacker who wants to examine a server's status.

sudo vs su

We've used sudo and su in all the chapters and this is time to have a closer look at them! **su** changes your account to something else. You get a new prompt with the new user account after successfully suing to that account: jadi@funlife ~$ **whoami**

```
jadi

jadi@funlife ~$ su -

Password:

root@funlife:~# whoami

root
```

```
root@funlife:~# su jadi -

bash: cannot set terminal process group (-1): Inappropriate ioctl for device

bash: no job control in this shell

jadi@funlife /root$ whoami

jadi

jadi@funlife /root$ exit

exit
```

```
root@funlife:~# whoami

root

root@funlife:~# exit

logout

jadi@funlife ~$ whoami

jadi

jadi@funlife ~$
```

Note that when running su you have to **provide the root password** to become root; or any other users password to become that user!

On the other hand, **sudo** asks for you own password and runs the command you gave it, with the root privileges. So sudo ls runs the ls command with the root privileges after asking for **your password**. Obviously you should have the *sudo right* to issue sudo. This is defined in **/etc/sudoers** file:

```
$ sudo cat /etc/sudoers

#

# This file MUST be edited with the 'visudo' command as root.

#
# Please consider adding local content in /etc/sudoers.d/ instead of

# directly modifying this file.

#

# See the man page for details on how to write a sudoers file.

#

Defaults env_reset
Defaults mail_badpass

Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"


# Host alias specification
```

```
# User alias specification

# Cmnd alias specification

# User privilege specification
root        ALL=(ALL:ALL) ALL


# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo      ALL=(ALL:ALL) ALL


# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Note the 2 important lines: how root gets the right to run all the commands and how the sudo and admin groups get rights to run commands as root. The ALL:ALL means these users can run as any use and any group. The last ALL tells the sudo that these users / groups can run ALL commands. It is possible to put /bin/ping in the last part to tell sudo that this user can run only ping as root.

The /etc/sudoers file is very important and breaking it will make major problems. To prevent you from adding un-interpretable lines in that file, the **visudo** command should be used instead of vi /etc/sudoers. This tool will check your edits to make sure that sudo command can understand them.

Now we know what sudo su - means. The sudo tells the system to run the su - command with the root access. It asks your password and runs the su - as the root if you have sudo access. The su commands changes your user to root and - switch load the root environment variables. This way you can become root using your own password via running su with sudo.

User limits

The resources on a Linux machine can be manages for users by the **ulimit** command. It is part of the PAM system. If you want to check the limits on the system run: ~$ **ulimit -a**

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 47457
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 47457
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

And you can change them like this:

```
$ ulimit -t 1
```

This will limit the CPU TIME of any process to 1 seconds. If you use more than that, the process will be killed automatically (by PAM module). Please note that clock time is different than CPU time. To see how much CPU time a process uses run it like this:

```
$ time firefox
```

Changing the **ulimit** as we did is a temporary thing. It only persists in that specific shell.

To change the **ulimits system-wide**:

```
$ cat /etc/security/limits.conf

# /etc/security/limits.conf

#
#Each line describes a limit for a user in the form:

#
#<domain>        <type> <item> <value>

#

#Where:

#<domain> can be:

#        - a user name

#        - a group name, with @group syntax

#        - the wildcard *, for default entry

#        - the wildcard %, can be also used with %group syntax,
```

```
#                for maxlogin limit
#        - NOTE: group and wildcard limits are not applied to root.
#          To apply a limit to the root user, <domain> must be
#          the literal username root.
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit (KB)
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues (bytes)
#        - nice - max nice priority allowed to raise to values: [-20, 19]
#        - rtprio - max realtime priority
#        - chroot - change root to directory (Debian-specific)
#
#<domain>       <type> <item>           <value>
```

```
#

#*              soft      core          0
#root           hard      core          100000
#*              hard      rss           10000
#@student       hard      nproc         20
#@faculty       soft      nproc         20
#@faculty       hard      nproc         50
#ftp            hard      nproc         0
#ftp            -         chroot        /ftp
#@student       -         maxlogins     4


# End of file
```

Soft limits can be changed by the user but hard limits are the real stop points.

Checking the users in the system

If you need to check who is in your system (and to some extend what they are doing) you can use these commands:

```
$ w
 22:03:37 up 3 days, 5:33, 13 users, load average: 1.48, 1.12, 1.19
USER     TTY      FROM             LOGIN@    IDLE  JCPU  PCPU WHAT
jadi     tty7     :0               Wed16     3days 2:30m 1.96s /sbin/upstart --user
jadi     pts/18   :0               Wed16     3:04m 1:02  1:02 /usr/bin/python manage.py runserver
0.0.0.0:8000
jadi     pts/19   :0               Wed16     1:11m 0.35s 0.35s /bin/bash
root     tty2                      Wed16     3days 0.07s 0.03s -bash
jadi     pts/21   :0               08:41     45:37    0.06s 0.06s /bin/bash
jadi     pts/23   :0               Thu11     46:49    0.25s 0.23s ssh startups
jadi     pts/21   funlife          Fri22     45:37  0.06s 0.06s /bin/bash
jadi     pts/25   :0               10:17     31:37    0.07s 5.81s /usr/bin/python /usr/bin/x-terminal-
emulator
jadi     pts/26   :0               21:39     0.00s 0.07s 0.00s w
jadi     pts/27   :0               21:55     8:09    0.01s 0.01s /bin/bash
```

You have a line for each logged in users (every single shell window is a separated login).

Another useful command is **who**. Let's check it:

```
$ who
jadi     tty7             2016-06-01 16:30 (:0)
jadi     pts/17           2016-06-0116:30 (funlife)
jadi     pts/2         2016-06-01 16:32 (:0)
jadi     pts/18           2016-06-0116:32 (:0)
jadi     pts/19           2016-06-0116:33 (:0)
root     tty2          2016-06-01 16:36
jadi     pts/21           2016-06-0408:41 (:0)
```

As you can see both these commands tell you when was the **time** that the user logged into the system but does not show the logged out people (because they are not on the system anymore!). If you need that data use the last command:

```
~$ last | head
jadi     pts/27       :0           Sat Jun 4 21:55      gone - no logout
jadi     pts/26       :0           Sat Jun 4 21:39      gone - no logout
jadi     pts/26       :0           Sat Jun 4 18:55 - 19:42 (00:46)
jadi     pts/25       :0           Sat Jun 4 10:17      gone - no logout
jadi     pts/26       :0           Sat Jun 4   09:25 - 09:26 (00:00)
jadi     pts/26       :0           Sat Jun 4   09:25 - 09:25 (00:00)
jadi     pts/25       :0           Sat Jun 4   08:52 - 09:27 (00:35)
jadi     pts/21       :0           Sat Jun 4   08:41     gone - no logout
jadi     pts/21       funlife         Fri Jun 3 22:22 - 08:41 (10:18)
```

There is a way to check the failed logins too: **last -f /var/log/btmp**

110.2 - Setup host security - *3*

Objectives

- Awareness of shadow passwords and how they work.
- Turn off network services not in use.
- Understand the role of TCP wrappers.

Terms and Utilities

- /etc/nologin
- /etc/passwd
- /etc/shadow
- /etc/xinetd.d/
- /etc/xinetd.conf
- /etc/inetd.d/
- /etc/inetd.conf
- /etc/inittab
- /etc/init.d/
- /etc/hosts.allow
- /etc/hosts.deny

Shadow passwords

The **/dev/passwd** is already discussed. It contains the passwords of the users but there is a logical problem: if a user should be able to change her own password, he should have access to this file and if this is the case, he can see other people's passwords. $ **ls -ltrh /etc/passwd**

```
-rw-r--r-- 1 root root 2.5K Jun 5 19:14 /etc/passwd
```

To prevent this the **/etc/shadow** file is introduces. In modern systems, we only show a * at

the **location of the password** in **/etc/passwd** and **store the real password** in **/etc/shadow**.

```
jadi@funlife ~$ grep jadi /etc/passwd

jadi:x:1000:1000:jadi,,,:/home/jadi:/bin/bash

jadi@funlife ~$ grep jadi /etc/shadow
```

grep: /etc/shadow: Permission denied

jadi@funlife ~$ **sudo grep jadi /etc/shadow**

[sudo] password for jadi:

jadi:6bp01DBX.$I6dt4pz8GeXJl6asgPeKhSdepf40bgepTz8zwB3HFmN56SdcsxjTETdZAmRt1
7biwMYOI7SoGFOXssHqeNFgw/:16963:0:99999:7::: jadi@funlife ~$ sudo ls -ltrh /etc/shadow

-rw-r----- 1 root shadow 1.5K Jun 11 17:36 /etc/shadow

/etc/nologin

This is a cool file! If you create and write something in it, the content will be shown to any person who tries to login into the system and **the login attempt will fail**. It is useful for maintenance time. Delete this file and the users will be able to login again.

Admin users will be able to login even in the presence of /etc/nologin

Turn off network services

Turning services off is easy but a bit different on different systems. In case you want to turn the **httpd service off**, you can do:

system	service manager	command
older linux systems	SysV	chkconfig httpd off sysv-rc-conf httpd off
Ubuntu	Upstart	update-rc.d httpd remove
newer linux distros	systemd	systemctl disable httpd

Please note that these commands prevent the service from starting on system boot.

Super-servers

A super-server or sometimes called a service dispatcher is a type of daemon run generally on Unix-like systems for security reasons. It starts other servers when needed, normally with access to them checked by a TCP wrapper.

This is a sample **xinetd** configuration:

```
service telnet
{
	disable		= no
	flags		= REUSE
	socket_type	= stream
	wait		= no
	user		= root
	server		= /usr/sbin/in.telnetd
	log_on_failure += USERID
	no_access	= 10.0.1.0/24
	}
```

If we change the disable to yes and restart the xinetd, the telnet daemon will start running.

There are a few files to control to xinetd related files.

/etc/hosts.allow & /etc/hosts.deny

These two files will **allow** or **deny access** from specific hosts. Its logic is like cron.deny and cron.allow. If something is **allowd**, everything else is denied but if you add something to the /etc/hosts.deny, only that specific thing is denied (and every other thing is allowed).

```
jadi@funlife ~$ cat /etc/hosts.allow

# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:  ALL: LOCAL @some_netgroup
#           ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#

# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#

telnet: 10.10.100.
```

Here the telnet service is only allowed from 10.10.100.* . It is possible to use ALL as the service name to allow or deny ALL services.

After changing this file, xinetd should be restarted

As mentioned, super servers are not being used anymore and most distributions use standalone services running on them. This is also called tcp wrapping since the tcp connections are being passwd from xinetd and xinetd decides what to do with them.

110.3 - Securing data with encryption - *3*

Objectives

- Perform basic OpenSSH 2 client configuration and usage.
- Understand the role of OpenSSH 2 server host keys.
- Perform basic GnuPG configuration, usage and revocation.
- Understand SSH port tunnels (including X11 tunnels).

Terms and Utilities

- ssh
- ssh-keygen
- ssh-agent
- ssh-add
- ~/.ssh/id_rsa and id_rsa.pub
- ~/.ssh/id_dsa and id_dsa.pub
- /etc/ssh/ssh_host_rsa_key and ssh_host_rsa_key.pub
- /etc/ssh/ssh_host_dsa_key and ssh_host_dsa_key.pub
- ~/.ssh/authorized_keys
- ssh_known_hosts
- gpg
- ~/.gnupg/

Key Pairs

In traditional cryptography the **symmetric keys** were used: both parties had a shared password; the files were encrypted with that password and then decrypted using the same password. These years the **Key Pairs** are becoming more and more common. When generating a key pair, we generate two keys using a computer algorithm in the way that any message which is encrypted using one, can be opened only using the other key. These are called **Public & Private key**. You publish the public key to your friends and even the strangers and if they need to send an encrypted message to you, scramble it using YOUR public key and send it to you. After receiving it, you open the file using your PRIVATE Key.

Great point about Public / Private Key is that the data can be transmitted over the internet with no fear of hackers or governments. You are publishing your key to the word, someone picks it and uses it to encrypt some data and sent the result to you. People can see that you are receiving "some data" but they cannot encrypt it because they do not have the private key needed to decrypt it.

ssh key pairs

Before using keys, we have to **generate** them. This task is as simple as running one command:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jadi/.ssh/id_rsa):
Created directory '/home/jadi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jadi/.ssh/id_rsa.
Your public key has been saved in /home/jadi/.ssh/id_rsa.pub.
The key fingerprint is:
bc:1c:1a:da:0b:8b:22:56:da:1f:68:14:5b:da:7e:2e jadi@localhost.localdomain
The key's randomart image is:
+-- [ RSA 2048]----+
|                 |
|                 |
|  . .            |
|       * .       |
|   +..S          |
| ..oo + o        |
| +oooo.o         |
|oo.o E+.         |
|o.. o.o.         |
+----------------          +
```

The above process asked for a passphrase. It is used to secure our key. You can create a password and it will be asked each time you want to use this key.

We have the option to generate keys using **different algorithms**, here I used the default RSA

and these two keys are created in ~/.ssh/:

```
[jadi@localhost ~]$ ls .ssh -ltrh

total 8.0K

-rw-r--r--. 1 jadi jadi 408 Jun 15 13:58 id_rsa.pubss -rw-
------. 1 jadi jadi 1.7K Jun 15 13:58 id_rsa
```

As soon as we ssh to any server, a file called **known_hosts** will save that sites public keys.

The next step is copying our public key there. For this task there is command called **ssh-copyid**.
It works like this:

```
 ssh-copy-id 10.0.2.15

jadi@10.0.2.15's password:

Now try logging into the machine, with "ssh '10.0.2.15'", and check in:

  .ssh/authorized_keys
```

To make sure we haven't added extra keys that you weren't expecting.

Now we will be able to **login** into that server **without providing our password**.

There is a **key pair** called **/etc/ssh_host_rsa_key** and **/etc/ssh_host_rsa_key.pub** on any server. These keys are used to **verify the security of the connectivity**. Server provides me its public key and uses its private key to keep the connection secure. This is the public key the client saved in its **~/.ssh/known_hosts**.

ssh tunnels

We have already discussed X forwarding.

Encryption using gpg

A software called **gpg** lets us use public and private keys to encrypt our data. At the beginning we have to **create a key pair**:

```
gpg --gen-key
```

Then we need to share our public key to other people. To **export our public key file** we need to run:

```
gpg --export name > gpg.pub
```

And the other party can **import our public key** into his gpg database by:

```
gpg --import gpg.pub
```

At this stage, if he wants to encrypt some data to us (say the file file.txt) he should run:

```
gpg --out file.txt.encrypted --recipient jadijdai@gmail.com --encrypt file.txt
```

And give the file.txt.encrypted to us. For opening it, we just need to:

```
gpg --out out.txt --decrypt file.txt.encrypted
```

As you saw, the public keys should be shared. The B party uses A parties public key to encrypt a data and A party uses his Private key to open it.

Revoking keys

What happens if you forget your key password or someone hacked your computer and acquired your private key? In this case you have to announce to the world that "I'm hacked! Do not use that public key of mine anymore!". This is called revoking. To **create a revoke key**, run:

```
gpg --output revoke.asc --gen-revoke jadijadi@gmail.com
```

This tells gpg to create a **revoke file** called **revoke.asc** for the identity jadijadi@gmail.com. If jadijadi@gmail.com needs to invalidate his public key, he have to publish this file to the internet or key servers.

Signing

In the previous section we used **gpg** to ecrypt the data. We used someone's Public key to sign and she used her own Private Key to decrypt the text. What happens if we do this in the reverse? I mean what happens if I encrypt something with my Private key, send it on the internet and everyone on the internet will be able to decrypt it using my Public key which is shared with everyone - since it is Public. This is signing!

By encrypting a document using your private key, you let everyone to try to open it using your public key and if they succeed, they will be sure that you have signed it using YOUR private key! gpg has a specific command to sign documents:

```
gpg --clearsign originalfile
```

Here the **--clearsign** tells the gpg to include the clear text message in the output file too. The output file will be **originalfile.asc**

And another one to **verify** that a document is singed correctly:

```
gpg --verify recievedfile
```

Bibliography

Bresnahan, C. (e.d.). *102.1 - Design Hard Disk Layout & Beyond*. Irkuprat minn https://static1.squarespace.com/static/510bcffce4b02b65f6e3d89f/t/52ec14f8e4b0296c217ae388/1391203576272/Objective+102-1+Design+Hard+Disk+Layout.pdf.

Bresnahan, C. (e.d.). *Objective 101-1: Determine and Configure Hardware Settings Study Sheet*. Irkuprat minn https://static1.squarespace.com/static/510bcffce4b02b65f6e3d89f/t/52ec113de4b034d1de3335b3/1391202621555/Objective+101-1+Determine+and+Configure+Hardware+Settings.pdf.

Izgurskii, D. (e.d.). *lpic-1-study-notes*. Irkuprat minn https://github.com/Dmitrii-I/lpic-1-study-notes/blob/master/exam-101.md.

Mirmirani, J. (e.d.). *LPIC1 exam guide in plain English*. Irkuprat minn https://jadi.gitbooks.io/lpic1/content/1011_determine_and_configure_hardware_settings.html.

Semaev, K. (e.d.). *lpic_1-101*. Irkuprat minn https://github.com/ksemaev/lpic_1-101.

Shields, I. (e.d.). *Learn Linux, 101: Manage user and group accounts and related system files*. Irkuprat minn https://developer.ibm.com/tutorials/l-lpic1-107-1/.

Shields, I. (e.d.). *Learn Linux, 101: Manage user and group accounts and related system files*. Irkuprat minn https://developer.ibm.com/tutorials/l-lpic1-107-1/.

Smith, R. W. (e.d.). *LPIC-1: Linux Professional Institute Certification Study Guide*.

Solutions, L. E. (e.d.). *Linux 101 Examination*. Irkuprat minn https://www.ledge.co.za/software/lpinotes/101-letter.pdf.