Linux

LPI 102

Exam
Preparation
Version - 2

91- LPI-102 -V2 -Exam Preparation

About this document:

This document is been produced to help candidates to pass the LPI 102 exam. I have created it essentially as a reference document and not as a tutorial. That's why in general, this document doesn't have many explanations for the subjects treated. I usually use this document in my courses designed as preparation to this exam. Although it covers, to my knowledge, the most important aspects of the topics asked in the exam, its layout and its content organization is not perfect. Helped by this document, most of my students, if they practiced enough, passed the exam. In some topic I have added more information than needed for the LPI 102 exam. When in doubt, just read again the description of the requirements located at the beginning of each topic.

I offer this document free. You may distribute, modify, improve, use personnaly or commercially as you whish. I don't take any responsibility of any kind for the accuracy of the information in this document as well as for the success or failure of the participants in passing the exam.
I would only appreciate that, if you make modifications to this document, that you send me a copy of it.
My first language being French, the english phrase constructions and vocabulary may suffer a bit in this document.

I do invite you to let me know of any errors or recommendations related to the accuracy of the information in this doicument, that would help me to improve it. Any feedback of any kind is also welcome. If anybody wants to contribute to this document, you're very welcome, please let me know.
My email: michel@linuxint.net
My Web Page: www.linuxint.net

I hope it will help you preparing for the LPI 102 exam and remember that, practice, practice, and more practice is the key.

Note: I'm still working on this document .
The section **1.112.1 Fundamentals of TCP/IP** is incomplete and might stay that way, at least for a while.
This section needs to be more understood with explanations which I don't have time to write. At the moment there are only protocol references.

LPI 102 Introduction:

This is a required exam for LPI certification Level 1. It covers basic system administration skills that are common across all distributions of linux.
Each objective is assigned a weighting value. The weights range roughly from 1 to 10, and indicate the relative importance of each objective. Objectives with higher weights will be covered in the exam with more questions.

Notes about LPI 102 Exam:

- **Number of Questions per topics**:

Topic	Nr. of Questions
Admin Taks	15
Network Fundamentals	10
Network Services	18
Security	6
Kernel	5
Boot,init,runlevels	4
Printing	3
Documentation	6
Shell,scripts,compilation	6
Total ----->	**73**

- **Special notes about the 102 Exam.**

  - Most questions that require that you fill-in the blank, don't require any options. eg. `cat` or `ls` or `cp` (without options)

  - Use your experience and common sense in deciding what is important and what is not when studying. When in doubt, just read again the description of the requirements located at the beginning of each topic.

  - It is recommended to create a checklist of topics for yourself and to review it once in a while to keep an overview. This can help avoiding spending too much time in one subject at the expense of other important subjects.

  - Pay attention to the weight of each topic and spend the equivalent time on them.

  - When doing the exam, it is recommended to answer first the questions that you are sure of and then go back to the other ones afterwards.

  - Read the questions thoroughly and make sure you understand them well. Then read ALL the answers carefully before answering. I almost got cought a few times, answering someting I was sure it can't be anything else but when I read the other answers I saw which one was really the correct answer.

  - The exam is difficult and needs concentration and a good memory. It is not recommended to eat a heavy meal before the exam.

  - No need to rush through the exam and risk overviewing something. There is more than enough time to answer all the questions. When you're finished and there is still time, take the time to review your answers once.

Weight per topics:

Topic	Weight
105: Kernel	
1.105.1 Manage/Query kernel and kernel modules at runtime	4
1.105.2 Reconfigure, build, and install a custom kernel and kernel modules.	3
106: Topic 106 Boot, Initialization, Shutdown and Runlevels	
1.106.1 Boot the system	3
1.106.2 Change runlevels and shutdown or reboot system	3
107: Printing	
1.107.2 Manage printers and print queues	1
1.107.3 Print files	1
1.107.4 Install and configure local and remote printers	1
108: Documentation	
1.108.1 Use and manage local system documentation	4
1.108.2 Find Linux documentation on the Internet	3
1.108.5 Notify users on system-related issues	1
109: Shells, Scripting, Programming and Compiling	
1.109.1 Customize and use the shell environment	5
1.109.2 Customize or write simple scripts	3
111: Administrative Tasks	
1.111.1 Manage users and group accounts and related system files	4
1.111.2 Tune the user environment and system environment variables.	3
1.111.3 Configure and use system log files to meet administrative and security needs	3
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	4
1.111.5 Maintain an effective data backup strategy	3
1.111.6 Maintain system time	4
112: Networking Fundamentals	
1.112.1 Fundamentals of TCP/IP	4
1.112.3 TCP/IP configuration and troubleshooting	7
1.112.4 Configure Linux as a PPP client	3
113: Networking Services	
1.113.1 Configure and manage inetd, xinetd, and related services	4
1.113.2 Operate and perform basic configuration of sendmail	4
1.113.3 Operate and perform basic configuration of Apache	4
1.113.4 Properly manage the NFS, smb, and nmb daemons	4
1.113.5 Setup and configure basic DNS services	4
1.113.7 Set up secure shell (OpenSSH)	4
114: Security	
1.114.1 Perform security administration tasks	4
1.114.2 Setup host security	3
1.114.3 Setup user level security	1

Topic 105: Kernel

- **1.105.1 - Manage/Query kernel and kernel modules at runtime**
  Weight: 4

  Description:
  Candidates should be able to manage and/or query a kernel and kernel loadable modules. This objective includes using command-line utilities to get information about the currently running kernel and kernel modules. It also includes manually loading and unloading modules as appropriate. It also includes being able to determine when modules can be unloaded and what parameters a module accepts. Candidates should be able to configure the system to load modules by names other than their file name.

  Key files, *terms*, and utilities include:

```
/lib/modules/$(uname -r)/modules.dep
/etc/modules.conf or
/etc/conf.modules
depmod          rmmod
insmod          modinfo
lsmod           modprobe
uname
```

- **Linux/UNIX Kernel:**

In UNIX or Linux Kernel version 1.x.x the kernel must be recompiled for new features or device drivers.
From Linux Kernel version 2.x.x external modules can be compiled separately from the kernel and dynamically loaded or unloaded. They are called Kernel Modules.

- **Mernel Options at Boot time**

The list of options supported by the current kernel can be found in :
`/usr/src/linux/Documentation/kernel-parameters.txt`.

- **Kernel Modules:**

The kernel modules are normally located in:

`/lib/modules/`*`kernel-version`*`/*` or
`/lib/modules/$(uname -r)/*`   `$(uname -r)` = kernel version

- Modules are files with the extention '**.**o' eg. `serial.o`
- Modules can depend on other modules to be loadable. The list of modules dependencies is located at: `/lib/modules/`*`kernel-version`*`/`**`modules.dep`**
  This file is produced by running the command: `depmod.` `depmod` will also generate various map files in this directory, for use by the *hotplug* infrastructure.
- Modules can be loaded in 2 different ways:
  - Manualy using the commands `insmod` and `modprobe`:

`insmod` *`modulename`*	Loads the module without checking for dependencies.
`modprobe` *`modulename`*	Checks the module's dependencies. Loads all the dependencies if needed and then loads the module.

- Automatically via:
  - The hotplug infrastructure (see LPI-101 Hardware section) (for filesystems etc.)
  - The `devfsd` daemon and an `alias` entry in `/etc/modules.conf`
    `devfsd` will load the module each time the device is accessed

```
syntax:    alias /dev/devicefile modulename
eg.        alias /dev/net/tun    tun
```

  - The kmod support in kernel (`CONFIG_KMOD`) and an `alias` entry in `/etc/modules.conf.` kmod uses `modprobe` to load modules.

```
Syntax:   alias DeviceInternalName  modulename
          alias block-major-NN[-nn] modulename
          alias char-major-NN-[nn]  modulename

eg.       alias eth0                8139too
          alias block-major-58      lvm-mod
          alias char-major-10-134   apm
          alias char-major-81       bttv
```

NN is The Device Major Number and the *nn* is the minor number.

```
eg.   ls -l /dev/apm_bios
crw-rw----  1 root root 10, 134 Jan 18 11:26 apm_bios
Entry in modules.conf:   alias char-major-10-134  apm
```

To create new devices in `/dev` directory use the following format:

```
     mknod -m modes /dev/newdev {c|b} majorNr. minorNr.
eg.  mknod -m 644   /dev/ttyS4   c      4        67
```

or use the script `MAKEDEV:`

```
eg.   cd /dev ; ./MAKEDEV ttyS
```

  - A runlevel script. The script can issue `modprobe` commands when the system boots-up to load modules ready to use.

- Note: the file `/etc/modules.conf` and `/etc/conf.modules` are the same. Which filename is used varies between distributions but `modules.conf` is newer.

- For a module to dynamically link to the kernel, a kernel symbol table with memory pointers is used. Such table can be seen at `/proc/ksyms`.

- **Programs used to control modules:**

Note: The *modulename* never contains the '`.`o' extention of its filename.

`lsmod` Lists the loaded modules. Same result as `cat /proc/modules`
Syntax: `lsmod`

`insmod` Loads a module (no dependency check)
Syntax: `insmod modulename [module_parameters]`
eg. `insmod ne io=0x300 irq=5`

`modprobe` Loads/Removes a module(with dependency check)
`modprobe` expects an up-to-date `modules.dep` file,
as generated by `depmod`.

```
Syntax:     modprobe [-vcniqo] module [module_params]
eg.  modprobe [-l] [-t dir.] [-a] [wildcard]
     modprobe -r module1 [module2] ...  (-r = remove)
```

Automatic try of all network card modules until success:

```
     modprobe -t net \*
```

Note: `modprobe` configuration file:(`/etc/modprobe.conf`) if exists.

`rmmmod` Removes(unloads) a module.

```
Syntax:     rmmod [-r] module1 [module2] ....
            -r = Removes recursively through dependencies
```

`depmod` Determines module dependencies and writes `modules.dep` file.

```
Syntax:     depmod [-abeFAn]      (-n=Writes only to screen).
eg.         depmod -av   Checks all and writes modules.dep
Note:  Run depmod -a after changes in /etc/modules.conf
```

`modinfo` Prints information about a module.

```
Syntax:     modinfo [-adlpn] [-F field] modulename
            -n = /path/filename  of module's file
            -F --field  Only print this field value, one per line.
Field names:
      author(-a), description(-d), license(-l),
      depends,and alias.
      param(-p) :  Shows which parameters are supported.
            Output format of -p:
            option type (valid-values) description
      Options [-adlp] are shortcuts for these above fields.
```

- **The file `/etc/modules.conf`** (or `/etc/conf.modules`):

This file is used by `kmod` to load the right modules automatically when certain devices are accessed or by `modprobe` to add needed options to modules and possibly run certain commands before and/or after loading and/or unloading modules. It can contain the following information:

- Module Parameters(options)

```
Syntax:     options modulename  options
eg.         options ne          io=0x300 irq=5
```

- Alias names for modules: Modules is then having 2 names.

```
Syntax:     alias aliasname modulename
eg.         alias eth0      3c509
            Makes it possible to do a -------> modprobe eth0
            which has the same result as --> modprobe 3c509
```

- Commands that should be run before and/or after a module is loaded

```
Syntax:     pre-install  modulename  command
            post-install modulename  command
eg.         post-install bttv        insmod tuner
```

- Commands that should be run before and/or after a module is un-loaded

```
Syntax:     pre-remove  modulename   command
            post-remove modulename   command
eg.         post-remove bttv         rmmod tuner
```

The command 'uname'

Syntax: `uname options`

This command is used to display information about the current system.

`uname -a`	Shows all information *in the following order*:
`-s, --kernel-name`	Print the kernel name
`-n, --nodename`	Print the network node hostname
`-r`	Print the current kernel release. eg. `/lib/modules/$(uname -r)/......` or `` /lib/modules/`uname -r`/....... ``
`-v, --kernel-version`	Print the kernel version (Build date)
`-m, --machine`	Print the hardware machine name
`-p, --processor`	Print the processor type
`-i, --hardware-platform`	Print the hardware platform
`-o, --operating-system`	Print the operating system

- **1.105.2 - Reconfigure, build, and install a custom kernel and kernel modules**
  Weight: 3

  Description:
  Candidates should be able to customize, build, and install a kernel and kernel loadable modules from source This objective includes customizing the current kernel configuration, building a new kernel, and building kernel modules as appropriate. It also includes installing the new kernel as well as any modules, and ensuring that the boot manager can locate the new kernel and associated files (generally located under /boot, see objective 1.102.2 for more details about boot manager configuration).

- **Key files, *terms*, and utilities include:**
  `/usr/src/linux/*`
  `/usr/src/linux/.config`
  `/lib/modules/kernel-version/*`
  `/boot/*`
  `make`
  make targets: **`config`**, **`menuconfig`**, **`xconfig`**, **`oldconfig`**, **`modules`**, **`install`**, **`modules_install`**, **`depmod`**

- **Kernel Source code**

  The kernel source code is normally located in `/usr/src/linux/*`.
  Normally this directory is a symbolic link to `/usr/src/`*`kernelname/`* directory.
  It contains also all the configuration files necessary to compile the kernel.

- **Configuring the kernel:**
  - Getting the source code and the current kernel configuration file.
    The source code is normally available from the current distribution disks or from the internet(`www.kernel.org`)

    After getting the source code installed in the system, the kernel needs to be configured before compiling it. This configuration process wil create the configuration file : `/usr/src/linux/.config`
    We have the choice of using an older configuration file as a template or create a totally new one from scratch.(not recommended)

    Before issuing any commands we need to change to the source code directory:

    ```
    cd /usr/src/linux
    ```

  - **Preparing the an old `.config` for a new kernel source.**
    Copy the old .config to `/usr/src/linux/` directory and run the command:
    `make oldconfig`
    This will scan the file and add the new items that were not existing in the old kernel but present in the new kernel.

  - **Configuration programs**
    The following 3 commands start programs that read the .config file, allow for changing the configuration and when finished, saves the new configuration in the same `.config` file, replacing the original.

`make config`	Older questions oriented.
`make menuconfig`	Text/Menu oriented.
`make xconfig`	Menu/Buttons Graphic Program

    The main work of kernel configuration is to decide for:

- Which features are supported in the kernel.
- Which modules will be either:
    - Integreated in the kernel or
    - Compiled as separate loadable modules
    - Not compiled seperately and not integrated in the kernel.

- **Preparing the compilation**
  Since 'make' doesn't compile already compiled parts of the kernel, in order to create completely new ones, some already compiled need to be deleted by issuing the command : `make clean`

  Before compiling the kernel, the dependencies file need to be created.
  This file is named: `/usr/src/linux/.depend`
  The command: `make dep`

- **Compiling the kernel**
  The long and complex compiling process can now start by issuing one of the following commands:

make zimage	Old command to create a small jernel which will be saved as: `/usr/src/linux/arch/i386/boot/zImage`
`make zdisk`	Old command that once compiled the kernel will be saved in a floppy as a boot floppy.
`make bzImage`	New command to create a big kernel which will be saved as: `/usr/src/linux/arch/i386/boot/bzimage`
`make bzdisk`	New command that once compiled the kernel will be saved in a floppy as a boot floppy.

- **Compiling the modules**
  The compiling of the modules is made by issuing the command:(also long)
  `make modules`

- **Installling the modules**
  Once compiled the modules need to be installed in the directory `/lib/modules/`*`kernelversion/`* by issuing the command:
  `make modules_install`

  The command `depmod -a` will be automatically by this above command.

- **Installing the new kernel**
  Once compiled the kernel and the system map file need to be copied to `/boot` directory and the boot manager configuration file modified to reflect the changes.
  This can be achieved by issuing the following commands:

```
cp /usr/src/linux/arch/i386/boot/bzimage /boot/vmlinuz
cp /usr/src/linux/System.map /boot/System.map.$(uname -r)
```

  (This `.map` file is a list of kernel symbols used mostly for debuging purposes, same as `/proc/ksyms`).

- **If an `initrd` is needed** then issuing the following command will do it:
  `mkinitrd` *`Options-needed`*

If using LILO	If using GRUB

`vi /etc/lilo.conf`
`lilo`
	`vi /boot/grub/menu.lst`

- **All kernel compiling commands in short**:

  - Install the kernel source in `/usr/src/linux/` directory

  - Copy the `.config` file from the current kernel in `/usr/src/linux/` directory

  - `make clean` Deletes all already compiled modules from source tree

  - `make oldconfig` Uses the current `.config` and creates a new one

  - `make xconfig` or `make menuconfig` or `make config`

    To configure the kernel options before compiling

  - `make dep` Creates the dependencies file `.depend`

  - `make bzImage` Compile the kernel

  - `make modules` Compile the modules

  - `make modules_install`

    Install modules in `/lib/modules/kernelversion/`

  - `cp /usr/src/linux/arch/i386/boot/bzimage /boot/vmlinuz`

    Copies the kernel in the `/boot` directory

  - `cp /usr/src/linux/System.map /boot/System.map.$(uname -r)`

    Copies the `.map` file in the `/boot` directory

  - If using an `initrd` file when booting: `mkinitrd` *`Options`*

  - If using LILO: `vi /etc/lilo.conf` (Edit `lilo.conf`) then `lilo`

  - if using GRUB: `vi /boot/grub/menu.lst` or
    `vi /boot/grub/grub.conf`

- **Safeguard against a non-working new kernel**:
  To make sure that the the old kernel is saved as an alternative to boot, in the case of the new kernel not working, it is advisable to change the name of the older kernel, its `initrd`, and *`System.map.`*`$(uname -r)`, and its `/lib/modules/`*`kernelversion/`* directory before copying the kernel or issuing the comand `make modules_install`.

  An alternative menu item in the boot manager config file for being able to boot the older kernel is also advisable.

Topic 106: Topic 106 Boot, Initialization, Shutdown and Runlevels

- **1.106.1 Boot the system**
  Weight: 3

  Description: Candidates should be able to guide the system through the booting process. This includes giving commands to the boot loader and giving options to the kernel at boot time, and checking the events in the log files.

  Key files, *terms*, and utilities include:
  `/var/log/messages`
  `/etc/conf.modules or /etc/modules.conf`
  `dmesg`
  `LILO &  GRUB`

- **Boot sequence and Runlevels** (or init levels)
  Here are the steps that Linux goes through till it waits for user interaction:
  - BIOS initializes its devices
  - The Boot Loader on MBR of Floppy/Hard Disd/CDROM/.... is read and executed
    At this point the Boot Loader may allow user to enter Kernel options.
  - The Kernel and maybe `initrd` is felched from the Floppy/Hard disk/CDROM.
  - The Kernel initializes its hardware environment, using modules compiled in kernel.
  - The Kernel starts its first process: `init` (PID=1)
  - The Kernel tests the root(`/`) and other partitions as per `fstab` and mounts them.
  - The Kernel initializes more hardware environment, using `/etc/modules.conf` and some boot scripts.
  - `init` reads its configuration file `/etc/inittab` and acts accordingly.
    `/etc/inittab` contains the list of processes that `init` should start like: console gettys, default runlevel, run levels definitions, etc.
  - `init` starts the default run level scripts and passes the control on to the getty on terminal 1 for user login.
  - If xdm/kdm/gdm display manager is started as part of the default runlevel, then the started display manager takes over the control of the display for graphic user login.

- **Giving kernel options to bootloader:**
  Before starting the loading of the kernel it is normally possible to give kernel options on the command line of the bootloader. These options ranges from:
  SCSI adaptor addresses, root partition, vga terminal mode, default runlevel, etc.
  The list of options supported by the current kernel can be found in :
  `/usr/src/linux/Documentation/kernel-parameters.txt`.
  The kernel options given are always readable from the file : `/proc/cmdline`
  Example of options given to LILO ,GRUB, SYSLINUX or LOADLIN bootloaders:
  `LILO boot:` `linux aha152x=0x300,10,7`
  Tells that the Adaptec scsi adaptor ist at address `0x300` IRQ `10` and SCSI-ID `7`
  If kernel options need to be permanent, they can also be written in the bootloader's configuration file.
  LILO under the keyword: `append=`. eg.
  ```
  append=vga=791 hdc=ide-scsi splash=verbose acpi=off
  ```
  GRUB or in GRUB's configuration file on the kernel definition line: eg.
  ```
  kernel (hd0,2)/boot/vmlinuz.2.4.20 root=/dev/hda3 vga=791 splash=verbose
  ```
  Note: Options are separated by a space but continuous within the option.(see above)

- **File `/etc/modules.conf`** (or `/etc/conf.modules`)
  Kernels can be of 2 types:

Monolitic	All device drivers are integrated in the kernel.
Modular	Some of the device drivers are compiled a loadable modules.

  For modular kernels the modules can be loaded/unloaded dynamically or automatically. The parameters needed for defining the addresses, irq, dma, etc. for a module as well as their system alias names, can and should be written in the file: `/etc/modules.conf`(new name) or `/etc/conf.modules`(old name).
  The syntax of these parameters is explained more in details in the previous topic no. 1.105.1: eg.
  ```
  alias eth0                ne
  options                   ne io=0x300 irq=5
  alias block-major-58      lvm-mod
  post-install bttv         insmod tuner
  post-remove bttv          rmmod tuner
  ```
  etc.

- **Boot Log files:**
  From the very start the kernel saves its log messages in an internal buffer which is readable by issuing the command `dmesg.`
  After the `syslogd` Daemon logging system has been started, the standard file where most of the system messages including kernel messages are stored is called: `/var/log/messages`. The comand `tail -f /var/log/messages` allows to read the last 10 lines of the log file and keep refreshing it every second.

- **`/etc/lilo.conf` Parameters**
  The `/etc/lilo.conf` file contains options and kernel image information.
  Popular directives are:

`boot`	The name of the hard disk partition that contains the boot sector.
`image`	Refers to a specific kernel file.
`install`	The file installed as the new boot sector.
`label`	Provides a label, or name, for each `image`.
`map`	Directory where the `map` file is located.
`prompt`	Prompts the user for input (such as kernel parameters or runlevels) before booting and without a keystroke from the user.
`read-only`	The root filesystem should initially be mounted read-only.
`root`	Used following each `image`, this specifies the device that should be mounted as root.
`timeout`	The amount of time, in tenths of a second, the system waits for user input.

1.106.2 Change runlevels and shutdown or reboot system

Weight: 3

Description: Candidates should be able to manage the runlevel of the system. This objective includes changing to single user mode, shutdown or rebooting the system. Candidates should be able to alert users before switching runlevel, and properly terminate processes. This objective also includes setting the default runlevel.

Key files, *terms*, and utilities include:
`/etc/inittab` **`shutdown`**
`init`

- **Runlevels**
  A runlevel is a software configuration of the system which allows only a selected group of processes to exist.
  Runlevels are identified by: `0 1 2 3 4 5 6 S` and `s`

  Description:
  `0` = Halt
  `1` = Single (root) login
  `2` to `5` = Usually define some kind of multiuser state, including an X login screen depending on Linux distributions.
  `6` = Reboot
  `S` & `s` = Scripts to run before entering runlevel `1` (single login).

- **The `/etc/init.d` directory**
  - The `/etc/init.d` directory contains initialization scripts and links controlling the boot process for many Linux distributions:
    `rc.sysinit` The startup script launched by init at boot time.
    `rc.local` A script for local startup customizations, started automatically after the system is running.
    `rc` A script used to change runlevels.
    `init.d` Directory containing scripts to start and stop system services.
    `rc0.d` through `rc6.d`
    Directories containing symlinks to scripts in `/etc/init.d`.
  - **Names of the links** are `[K|S][`*`nn`*`][`*`init.d_name`*`]`:
    - `K` and `S` prefixes mean `kill` and `start`, respectively.
      The scripts names starting with `S` are run with the argument `start`, and the ones with `K` are run with the argument `stop`.
      Upon entering a new runlevel:
      First the `K` scripts are run if their equivalent `S` scripts had been started in the previous runlevel, and then the `S` scripts are run if they had not already been started in the previous runlevel. Therefore on each change of runlevel, the `rc` script checks the scripts of the previous and new runlevels to determine which of the `K` or `S` scripts of the new runlevel should be run.
    - *nn* is a sequence number controlling startup or shutdown order.
    - *`init.d_name`* is the name of the script being linked.

- **Displaying the current runlevel**
  The command runlevel displays the Previous ('N' if None) and the current runlevel:
  eg. `# runlevel`
  `N 3` **<-----** The previous runlevel was None (After Booting) and present: 3

- **Changing runlevel**
  The command: `telinit` *`newrunlevel`* is used to change the current runlevel. `/sbin/telinit` is linked to `/sbin/init`. This means one can also use `init` instead. It takes a one-character argument and signals `init` process to perform the appropriate action. The following arguments serve as directives to `telinit`:

0,1,2,3,4,5 or 6	tells `init` to switch to the specified run level.
a,b,c	tells `init` to process only those `/etc/inittab` file entries having runlevel a,b or c.
Q or q	tells `init` to re-examine the `/etc/inittab` file.
S or s	tells `init` to switch to single user mode.
U or u	tells `init` to re-execute itself (preserving the state). No re-examining of `/etc/inittab` file happens. Run level should be one of `Ss12345`, otherwise request would be silently ignored.

  `telinit` (or `init`) can also tell `init` process how long it should wait between sending processes the SIGTERM and SIGKILL signals when shutting down a runlevel service.
  The default is 5 seconds, but this can be changed with the `-t` *`sec`* option.

- **`/etc/inittab` file format**
  Each line starting with '#' is a comment.
  Each entry uses one line. Each entry's syntax is as follows:

  `id``:`*`runlevels`*`:`*`action`*`:`*`process`*

`id`	Is a unique sequence of 1-4 characters which identifies an entry in inittab. Note: For gettys or other login processes, the id field should be the tty suffix of the corresponding tty, e.g. 1 for `tty1`. Otherwise, the login accounting might not work correctly.
`runlevels`	Lists the runlevels for which the specified action should be taken. The runlevels field may contain multiple characters for different runlevels. For example,123 specifies that the process should be started in runlevels 1, 2, and 3.
`action`	Describes which action should be taken (see below).
`process`	Specifies the process (or command) to be executed.

  - <u>Most common `actions`:</u>

`respawn`	The process will be restarted whenever it terminates (e.g. `getty`).
`wait`	The process will be started once when the specified runlevel is entered and init will wait for its termination.
`once`	The process will be executed once when the specified runlevel is entered.
`boot`	The process will be executed during system boot. The runlevels field is ignored.
`bootwait`	The process will be executed during system boot, while init waits for its termination (e.g. `/etc/rc`). The runlevels field is ignored.
`off`	This does nothing.

`initdefault`

An initdefault entry specifies the runlevel which should be entered after system boot. If none exists, `init` will ask for a runlevel on the console. The process field is ignored.

`ctrlaltdel`

The process will be executed when init receives the SIGINT signal. This means that someone on the system console has pressed the CTRL-ALT-DEL key combination. Typically one wants to execute some sort of shutdown either to get into single-user level or to reboot the machine. Often used to reboot the machine in many distributions.

See `man inittab` for more info on other actions like:
`sysinit`, `powerwait`, `powerfail`, `powerokwait`, `powerfailnow`, `resume`, `kbrequest`, `ondemand`.

- **Shutting down the system properly.**
  Before the system is turned off, it needs to properly shut down every current runlevel service and unmount all the partitions. This is done with the commands:

```
init 0 or            | These 3 commands will shutdown the system.
shutdown -h now or   |
halt                 |

init6 or             | These 3 commands will reboot the system.
shutdown -r now or   | If /etc/inittab is set accordignly , pressing
reboot               | the key combination <Ctrl-Alt-Del> will also reboot the
                     | system.

shutdown -c          Cancels the already scheduled shutdown.
```

  Note: The `reboot`, `poweroff` and `suspend` are a symbolic links to `halt`.

- `shutdown` command:
  Syntax: `shutdown [`*`options`*`]` *`time`*

  Options:

`- c`	Cancels a shutdown
`- f`	Will not run fsck on the reboot
`- F`	This WILL run fsck on reboot
`- h`	Halts system after shutdown
`- k`	Sends warning / does not shutdown
`- n`	Shuts down without calling init
`- r`	When shutdown, will reboot
`- t {Seconds}`	Delay time after killing process (before init)

  Time format:

`now`	Well...NOW!
`+2m`	In 2 minutes
`+4`	In 4 minutes
`hh:mm`	At this time

  Command access rights:
  The file `/etc/shutdown.allow` may contain the user names (one per line) to which the permission is given to run the `shutdown` command.

Topic 107: Printing

- **1.107.2 Manage printers and print queues**
  Weight: 1

  Description: Candidates should be able to manage print queues and user print jobs. This objective includes monitoring print server and user print queues and troubleshooting general printing problems.

  Key files, *terms*, and utilities include:

  ```
  /etc/printcap
  lpc +options          lprm + options
  lpq +options          lp
  ```

- **How `lp` printing works:**

  So how does it fit together? The following things happen:

1. At boot time, **`lpd`** is run. It waits for connections and manages printer queues.
2. A user submits a job with the **`lpr`** command or, alternatively, with an `lpr` front-end like GPR, PDQ, etc. **`Lpr`** contacts **`lpd`** over the network (localhost or remote) and submits both the user's data file (containing the print data) and a control file (containing user options).
3. When the printer becomes available, the main **`lpd`** spawns a child **`lpd`** to handle the print job.
4. The child **`lpd`** executes the appropriate filter(s) (as specified in the `if` attribute in `/etc/printcap`) for this job and sends the resulting data on to the printer.

The `lp` system was originally designed when most printers were line printers - that is, people mostly printed plain ascii. By placing all sorts of magic in the `if` filter, modern printing needs can be met with **`lpd`** (well, more or less; many other systems like CUPS do a better job).

-
- **Print related commands list:**

`lpc`	Printer control
`lpd`	Print daemon that should be started. Normally as a runlevel service.
`lpq`	Print spool control . Display the print jobs in the print queue
`lpr`	The print spooler. Sends print jobs to `lpd`
`lprm`	Print job removal. Erases print jobs from the print queue
`pac`	Print account report generation

- **Principle of print queues under Linux:**
  The spool is a directory where print jobs get saved and then erased when finished.
  For each print job there is 2 files:

Control file	of which its name starts with `cf`. Contains the information about the print job.
Data file	of which its name starts with `df`. Contains the actual data to send to the printer.

  This directory is watched by a print queue Daemon and jobs are printed sequencially if the printer is available. If the printer is not available for a while, then these files will stay there until the printer is made available again and the jobs printed.

- **Definition of print queues in `/etc/printcap`.**
  Each created printer get a spool directory: `/var/spool/lpd/`*`printername.`*
  The permissions of this directory are: `drw--S---`
  The printer has also a queue control lock file: `lpd.lock.`*`printername`*`.printer`
  Each recognized local or network printer queue is having an entry in `/etc/printcap`.
  These entries will be explained more in details in section 1.107.4.
  eg.

  ```
  lp|hplaser|PS;r=600x600;q=medium;c=gray;p=a4;m=auto:\
  :sd=/var/spool/lpd/lp:\
  :lf=/var/spool/lpd/log:\
  ................
  ................
  ```

  Important here is:
  - The `lp` where `lp` means the default printer queue.
  - The `hplaser` is the normal name for the printer queue.
  - The `sd=/var/....` (**s**pool **d**irectory) defines the print queue directory.

- **Extra files**

`/etc/lpd.perm`	Permissions database. Afftect the behaviour of `lpd`, `lpc` and `lpq`. Controls allowances of local and remote print jobs.
`/etc/lpd.conf`	Extensive configuration file for the `lpd` Daemon.

- **Controlling the print queues:**
  Command syntax: `lpc {`*`command`*`} {`*`value`*`}`

`?`	Prints help about a command
`Abort`	Kill active print daemon
`Clean`	Removes unprintable files
`Disable`	Turn off printers queue
`Down`	Turn off printers queue
`Enable`	Turn on printers queue
`Help`	Prints help about a command
`Restart`	Shuts down current session - starts a new one
`Start`	Turns printing ON
`Status`	Gives a status of queue
`Stop`	Shuts off the spooling daemon
`Topq`	Moves jobs to top of queue
`Up`	Turns ON printer queue

If `lpc` is given without parameters then `lpc` gets into 'lpc command line' mode:
eg.(italics are the responses from `lpc`)

```
lpc
lpc> up lp
lp:
     printing enabled
     daemon started
```

- **Displaying print jobs:**
  Command syntax: `lpq [options] [job] [user]`
  Options:

`-P`*`printqueue`*	Name of print queue(*`printqueue`*) jobs to list. The default is the default print queue(`lp`).
`-l`	Requests a more verbose (long) reporting format.
`-a`	Reports jobs on all printers

  This command shows also the status and warnings of the print queue.

- **Deleting Print Jobs:**
  <u>Deleting print jobs</u>
  Command syntax: `lprm [options] job [user]`
  Options:

`-P`*`printqueue`*	Name of print queue(*`printqueue`*) jobs to delete. The default is the default print queue(`lp`).
`-`	Single dash(`-`)will remove all jobs.
`user`	(Optional). Deletes all jobs of a user. eg. `lprm - harry`

Note: Make sure you are familiar with the following:
- `lpq`, `lprm` and `lpc` commands and options
- The option `-P` is used in `lpq`, `lpr` and `lpc` to specify the printer's name
- `lpc`'s syntax can work off the command line or in interactive mode.
- `lpc`'s syntax:
  - it needs the printer(s) to be specified: `all` or *`printer`*
  - `enable` and `disable` controls the incoming jobs to the printing queue
  - `stop` and `start` control the sending of printing jobs to the printer and `lpd`'s child processes.
  - `up` and `down` controls all of the above.

- **1.107.3 Print files**
  Weight:1

  Description: Candidates should be able to manage print queues and manipulate print jobs. This objective includes adding and removing jobs from configured printer queues and converting text files to postscript for printing.

  Key files, *terms*, and utilities include:
  `lpr   lpq   mpage`

- **Controllling print queues.**

<u>Sending a print job:</u>
`lpr` submits files for printing. Files named on the command line are sent to the named printer (or the system default destination if no destination is specified).
If no files are listed on the command-line `lpr` reads the print file from the standard input. In fact lpr doesn't send the print job directly to the printer, its send it to the `lpd` daemon.

Command syntax: `lpr [`*`options`*`]` *`FileToPrint`*
This utility prints given files. For its printer destination, 2 Environment variables may be used: `LPDEST` or `PRINTER`.
Options:

`-P`*`printqueue`*	Name of print queue(*`printqueue`*) to send job to. The default is the default print queue(`lp`).
`-\#`*`n`*	Number(n) of copies to print (from 1 to 100).
`-K`*`n`*	Same as above `-\#n`
`-Q` *`spoolqueue`*	Selects a spool queue instead of default.
`-R` *`remoteaccnt`*	Identifies the remote account name when sending remote jobs.
`-w` *`width`*	Defines the width of the page in characters.(default=72)
`-h "`*`header`*`"`	Defines the page header to print instead of the default.
`-l lines`	Defines the number of lines per page.(default=66)
`-C string`	Replace system name on the burst page with string.
`-J name`	Replace the job name on the burst page with name. If omitted, uses the first file's name.
`-T title`	Use title as the title when using `pr`.
`-i [cols]`	Indent the output. Default is 8 columns. Specify number of columns to indent with the cols argument.
`-d`	Double the line spacing.
`-m`	Send mail to notify of completion.
`-b`	Does not print a banner or a header.
`-F`	Allows to specify one of the following print formats:
`-F`**`b`**	File has binary content and should be processed anyway.
`-F`**`d`**	Accept the file as being written by the `tex` editor.
`-F`**`n`**	Accept output from `troff`
`-F`**`t`**	Same as `-n`
`-F`**`p`**	Use `pr` to format the file before printing.
`-F`**`r`**	Deletes the file after spooling.
`-F`**`v`**	Assume a raster image.

see `man lpr` for more options.

- **Print engine lpd Daemon**:
  This Daemon process is normally started at boot time and watches the print queues for incoming printing jobs.
  Syntax for `lpd`: `lpd [`*`options`*`] [`*`port`*`]`
  Options:

- **Special file types converters for printing.**

`a2ps` Converts ASCII text files to Postscript format.
Default options results in:
- Printing 2 pages in one
- Each page is framed incl. filename, username and print date.

Options:

`-p printername`	Sends the output to a printer
`-o filename`	Saves the output to a file.
`-o -`	Sends the output to to `STDOUT` (standard output).
`-E`	Pretty-Printing for C code, Bash scripts, etc.

`enscript` Same functions as `a2ps` plus a few more including:
- Control of the output Pretty-Printing
- Can also output : HTML, ANSI and RTF
- Can output 1,2,4,or 8 pages per printed page.

`mpage` Reads a text or Postscript file and prints multiple pages in one page. The difference to other above tools is that it reads PostScript as well including graphics.

Note: Make sure that you understand the functions of the `lpr` and `lpd`. `lpr` sends print jobs to `lpd` and `lpd` send the jobs to the printer. Also get familiarized with the option used with `lpr`.

- **1.107.4 Install and configure local and remote printers**
  Weight: 1

  Candidate should be able to install a printer daemon, install and configure a print filter (e.g.: `apsfilter`, `magicfilter`). This objective includes making local and remote printers accessible for a Linux system, including postscript, non-postscript, and Samba printers.

  Key files, *terms*, and utilities include:
  `/etc/printcap`
  `/etc/apsfilter/*`
  `/var/lib/apsfilter/*/`
  `/etc/magicfilter/*/`
  `/var/spool/lpd/*/`
  `lpd`

- **Printer definitions file: `/etc/printcap`**
  This file can contain the definitions of local and remote printers printers.
  Entries in this file are in reality only one line per printer, the '\' at the end of each line simulates the single line like in bash scripts. Except for the name of the printer, which starts without '`:`', each item starts and ends with a '`:`'. See examples below:

  Keywords:

`lp|ljet4:`	`lp`(default) or `ljet4` are 2 aliases names of the printer.
`:af=`*`Filename`*`:`	**A**ccount **F**ile for the printer
`:if=`*`FilterName`*`:`	**I**nput **F**ilter Name
`:lp=`*`PrinterDevice`*`:`	**L**ocal **P**rinter device, such as `/dev/lp0`.
`:lf=`*`Log_File`*`:`	Error message **l**og **f**ile.
`:mx=`*`Max_Size`*`:`	**M**aximum size of a print job in blocks. 0 = no Limit
`:rm=`*`RemMachineName`*`:`	**R**emote **M**achine. Printer server name if remote used.
`:rp=`*`RemPrinter`*`:`	**R**emote **P**rinter Name on the remote machine.
`:sd=`*`Spool_directory`*`:`	**S**pool **D**irectory under `/var/spool/lpd`.
`:sh:`	**S**uppress **H**eader pages for a single printer definition.

- **`/etc/printcap` Examples:**

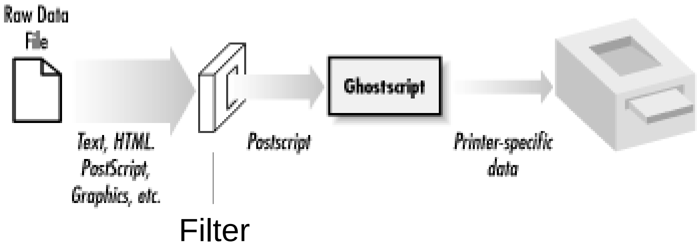
```
lp|hplaser:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lp:\
        :mx#0:\
        :lf=/var/spool/lp/hp-log:
```

Here (above) the printer device is local (`:lp=/dev/lp0:`).
The next example doesn't us the keyword `lp=`, instead it suses the `rm=` for remote:

```
lp1|remote printer on marvin:\
        :sd=/var/spool/lp1:\
        :rm=marvin.mydomain.net:\
        :rp=lp:\
        :sh:mx#0:
```

Note: Pay attention to the function of the `/etc/printcap` and its syntax including the variable's keywords.

- **Using Filters:**
  Filters are used to convert different document formats(Txt, HTML, Postscript, Graphics, etc) into Postscript format(ps). It is then passed-on to GhostScript which (if needed) converts it into a raster format(Printer-specific data) and then sent to the printer.



Filters looks at the 'Magic Code' at the beginnning of a document to determine the type. If it is already a PostScript document it will be sent to GhostScript without changes.
Two of the most popular filters used in Linux (that we need to know for the LPI-102) are: Apsfilter and Magicfilter.

- **apsfilter**
  Its configuration file: **/etc/apsfilter/apsfilterrc**
  Its configuration tool: **apsfilterconfig**
  Its location: **/usr/lib/apsfilter/filter/***

  This popular filter program accepts files in the PostScript, TeX DVI, ASCII, PCL, GIF, TIFF, Sun Raster files, FIG, PNM (pbmplus), HTML, and PDF formats.
  It takes care itself of sending its output(ps format) to GhostScript.
  Here are some examples of configuration of the printcap file using `apsfilter`:

```
ascii|lp1|ljet3d-letter-ascii-mono|ljet3d ascii mono:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/ljet3d-letter-ascii-mono:\
        :lf=/var/spool/lpd/ljet3d-letter-ascii-mono/log:\
        :af=/var/spool/lpd/ljet3d-letter-ascii-mono/acct:\
        :if=/usr/lib/apsfilter/filter/aps-ljet3d-letter-ascii-mono:\
        :mx#0:\
        :sh:

lp|lp2|ljet3d-letter-auto-mono|ljet3d auto mono:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/ljet3d-letter-auto-mono:\
        :lf=/var/spool/lpd/ljet3d-letter-auto-mono/log:\
        :af=/var/spool/lpd/ljet3d-letter-auto-mono/acct:\
        :if=/usr/lib/apsfilter/filter/aps-ljet3d-letter-auto-mono:\
        :mx#0:\
        :sh:

raw|lp3|ljet3d-letter-raw|ljet3d auto raw:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/ljet3d-raw:\
        :lf=/var/spool/lpd/ljet3d-raw/log:\
        :af=/var/spool/lpd/ljet3d-raw/acct:\
        :if=/usr/lib/apsfilter/filter/aps-ljet3d-letter-raw:\
        :mx#0:\
        :sh:
```

As you can see, the installation creates three printer definitions, each with multiple aliases and each using the same output device. This allows some degree of control over the filter, because the selection of the queue implies specific print parameters. The first definition (`ascii`) is intended to allow the user to force the printing of plain text even if the data is a PostScript program. The second entry (`lp`, the default) is the standard magic APSfilter, which tries to identify the data type itself. The last definition allows users to force APSfilter to send raw data directly to the printer with no intervention. This can be useful, for example, if you wish to print a PostScript file's programming instructions.

- **Magicfilter**
  The magicfilter is expandable and automatic. It loads the proper filter dynamically according to the Magic-Number located the beginning of the data to print.
  Since the printcap file doesn't support options, the entry if= in /etc/printcap should point to one of the preconfigured scripts for the appropriate printer type in the /etc/magicfilter/ directory. Each one of these scripts starts with the line:
  #!/usr/sbin/magicfilter

  Which will run (for setting magicfilter options) using the magicfilter script interpreter.
  The format of the scripts is:

	FileOffset	MagicNumber	WhatToDo
eg.	0	GIF87a	pipe /usr/bin/gif2pnm 2>/dev/null
	0	GIF89a	pipe /usr/bin/gif2pnm 2>/dev/null

Which tells magicfilter that if the FileToPrint starts with the characters GIF87a or GIF98a then convert the file to a PNM format before sending it to GhostScript.
To facilitate the process of configuring these scripts, a configuration script is provided with the magicfilter called magicfilterconfig.
Here is an example of an entry of the magicfilter in printcap:

```
lp|hplj4l|HP Laserjet 4L:\
        :lp=/dev/lp1:sd=/var/spool/lpd/hplj4l:\
        :sh:pw#80:pl#72:px#1440:mx#0:\
        :if=/etc/magicfilter/ljet4l-filter:\
        :af=/var/log/lp-acct:lf=/var/log/lp-errs:
```

The pw#, pl#, px# and mx# are settings of:

PageWidth(pw#) in characters:	80 Chars
PageLength(pl#) in lines:	72 Lines
PageWidth(px#) in pixels:	1440 Pixels
Maximum File Size(mx#):	Unlimited file size(0)

- **Linking to a remote Windows print server.**
  It is possible to point the printing destination to a printer share installed on a remote Windows or Samba Print Server. Since the entry of the InputFilter (if=) in printcap is used to start a particular program to handle the printing, a script using the smbclient program can be used to send the job to a SMB Print server. For this to work we need to install the package where smbclient is located. Here we call the script smbprint.
  eg.

```
lp2|remote-smbprinter:\
        :lp=/dev/null:sh:\
        :sd=/var/spool/lp2:\
        :if=/usr/local/sbin/smbprint:
```

This script (/usr/local/sbin/smbprint) must have the following entries:
- Host Name/IP of the print server
- Printername on the server
- Username and password on the printer server

Syntax: /usr/bin/smbclient *//Server/PrinterName Password* -U *UserName*
eg. /usr/bin/smbclient //prntsrv1/lpdj4 mot3tl6i -U barbara

- **`lpd` print Daemon**
  For this printing system to work a printing process must be started. In this case the lpd daemon should be started, normally at boot time.
  Once started its reads`/etc/printcap` and watches the print queues in `/var/spool/lpd/`*`printername/`*.
  The control od which host is allowed to use the printers is defined in: `/etc/hosts.lpd`

  `lpd` syntax: `lpd [-FV] [-D` *`dbglvl`*`] [-L` *`logfile`*`]`
  Options:

  ```
  -D dbglvl        - Sets debug level and flags eg.  -D10,remote=5
                     sets debug level to 10, remote flag = 5
  -F               - Run in foreground, log to STDERR
  -L logfile       - Append log information to logfile
  -V               - Show version info
  ```

  See `man lpd` for more info.

Topic 108: Documentation

- **1.108.1 Use and manage local system documentation**
  Weight: 4

  Candidates should be able to use and administer the man facility and the material in `/usr/share/doc/`. This objective includes finding relevant man pages, searching man page sections, finding commands and man pages related to them, and configuring access to man sources and the man system. It also includes using system documentation stored in `/usr/share/doc/` and determining what documentation to keep in `/usr/share/doc/`.

  Key files, *terms*, and utilities include:
  MANPATH **man**
  apropos
  whatis

- **Different methods fo getting help under Linux:**
  - **man, xman** eg. **man *topic*** (q for quit)
    xman -notopbox
  - **info, xinfo** eg. **info *topic*** (q for quit)
  - **/usr/share/doc/howto**
  - **/usr/share/doc/packages/**
  - **SuSE Help system**
  - Linux Documentation Project which is responsible for:
    - ManPages
    - FAQs
    - HOWTOs
    - Tutorials

Finding Help on topics(locally)	**Finding Help using Internet**
apropos *theme*	Linux Document Project at www.tfpd.org
whatis *command*	Internet Linux sites. eg. www.linux.org
man [*type*] *command*	Internet search engines eg. www.google.com/linux etc.
Docs in /usr/share/doc/*	
rpm -qi *packagename*	
FAQs documents (Frequently Asked Questions)	
HowTO's in /usr/share/doc/howto	

apropos *topic* Searches for the topic in the keywords and short descriptions, of the `whatis` (`/usr/share/man/whatis`) database and displays them all. Same result as: `man -k` *topic*

whatis *command* Searches the man pages keywords and presents the first short description of the command. The exact command must be found otherwise nothing is displayed. It displays the single line descrtiption found in the manpage. It first searches in the man page index and

then in its own database if the man page index file is not found.
Note: The `whatis` database is `/usr/man/whatis`.
or `/usr/share/man/whatis` or `/var/cache/man/whatis`
It is created/updated using the `makewhatis` command.

`whatis -r` ***`topic`***

Same as above `apropos` except the *`topic`* is searched only in the keywords and not in the short descriptions. It shows all the occurences found. Same result as **`man -f`** ***`topic`***
eg. **`whatis -r isdn`**

- **Man pages**
  Man pages (man command) are used to look-up certain commands and their use.
  Man pages are divided in 9 types(sections).

- **Syntax:** `man [`*`options`*`] [`*`type`*`]` *`commandname`*
  `type` (optional)
  1 Executable programs or shell commands
  2 System calls (functions provided by the kernel)
  3 Library calls (functions within program libraries)
  4 Special files (usually found in `/dev`)
  5 File formats, configuration files and conventions eg. `/etc/passwd`
  6 Games
  7 Miscellaneous (including macro packages and conventions), e.g. `man`(7), `groff`(7)
  8 System administration commands (usually only for `root`)
  9 Kernel routines [Non standard]
  - Note: When no type is given, the type search sequence until one is found is:
    `1,8,2,3,4,5,6,7,9`

  `commandname` : any command that has man pages. eg. `man ls`

- **Files, programs and variables:**

`/usr/bin/mandb`	Program to create or update the manpages caches.
`MANPATH`	Contains the PATHs where `mandb` looks while indexing pages.
`/usr/bin/manpath`	Program to display the paths searched for manpages.
`/etc/manpath.config`	`mandb` configuration file.
`/usr/share/man/index.`(`bt`|**`db`**|`dir`|`pag`) or	
	A traditional global index database cache.
`/var/cache/man/index.`(`bt`|**`db`**|`dir`|`pag`)	
	Alternate/FHS compliant global index database cache.

- **Locations of man pages:**

`/usr/man/*`	Old location of man pages
`/usr/share/man/*`	A global manual page hierarchy.
`/usr/local/man/*`	Extra user's commands man pages.
`/usr/local/share/man/*`	"" ""
`/usr/X11R6/man/*`	X11 Applications man pages
`/opt/gnome/man/*`	Gnome Desktop applications man pages
`/opt/kde3/man/*`	KDE Desktop applications man pages
`/usr/openwin/man/*`	Openwindows Desktop applications man pages
`/var/cache/man/*`	`catman` pages files and Index of manpages

Note: Most man pages in these directories are classified in subdirectories by their respective type (sections) eg. ..../man1/ ..../man2/
The man pages are normally in compressed(.gz) GROFF source format.
are decompressed automatically before the page is displayed.
The cat pages are preformatted Text man pages including the formatting characters.
Thex are normally saved in .../cat1 .../cat2 .... directories

- manpath

  The program manpath can be used to display the PATH used to search the man pages. If MANPATH is set, manpath will simply display its contents and issue a warning. This program is also used to determine the paths to search if the MANPATH variable is not set.

eg:
```
> manpath
manpath: warning: $MANPATH set, ignoring /etc/manpath.config /
usr/local/man:/usr/share/man:/usr/man:/usr/X11R6/man:/usr/openwin/man
```
If not, manpath will determine a suitable manual page hierarchy search path from the configuration file (/etc/manpath.config)and display the results.

- **Pager**

  The man pages use the **pager** (usually /bin/less) to display the page.
  It can be changes by changing the environment variable **PAGER**.
  eg. `export PAGER=/bin/more or export PAGER=/bin/nroff`
  or: `man -P"less -X"` *command*
  Uses less -X as Pager. This displays the man page but leaves the X-terminal content as-is when leaving man.

- **man command Examples:**

`man n command` Display the man page for the *command* in the section *n*
eg. **man 8 mount** (displays the mount administration command)
man 2 mount (displays the mount system call)

`man -a command` Display **all** the man page for the *command* . They are displayed one after the other, each one being terminated with 'q'.

`man -k command` Same as **apropos**. Displays all the man titles subjects relating to this topic. The *command* is searched in the keywords as well as in the short descriptions eg. **man -k isdn**

- **man pages filters and GUIs:**

GUIs:	tkman and xman
Filters:	rman

- **Filters Examples:**

- To convert a man page to HTML format of command ls
```
zcat $(whereis -m ls | cut -d" " -f2) | rman -n ls -f HTML \
> ls.1.html
```

- To show it in w3m browser instead of savingit as a file:
```
zcat $(whereis -m ls | cut -d" " -f2) | rman -n ls -f HTML \
| w3m -T text/html
```

- To convert a man page to PDF format of a command ip
```
zcat $(whereis -m ip | cut -d" " -f2) | groff -mandoc\
| ps2pdf - - > man.ls.1.pdf
```

- To show it in GhostView instead of saving it in a file:

```
zcat $(whereis -m ip | cut -d" " -f2) | groff -mandoc\
| ps2pdf - - | gv -
```

- To convert a man page in plain text format plain ASCII text version of the man page without excape characters or character formatting or colors etc.

```
man command | col -b
```

man command options:

```
-a, --all                    find all matching manual pages.
-d, --debug                  emit debugging messages.
-e, --extension              limit search to extension type `extension'.
-f, --whatis                 equivalent to whatis.
-k, --apropos                equivalent to apropos.
-w, --where, --location      print physical location of man page(s).
-l, --local-file             interpret `page' argument(s) as local filename(s).
-u, --update                 force a cache consistency check.
-r, --prompt string          provide the `less' pager with a prompt
-c, --catman                 used by catman to reformat out of date cat pages.
-7, --ascii                  display ASCII translation of certain latin1 chars.
-D, --default                reset all options to their default values.
-M, --manpath path           set search path for manual pages to `path'.
-P, --pager pager            use program `pager' to display output.
-S, --sections list          use colon separated section list.
-m, --systems system         search for man pages from other unix system(s).
-L, --locale locale          define the locale for this particular man search.
-V, --version                show version.
-h, --help                   show this usage message.
```

- **INFO Pages**

  Info pages are supposed to have more information than the man pages. Some individuals write a short description of their programs in the man pages and a longer one in the info pages.

  Syntax:

  `info [OPTIONS] [command] [subsection]`

Navigation through info pages

d	directory	***<space>***	- move forward, Page-Down-Key
h	help	***<Backspace>***	move backward, Page-Up-Key
b	begin of node	**u**	up node
e	end	**n**	next node
s	search * find	**p**	previous node
l	last text displayed		

- **catman**

  Creates or updates the pre-formatted manual pages.

  catman is used to create an up to date set of pre-formatted manual pages known as cat pages. Cat pages are generally much faster to display than the original manual pages, but require extra storage space. Normally the man pages are in GROFF format, normally man searches for a preformatted cat page, if not found it then must convert the man page into a format readable by and adjusted to the present terminal. When catman runs, it formats the man pages documents, making the displaying of large man pages quite faster but at the expenses of HD space. The decision to support cat pages is that of the local administrator, who must provide suitable directories to contain them. catman works with the variables MANSECT and MANPATH, if MANSECT is not set.

  Syntax:

  `catman [-dhV] [-M path] [section] ...`

- **1.108.2 Find Linux documentation on the Internet**
  Weight: 3

  Description: Candidates should be able to find and use Linux documentation. This objective includes using Linux documentation at sources such as the *Linux Documentation Project* (LDP), vendor and third-party websites, newsgroups, newsgroup archives, and mailing lists.

  Key files, *terms*, and utilities include:
  not applicable

- `www.linuxdoc.org`(old) or `www.tldp.org`(new)
  The Site for the Linux Documentation Projects. Handbooks, Books, HOWTOs, FAQs and lots more.
- `rute.sourceforge.net`
  A complete course on Linux.
- `www.linux.org/docs/`
  The official Linux web site with more Documentation and links to other Linux web sites.

Newsgroups

- `comp.os.linux.advocacy`
  General discussions about the advantages of using Linux vs. other OS.
- `comp.os.linux.announce`
  Commented Linux news
- `comp.os.linux.answers`
  Comented sending of Linux FAQ's. HOWTO's, and README's.
- `comp.os.linux.apps`
  General discussions about Linux Applications.
- `comp.os.linux.development.apps`
  Discussions about programming and porting applications for Linux.
- `comp.os.linux.development.system`
  Discussions about the Linux kernel, device drivers und loadable modules.
- `comp.os.linux.hardware`
  General discussions about questions on Linux hardware compatibility.
- `comp.os.linux.misc`
  Different themes about Linux which are not found in other newsgroups.
- `comp.os.linux.networking`
  General discussions about questions on networking and communications.
- `comp.os.linux.setup`
  General discussions about Linux installation und System Administration.
- `comp.os.linux.x`
  Discussions about The X Window System under Linux.
- alt.os.linux
  Generelle Diskussion zum Thema Linux.

Newsgroup Archive

- `www.dejanews.com`
  Archives of all Newsgroups. Google has since then overtaken this function.

Mailinglists

The following mailing lists are running off a central Majordomo server. To subscribe to one of these mailing list, send an email to `majordomo@vger.kernel.org` with the mail text body being:

```
  subscribe ListName
```

`ListName` = One of the mailing lists below. The text in the subject area is ignored.

- `linux-8086`
- `linux-admin`
- `linux-alpha`
- `linux-apps`
- `linux-arm`
- `linux-bbs`
- `linux-c-programming`
- `linux-config`
- `linux-console`
- `linux-diald`
- `linux-doc`
- `linux-fido`
- `linux-fsf`
- `linux-ftp`
- `linux-gcc`
- `linux-gcc-digest`
- `linux-hams`
- `linux-hppa`
- `linux-ibcs2`
- `linux-ipx`
- `linux-isdn`
- `linux-japanese`
- `linux-kernel`
- `linux-kernel`
- `linux-kernel-digest`
- `linux-kernel-patch`
- `linux-laptop`
- `linux-linuxss`
- `linux-lugnuts`
- `linux-mca`
- `linux-mips`
- `linux-msdos`
- `linux-msdos-digest`
- `linux-msdow-devel`
- `linux-net`
- `linux-new-lists`
- `linux-newbie`
- `linux-newbiew`
- `linux-nys`
- `linux-oasg`
- `linux-oi`
- `linux-opengl`
- `linux-pkg`
- `linux-ppp`
- `linux-pro`
- `linux-qag`

- linux-raid
- linux-scsi
- linux-serial
- linux-seyon
- linux-smp
- linux-sound
- linux-standards
- linux-svgalib
- linux-tape
- linux-term
- linux-training@lists.iphil.net
- linux-userfs
- linux-word
- linux-x11
- linux-x25
- sparclinux
- ultralinux

All of the above themes can be subscribed to and more is available from the:

Linux Mailing Lists
`http://oslab.snu.ac.kr/%7Edjshin/linux/mail-list/index.shtml`

- **1.108.5 Notify users on system-related issues**
  Weight: 1

  Description: Candidates should be able to notify the users about current issues related to the system. This objective includes automating the communication process, e.g. through logon messages.

  Key files, *terms*, and utilities include:
  `/etc/issue`
  `/etc/issue.net`
  `/etc/motd`

- **Login Sequence:**
  When the system boots-up, right at the end of its default runlevel, init starts the program mingetty for each virtual console defined in `/etc/inittab`.
  So here is the sequence of events:
  - The `mingetty` (getty process) displays the content of the file `/etc/issue`.
  - Then its displays the prompt *`HostName`* `login`: and waits for the username.
  - When the username is given and the user presses `<enter>` `mingetty` replaces itself with the program `login` and `login` gets the username from mingetty and waits for the password from the user.
  - If this authentication succeeds, `login` process then starts the shell(`bash`).
  - `bash` reads its configuration scripts (`/etc/profile` etc.) and displays the content of the file `/etc/motd` (**m**essage **o**f **t**he **d**ay), then displays its prompt.

  So: `mingetty` -----------------------------------> `login` ----------------------------> `bash`
  Shows `/etc/issue` waits for username   Waits for passwd   Shows `/etc/motd` and prompt

- **The file `/etc/issue`:**
  This file contains the message sent to the users consoles(by `mingetty`) before he logs in. Its content is normal text including special escaped characters which will be converted to their meaning before the file is displayed.
  These escaped characters are

`\b`	Baudrate of terminal connection (only for serial terminal connection)
`\d`	Today's date
`\s`	Operating System Name (eg. 'Linux')
`\l`	Name of the currrent TTY
`\m`	System Architecture(eg. i486)
`\n`	Hostname
`\o`	Domain name
`\r`	Release number of the Kernel
`\t`	Present time.
`\u`	Delay since login of the user
`\U`	The word `User(s)` and the delay since login of the user.
\v	Kernel version (Buils Date)

eg. The content of /etc/issue can look like this:

```
  Welcome at \n.\o (\s \m \r)
```

This would display the following:

```
  Welcome at marvin.mydomain.org (Linux i686 2.4.18)
```

Topic 109: Shells, Scripting, Programming and Compiling

- **1.109.1 Customize and use the shell environment**
  Weight: 5

  Description: Candidate should be able to customize shell environments to meet users' needs. This objective includes setting environment variables (e.g. PATH) at login or when spawning a new shell. It also includes writing bash functions for frequently used sequences of commands.

  Key files, *terms*, and utilities include:

```
~/.bash_profile     ~/.bashrc
~/.bash_login       ~/.bash_logout
~/.profile          ~/.inputrc
```

  `function` (Bash built-in command)
  `set` (Bash built-in command)
  `unset` (Bash built-in command)
  `env`
  `export`

Login vs. Non-login shell

Login shell:	Shell started either by `login` prgm or `bash -l` or or `su -`
Non-Login shell:	Shell started in any other way other than `login` or `su -`

Reason for 2 types of shells:
The login shell reads a series of configuration file as it is started. The non-login shells enherit settings (Environment variables) from the parent program which started it.

Variable enheritance
- If a shell variable is declared inside a shell, it is called a 'shell variable' and will only be enherited by its child/children processes if the variable tag for export is ON.(Environment variables).
- If an exported variable is changed in the child process, the value of the variable in the parent shell is not changed, but this new value will be exported to to any subchild processes.
- All shell variables tagged for export keep their export settings in the child process.
- If a shell script is called from within a shell a new child non-login shell is started.
- If a shell script is started with the '`.`' command within a shell, then the script is run within that current shell. eg. `. /home/joe/bin/myscript`
  Warning:
  If the called script runs the command `exit`, the current shell will be terminated!

Interactive and NON-Interactive shells

Interactive shell:	Shell where the user types commands.
Non-Interactive shell:	Shell started by calling a shell script or by the command: `sh -c '`*`command...`*`'`

Sequence of events when bash starts

Files read	**Interactive-login Bash** (eg. `bash --login` or `su - `*`username`* or from `login`)
`/etc/profile`	Executed first from interactive login shell. It contains system-wide environment settings. If existent, it is read in and executed before `$HOME/.profile`.
`~/.bash_profile`	Individual users shell settings. If exist is executed after `/etc/profile.`
`~/.bash_login`	Executed if `~/.bash_profile` doesn't exist.
`~/.profile`	Executed if `~/.bash_login` or `~/.bash_profile` doesn't exist.
	Interactive NON-Login Bash (eg. `su `*`username`* or `bash -c `*`command`*)
`~/.bashrc`	The only script executed when started. And enherits from parent bash environment.
	NON-Interactive NON-Login Bash(forked when scripts are run)
	No above scripts are executed but enherits env. from parent.
`BASH_ENV`	Reads file in the variable `BASH_ENV`.
`ENV`	Reads file in the variable `ENV` if `BASH_ENV` doesn't exist.
	Extra files
`/etc/inputrc`	System bash line editing(`readline`) configuration file
`~/.inputrc`	Individual bash line editing(`readline`) configuration file
`~/.bash_logout`	Executed (if exists) when a login shell exits.

Commands for shell/environment variables

`Variablename=value`

Assign a value to a set(existing) or non-set variable.

`export `*`Variablename`* or
`declare -x `*`Variablename`*

Sets the export tag ON for an existing shell var.

`export `*`Variablename=value`* or
`declare -x `*`Variablename=value`*

Assign a value to a set(existing) or non-set variable and sets its export tag ON, all in one command.

`env` Displays all the environment variables(export tag ON)

`export` Same as env command except the display format is different.
eg. `declare -x PAGER="less"`

Aliasses

- Aliases are normally used to create command shortcuts(short names).
- <u>Aliasses are NOT exportable:</u> not passed-on to subshells or child process.
- Aliasses anre not recognized in scripts.
- An alias can call another alias within the command.
  eg. `alias li="ls -l"; alias al="li -a"` : `al` calls the alias '`li`'
- Parameters added to `alias` will be added at the end of the real command.
- The parameters variables ($1, $2, $3 ...etc) cannot be used within aliasses.
- Aliases are often defined in a file run within a script
  (eg. `~/.bashrc` or `~/.profile`) with the dot '`.`' command.
- <u>Alias commands:</u>

`alias`	Displays all the current shell aliasses.
`alias` *`AliasName`*`="`*`command`*`(s)..."`	Sets a new alias value
eg. `alias cp="cp -i"`	replaces the original command `cp` with `cp -i` for interactive copying.(asks before overwriting files)
`unalias` *`AliasName`*	Unsets(deletes) the `alias`.

Functions

- They are normally used like fast local mini-scripts within a shell which need to be called more than once within the interactive shell or script.

- Variables can be passed-on to functions and will be recognized as `$1 $2 $3` etc. In fact the following variables are local within a function:

`$1 - $9`	Positional parameters
`$#`	Number of positional parameters
`$*`	`"$1 $2 $3 ..."`
`$@`	`"$1" "$2" "$3" ...`

- The Positional parameter `$0` and all other variables stay global within the shell unless the command `local` *`VariableName`* is given within the function. Within a function, the variable `FUNCNAME` is used instead of the `$0.`

- Global shell or exported variables can be changed within the function.

- Functions do not return variables except for the `return` number, eg. `return 5`
  `return` command will also <u>terminate the function</u> immediately.
  The `return` number can then be read as a normal *exit number* using the `$?`.

- In scripts normally functions are included at the top so that they are read in first.

- Environment functions can be put into a file and read in with the `.` command.

- Functions may be recursive. No limit is imposed on the number of recursive calls.

- Functions can be exported, using the command: `export -f` *`FunctionName`*

- Function syntax:

```
   FunctionName () {
    command ;
    command ;

   }
```

or

```
function FunctionName () {
  command ;
  command ;

 }
```

- The command: `unset -f` *`FunctionName`* Deletes an existing function.

- **Command search priority.**

  When a command is run, bash tries to find the command in the following sequence:
  - Aliasses
  - Functions
  - Builtin commands
  - PATH

  the first command found is the one which is run.
  To force using a builtin command instead of an alias or a function (in the case the same command name exists as alias or function), use the command `builtdin`.
  eg. `builtin cat /etc/fstab`

- **`set` and `unset` commands**

  `set`

  Syntax: `set [--abefhkmnptuvxBCHP] [-o` *`option`*`] [arg ...]`
  The `set` command is used to:
  - Set bash operating attributes(using options)
  - To assign values to positional parameters: eg.

  ```
  set -a
  ```
  Automatically mark variables and functions which are modified or created for export to the environment of subsequent commands.

  ```
  set aaa bbb ccc
      $1  $2  $3
  ```
  Assigns the value `aaa` to `$1`, `bbb` to `$2` and `ccc` to `$3`.

  `unset`

  Syntax: `unset [-fv] [name ...]`

  For each name, remove the corresponding variable or function.
  Each unset variable or function is removed from the environment passed to subsequent commands. If any of `RANDOM`, `SECONDS`, `LINENO`, `HISTCMD`, `FUNCNAME`, `GROUPS`, `DIRSTACK` are unset, they lose their special properties, even if they are subsequently reset.
  The exit status is true unless a name does not exist or is readonly.

  `-v` If no options are supplied, or the `-v` option is given, each name refers to a shell variable. Read-only variables may not be unset.

  `-f` Each name refers to a shell function, and the function definition is removed.

  eg. `unset DISPLAY` : Deletes the variable `DISPLAY`
  `unset -f startx` : Deletes the function `startx`

- **1.109.2 Customize or write simple scripts**
  Weight: 3

  Description: Candidate should be able to customize existing scripts, or write simple new (ba)sh scripts. This objective includes using standard sh syntax (loops, tests), using command substitution, testing command return values, testing of file status, and conditional mailing to the superuser. This objective also includes making sure the correct interpreter is called on the first (`#!`) line of scripts. This objective also includes managing location, ownership, execution and suid-rights of scripts.

  Key files, *terms*, and utilities include:

```
while     test
for       chmod
```

What is a shell script?

A shell script is a text file that tells the shell what to do.
It contains the name of the program that is used as the interpreter for the rest of the content of the script.
The line starting with `#!ProgramPath+Name` (normally the first line) designate this interpreter to be used. eg.

```
#!/bin/bash      or
#!/bin/sh        or
#!/usr/bin/perl -w
```

In reality when the system is asked to start a script, the line with the #! is read and the appropriate script interpreter is started which in turns reads the script and executes its commands included in it.

Conditions for running a script:

- the script file must be runnable by the user running it (`chmod` ....)
- The interpreter must be where it says it is: the default is to call `bash`.

Language used in shell script

The language depends on the script interpreter used. eg. `#!/usr/bin/perl -w`
`bash` has its own syntax which can be used interactively or in a script.

Passing parameters to a script

Scripts can be given up to 9 positional parameters(for all interpreters) or up to 99 parameters with bash.
Inside the script each parameter will be identidied as $1 to $9 or ${10} to ${99}

```
eg. scriptname param1 param2 param3 param4 param5 ..... param57.....
      $0          $1     $2     $3     $4     $5      ${57} .....
```

Other containers of positional parameters:
Some special parameters are automatically set by the Bourne shell, and usually cannot be directly set or modified.
The $n can be modified by the command `set` *aaa bbb ccc*... inside the script.

```
                                   $1  $2  $3
```

Special Parameters

`$`*n*	Positional parameter *n* max. n=9 (`$0` is the name the shell script)
`${nn}`	Positional parameter *nn* (for *nn*>9)
`$#`	Number of positional parameters (not including the scriptprgm)
`$@, $*`	All positional parameters
`"$@"`	Same as `"$`**`1`**`" "$`**`2`**`"` . . . `"$`*n*`"`
`"$*"`	Same as `"$`**`1`***c*`$`**`2`***c* `. . . $`*n*`"` *c* = content of **`$IFS`** (default is space)

`$?`	Exit status of the last command
`$$`	Process ID of the current shell
`$-`	Current options in effect
`$!`	Process ID of the last background command
`$is`	Name of the curent shell (in this case 'bash')

The `shift` command:
The command shift moves the assignment of the positional parameters to the left.
eg. `script1 aaa bbb ccc ddd`
(*inside the script*)

```
> echo $1 $2 $3 -------> result aaa  bbb  ccc
                                $1   $2   $3
> shift
> echo $1 $2 $3 -------> result bbb  ccc
                                $1   $2   $3
```

The command `set` and `unset`:
The comand `unset` is normally used for unsetting values of variables, and the command set for assigning values to positional parameters from inside a script. Very usefull if for example a script has been started without positional parameters and after verifying this the script assigns dafault values to them.
eg. `set aa bb cc dd` willl assigns aa to $1, bb to $2, cc to $3 and dd to $4'
The commmand set is also useful for changing properties of `bash`'s behaviour.

The `if` conditional branching directive:
The if allows to execute certain commands only if certain contitions are met.
Syntax: (see also later in this topic the section 'CONDITIONAL EXPRESSIONS')

```
if condition_is_true ; then
     run_these_commands
     .................
elseif condition_is_true ; then
     (if above conditions are not met and this one is met then:
     run_these_commands
     .................
else (otherwise if all above conditions are not met then:)
     run_these_commands_instead
     .................
fi (end of if  directive block)
     condition_is_true can be of the following type:
     - Testing the status of files or directories.
          eg.   if test -e /etc/fstab ; then
          or    if [ -e /etc/fstab ] ; then
     - Command or script exit code.
          eg.   if (ifconfig | grep 'ppp0') ; then
     - The content of a variable correspond to a certain value:
          eg.   if  $1 ; then        : true if $1 has a value in it
          or    if [ "$net" = "eth0" ] ; then (string testing)
          or    if test "$#" -eq 5 ] ; then (integer testing)
```

Below is a list of the most used conditional directives:

The case conditional branching directive:

The case is normally used for conditionnally branching to one of multiple choices dependant on the content of a variable.
Syntax:

```
     case Variable in
          choice1)   commands to run
                     ......
               ;;
          choice2)   commands to run
                     ......
               ;;
          choice3)   commands to run
                     ......
               ;;
               *)    commands to run if none of the above answers
                     are suitable.
                     ......
               ;;
     esac (end of case directive block)
```

Looping in scripts

Whenever a sequence of commands need to be repeated as offen as possible a for a while until a condition is met then we normally use looping directives.

The while conditional loop directive.

The while directive keeps looping and running the commands in its block for as long as its condition(s) (defined in the while statement) is/are met.
Syntax:

```
     while condition_is_true ; do
          run_these_commands
          ................
     done (end of while directive block)
```

Note: While is often used to ask the user for a keyboard entry of some sort and if the response is not adequate then the request is repeated until the proper information is entered. The while loop is then exited and the program further resumes its execution.

The until conditional loop directive:

The until loop works exactly the same way as the while loop except that the logic is the opposite: The loop continues until condition(s) is/are met.
Syntax:

```
     until condition_is_true ; do
          run_these_commands
          ................
     done (end of until directive block)
```

The `for` loop directive

The `for` directive allows to loop through a set of commands for as many times as there are items in a given list. Each time the loop runs through, the content of a specific variable becomes value of the current item in the given list.

Syntax:

```
     for variable in list ; do
          run_these_commands
          ................
     done (end of for directive block)
```

`variable` = the variable name which will have its content become the current item on each loop round in the given list (*`list`*)

The *`list`* can also be a variable which contains a list of items.

eg.

```
     for item in ~/file1 ~/file2 ~/file3 ; do
          echo "----------- Content of $item ----------"
          cat $item >> ~/allfiles
     done
```

Shell functions:

Shell functions allows to repeat a series of commands multiple times within the script without having to copy the commands each time.

Parameters can be passed to functions via positional parameters.

The positional paramaters($1, $2, $3 ...), which will become local to the function.

They use the same syntax as for a script except that the first (`$0`) stays global.

For the same purpose as the `$0` the variable `FUNCNAME` is used instead.

Special variables like `$#`, `$*`, `$@`, are also local within the function.

All other variables are global to the script and can be modified by the functions.

The command `return` *`x`* (*x*=return code) can be used as a function exit function command and as to assign a function return code.

Syntax:

```
FunctionName () {
 command ;
 command ;
}
```

or

```
function FunctionName () {
  command ;
  command ;
}
```

See functions in the previous section (1.109.1 Customize and use the shell environment) for more details on shell Functions.

Exit codes and the variable $?

All programs including scripts, when ending their process, modifies the exit code which helps determining the success or the failure of the program or the script. This exit code can then be read via the special variable `$?` and be used to make decisions further in the script. Generally the exit code of '`0`' means success and any other code `1-255` means some sort of failure. It is also often refered as the error code.

The && and || conditional branching

The exit code can be used to execute another command (only one) depending upon its success or its failure. The double ampersand '&&' is used to designate the command to run if the exit code is success(0) and the double pipe '||' to designate the command to run if the exit code is not a success (1-255).
eg.

```
 ifconfig ppp0 && echo "pppd running" || echo "pppd not running"
```

If the command `ifconfig ppp0` succeeds then
the command `echo "pppd running"` will be executed(&&)
otherwise the command `echo "pppd not running"` will be executed(||).

Mailing messages to root from a script.

Sometimes it is useful to mail a message to `root` or to anybody announcing some annomalies or success in the running of an automated srcipt. The program normally used is '`mail`'. See `man mail` for getting all the options it uses.
Syntax1:

```
     mail -s "subject" destination_mail_address "message.."
```

Syntax2:

```
     program | mail -s "subject" destination_mail_address
```

Syntax3:

```
     mail -s "subject" destination_mail_address <<EOM
          message body.......
          EOM
```

eg. `df | mail -s "HD Space on $(date)" root`
Mails the result of the command `df` to the local root user.

Location and security for bash scripts

Normally unless they are only for user's personal use, the scripts for administration are stored in the PATH which is either `/usr/local/bin` or `/root/bin`. Their normal access rights are 755(rwx r-x r-x) or for more protection by preventing any other user than `root` to run it: 700(rwx --- ---). Although the SUID set on scripts doesn't have any effect on scripts, very old version of linux may be affeted by SUID being set.

CONDITIONAL EXPRESSIONS

The **`test`** and `[...]` commands are used to evaluate conditional expressions with file attributes, strings, and integers. The basic format is:

```
    test expression
       or
       [ expression ]
```

Where *expression* is the condition you are evaluating. There must be whitespace after the opening bracket, and before the closing bracket. Whitespace must also separate the expression arguments and operators. If the expression evaluates to true, then a zero exit status is returned, otherwise the expression evaluates to false and a non-zero exit status is returned.

Test File Operators

-a *file* True if file exists.

-b *file* True if file exists and is a block special file.

-c *file* True if file exists and is a character special file.

-d *file* True if file exists and is a directory.

-e *file* True if file exists.

-f *file* True if file exists and is a regular file.

-g *file* True if file exists and is set-group-id.

-h *file* True if file exists and is a symbolic link.

-k *file* True if file exists and its ``sticky'' bit is set.

-p *file* True if file exists and is a named pipe (FIFO).

-r *file* True if file exists and is readable.

-s *file* True if file exists and has a size greater than zero.

-t *fd* True if file descriptor *fd* is open and refers to a terminal.

-u *file* True if file exists and its SUID bit is set.

-w *file* True if file exists and is writable.

-x *file* True if file exists and is executable.

-O *file* True if file exists and is owned by the effective UID.

-G *file* True if file exists and is owned by the effective GID.

-L *file* True if file exists and is a symbolic link.

-S *file* True if file exists and is a socket.

-N *file* True if file exists and has been modified since it was last read.

file1 **-nt** *file2* True if *file1* is newer (according to modification date) than *file2*, or if *file1* exists and *file2* does not.

file1 **-ot** *file2* True if file1 is older than *file2*, or if *file2* exists and *file1* does not.

file1 **-ef** *file2* True if *file1* and *file2* refer to the same device and inode numbers.

-o *optname* True if shell option *optname* is enabled.
See the list of options under the description of the -o option to the set builtin below.

Test String Operators

-n *string* Tue if length of *string* is not zero

-z *string* Tue if length of *string* is zero

string Tue if *string* is not set to null

string1 = *string2*	Tue if *string1* is equal to *string2*
string1 == *string2*	“ “ “ “ “ “ “ “
string1 != *string2*	Tue if *string1* is not equal to *string2*
string1 < *string2*	True if *string1* sorts before *string2* lexicographically in the current locale.
string1 > *string2*	True if *string1* sorts after *string2* lexicographically in the current locale.
string = *pattern*	True if *string* matches *pattern*
string != *pattern*	True if *string* does not match *pattern*

Test Integer Operators

exp1 **-eq** *exp2*	True if *exp1* is equal to *exp2*	eg. `[ "$#" -eq 4 ]`
exp1 **-ne** *exp2*	True if *exp1* is not equal to *exp2*	eg. `test "$#" -ne 3`
exp1 **-le** *exp2*	True if *exp1* is less than or equal to *exp2*	
exp1 **-lt** *exp2*	True if *exp1* is less than *exp2*	
exp1 **-ge** *exp2*	True if *exp1* is greater than or equal to *exp2*	
exp1 **-gt** *exp2*	True if *exp1* is greater than *exp2*	

Other test Operators

! *exp*	True if the given expression is false eg. `[ ! -r /etc/motd ]`
exp1 **-a** *exp2*	True if both *exp1* and *exp2* evaluate to true (see example below)
exp1 **-o** *exp2*	True if either *exp1* or *exp2* evaluate to true
\(*exp* \)	True if *exp* is true; used to group expressions (\ used to escape parentheses) Use space

eg :

```
[ "$A" = "$B" -a \( "$C" = "$D" -a "$E" = "$F" \) ]
 ^              ^  ^                           ^  ^
```

Note: always use a space between the `[ ] \( \)` and the expressions like seen in the above example pointed by '^'.

Example of logical AND of commands

```
if ( cat /etc/motds &>/dev/null && cat /etc/fstabs
&>/dev/null ) ; then echo "all OK" ; fi
```

Topic 111: Administrative Tasks

- **1.111.1 - Manage users and group accounts and related system files**
  Weight: 4

  Description: Candidate should be able to add, remove, suspend and change user accounts. Tasks include to add and remove groups, to change user/group info in passwd/group databases. The objective also includes creating special purpose and limited accounts.

- **Key files, *terms*, and utilities include:**
  `/etc/passwd`
  `/etc/shadow`
  `/etc/group`
  `/etc/gshadow`
  `useradd`
  `usermod`
  `userdel`
  `passwd`
  `chage`
  `groupadd`
  `groupmod`
  `groupdel`
  `gpasswd`
  `pwconv`
  `pwunconv`
  `grpconv`
  `grpunconv`

- **The `/etc/passwd` file purpose and format**
  This file contains the users account info. One per line. Fields are separated by '`:`'.
  File Format:

```
 username : x : userID : GroupID : UserInfo : HomeDir : Shell
    1       2     3        4          5          6         7
```

  Field 2= Password, x=reference to `/etc/shadow`, empty=no password, `*`or `!`=no loging possible

- **The `/etc/shadow` file purpose and format**
  If the shadow password system is installed, this file contains the encrypted passwords for each user and their expiry parameters.

  Line fields: The fields are separeted by the char. '`:`'.
  Here are the fields' sequence:

  1. User login name
  2. Encrypted password **:** (empty=no Passw, *****=no login possible)
  3. Days since Jan 1, 1970 password was last changed **:** (never empty)
  4. Days until change allowed **:** (0=always allowed to c hange)
  5. Days before change required **:** (Normal is 10000 days)
  6. Days warning for expiration **:** (empty=no warning)
  7. Days before account inactive **:** (empty= never inactive)
  8. Days since Jan 1,1970 when account will be disabled **:**
     (empty = will never be disabled)
  9. Reserved for future use

- **User accounts administration.**
  The user accounts are located in the file `/etc/passwd`, their encrypted passwords are in `/etc/shadow` (if the shadow password system is installed). When a new user account is created (using `useradd`) the default template (`-m` option) used to create the user's personal and work directory is `/etc/skel`.

  Users admin commands:

  `useradd [`*`OPTIONS`*`]` ***`username`*** Adds a user to the system

  options

  `-b` *`default_home_directory_path`*
  (Note:the username will be added to the path

  `-d` *`default_home_directory_path`*
  (Note: *username* will NOT be added to the path)

  `-e` *`default_expire_date`*
  The date on which the user account is disabled.

  `-f` *`default_inactive`* The number of days after a password has expired before the account will be disabled.

  `-g` *`default_group`* The group name or GID for a new user's main group.

  `-s` *`default_shell`* The name and location of the new user's login shell.

  `-m` Copies the directory `/etc/skel` to user home directory

  `-k` *`template_dir`* Combined with `-m` will copy use the *`template_dir`* instead of `/etc/skel`.

  Note: When certain defaults are not given via options then they are taken from the file **`/etc/default/useradd`** file. These default parameters can also be seen using : `useradd -D`

  `/etc/login.defs` This file is for setting the extra defaults after login.

  `usermod` *`username`* Modifies the existing user's login parameters

  ```
  [-c comment]       [-d home_dir [ -m]]
  [-e expire_date]   [-f inactive_time]
  [-g initial_group] [-G group[,...]]
  [-l login_name]    [-p passwd]
  [-s shell]         [-u uid [ -o]] [-L|-U]
  ```

  `userdel` *`username`* Deletes a user from the system
  `-r` Deletes the user's home directory as well !!!

  `passwd` `[`*`username`*`]` Changes the password of user.
  Allowed characters in password are:
  `# * , . ; : _ - + ! $ % & / | ? { [ ( ) ] }`

  `chage` `[`*`options`*`]` *`username`*
  Used to list (`-l`) or th change the user's password expiry parameters. Options:

  ```
  [-D binddn]     [-P path]     [-m mindays]
  [-M maxdays]    [-d lastday]  [-I inactive]
  [-E expiredate] [-W warndays]
  ```

  `newusers` *`Filename`* Update and create new users in batch mode.

  `chpasswd` *`UserSPaSSFILE`*
  Modifies the password of multiple users in batch mode

Extra and login user related commands:

`id -ng [`*`username`*`]`	Shows the Present Effective group of a user.
`id -nG [`*`username`*`]`	Shows all the groups the present user belongs to.
`groups` `[`*`username`*`]`	"" "" "" "" "" ""
`id -nu [`*`username`*`]`	Shows the current username of a user.
`echo $USER`	"" "" "" ""
`id -u`	"" "" "" user ID of a user.
`users`	users presently logged in locally (short format)
`who`	"" "" " "" "" (long format)
`w`	"" "" " locally (long format)
`finger [-l `*`username`*`]`	"" "" " locally or remotely (long format) (Information in `[room_no]` below will not be shown)

`chfn [`*`options`*`]`	Changes the users info(field 5) in the `/etc/passwd`.
	(for scripting purposes) Options:
	`[-f full_name] [-r room_no] [-w work_ph]`
	`[-h home_ph] [-o other] [user]`
	Each field will separated with comas (`,`)
	Characters NOT allowed are: `,` `:` `=` or Ctrl chars.

Note: Information of the user can be displayed with the command:

```
finger -l username
```

`lastlog`	Shows the last logins that happened since the logfile `/var/log/lastlog` (binary format!!) was created . The list includes booting and shutdowns and loging in previous days. [-u *`username`*] [`-t` *`days_before`*]
`last`	Displays all the proper logins that happened since the last creation of the (binary format!!) log file `/var/log/wtmp`.. This file is regularly compressed and made new.
`lastb`	Displays all the proper logins that happened since the last creation of the (binary format!!) log file `/var/log/btmp`. This file is regularly compressed and made new.

- **Groups Administration:**
  - A user can be participant to more than one group at the same time.
  - A user who is member of a group can change to that group without password but a user NOT member can only change to that group if the group password exist and the user gives it.
  - One or more users can become group administrators for specific groups.
    - Group Administrators can:
      - ➢ add/change/delete the password of the group
      - ➢ add/delete users to the group
      - ➢ reserve the group to members-only

Groups administration commands:

`groupadd` `[`*`options`*`]` *`group`*

System administrator (root) adds a group to the system. Options:

`-g gid` The numerical value of the group's ID.
Value must be non-negative.
This value must be unique, unless the `-o` option is used.

`-o` Allows to assign an existing ID to a group.
The default is to use the smallest ID value equal or greater than `GID_MAX` from `/etc/login.defs` and greater than every other group. Values between `0` and lower than `GID_MIN` are typically reserved for system accounts.

`-r` This flag instructs groupadd to add a system account. The first available `gid` lower than `GID_MAX` will be automatically selected unless the `-g` option is also given on the command line.

`groupmod` `[-g` *`newgid`*`] [-n` *`newname`*`]` *`group`*

System administrator modifies a group settings.

`-g` *`newgid`* changes the gid of the group.
`-n` *`newname`* changes the name of the group.

`groupdel` *`group`* System administrator deletes a group to the system

System administrator(root) `gpasswd` options:

`gpasswd` `[`*`options`*`]` *`group`*

adds/changes the group's password.
Note: The group's password is only needed if a user, which is not a member of the group, wants to temporarily become one and have it as its effective group.
He will be prompted to give the group's password.
Options:

`-R` Makes the group reserved to members-only.
Result: No change of group through `sg` or `newgrp` is allowed for non-members.
The password in `/etc/gshadow` becomes `'!'`

`-A` *`user,...`* adds Group **A**dministrator(s) to a group.
`-M` *`user,...`* adds Group **M**ember(s) to a group.
`-r` *`group`* Removes the password for the group.
The group is then also reserved for members-only
Password in `/etc/gshadow` is simply deleted.

Group administrators `gpasswd` options:

`gpasswd [`*`options`*`]` *`group`*

Adds a new password to a group. Options:

`-a` *`user`* Adds permanently a user to a group
`-d` *`user`* Deletes permanently a user from a group.
`-r` *`group`* Removes the password for the group.(same as with root)

`newgrp` *`group`* or `sg` *`group`* — A user changes itself temporarily to a new group. If the user is not a permanent member, password is asked. The user will be denied access if the group password is empty and the user is not a permanent member.

`sg` *`group`* `-c` *`command`* — Runs a command as participant of the given group and returns to normal after the command finishes.

`grpck` *`group`* — System administrator checks a group.

The groups configuration files:

`/etc/group` — Where all the users for each groups are listed. (Note: The default (main)group of users does not contain the users names)

Format:

```
groupname : Password or x or ! : GID :  Memberlist
```

(eg. user1,user2.. Users' list are separeted by comas)

`/etc/gshadow` — Where the groups passwords are kept

Format:

```
groupname : Password or ! : AdminUsersList  : MemberUsersList
```

(Admin and Users' list are separeted by comas)

- **Converting to/from the standard(older) to shadow(newer) password system.**
  In the older times of Unix, this file was containing the whole incrypted (DES) password in the second field. Because of new faster computers being capable of reverse decoding the passwords, the shadow password system got introduced. If the shadow password system is installed and activated, the second field of `/etc/passwd` file contains an 'x', to indicate that the user has a password and it is located in `/etc/shadow` file.

To convert from one password system to another the following commands are used:

`pwconv` — Converts all the users passwords from the older system to shadow password system. It creates the file `/etc/shadow`.

`pwunconv` — Converts all the users passwords from the shadow system to older system. The `/etc/shadow` is then erased.

`grpconv` — Similar to `pwconv` except that it applies it to the groups. Converts all the group passwords from the older system to shadow password system. It creates the file `/etc/gshadow`.

`grpunconv` — Similar to `pwunconv` except that it applies it to the groups. Converts all the group passwords from the shadow system to older system. The `/etc/gshadow` is then erased.

- **Checking the consistence of passwords and groups files:**

  Two tools are available for checking the consistence of the user's and groups accounts files.

  `pwck [`*`options`*`]` Checks the user's accounts files for consistency.
  (`/etc/passwd` and `/etc/shadow`)
  Checks are made to verify that each entry has:
  - the correct number of fields
  - a unique user name
  - a valid user and group identifier
  - a valid primary group
  - a valid home directory
  - a valid login shell.

  Options:
  `-r` This causes all questions regarding changes to be answered 'no' without user intervention.
  `-s` Sorts entries in `/etc/passwd` and `/etc/shadow` by UID.

  `grpck [`*`options`*`]` Checks the group's accounts files consistency.
  (`/etc/group` and `/etc/gshadow`)
  Checks are made to verify that each entry has:
  - the correct number of fields
  - a unique group name
  - a valid list of members and administrators.

  Options:
  `-r` This causes all questions regarding changes to be answered 'no' without user intervention.
  `-s` Sorts entries in `/etc/passwd` and `/etc/shadow` by UID.

- **Tips & Tricks**
  - Showing all the registered users and their groups in details:
    (For systems where their UID starts at 500).

    ```
    grep ':[5-9]..:' /etc/passwd | cut -d: -f1) ; \
         for user in $users; do id $user ; done ;
    ```

  - Producing an encrypted password through `crypt()` function.

    ```
    echo ClearTextPassword | mkpasswd -s
    ```

  - Disabling a User account without deleting anything:
    - by Adding a **`*`** or a **`!`** to the encrypted password in `/etc/shadow` file**.**

  - Preventing a user from loging in to a shell:
    - by changing the shell of the user to `/bin/false`
    eg. `usedmod -s /bin/false username`

1.111.2 - Tune the user environment and system environment variables

Weight: **3**

- **Description:** Candidate should be able to modify global and user profiles. This includes setting environment variables, maintaining skel directories for new user accounts and setting command search path with the proper directory.

  Key files, *terms*, and utilities include:

`/etc/profile`	**`env`**	**`export`**
`/etc/skel`	**`set`**	**`unset`**

- **Configuration of the shell.**
  Each time a user logs-in (via the login program) a shell is started. Which shell will be started is defined in the last field of the file `/etc/passwd`. Depending on the shell started, some configuration files will be executed to prepare the shell's environment. Some configurations are system-wide and others are in unique for each individual user. Here is the list of configuration files read for different login-shells:

	Bourne Again Shell	Korn Shell	TC-Shell
System wide	`/etc/profile`		`/etc/csh.cshrc` `/etc/csh.login`

Individual Users	`~/.bash_profile`
	`~/.bash_login`
			`~/.tcshrc or ~/.cshrc`
			`~/.history`
			`~/.login`
			`~/.cshdirs`

	`~/.profile`		

Note: The non-login bash shell reads only the `~/.bashrc` and enherits all the environment parameters from the parent process.
Although already mentionned in the previous chapters, there is also a difference between an interactive shell and a non-interactive shell:

Interactive: Presents a prompt for the user to enter commands.
The shell will read its configuration files depending being a login or non-login shell.(see above)

non-interactive: Is normally run as a sub-shell and will not read any configuration file. It depends entirely on enheritance from the parent process.

- **Parameters defined for bash in configuration files:**
  The following parameters are defined either in /etc/profile for all users or in one of the files defining the individual bash settings:
  `~/.bash_profile, ~/.bash_login and ~/.profile`

  Some of the Environment Variables are very important to be set properly before the shell is ready to respond to user commands. Here is a short list of them:
  Environment Variables:

  `PATH` This variable defines the directories that will be searched by bash to lool for the programs that are typed without their path.

  `MANPATH` This variable defines where the program `man` will look for man pages.

`INFODIR` and `INFOPATH` These two variables contains the paths where the program info searches for the info pages.

`PAGER` This variable should contain the name and path of the program used to display man pages.

`EDITOR` This variable normally contains the path and program name of the editor used by some programs like `less`.
eg. in `less` the 'v' starts the editor. Default is `/bin/vi`.

`PS1` & `PS2` These 2 variables define the prompt displayed by bash when waiting for a user command.

`LS_OPTIONS` `LESSCHARSET` These 2 variables often contain the list of options and parameters that the `less` programm uses to display the content of files.

- Functions and aliasses are also defined in these files.
- When bash exits, either after the command `logout`, or `exit`,
  or the key combination Ctrl-D, it runs the file script `~/.bash_logout` if it exists.

- **Environment Variables.**
  The <u>environment variables</u> are simply the variables held by bash that are tagged for export, i.e. variables that will be passed-on to children processes.
  All variables enherited are exported further to children processes.
  Variables that are NOT tagged for export are called <u>shell variables</u>.
  To set the export flag of a shell variable the command `export` or `declare -x` are used. eg.

  ```
  export DISPLAY or declare -x DISPLAY
  ```

  A shell variable can also be created, filled with a value and tagged for export in one command: eg.

  ```
  export DISPLAY=localhost:0.0
  ```

  Here are some commmands (already seen in other chapters) to manipulate the variables:

  `variable=value` Defines the value of a shell or exported variable.
  `export` Lists all the exported (environment)variables
  `declare -x` ""   ""   ""   "
  `set` Lists all shell and exported variables including functions.
  `unset` *`variablename`* Deletes a vartiable and its value.
  `set` Used to define positional variables or define bash operational options.

- **The directory `/etc/skel`**
  This directory contains all the files and subdirectories that are used as template when creating a new user's home directory by usingthe command:
  `useradd -m` *`username`*.
  Because of this, any changes made to the content of this directory will be reflected in the new user's home directories created afterwards. The files and directories in this directory belong to `root` but their copies will belong to their respective owners when their home directory is created . This is the default template directory.
  Another template directory can also be used by issuing the command:
  `useradd -mk` *`/template/dir`*.

- **1.111.3 - Configure and use system log files to meet administrative and security needs.**
  Weight: 3

  Description: Candidate should be able to configure system logs. This objective includes managing the type and level of information logged, manually scanning log files for notable activity, monitoring log files, arranging for automatic rotation and archiving of logs and tracking down problems noted in logs.

  Key files, *terms*, and utilities include:

  ```
  /etc/syslog.conf         logrotate
  /var/log/*               tail -f
  ```

- The `syslodg` Daemon.
  - This daemon process runs in the background, receives log information from the kernel and from other applications or Daemons and, according to the configuration in `/etc/syslog.conf`, it distributes the log information either to files or send it to a console or even can send it to a syslog server via network.
  - Each log information contains only a one line message.
  - The same log information can be sent to many files or other destinations.
  - The log information is received with the following content:
    - Facility: `auth,authpriv,cron,daemon,kern,lpr,mail,news,` `local`*`N`*`,user`
    - Priority Level: `debug,info,notice,warning,err,crit,alert,emerg`
    - Tag: (Title) and maybe the PID of process.
    - Log Text : (actual message).
  - The saved or sent information(one line) is having the following content:

  ```
  Date Time Hostname Tag [Process ID] Message
  ```

- **Format of `/etc/syslog.conf`**:

  facility.level;[*facility.level*];...... *destination*

  IMPORTANT: Remember to use TABs and NOT spaces in your `syslog.conf` file.
  Otherwise it won't work.

  Facilities: (For multiple facilities separate by ',' eg. `auth,cron.*`**)**

`*`	All Facilities
`auth`	General authentications
`authpriv`	Login authentication
`cron`	cron subsystem
`daemon`	System server processes
`kern`	Linux Kernel
`lpr`	Spooling subsystem
`mail`	Mail subsystem
`news`	News subsystem
`local`*`N`*	Locally defined syslog facilities *N*=0 to 7
`user`	Users system messages

Levels: (Priority) From the least to the most important level

	`none`	Exclude messages of this facility eg. `mail.none` = no mail messages
	`*`	All messages
least important	`debug`	Lots of messages. For debugging purposes
|	`info`	Information
|	`notice`	
|	`warning`	
|	`err`	Errors
|	`crit`	Critical
V	`alert`	Very critical
most important	`emerg`	High emergency

Destinations:

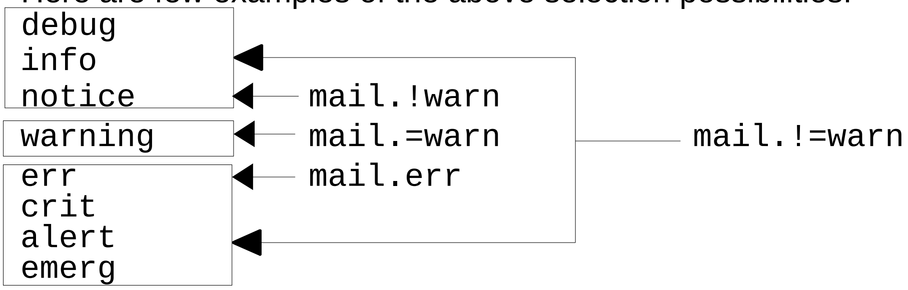
- files	eg. `/var/log/cron.msgs`
- devices	eg. `/dev/tty6` (to virtual console 6)
- usernames	eg. `root` (to root) or `*` (to all logged-in users)
- computer	eg. `@moon` (to 'moon' host....syslog server)
- named pipes	eg. `|/dev/xconsole` (to the virtual x console)

`'-'` before the filename means: buffered before writing.
eg: `*.*;mail.none;auth.none  -/var/log/messages`

Messages exclusive logging: (examples)

`mail.warn`	log the mail warning or more serious messages
`mail.!warn`	log ONLY the messages less serious than warn
`mail.=warn`	log ONLY the mail warning messages
`mail.!=warn`	log all mail messages except the warning ones

Here are few examples of the above selection possibilities:



General Examples:

```
kern.=warn;*.err;authpriv.none  -/var/log/warnings
```

- send Kernel warnings only
- error messages from all facilities
- BUT none from the `authrpiv`
- to the file `/var/log/warnings`

(`'-'` before the filename means buffered before writing to disk)

```
*.emerg;user.none   *
```

- send to whosoever is now logged-in about real emergencies
- BUT not the messages concerning users

Syslog server (hosts messages logging center)

```
*.info          @mainlogger.gdf.local
```

Send all info level messages and more serious to the host '`mainlogger`' using: Prot: `UDP` port: `514`

The host `mainlogger.gdf.local` will log these messages according to its `/etc/syslog.conf` configuration file as if coming from is local processes. Messages will include the IP of sending host though.

IMPORTANT: The receiver host must start the `syslogd` daemon with the `-r` option (`/usr/sbin/syslogd -r`)
in the `/sbin/init.d/syslog` script to enable the receiving logging messages on port `514` from other hosts.

Special for SuSE distribution 8.0 and up:
in `/etc/sysconfig/syslog` the parameter `SYSLOGD_PARAMS` should include the " `-r` " (`SYSLOGD_PARAMS="-r "`)
or use YAST (`/etc/sysconfig/` editor) and Search for `SYSLOGD`

Real-time watching the content of a log file.
The following command allows to watch real-time the content of a log file.

```
eg.     tail -f /var/log/messages
or      less +F /var/log/messages
```

Other GUIs like `xtail` and `xlogmaster` can also do the same more elegantly.

Generating log messages from command line or scripts:

```
logger -p facility.level -i -t "MessageTitle" "Message"
```

('`*`' not allowed as facility or level here) (`-i` option for adding the process PID)

Command to show the very start of kernel mesages at boot-up:

```
dmesg | less     All kernel messages (including booting messages)
cat /proc/kmsg ""    ""    ""          ""    ""    ""
```

Stop generation of the `-----MARK-----` Lines in the log files
(good for laptops: to stop the frequent harddisk access in idle)
start the `syslogd` with the option `-m 0`

- **The `logrotate` program**
This program allows to save and compress the log files regularly based on their age or their size. These parameters are defined in its configuration file: `/etc/logrotate.conf` The parameters in this file will decide which files will be backed-up and according to which criteria.
(The content of this configuration file is not needed for the LPI 102 exam.)
Example of rotation and compression instructions:

```
compress
/var/log/messages {
    rotate 5      Make 5 weekly rotations of the file before deleting old ones.
    weekly        Rotate weekly.
    postrotate    Run the following script after rotating
          /sbin/killall -HUP syslogd
    endscript
}
```

Other possible log files(SuSE only):

`/var/log/boot.msg`	System hardware initialization log at boot-up. It records lilo and kernel boot messages till end of default runlevel initialization. Also
`Ctrl-Alt-F10`	Shows the kernel modules messages.

Examples of default configured log files:

`/var/log/messages`	All messages except mail and auth
`/var/log/faillog`	System failures log file
`/var/log/warn`	System warnings log file

Log files viewer under X-Windows (kde):

`xtail`	Very good
`xlogmaster`	Very good
`kwatch`	hummm... buggy!! ..till now

- **1.111.4 Automate system administration tasks by scheduling jobs to run in the future**
  Weight: 4

  Description: Candidate should be able to use **cron** or **anacron** to run jobs at regular intervals and to use **at** to run jobs at a specific time. Task include managing **cron** and **at** jobs and configuring user access to **cron** and **at** services.

  Key files, *terms*, and utilities include:
  `/etc/anacrontab`
  `/etc/at.deny`
  `/etc/at.allow`
  `/etc/crontab`
  `/etc/cron.allow`
  `/etc/cron.deny`
  `/var/spool/cron/*`
  `at`
  `atq`
  `atrm`
  `crontab`
  `atd`
  `crond`

- **`at` and `cron` services**
  `at` is a service that checks for pre-programed a one-shot job, executes it on schedule and then erases it from the job queue.
  `cron` is a service that checks every minute for scheduled repetitive jobs, executes them on schedule.
  In short, `at` does one time jobs like a spool system
  and `cron` does jobs to be repeated at specific times.

- **The `cron/crontab` service**
  `cron` executes specific commands in *`crontabs`* on a regular basis based on configuration created by root or users.

- **Types of crontabs**
  Crontabs are the configuration files read by the `crond` Daemon defining which jobs should be run when.

  - **Users crontabs**
    The users crontabs are created by the user issuing the command `crontab -e`.
    It is not recommended to edit the user crontabs using an editor directly instead of passing by the above command. Each user gets their own crontab file.
    Once written and saved, users crontabs are located in:
    `/var/spool/cron/tabs/`*`username`* (SuSE)
    `/var/spool/cron/crontabs/`*`username`* (RedHat, Debian, etc) **?**
    These crontab files contain only 6 fields.

  - **System wide crontab file: `/etc/crontab`**
    The file `/etc/crontab` is used as the main system crontab.
    In this file `crond` Daemon will execute the scheduled tasks and run them as the user specified for that task. For that an extra 'user' field (field 6) is used.
    Therefore this crontab file contains 7 fields.

  - **The `/etc/cron.d/` directory**
    This directory contains extra crontabs of the same format as the `/etc/crontab` and will be recognized by `crond`.

  - **The `/etc/cron.{hourly,daily,weekly,monthl}` directories**
    These directories contain scripts that will be run regularly according to their names.
    They are not directly read by `crond` but are usually run from `/etc/crontab`.

- **File format of the user crontab**
  This file may contain the following types of entries:
  - Comments. These lines start with a '#'
  - Environment variables definitions. eg. MAILTO=michel
  - user cron schedule entries: made of 6 fields per line:

minute	hour	day of the month	month	day of the week	Command
0-59	0-23	1-31	1-12	0-7(1=Monday)	/home/michel/bin/myscript
			jan	mon	
			feb	tue	
			et...	etc..	

- **File format of the system crontabs**
  This file format applies to `/etc/crontab` and `/etc/cron.d/`*`xxxxxx`*
  This file may contain the following types of entries:
  - Comments. These lines start with a '#'
  - Environment variables definitions. eg. MAILTO=admin
  - user cron schedule entries: made of 7 fields per line:

minute	hour	day of month	month	day of the week	[username]	Command
0-59	0-23	1-31	1-12	0-7(1=Monday)	root	/usr/lib/cron/run-crons
			jan	mon		
			feb	tue		
			et...	etc..		

Rules for CRONTAB files

- Cron jobs are checked every minute by the `crond` daemon.
- A Crontab file can be a system or a user crontab.
- System crontabs include the username field and the users crontabs don't.
- A crontab entry(line) will be either a comment, an environment setting or a cron command.
- A line starting with `#` is considered a comment.
- An environment setting is in the form: ***`VariableName = value`***
- Each cron entry can be very long but MUST exist on a single physical line.
- If the first character of a crontab line is a '**-**' then cron will not send a message to syslog each time the command is executed, otherwise it will.
- A **`*`** in a field means the command is executable on every instance possible of that field.
- Days of the week are numbers (0 to 7)and start on sunday(0 or 7)(monday=1).
  Names of the weekdays can also be used: `mon,tue,wed,thu,fri,sat,sun`
- '`/`' char. with a number, defines the steps size (default = 1).
  eg. `*/10` is every 10 Minutes `10-18/2` is from 10:00 to 18:00 at every 2 hours
- Hours field is using the range of 0-23
- Any output (incl. errors) from the scheduled jobs are sent to the local user's mail.
  - To avoid to generate any mail then redirect the output (`STDERR` + `STDOUT`) of commands to `/dev/null`. eg. `*/2 * * * * ping -c www.suse.de &> /dev/null`
  - or to avoid generating any mail for ALL of the processes started within this crontab: set the variable MAILTO to "". eg. `MAILTO=""`
- The jobs will be executed by `/bin/sh` or the content of the Variable `SHELL` only if:
  minute & hour & month & (day of the week or day of the month) are matching.

- **Crontab commands:**

`crontab -e`	to create/edit a user crontab(scheduling) file (incl. root) The crontab program will save this file under the user's name in `/var/spool/cron/tabs/` directory. eg. `/var/spool/cron/tabs/joe`
`crontab -l`	Displays user's crontab file.
`crontab -r`	Deletes user's crontab file.
`crontab -e -u` *`username`*	Working on a user's crontab file (need to be root)

- **Access control of cron scheduling service.**
  The crontab scheduling can be restricted to certain users and blocked for all others. This control is done via the files `/etc/cron.allow` and `/etc/cron.deny`.
  - If `cron.allow` exists then all users are NOT allowed to use the cron service except the ones listed in `cron.allow`. `cron.deny` if it exists is then ignored.
  - If `cron.allow` doesn't exist and the `cron.deny` does exist, then all users are allowed to use the cron service except the ones listed in `cron.deny`.

Examples of user crontabs

- **Range of numbers** are expressed with a '-' between numbers:

eg.

```
0  8-18   * * *  /usr/X11R6/bin/xmessage -display :0 "Take a break"
```

Get the message **`"Take a break"`** every hour from 8 to 18 hours every day.

- **Commands can be grouped together** inside **`()`** for redirections of standard output.

eg.

```
0,15,30,45   *   *   *  *   (echo -n "---";date) > /dev/console
```

- **Standard input entries** can be expressed via the '**%**' in the command.
  Any additional '**%**' in the command line produces a new line (like **`/n`** in bash)
  use **\%** to enter a litteral '**%**':

eg.

```
30 11 31 12 * /usr/bin/wall%Happy new Year!%Lets meet and enjoy!
```

- **Range Interval settings** using `'/'` character.
  When a range of time-time or date-date is given, an interval between repetition can be set with the `'/'` character.

eg.

```
30 6-16/4 * * * /usr/X11R6/bin/xmessage -display :0 "Take medicine"
```

Gets a message **`"Take Medicine"`** every 4 hours(**`/4`**) on the half hour (**`30`**)
This means at : 6:30 , 10:30, 14:30, 16:30 every day

- Other simple examples:

eg1.

```
-0   1    *    *    1-5  updatedb
```

Runs the `updatedb` program every day, Monday to Friday at 01:00, and doesn't report to syslog ('**`-`**' at start of line)

eg2.

```
0,10,20,30     12    *     *     1,2   updatedb
```

Runs the `updatedb` program every Monday and Tuesday at: 12:00,12:10,12:20,12:30

eg3: (Username entry is only in `/etc/crontab`)

```
*/10 *    *    *    *    root ping -c1 www.suse.de
```

Sends a ping once every 10 minutes (**`*/10`**)to `www.suse.de` as user **root**

The `anacron` service.

`Anacron` is a similar service like `cron`, except that its frequency is expressed in days and not in minutes. Unlike `cron`, it does not assume that the machine is running continuously. Hence, it can be used on machines that aren't running 24 hours a day, to control daily, weekly, and monthly jobs that are usually controlled by `cron`.

When executed, `Anacron` reads a list of jobs from a configuration file, normally `/etc/anacrontab`. This file contains the list of jobs that Anacron controls. Each job entry specifies a period in days, a delay in minutes, a unique job identifier, and a shell command.

For each job, Anacron checks whether this job has been executed in the last *n* days, where *n* is the period specified for that job. If not, Anacron runs the job's shell command, after waiting for the number of minutes specified as the delay parameter.

After the command exits, `Anacron` records the date in a special timestamp file for that job, so it can know when to execute it again. Only the date is used for the time calculations. The hour is not used.

The file `/etc/anacrontab`

The file `/etc/anacrontab` describes the jobs controlled by `anacron`.

Its lines can be of four kinds:

- job-description lines, environment assignments, empty lines or comments.
- Environment assignments, blank line sand comments are the same as for crontab.

- Job-description lines are of the form:

    `period delay job-identifier command`

  - The period is specified in days
  - The delay is specified in minutes.
  - The job-identifier can contain any non-blank character, except slashes'`/`'. It is used to identify the job in Anacron messages, and as the name for the job's timestamp file.
  - The command can be any shell command.

eg.
```
1  5  cron.daily   nice run-parts --report /etc/cron.daily
7  10 cron.weekly  nice run-parts --report /etc/cron.weekly
30 15 cron.monthly nice run-parts --report /etc/cron.monthly
```

In this example, `anacron` :

- runs all the scripts contained in the directory `/etc/cron.daily`, (using default nice value) every day as soon as the day has changed after it has waited for 5 minutes (delay).
- runs all the scripts contained in the directory `/etc/cron.weekly`, every week as soon as the 7th day has come after it has waited for 10 minutes.
- runs all the scripts contained in the directory `/etc/cron.monthly`, every month as soon as the 30th day has come after it has waited for 15 minutes.

- **The `at` spool service**
  `at` is a service(`atd` Daemon) that checks every minute its job queue and executes the ones that are programmed to be run on schedule.

`at`	Runs a command only once at a predetermined time.
`batch`	command is only a script and does the same as **<u>at</u>** except that it executes commands when system load levels permit; in other words, when the load average drops below 0.8

Syntax:

```
  at [options] time
  batch [options] time
```

How to launch an AT job:

1. Type **at** and any **options** and **time** of execution
2. Enter the command(s) you want to run (press Enter after each command)
   Note: The commands MUST contain the full path of the command.
3. Press Ctrl-D to save the job(s).

- **`at` options:**
  - `-b` Run command only when the system load is low
  - `-d` *`jobNr`* Delete an at job from queue (same as atrm)
  - `-f` *`filename`*
    Read job from a specified file
  - `-l` List all jobs for that user. If user is root, shows all jobs.
  - `-m` Mails user when job completes.
  - `-q queueName`
    Send the jobs to a specific queue.
    Queue names are single letters: **a** to **`z`** and **A** to **`Z`**
    a is the default and set highest AT priority (nice=2).
    b is the default for batch command queues.
    The higher the letter the higher the <u>nice</u> value
    (less priority). <u>nice</u> of: a=2 b=4 c =6 d=8 etc.
- **Time formats**

`now`	Well.... NOW!
`17:00`	At 17:00 hours
`+3 hours`	In 3 hours from now
`+2 minutes`	In 2 minutes from now
`+2 days`	Same time as now but In 2 days
`+3 months`	Same time as now but In 3 months
`19:15 12.03.01`	On 12 march 2001 at 19:15 allowed formats: `MMDDYY, MM/DD/YY, DD.MM.YY`
`now,noon,teatime (4pm),midnight,today,tomorrow`	
`4PM`	At 4 pm today if not too late otherwise tomorrow at 4PM
`16:00 + 3 days`	At 16:00 hous in 3 days
`mon`	Next Monday same time. Days are: `non,tue,wed,thu,fri,sat,sun`
`18:25 31 march 2001`	On <u>march 31 2001</u> at <u>18:25</u>

- **`atq`** Show the AT jobs queue of user. Incl jobs Numbers
  If user is root, shows all jobs. Same as `at -l`
- **`atrm`** *`jobNr [jobNr] ...`* Deletes an AT job from the jobs queue

AT Command on one line(useful in scripts)**:**

```
echo "command1;comman2;..." | at time
```

eg.

```
xhost + localhost
echo "xmessage -display :0 'It works'" | at +1 minutes
watch -n1 atq
```

Files involved:

`/var/spool/atjobs` Where the jobs are stored.
`at` Produces jobs starting with a if queue is not given (`-q`).
These jobs is a snapshot of environment variables plus the commands given.

`/var/spool/atspool` Unknown!!

`/proc/loadavg` Average system load value that gives a one line display of the following information: The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

`/var/run/utmp` Uptime of the processes list in binary format. Used to determine the system load averages.

`/etc/at.allow` List of users that are allowed to use the commands `batch` and `at`.
If present: `/etc/at.deny` is ignored (If at all present) and all the users that are not listed here are NOT allowed.

`/etc/at.deny` List of users that are NOT allowed to use the command batch and at.
If present and `/etc/at.allow` is NOT present, then all users are allowed except the ones listed here.

Alternatives to **`at`** and **`Batch`:** consider another batch system like **`nqs`**.

- **1.111.5 Maintain an effective data backup strategy**
  Weight: 3
- **Description:**

  Candidate should be able to plan a backup strategy and backup file systems automatically to various media. Tasks include dumping a raw device to a file or vice versa, performing partial and manual backups, verifying the integrity of backup files and partially or fully restoring backups.

  Key files, *terms*, and utilities include:

  ```
  cpio      restore
  dd        tar
  dump
  ```

- **Types of backups**
  Depending on the needs, 3 different methods of backups are mostly used:
  <u>Full Backup:</u>
  The Full backup backs-up all the files, independently of whether it was previously backed-up or not. This method uses the most media space. In this case it is recommended to use compression like `gzip` or other methods to reduce the media space needed.

  <u>Differential Backup:</u>
  Saves only files that have been modified or created since the last Full Backup. Normally a Full backup is made and then regular differential Backups are performed.
  Advantages: Only the full backup and the last good differential backup are needed to restore the whole of the data.
  Disadvantages: Takes longer to make than incremental backups and needs larger medias.

  <u>Incremental Backup:</u>
  Saves only the files that have been changed or created since the last backup.

  Advantages: It is shorter to make than the differential backups and needs smaller medias size.

  Disadvantages: All of the incremental backups, up to the last known good one, and the full backup are needed for restoring the whole data. If one of the incremental backups has some media fault, the backup may be unreliable.

  Note: Normally a full backup is coupled with either periodic differential backups or periodic incremental backups.

- **Restoring data**:
  With differential backups:
  The Full backup and the last good differential backup is needed.
  1 - Read the full backup
  2 - Read last good differential backup.
  With incremental backups:
  The Full backup and ALL of the incremental backup are needed.
  1 - Read the Full Backup
  2 - Read sequentially each incremental backup up to the last good one.

- **Backup media devices files**
  To create backups, external media devices are needed. Here are some common one used under Linux:

`/dev/st0`	First SCSI Tape Drive
`/dev/ft0`	First floppy-controller tape drive
`/dev/fd0`	First floppy disk drive
`/dev/hd`*`x`*	May be an ATAPI Zip or other removable disk

- **Basic backup programs:**

`tar` (Tape ARchive)

Recursively creates archives of files and directories including file properties.
It requires at least one mode option to function properly.
Basic Mode options:

`-c`	Create a new archive.
`-t`	List the content of the archive
`-x`	Extract files from the archive.

Often used Options:
Basic mode options

`f tarfile`	Unless tar is using standard I/O, use the '`f`' option with tar to specify the tarfile. This might be simply a regular file or it may be a device such as `/dev/st0`.
`v`	Verbose mode. By default, tar runs silently. When '`v`' is specified, tar reports each file as it is transferred.
`w`	Interactive mode. In this mode, tar asks for confirmation before archiving or restoring files. This option is useful only for small archives.
`z`	Enable compression. When using '`z`', data is filtered through the gzip compression program prior to being written to the tarfile, saving additional space. The savings can be substantial, at times better than an order of magnitude depending on the data being compressed. An archive created using the '`z`' option must also be listed and extracted with '`z`'; tar will not recognize a compressed file as a valid archive without the '`z`' option. Tarfiles created with this option will have the `.tar.gz` file extension.
`j`	BZ2 Compression. Similar to the '`z`' compression except that its compression method is a bit more efficient on the media space used. The filename of the archive should then have the extention `.tar.bz2`
`N` *`date`*	Store only files newer than the `date` specified. This option can be used to construct an incremental or differential backup scheme.
`V "`*`label`*`"`	Adds a *`label`* to the `.tar` archive. Quotes are required to prevent the label from being interpreted as a filename. A label is handy if you find an unmarked tape or poorly named tarfile.

dump

From the world of the BSD UNIX, allows to backup a whole partition or a full directory. But Linux `dump` is unique and written specially for `ext2`.
Now a version for `ReiserFS` is also available.
`dump` search through files and decide which ones files should be written.

Output of `dump:`	*Hard Disk*, or *Tape* or *File*(Option `-f`)
Feature:	Span files on multiple medias.(medium change).
Max. Backup Levels:	10 (0-9)
Level 0:	Full backup
Level 1-9:	Incremental backup relative to the lower level backup.

Syntax: `dump [-level] [-ua] [-f` *`BackupFile`*`]` *`Source`*

Options:

`-level`	0 to 9
`-u`	Update. Uses the file `/etc/dumpdates` to know which update to do.
`-a`	Automatically asking for next medium change.
`-f` *`BackupFile`*	Name of destination filename. eg. `/dev/st0` Tape drive. `/bckp/backup01.dump` Normal file.
`Source`	Device(partition) or Directory name to backup.

The file `/etc/dumpdates` contains a list of already done backups.
Format: *`Source BackupLevel Date_Time`*
eg.

```
/dev/sda5 0 Sat May 18  23:55:32 2003
/dev/sda5 1 Mon May 20  23:54:13 2003
```

Shows that on Sat. May 18 a Full backup was made using the command:

```
dump -0ua -f /dev/tape   /dev/hda5
```

and an incremental backup relative to the Full backup using the command:

```
dump -1ua -f /dev/tape   /dev/hda5
```

restore

This program is the counterpart of the backup program `dump`. It is not only used for restoring but also to compare the backed-up data with the current original data.

Commonly used options are:

```
restore -C -f BackupDevice
```

Will compare(`-C`) the content of the BackupDevice(eg. `/dev/st0`) to the original and the differences will be shown.

```
restore -i -f BackupDevice
```

Will start in interactive(`-i`) mode and waits for commands relating to the list of files to restore. The most important commands are:

`cd` *`Directory`*	Changes to another directory on the backed-up medium.
`ls [`*`Directory`*` |file]`	Lists the current directory (like bash's `ls`)
`add ` *`Directory`*`|`*`File`*	Add the Dir. or File in the list to restore.
`delete` *`Directory`*`|`*`File`*	Delete the Dir or File in the list to restore.
`extract`	Start the restoring the files listed.
`quit`	Exit restore program.

Important: When the restore is activated, its restores the files in the current directory. Therefore if files backed-up to `/dev/st0` need to be restored to `/dev/hda8` which is mounted on `/mnt/data`, you need to change the current directory to the mountpoint:

eg.

```
cd /mnt/data
restore -r -f /dev/st0
```

Here the full Backup located in `/dev/st0` will be restored to the directory `/mnt/data` which is the mountpoint of `/dev/hda8` partition.

Restoring single files:

```
restore -x -f BackupDevice File1 File2 File3 ....
```

Restores *`File1 File2 File3`* etc. from the *`BackupDevice`* to the current directory.

`cpio`

This back-up utility can handle different types of backup format including the TAR format. Its advantage over tar is that, it takes the list of the files to backup from the STDIN instead of from the command line. This way it facilitates the use of the program `find` to feed the list of files to backup.

`cpio` Modes of operation:

*copy-**out*** (-o)
The output of the program is an archive: Backup

*copy-**in*** (-i)

Files are extracted from the archive: Restore

*copy-**pass*** (-p)
Simple copy of files from one location to another: Copy

`cpio` works with a very large amount of options. They are not the goal of LPI-102.

Here are some commonly used `cpio` options to remember:

`-d`	Create directories if needed.
`-f`	Specifies a filename
`-t`	Shows the contents of an archive.
`-u`	Overwrites existing files
`-v`	Runs in verbose mode

eg1. `cpio -iv < /dev/tape`
The above command reads in files from a tape and displays them as it is operating.

eg2. `find / -name mart* | cpio -pdv /home/martin/backups`
Copy all files from the whole system which start with `mart` to the home subdirectory of `martin`, creating all the needed subdirectories(`-d`),
using the verbose mode(`-v`).

eg3. `find . -name "*.old" | cpio -ocBv >/dev/st0`
Backup(`-o`) all files with ext. `.old`, using the new (SVR4) portable format(`-c`) and the block size of 5120 Bytes(`-B`) to a tape drive (`/dev/st0`),
using the verbose mode(`-v`).

eg4. `cpio -icdv "*.c" < /dev/st0`
Restore(`-i`) all the `*.c` files using the new (SVR4) portable format(`-c`), creating new subdirectories if needed(`-d`) from the tape drive (`/dev/st0`), using the verbose mode(`-v`).

eg5. `find . -depth | cpio -pd /tmp/newdir`
Copy(`-p`) recursively all files in current directory(`.`) to `/tmp/newdir`, creating all the needed subdirectories(`-d`).

dd

This DiskDump program allows to read and write directly to and from a block device as well as normal files. It is very often used program. It can copy directly data blocks from a device to a file and vise versa as well as device to device.
Syntax:
`dd if=`*`InputFile`* `of=`*`OutputFile`* `bs=`*`BlockSize`* `count=`*`NumberOfBlocks`*
Extra options:

```
ibs=InputBlockSize
obs=OutputBlockSize
```

Allows to set the input block size and the output block size when they differ.
Mostly used

The default for `bs` is the original block size of `if=`*`InputFile`*
The default for `count` is the whole device or file.

Examples:

To copy a full partition to a file:

```
dd if=/dev/hda4 of=/tmp/hda4_Image.img
```

To backup the current MBR to a file:

```
dd if=/dev/hda of=/var/backup/MBR.img bs=512 count=1
```

To create a CD image file from a CD-ROM.

```
dd if=/dev/cdrom of=/home/martin/images/cdrom2.img
```

To create a bcakup of partition to a Streaming Tape:

```
dd if=/dev/hda4 of=/dev/st0
```

To restore the above backup:

```
dd if=/dev/st0 of=/dev/hda4
```

1.111.6 Maintain system time

Weight: 4

Description:

Candidate should be able to properly maintain the system time and synchronize the clock over NTP. Tasks include setting the system date and time, setting the BIOS clock to the correct time in UTC, configuring the correct timezone for the system and configuring the system to correct clock drift to match NTP clock.

Key files, *terms*, and utilities include:

```
/usr/share/zoneinfo      date
/etc/timezone            hwclock
/etc/localtime           ntpd
/etc/ntp.conf            ntpdate
/etc/ntp.drift
```

Time clocks under Linux:

There are 2 types of clocks in Linux which play an important role:

Hardware Clock: aka: RTC, RealTimeClock, CMOS Clock, BIOS Clock. Runs independant to the Operating System and runs even when the computer is turned OFF, as long as the CMOS Battery lasts.

Software Clock: aka: System Clock. Runs via the system timer interrupt. Counts the number of seconds since 1st. Jan. 1970. Is the main clock under Linux. At boot time it reads the hardware clock and continues alone from there.

Clock control programs

Under Linux 2 main programs are used to control the 2 clocks.

`hwclock`	Controls the Hardware Clock
`date`	Controls the System Clock

Time settings and zones

There are 2 possible standard ways to set the clock.

Local Time	Time of geographic location
UTC	'Greenwich Mean Time'. Normal way of setting the time from which a time zone offset is given to calculate the Local Time.

Setting the time in Linux

The procedure is relative simple:

1) Set the Hardware clock to UTC via the BIOS setup.

2) Set the Environment variable TZ to the proper time zone using the script:

```
tzselect
```

Alternative step 2:

- Use the program tzconfig which will set a symbolic link in the for of:

```
ln -s /usr/share/zoneinfo/Europe/Berlin /etc/localtime
```

3) To tell Linux that our Hardware clock and system clock are set to UTC run:

```
hwclock --utc --hctosys
```

The Program hwclock

This program is used to display or set the Hardware clock.

Syntax:

```
     hwclock [option]
```

Options:(only one of the following options is used at one time)

`--show`	Read the Hardware Clock and print the time on Standard Output. The time shown is always in local time, even if you keep your Hardware Clock in Coordinated Universal Time (UTC).
`--set`	Set the Hardware Clock to the time given by the `--date` option.
`--hctosys`	Set the System Time from the Hardware Clock. Also set the kernel's timezone value to the local timezone as indicated by the TZ environment variable and/or `/usr/share/zoneinfo`. This is a good option to use in one of the system startup scripts.
`--systohc`	Set the Hardware Clock to the current System Time.
`--adjust`	Add or subtract time from the Hardware Clock to account for systematic drift since the last time the clock was set or adjusted.
`--utc`	Indicates that the Hardware Clock is kept in Coordinated Universal Time .
`--localtime`	Indicates that the Hardware Clock is kept in Local Time. It is your choice whether to keep your clock in UTC or local time, but nothing in the clock tells which you've chosen. So this option is how you give that information to `hwclock`.

The Program date

This program is used to show or set the System time.

Syntax:

```
     date [options]
```

Options:

+text_and_metacodes

Allows to control the display of the current time /and/or date.

eg. `date "+It is now %H Hours and %M Minutes"`

Will have the following result: `It is now 14 Hours and 33 Minutes`

MMDDhhmm	Set the System time to a specific value:
MMDDhhmmYYYY.[*ss*]	Set the System time to a specific extended value.

MM	Month (01-12)
DD	Day(01-31)
hh	Hours(00-23)
mm	Minutes(00-59)
YYYY	Year(1900-2099)
[*.ss*]	Optional Seconds(00-59)

Note: In the file `/etc/adjtime` the correction factor can be saved to keep the clock as accurate as possible.

- **Network Time Service**

This service is used to set the clients clocks to a very precise clock.
The service allows to calculate the time delay for TCP connection so that the received time at the client's computer is anyway accurate.
Protocol: NTP
Methods:
1) Cron job asks the Time server using the `ntpdate` program.
2) Local daemon(`ntpd or xntpd`) runs on client and polls the Time server.
Note: This solution transforms the client also as Time Server.

The program `ntpdate`

This program connects with a Time Server and sets the System time.
Syntax:

```
ntpdate TimeServerName
```

Normally it is regularly called from a cron job. eg:

```
10 * * * *     root /usr/sbin/ntpdate ntp3.fau.de
```

Note: A list of time servers in Internet is located at:
`http://www.eecis.udel.edu/~mills/ntp/clock1a.html`

The Daemon `ntpd or xntpd`

These daemon polls one or more Time Server(s) each 5 minutes and sets the system Time.

Configuration file: `/etc/ntp.conf`

eg.

```
server ntp3.fau.de
driftfile /etc/ntp.drift
```

This `driftfile` will store the local Hardware Clock drift and will be used at boot time to set local System Clock to a more accurate time till a connection to a Time Server is achieved.

Note1: If the local time has drifted of more than 1000 seconds then a syslog message is generated and the clock must be set manually.

Note 2: It is also not possible to use both methods: `ntpd` and `ntpdate` at the same time.

Topic 112: Networking Fundamentals

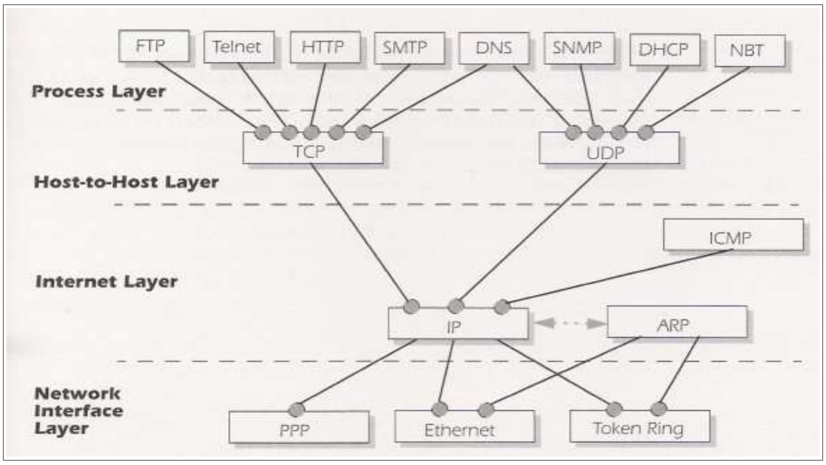
- **1.112.1 Fundamentals of TCP/IP**
  Weight: 4

  Description: Candidates should demonstrate a proper understanding of network fundamentals. This objective includes the understanding of IP-addresses, network masks and what they mean (i.e. determine a network and broadcast address for a host based on its subnet mask in "dotted quad" or abbreviated notation or determine the network address, broadcast address and netmask when given an IP-address and number of bits). It also covers the understanding of the network classes and classless subnets (CIDR) and the reserved addresses for private network use. It includes the understanding of the function and application of a default route. It also includes the understanding of basic internet protocols (IP, ICMP, TCP, UDP) and the more common TCP and UDP ports (20, 21, 23, 25, 53, 80, 110, 119, 139, 143, 161).

  Key files, *terms*, and utilities include:
  /etc/services
  ftp
  telnet
  host
  ping
  dig
  traceroute
  whois

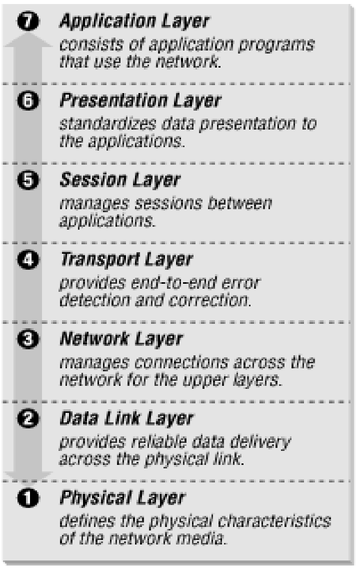
OSI Model	**TCP/IP Stack**	**Protocols**
Application	Process	FTP,Telnet, SSH, HTTP,...
Presentation		
Session	Host to Host	TCP, UDP
Transport		
Network	Internetwork	IP, *ICMP*, *ARP*, *OSPF*, *EGP*
Data Link	Network Interface	Ethernet, FDDI, AAL5, PPP, PPPoE
Physical		Ethernet Frame

Example:



- IEEE-802.3=Ethernet

- **OSI Model**



7 **Application Layer** consists of application programs that use the network.

6 **Presentation Layer** standardizes data presentation to the applications.

5 **Session Layer** manages sessions between applications.

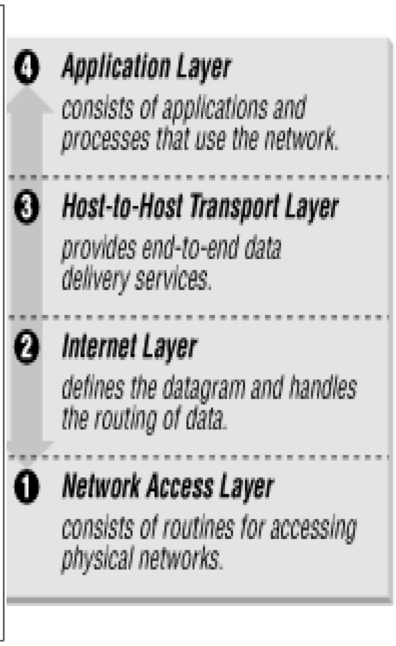
4 **Transport Layer** provides end-to-end error detection and correction.

3 **Network Layer** manages connections across the network for the upper layers.

2 **Data Link Layer** provides reliable data delivery across the physical link.

1 **Physical Layer** defines the physical characteristics of the network media.

TCP/IP Stack



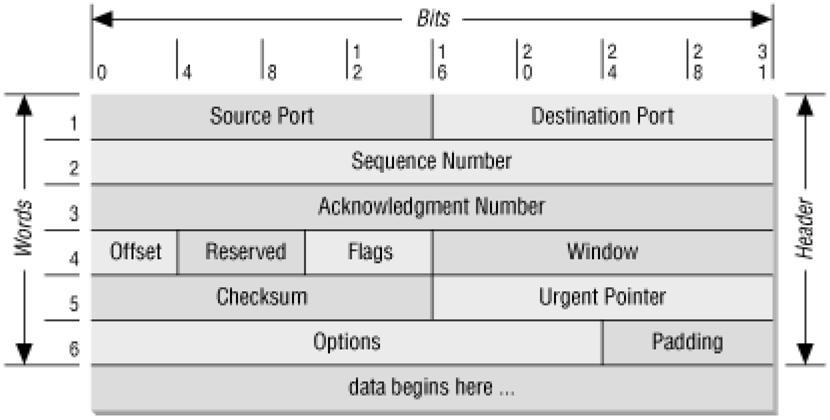
4 **Application Layer** consists of applications and processes that use the network.

3 **Host-to-Host Transport Layer** provides end-to-end data delivery services.

2 **Internet Layer** defines the datagram and handles the routing of data.

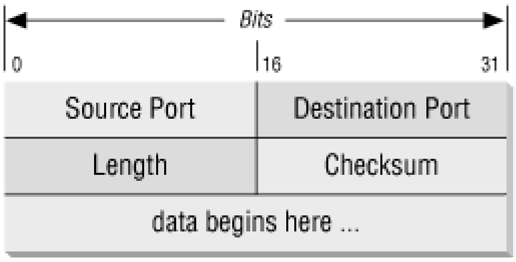
1 **Network Access Layer** consists of routines for accessing physical networks.

- **TCP Protocol Header**



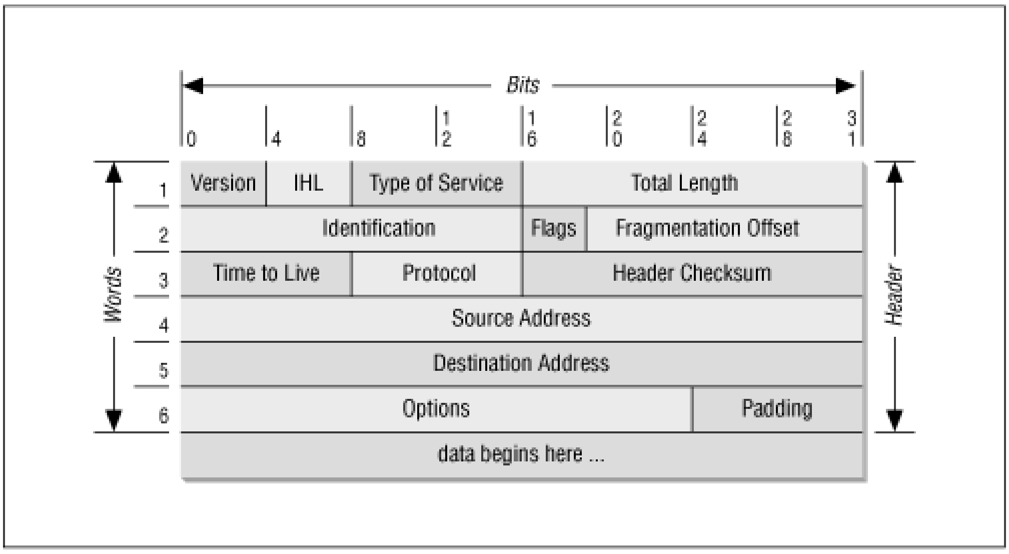
1	Source Port			Destination Port
2	Sequence Number			
3	Acknowledgment Number			
4	Offset	Reserved	Flags	Window
5	Checksum			Urgent Pointer
6	Options			Padding
	data begins here ...			

- **UDP Header**

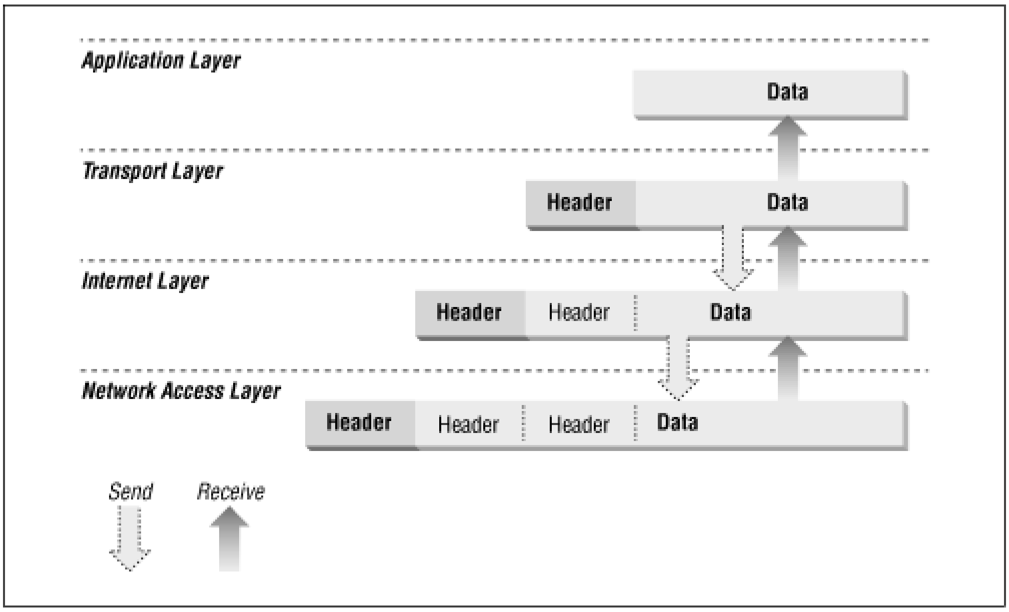


Source Port	Destination Port
Length	Checksum
data begins here ...	

- **IP Header**



- **TCP/IP data Encapsulation**



- **TCP/IP Network Tools**

`ping` Sends an ICMP Packet (type 8) to verify the presence of a remote host. The remote host normally sends an ICMP packet (Type 0) back.

`traceroute` Displays the Names/IP of routers encountered to a remote destination.

`whois` Asks a whois server(RFC 812) for the owner and administrator of a Domain.

`host, nslookup, nsquery,dig`

Ask a DNS (Name Server) to translate an FQDN to an IP or reverse.

eg. `dig . ns`

Displays the list of all ROOT DNS Servers.(hint types servers)

- **IP Addresses Classes**

Normal Internet Addresses(Unicast=Single machine)

Class	Start Addr.	End Addr.	Netmask
A	1.0.0.0	127.255.255.255	255.0.0.0
B	128.0.0.0	191.255.255.255	255.255.0.0
C	192.0.0.0	223.255.255.255	255.255.255.0

Reserved Addresses:(Internet Non-Routable Addresses=Reserved for Intranet) (RFC 1597)

A	10.0.0.0	10.255.255.255	255.0.0.0
B	172.16.0.0	172.31.255.255	255.255.0.0
C	192.168.0.0	192.168.255.255	255.255.255.0

Special Addresses:(Reserved)

D		224.0.0.0	239.255.255.255	(Multicasting-Groups)
	eg.	RIPv2	224.0.0.9	All RIPv2 Routers
		OSPF	224.0.0.5	All OSPF Routers
		OSPF	224.0.0.6	Some OSPF Routers
E		240.0.0.0	255.255.255.255	(Internet Administration)

ICMP Messages:

Error Messages:

3	Destination unreachable
4	Source quench
5	Redirect
11	Time exceeded
12	Parameter Problem

Information Messages:

0	Echo reply
8	Echo request
13	Time stamp
14	Time stamp reply
15	Information request
16	Information reply
17	Address mask request
18	Address mask reply

PCP/IP Services:
(Also found in `/etc/services`)

Port	Protocol	Transport Protocol	Description
20	FTP-Data	TCP	Data Channel of FTP Connection..
21	FTP	TCP	Control Channel of FTP Connection.
22	SSH	TCP or UDP	Secure Shell
23	TELNET	TCP	Terminal Emulation over Network
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP or mostly UDP	Domain Name Server
80	WWW/HTTP	mostly TCP or UDP	Hypertext Transfer Protocol
110	POP3	TCP or UDP	Post Office Protocol
119	NNTP	TCP	Net News Transfer Protocol
139	NetBIOS-SSN	TCP or mostly UDP	Windows Network Session Service
143	IMAP2	TCP or UDP	Interim Mail Access Protocol (Encrypted)
161	SNMP	UDP	Simple Network Management Protocol

- **Extra subjects treated in LPI 102 but not described here:**
  - Netmask and Subnetting (CIDR)
  - Routing fundamentals including default gateway
  -

- **1.112.3 TCP/IP configuration and troubleshooting**
  Weight: 7

  Description: Candidates should be able to view, change and verify configuration settings and operational status for various network interfaces. This objective includes manual and automatic configuration of interfaces and routing tables. This especially means to add, start, stop, restart, delete or reconfigure network interfaces. It also means to change, view or configure the routing table and to correct an improperly set default route manually. Candidates should be able to configure Linux as a DHCP client and a TCP/IP host and to debug problems associated with the network configuration.

  Key files, *terms*, and utilities include:
  `/etc/HOSTNAME` or
  `/etc/hostname`
  `/etc/hosts`
  `/etc/networks`
  `/etc/host.conf`
  `/etc/resolv.conf`
  `/etc/nsswitch.conf`
  `ifconfig`
  `dhcpcd, dhcpclient, pump`
  `host`
  `hostname (domainname, dnsdomainname)`
  `netstat`
  `ping`
  `traceroute`
  `tcpdump`
  `route`
  and *The network scripts run during system initialization.*

- **Networking configuration files and tools**
  Configuration files:

  `/etc/HOSTNAME` or `/etc/hostname` Contains the name(FQDN) of the present host

  `/etc/hosts` Contains a translation table of IPs and equivalent Hostnames. A short and/or a long name(s) per IP. Was primarily(beginning of internet) used before DNS was devellopped.

  `/etc/networks` Contains a translation table of Network Addresses and equivalent Network name(s)

  `/etc/host.conf` Contains the sequence that the local resolver should follow for translation a name to an IP(resolving).
  eg.
  `multi on` Multiple names are allowed per host
  `order host bind` First try to resolve using /etc/hosts then use DNS queries

  `/etc/nsswitch.conf` Same but more refined function as `/etc/host.conf`.

  `/etc/resolv.conf` List of search domains for resolving short names and list of (max 3) of IP of NameServers to resolve FQDNs.
  eg.
  ```
  search linux.local
  domain linux.local (deprecated)
  nameserver 194.25.2.129
  nameserver 192.76.144.66
  nameserver 145.253.2.171
  ```

- Networking tools

`ping` Sends an ICMP Packet (type 8) to verify the presence of a remote host. The remote host normally sends an ICMP packet (Type 0) back.

`traceroute` Displays the Names/IP of routers encountered to a remote destination.

`whois` Asks a whois server(RFC 812) for the owner and administrator of a Domain.

`host, nslookup, nsquery, dig`
Ask a DNS (Name Server) to translate an FQDN to an IP or reverse. eg. `dig . ns`
Displays the list of all ROOT DNS Servers. (hint types servers)

`hostname` Displays different parts or all of the local host FQDN.

`domainname` Displays the local NIS domain name (different than DNS name)

`dnsdomainname` Displays the local DNS Domain Name.

`ifconfig` Tool to configure or turn OFF the network interface.
eg. `ifconfig eth0 192.168.100.60 up`

`route` Tool to display and set and erase entries in the routing table

`netstat` Tool to display a variety of network informations incl:
- Routing Table
- UNIX and TCP/IP Sockets
- Ports in listening mode
- Present TCP/UDP connections status

`tcpdump` A network sniffer program to display the content of network packets.

`dhcpcd` DHCP client program(The one used by SuSE)

`pump` DHCP client program(The one used by RedHat)

`dhclient` ISC DHCP client program. With extended functions compare to the above two DHCP clients.

- **Boot time scripts**

These scripts are part of the runlevel system and are run at boot time to configure the network devices and services ready for operation. They are normally located in:

```
/etc/init.d/*
```

These scripts often use configuration files located in /etc/ or subdirectories of /etc.
eg. `/etc/sysconfig/network/*`
Frontends for `ifconfig` and `route (ifup` and `ifdown)` are often used to configure more easily the network interface.

- **Extra subjects treated in LPI 102 but not described here:**
  - Parameters and options of all the network tools
  - `ftp` commands
  - Paramters of `/etc/nsswitch.conf`
  - Parameters of `ifconfig`
  - Parameters of `route`
  - Options of `netstat` and their results meanings.
  - How `ifup` and `ifdown` get called and how they work
  - DHCP server configuration
  - DHCP Clients parameters
  - Parameters of `/etc/host.conf`
  - Parameters of `/etc/resolv.conf`
  - Options of `tcpdump`
  - Function and configuration of DHCP
-

- **1.112.4 Configure Linux as a PPP client**
  Weight: 3

  Description: Candidates should understand the basics of the PPP protocol and be able to configure and use PPP for outbound connections. This objective includes the definition of the chat sequence to connect (given a login example) and the setup commands to be run automatically when a PPP connection is made. It also includes initialisation and termination of a PPP connection, with a modem, ISDN or ADSL and setting PPP to automatically reconnect if disconnected.

  Key files, *terms*, and utilities include:

```
/etc/ppp/options.*        /etc/ppp/ip-up
/etc/ppp/peers/*          /etc/ppp/ip-down
/etc/wvdial.conf          wvdial
                          pppd
```

- **Connection protocols:**
  Most TCP/IP connections use one of the following connection protocols:
  - Ethernet with MAC address (ARP protocol)
  - SLIP (Serial Line IP): Older and almost not any more used
  - PPP (Point to Point Protocol) Mostly used.

- **PPP Protocol.**
  This protocol allows to connect from a host to another via a Point-to-Point Protocol. Although it can be used to do so only it can also be used to connect to a host which is a gateway to the Internet. This is the way it is mostly used these days.
  PPP Protocol comes in different versions:

- PPP	for Analog Modems
- syncPPP	for ISDN
- PPPoE	for SDL

- **Sequence of PPP connection build-up:**
  1 - Modem connection build-up
  2 - Login chat connection build-up
  3 - PPP Connection build-up

  - **1 - Modem connection build-up**
    The modem connection build-up is usually done by controlling the analog/ISDN/DSL Modem in order to establish a contact between the local and a remote modem. With analog modems this control is achieved via 'AT' (Hayes compatible) commands sent to the modem. After each command, if it is successful, the modem answers with an 'OK'.

eg.	ATZ	Reset the modem to User Settings
	AT&F	Reset the modem to Factory Settings
	ATD0017853	Dial the number 0017853

    Once the Modems have synchronized the local modem sends a message that contains the word CONNECT(eg. CONNECT 28800) to the dialer. Once the modems have connected and synchronized, they become transparent and simulate a simple serial cable connection between the 2 hosts.
    The modem connection is then established.

- **2 - Login chat connection build-up**
  Once the modem connection is completed, the remote *getty* program(in the ppp server) will then send the message `Login:` and wait. At this point the pppd daemon need to be started with its positional parameters which sets:
  - The Login program and parameters
  - The device connected to the modem
  - The speed of connection

  eg. `pppd "chat -f  /etc/ppp/provider" /dev/ttyS1 38400`

  Using the Login script (`/etc/ppp/provider`) the local `chat` program will start answering the remote *getty* with the Login Name and its respective password. When the `chat` program finishes its script successfully, the login connection is established.

- **3 - PPP Connection build-up**
  As soon as the login connection is established, the pppd starts the shell script `/etc/ppp/ip-up` including the following list of positional parameters.
  Syntax:
  `ip-up` *`InterfaceName Device Speed Local_IP Remote_IP`*
  eg.
  `ip-up /dev/ttyS1 /dev/ppp0 38400 136.36.27.93 42.94.78.35`

  This script will take care of some of the necessary preparations incl:
  - Writing the proper nameserver IPs in `/etc/resolv.conf` if needed
  - Start the firewall if needed
  - Starting any other needed process if needed.

  The PPP connection is then established.

  Note: After the pppd has shut down the connection, it starts the `/etc/ppp/ip-down` script, which is a symbolic link to `ip-up`.

- **The dialer `wvdial`**
  This dialing program will take charge of:
  - Dialing the modem with AT commands
  - Answering the remote getty with Name and Password
    (The `chat` program is not needed)
  - Start the pppd daemon. The pppd daemon is responsible for starting the `ip-up` and `ip-down` scripts.

  Its configuration file: `/etc/wvdial.conf`
  Content of `/etc/wvdial.conf`:
  `[Dialer Defaults]`
  Defaults dialing parameters for all connections.
  eg.

```
[Dialer Defaults]

    Modem = /dev/modem
    Baud = 57600
    Init1 = ATZ
    Dial Command = ATDT
    Idle Seconds = 360
    Phone = 0192479264
    Username = michel
    Password = mypasswd
```

`[Dialer` *`ProviderName`*`]`
Dialing parameters for this provider connection.

eg.

```
[Dialer provider1]

     Phone = 0987654321
     Username = hans
     Password = hanspasswd

[Dialer provider2]

     Phone = 0918273645
     Login Prompt = mariette:
     Username = imueller
     Password = pw5Xvg$
```

Extra `wvdial` options for pppd

With some of the latest `pppd` daemons the presence of `wvdial` options file is also needed: `/etc/ppp/peers/wvdial` to add a few option to the `pppd`.
Example of content of `/etc/ppp/peers/wvdial`

```
noauth
name wvdial
replacedefaultroute
```

pppd options

`pppd` daemon uses the general options file `/etc/ppp/options`.
It uses also the individual network interface options file `/etc/ppp/options.`*`InterfaceName`* if it exists.
eg. `/etc/ppp/options.modem` for the `/dev/modem` interface.

Content of `/etc/ppp/options` file:

This file contains the pppd operating options. These options can also be given on the command line witht the `pppd` command.
See `man pppd` for a list of options available.

Shutting down a ppp connection

To shut down a ppp conection, we only need to kill the `pppd` process.
The most appropriate way to do this is to send the signal `-INT` to `pppd`.
eg. `kill -INT $(cat /var/run/ppp0.pid)`
Kills the `pppd` process responsible for the `ppp0` connection.

Topic 113: Networking Services

- **1.113.1 Configure and manage inetd, xinetd, and related services**
  Weight: 4

  Description: Candidates should be able to configure which services are available through inetd, use tcpwrappers to allow or deny services on a host-by-host basis, manually start, stop, and restart internet services, configure basic network services including **telnet** and **ftp**. Set a service to run as another user instead of the default in inetd.conf.

- **Key files, *terms*, and utilities include:**

```
/etc/inetd.conf      /etc/services
/etc/hosts.allow     /etc/xinetd.conf
/etc/hosts.deny      /etc/xinetd.log
```

- **The superdaemons inetd and xinetd.**
  Certain services are made available via daemons running in the background. Certain other set of services are made available via a single daemon which watches the service ports and starts the appropriate program when a client of that service requests attention. This watching demaon is called 'superdaemon'. There are two versions of this type of superdaemons: inetd and xinetd(more recent).

- **inetd Superdaemon**
  This daemon uses the settings in its configuration file /etc/inetd.conf to determine which service ports will be watched and which service programs are associated with them. When a service port receive a request from a client on a particular port , inetd can be configured to use a tcpwrapper which will check if the client host is allowed to use this service before the service program is started.

- **The configuration file inetd.conf**
  Each port that need to be watched gets one configuration line. The parameters are separated with spaces or TABs. Here is the configuration line format:

```
service    socketType protocol wait       user  program          arguments
eg.
ftp        stream     tcp      nowait     root /usr/sbin/tcpd wuftpd
telnet     stream     tcp      nowait     root /usr/sbin/tcpd telnetd
```

service: name of the service referenced in the file /etc/services

socket: can be stream, dgram, raw, rdm or seqpacket

stream:	TCP
dgram:	UDP
raw:	raw format
rdm:	Reliable Delivered Message
seqpacket:	Sequenced Packet Socket

wait: Can be wait or nowait
Tells inetd whether it should wait for the server to come back before accepting another client connection.
nowait is used for multi-threaded services(most services)
wait is used for single-threaded services(some UDP services)
eg: comsat, biff, talkd and tftpd

user: Which local user will be the owner of the service process.
program: Program to start to provide the service
(normally the `tcpd` tcpwrapper)

arguments: Either the service program as arguments for the `tcpd` tcp wrapper
or the service program itself without tcpwrapper...NOT recommended.

- **`xinetd` Superdaemon**
  This more recent Superdaemon allows for more flexibility and security. It uses one main configuration file `/etc/xinetd.conf` which can be extended to multiple service definition files via the parameter `includedir`.
  eg. `includedir /etc/xinetd.d .` (All files in the `/etc/xinetd.d/` directory).
- **Advantages of xinetd over inetd:**
  - `xinetd` uses the control files(`hosts.allow` and `hosts.deny)` directly without the need to use the tcpwrapper `tcpd`.
  - Limits the connections either general, per client or per service
  - Certain clients can be given certain services vs. others
  - Protection against DenialOfService
  - Produces its own log files independantly from syslog
  - Possibility to redirect incoming requests to another server(eg. in a DMZ)
  - Full support of IPv6
  - Interaction with the client: Messages different for success vs. failure to connect.
- To convert the service definition parameters from `inetd` format to `xinetd` format, the tool `xconv.pl` can be used. It is delivered with the `xinetd` package.
  The `xinetd.conf` contains the default and per-service definitions.
  The default definitions are used for all of the services. In case of conflict with the per-service definition, the per-service prevail. For example for defaults and ftp service:

```
defaults
{
      instances   = 15
      log_type          = FILE /var/log/xinetd.log
      #log_type         = SYSLOG daemon info
      log_on_success    = HOST PID USERID DURATION EXIT
      log_on_failure    = HOST USERID RECORD
      only_from         = 192.0.0.0/8
      disabled          = shell login exec comsat
      disabled          += telnet ftp
      disabled          += name uucp tftp
      disabled          += finger systat netstat
}

service ftp
{
      socket_type       = stream
      wait              = no
      user              = root
      server            = /usr/sbin/in.ftpd
      server_args       = -l
      instances         = 4
      access_times      = 7:00-12:30 13:30-21:00
      nice              = 10
      only_from         = 192.168.1.0/24
      disabled          = yes
}
```

Deactivated parameters starts the line with a '#'. The parameters meanings are somewat similar to the inetd.conf but allows for more flexibility. The service definition block starts with the word service followed by the service name, then all of the parameters for this service are enclosed within curly brackets.'{....}'. The parameter disable = yes tells that the service is disabled. It must be set to no to enable it.
= sets the value,
+= adds the value (to default values) ,
-= deletes the value (from default values)

- **Tcpwrappers**

The tcpwrappers are programs that uses configuration files to check if the client host is allowed to use the requested service. One commonly used tcpwrapper is tcpd. It uses the /etc/hosts.allow and /etc/hosts.deny files for this purpose. They contain a listing of the hosts concerned for each requested service. Here is the logic: If none of the two files exists, then all of the hosts are allowed to use all watched services.

The access control software consults two files. The search stops at the first match:

- Access will be granted when a (daemon,client) pair matches an entry in the /etc/hosts.allow file.
- Otherwise, access will be denied when a (daemon,client) pair matches an entry in the /etc/hosts.deny file.
- Otherwise, access will be granted.

Format of hosts.allow and hosts.deny:

Syntax:

```
daemon: [client1].... [EXCEPT client2 [client3] ....]
```

eg.

```
ALL:           LOCAL @some_netgroup
ALL:           .foobar.edu EXCEPT terminalserver.foobar.edu
in.fingerd:    .mydomain.com EXCEPT hacker.mydomain.com
vsftpd:        .mylocal.domains
```

Wildcards

The access control language supports explicit wildcards:

ALL The universal wildcard, always matches.

LOCAL Matches any host whose name does not contain a dot character.

UNKNOWN Matches any user whose name is unknown, and matches any host whose name or address are unknown. This pattern should be used with care: host names may be unavailable due to temporary name server problems. A network address will be unavailable when the software cannot figure out what type of network it is talking to.

KNOWN Matches any user whose name is known, and matches any host whose name and address are known. This pattern should also be used with care for the same reasons as for UNKNOWN.

PARANOID Matches any host whose name does not match its address. When tcpd is built with -DPARANOID (default), it drops requests from such clients even before looking at the access control tables. Build without -DPARANOID when you want more control over such requests.

See man hosts_access for more information.

- **1.113.2 Operate and perform basic configuration of sendmail**
  Weight: 4

  Description: Candidate should be able to modify simple parameters in sendmail configuration files (including the "Smart Host" parameter, if necessary), create mail aliases, manage the mail queue, start and stop sendmail, configure mail forwarding and perform basic troubleshooting of sendmail. The objective includes checking for and closing open relay on the mail server. It does not include advanced custom configuration of Sendmail.

  Key files, *terms*, and utilities include:

  ```
  /etc/aliases or          mailq
  /etc/mail/aliases        sendmail
  /etc/mail/*              newaliases
  ~/.forward
  ```

  LPI-102 Doesn't require deep knowledge of sendmail. Just the configuration of a simple mail server.

- **Mail system components**

  - **MTA - M**ail **T**ransfer **A**gent
    Programs under Unix/Linux: `postfix`, `sendmail`, `qmail`, `exim`, `smail`
  - **MDA** - Mail Delivery Agent oder **LDA** - **L**ocal **D**elivery **A**gent
    Programs under Unix/Linux: `mail`, `procmail`, `local` (Postfix), `qmail-local`
  - **MUA** - Mail User Agent
    - MUAs under Unix/Linux: `mail`, `pine`, `mutt`, `kmail`(kde), `balsa` (gnome) `evolution` (gnome)
    - MUAs under Windows: `Eudora`, `Outlook Express, MS Outlook, Netscape Composer`

- **SMTP Principle**
  The principle of an SMTP server is that the SMTP daemon runs in the background and watches the port 25. A requesting client initiates the contact and then sends the mails. The server will then either save the mails in the local mailboxes or relay them to their destinations by forwarding to the proper remote SMTP server. In the process of reception of mails, a series of checks as well as mail headers content changes can be performed. It all depends on the configuration of the mail server.

- **`sendmail` configuration file**
  The main configuration file of sendmail is `/etc/sendmail.cf`
  because of its complexity, this file is normally not manually created or edited. Some minor changes to this can be anyway done without complications.

- **`sendmail` mailboxes**
  There are 2 types of mailboxes for `sendmail`:

  - Outgoing queue: `/var/spool/mqueue` or `/var/mqueue`
    Used by MUAs for sending mails. Their content is checked regularly and sent to their destinations. The program `mailq` (symlink to `sendmail`) is used to display the content of this mail queue.

  - Incoming queue: `/var/spool/mail/`*`UserName`*
    Used by the MTA to store the incoming mails.
    The local MUA also looks in this directory (belonging to the user running the MUA) and displays it. The POP and IMAP servers are also looking in this directory to pic-up the mail and send it to the requesting client.

- **Starting `sendmail`**
  The main purpose of sendmail is to receive mail and save the mail in local mailboxes or send the mail to another mail server(relaying). As far as relaying is concerned, there are two possibilities: either the host is permanently connected to the Internet where it has a FQDN name, a fixed IP address or it is temporarily connected to the Internet and used only to send mails.

  Permanent Internet connection:
  In this case sendmail has 2 functions:
  - Receiving mails via SMTP protocol
  - Sending mails via SMTP protocol.

  For receiving mails, sendmail need to be running as daemon in the background and watch the port 25(SMTP). The incoming mails are always received immediately. Normally a command line option is be given for sendmail to regularly check and process the mails in the outgoing mail queue.(`mqueue`)
  eg. `sendmail -bd -q15`
  Sendmail is started as daemon(`-bd`) and checks the outgoing mail queue every 15 minutes.(`-q 15`)

  Temporary Internet connection:
  I this case sendmail is used only to send mails. It is normally called after the Internet connection is been established. The incoming mails are handled by `fetchmail` or other mail retrieving programs.
  eg. `sendmail -q`
  Sendmail is started and checks the mails in the outgoing queue, sends them if any are present and exits when finished. This command can also be regularly be called by a `cron` job.
  Sendmail can also be started in daemon mode to receive incoming mails.
  eg. `sendmail -bd`
  Sendmail starts as daemon watches the SMTP port and receives incoming mails. Outgoing mails can still be regularly be sent via a cron job as above:
  eg. `sendmail -q`

- **Mail Aliases**
  It is possible to redirect incoming mails meant for a user, which doesn't exist in the host, to an existing local user. They are then called aliases to the real user.
  eg. `martin.hoofer@mybestmail.com` sent to the local user `martinh`
  These aliases are written is: `/etc/aliases` or `/etc/mail/aliases`.
  <u>`aliases` file syntax:</u>
  `AliasName: RealUserName`

  ```
  eg.    martin.hoofer: martinh
         webmaster:     root
         abuse:         root
  ```

  Note1: Aliases is only applicable for incoming mails that are meant for local users.
  Note2: When changes are made in this file, you need to issue the command: `newaliases` as well as restart sendmail daemon if needed.

- **Piping mails to programs:**
  It is also possible to send the incoming mail to a specific local program. It is also done using the same above file: `/etc/aliases`.
  eg.

  ```
  harry: "| /usr/bin/mail -s 'Forwarded mail' harry2@remoteserver.com"
  ```

  The mail is sent to the local mail program which sends it further to its proper destination.

- **Redirecting incoming mail**
  The previous example shows one method of redirecting incoming mail to a remote mail server. Another method is via the file `~/.forward`. This file is written in the home directory of the local user and his incoming mails will all be sent to:
  - to another local user, or
    eg. `hans`
  - appended to a local file(the file must be writable by the user) or
    eg. `"/var/spool/mail2/peter"`
  - piped to a local program (like in the above example of `/etc/aliases`)
    eg. `"| /usr/bin/antiviruschk"`

- **Outgoing mail server:**
  Sendmail can be configured to send mail to 2 different types of remote hosts:
  - Direct to the destination address or
  - Via a 'Smart Host' which will take care of sending the mail to its destination. This method is very often used by Internet Mail providers.

  - <u>Direct to the destination address:</u>
    The achieve this, a proper DNS must be reachable to reach the proper destination hosts.
  - <u>Via a 'Smart Host'</u>
    In this case all of the outgoing mails are sent to this host. In turns this 'Smart Host' must have the proper configuration of a DNS to reach the proper destinations.

    The choice between the 2 above method requires an entry in the main sendmail configuration file `/etc/sendmail.cf`:

`DS`*`SmartHostName`*	To send mail via a 'Smart host'
`DS`	To send mail directly to their destinations.

- **Files in `/etc/mail/` Directory**
  Files in this directory are meant to configure sendmail. They are of 2 types:
  - Configuration files in text format.
  - Database files in binary format.(*xxxxx*`.db`)

  The databases are in fact the configuration files converted in binary(`.db`) format. Sendmail doesn't read directly the text configuration files in this directory. They need to be converted using the `makemap` command:
  eg.

  ```
  makemap hash -f /etc/mail/virtusertable.db < /etc/mail/virtusertable
  ```

- **Access control of incoming mail**
  The control of which host is allowed to send mail to the local mail server is done via the file:
  `/etc/mail/access`
  Syntax:

  ```
  Hostname        AccessType
  Domain          AccessType
  IPNo.           AccessType
  ```

  AccessTypes: `OK`, `RELAY,` etc
  eg.

  ```
  127                   RELAY
  192.168.100.46        OK
  localdomain.com       OK
  ```

  Here the localhost sends it mail only from here to outside, the host 192.168.100.46 is allowed to send its mails via this mail server (relaying) as well as the hosts belonging to the domain `localdomain.com`. All other hosts will not be allowed to relay their mails via this server.

- **Converting the sender name**
  The sender address can be modified to reflect a proper known Internet email address before being sent. For example: A mail sent from local `root` user would be written as `root@myhost.local` as sender. This name being unknown to the Internet, it needs to be converted to a known name like `marty@totsi.com` .
  To achieve this the one needs entries in file `/etc/mail/genericstable`.
  Syntax:

  ```
  LocalUserName             InternetMailAddress
  ```

  eg.

  ```
  root                      marty@totsi.com
  root@myhost.local         marty@totsi.com
  michael                   michael.harmon@goofo.de
  michael@server.local      michael.harmon@goofo.de
  ```

- **Mail delivery control**
  When sendmail sends the outgoing mail, it is possible to comntrol via which mail server and protocol the mails will be sent. It is done via entries in the file:
  `/etc/mail/mailertable`
  Syntax:

  ```
      DestinationDomain              Protocol:RemoteServer
  eg: serveroof.com                  SMTP:mail.serveroof.com
      educ.tim.fr                    SMTP:post.mitgoo.com
  ```

  Here the mail to `martin@educ.tim.fr` would be sent to `post.mitgoo.com` using the SMTP protocol.

- **Virtual mail domains or redirections**
  To allow sendmail to support multiple mail domains as destination addresses, as if the local host had many FQDN mail names, or to redirect some mails to another mail address, entries in the file `/etc/mail/virtusertable` are used.
  Syntax:
  ```
      DestEmailAddress    LocalUserName or RemoteMailAddress
  ```
  eg.
  ```
      lorie@tango.com          lorieanne
      mitchell@parto.de        mitchell.soubir@sidoune.com
  ```

  Here mails received for `lorie@tango.com` will be saved in the mailbox of the local user `lorianne`, and mails received for `mitchell@parto.de` will be sent further to the `mitchell.soubir@sidoune.com` destination.

- **Important Note:**
  Whenever changes to any of the above configuration files in `/etc/mail` directory are made, a new corresponding database file needs to be generated using the command `makemap` as described previously in this chapter.
  eg.
  ```
  makemap hash -f /etc/mail/virtusertable.db < /etc/mail/virtusertable
  ```

- **1.113.3 Operate and perform basic configuration of Apache**
  Weight: 4

  Description: Candidates should be able to modify simple parameters in Apache configuration files, start, stop, and restart **`httpd`**, arrange for automatic restarting of **`httpd`** upon boot. Does not include advanced custom configuration of Apache.

  Key files, *terms*, and utilities include:
  `httpd.conf` **`apachectl`** **`httpd`**

- **IMPORTANT NOTE:**
  This section will only list the items that are recommended to learn. There will be some times short descriptions and no more. The explanations will need to be taken from other sources of Apache documentation. For example my Apache course at:
  `http://www.linuxint.net/PDF/63_Apache_Web_Server.pdf`
  The reason for this is that, to properly understand Apache to the required degree for LPI-102, a fair amount of explanations need to be given. Since I had written this above document already, I don't find a proper reason and motivation for creating a reduced version of it.

- **Configuration files:**
  Apache, because of the wish to be backward compatible, possesses 3 main configuration files. They are usually found in: `/usr/local/etc/httpd/conf/` directory. This location is not conform to the standard Linux File Locations. That's why the main daemon is normally started with the option `-f` *`ConfigFilePath`*. Very often it is `/etc/httpd/httpd.conf`.
  The configuration files are:

`httpd.conf`	The main and mostly the only configuration file used today.
`srm.conf`	The resources configurations.
`access.conf`	The configurations for controlling the access of Apache.

- **Directives of Apache configuration file:**
  Apart form the main Apache core kernel directives, most directives used in the configuration file are controlling the functions that comes with each separate module. A module might have only one or more directives that will dictate the way the module is used. Modules can be compiled internally to the main Apache daemon or compiled separately and loaded dynamically when the daemon is started. In a way similar to the Linux kernel, just that the modules are all loaded when Apache daemon starts and they are never unloaded afterwards.
  The thing to remember here is that, if a module is not loaded at the start of apache, then its corresponding directives, if found in the configuration file, will be seen as junk and Apache will not start at all!!! Unfair but true.

- **List of directives to remember.**
  <u>Main server directives:</u>

  `ServerType standalone|inetd`
  Start the server as Daemon or via `inetd`
  `ServerRoot "/usr/local/httpd"`
  Directory where files that are needed by Apache reside.
  `LockFile /var/lock/subsys/httpd/httpd.accept.lock`
  File to prevent Apache running multiple main daemons.

`PidFile /var/run/httpd.pid`

Where the process ID of Apache Daemon is stored.

`Timeout 300`

Number of seconds of no TCP response before stopping the data stream with client.

`KeepAlive On`

Keep the TCP connection with the client after a request.

`MaxKeepAliveRequests 100`

Max 100 simultaneous requests per client on same TCP connection before requiring new TCP handshake.

`KeepAliveTimeout 15`

If idle for 15 sec. end the TCP Connection with client.

`StartServers 1`

Start 1 server ready for requests when daemon starts.

`MinSpareServers 3`

Always keep minimum 3 spare servers for new clients

`MaxSpareServers 6`

Never keep more than 6 spare servers running.

`MaxClients 150`

Maximum 150 requesting clients simultaneously

`MaxRequestsPerChild 0`

Umlimited(0) number of simultaneous requests per client

`Port 80`

Use the standard port 80 for clients HTTP requests.

`User wwwrun`

Servers processes are owned by the user `wwwrun`

`Group nogroup`

Servers processes are owned by the group `nogroup`

`Listen 80`

Same as port 80 but used to assign more than one port to listen to. Normally used to assign the SSL port 443 as well as port 80.

`ServerAdmin root@localhost`

Email address of the Apache server administrator.

`DocumentRoot "/usr/local/httpd/htdocs"`

Root directory for the documents served to clients.

`DirectoryIndex index.html`

Document sent from a directory when only the directory is specified in the HTTP request.

`ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"`

Specify the real path where CGI scripts are stored.
It is almost like an internal Apache symbolic link from `/cgi-bin/` to the real path.

`Options Indexes FollowSymLinks ....`

Options controlling the access to documents in system.

`Order Allow,Deny`

Select the order in which the allowing or denying directives will be treated. The first is normally seen as general rule and the second as exception to the general rule. See the next 2 directives below.

`Allow from 192.168.100.0/24`

Allow requests from clients belonging to this subnet

`Deny from 192.168.100.56`

Deny requests from client having this address

Configuration of Virtual Servers

(Minimum directives)

```
NameVirtualHost 10.230.1.101
<VirtualHost 10.230.1.101>
     DocumentRoot /www2
     ServerName virtual1.mydomain.com
</VirtualHost>
```

- **Start and stop of Apache main Daemon**
  `apachectl` *`command`*
  `Commands are:`

`start`	Starts the Apache daemon: `httpd`
`stop`	Stops the Apache daemon: `httpd`
`restart`	Same as a stop and then a start command .
`graceful`	Sends a HUP Signal to the Daemon to order a re-read of the configuration file.
`configtest`	Checks the syntax of the configuration file. Response: `Syntax Ok`

- **The `httpd` daemon command line options**
  Syntax: `/usr/sbin/httpd` *`-options`*

Options: (Important ones for LPI-102 are highlighted in grey)

`-f` *`ConfigFile`*	Specifies an alternate configuration file.
`-v`	Display Apache's version number
`-L`	List core configuration directives
`-S`	Show virtual hosts settings
`-t`	Run syntax test for configuration files only.
`-X`	Single process foreground debugging mode
`-R`	Specify an alternate location for loadable modules
`-C D`*`irective`*	Processes this directive before reading config files
`-c D`*`irective`*	Processes this directive after reading config files
`-h`	List valid command line options
`-l`	List compiled-in modules
`-d` *`ServerRootDir`*	Specifies an alternate initial `ServerRoot` directory.
`-D` *`name`*	Defines a name for use in `<IfDefine` *`name`*`>` directives. `<IfDefine` *`name`*`>` is used to define different server global settings and chose which one will be read at start-up of Apache.

- **1.113.4 Properly manage the NFS, smb, and nmb daemons**
  Weight: 4

  Description: Candidate should know how to mount remote filesystems using NFS, configure NFS for exporting local filesystems, start, stop, and restart the NFS server. Install and configure Samba using the included GUI tools or direct edit of the `/etc/smb.conf` file (Note: this deliberately excludes advanced NT domain issues but includes simple sharing of home directories and printers, as well as correctly setting the `nmbd` as a WINS client).

  Key files, *terms*, and utilities include:

  ```
  /etc/exports         mount
  /etc/fstab           umount
  /etc/smb.conf
  ```

- **NFS - Network File System**
  The NFS is a File system that allows to mount locally directories existing in remote hosts. Once mounted the remote directory is seen as a local directory by all applications. The difference is that it might take longer to read and write into it. Mostly the NFS is used exclusively between variations of Unixes OS.

- **Mounting an NFS remote directory**
  The method of mounting an NFS is very similar to mounting a local device onto a directory. The command used is also mount.

  ```
  Syntax: mount [-t nfs] RemoteHost:RemoteDir LocalMountPoint \
                     [-o MountOptions]
  eg.     mount -t nfs nfsserver:/public /mnt/public -o ro
  ```

  This command will mount the remote directory `/public` located on the remote host `nfsserver` to the local mountpoint `/mnt/public` with the option ReadOnly(`ro`).

  Note: The same functions apply to this type of mounting as for local device mounting regarding the possibility to insert the above mounting information into the `/etc/fstab`. eg.
  in `/etc/fstab`

  ```
        nfsserver:/public   /public   nfs   ro   0 0
  ```

  To mount it the 1 of the following 2 commands can then be given:

  ```
        mount  nfsserver:/public      or
        mount /public
  ```

- **Setting-up the NFS server**
  To allow a client host to mount a server directory(NFS share) on his directory tree, the server host needs to set the allowance via an NFS server process.
  The configuration file of the NFS server: `/etc/exports`
  This file provides the NFS server with the following information:
  - Local directory(NFS share) allowed to be mounted
  - Which remote hosts are allowed to mount it
  - Mount options for the allowed hosts

  eg: in `/etc/exports`

  ```
        /public 192.168.10.0/255.255.255.0(ro) 192.168.10.45(rw)
  ```

  All the hosts residing in the IP range from 192.168.10.0-192.168.10.255 will be allowed to mount the server's `nfs` share with the option ReadOnly(`ro`) except the host having the IP 192.168.10.45 will have the ReadWrite(`rw`) mount privileges.

- **NFS Server processes**
  On the NFS server host in order to offer NFS shares to clients, some processes need to be constantly running as daemons. In newer kernels the kernel based nfs daemon `knfsd` is normally used. This kernel process just needs to be started using the appropriate provided tools. Another and older method is to run a user space daemon called `rpc.nfsd`. Since nfs is an RPC type service a extra and necessary daemon process needs also to be running: the portmapper `portmap` daemon. Without going into details of the RPC based services and the portmapper's functions, here is how it works:
  When the NFS daemon starts, it registers its name and listening port number to the portmapper. When a client needs to connect with the NFS server daemon, it connects first to the portmapper on port `111` and asks for the port number where the NFS server daemon listen on, and then connects to it; just like a telephone directory assistance. Another daemon which also needs to be running in the background on the server host is the `rpc.mountd`. This daemon works together with either the kernel `knfsd` or the user space daemon `rpc.nfsd` to implement fully the network mount protocol of the NFS service.

  So to resume: To implement the NFS service in a server host, 3 processes need to be running:
  - RPC Portmapper daemon `portmap`.
  - Kernel `knfsd` or userspace `rpc.nfsd` daemon.
  - Mount protocol daemon `rpc.mountd`

  Both NFS and mount daemons use the configuration file `/etc/exports` to identify and control the access to the server NFS shares.
  Here are some other examples of the possible share entries and their options:

```
/           *.berlin.de(rw,no_root_squash)
/cdrom      *(ro) 192.168.10.100(rw)
/home       192.168.0.0/255.255.0.0(rw)
/public     *(rw,sync) *.nebbo.com(ro,sync)
/transfer   192.168.0.0/24(ro,intr)
```

  Note: Make sure that there is no space between the hosts definitions and their corrresponding mount options. A space in this area is used for another host/options pair definition like the NFS definition of the `/public` share above.

- **UID and GID in NFS mounted shares**
  When a client writes a new file or directory into a remotely mounted share, assuming it is mounted with the ReadWrite(`rw`) option, the NFS server will assign the UID and GID of the file or directory being written to the UID and GID of the client user writing it. It means that if the user marry with UID=500 on the client host writes a file in a mounted share, this file will be effectively be written into the servers share directory. The UID on the server's host might be the UID of the user john....Outch!! To be careful with this. One solution would be to make sure that each user on clients hosts also has an account on the NFS server host and that both UID and GIDs of users are the same on the client's host as on the servers host.

- **Squashing UID and GID for ALL**
  On that above principle another solution would be to select which UID and GID all of the written files and directories would end-up having. This is done also through a 'squashing' function in mount options of `/etc/exports` file:

`all_squash`	All files and directories get the nobody's UID and GID
`anonuid=1000`	Sets the NFS's nobody's UID to 1000
`anongid=2000`	Sets the NFS's nobody's GID to 2000

  eg.
  ```
   /public *.dept1.com(rw,all_squash,anonuid=1000,anongid=2000)
  ```

- **Selecting which UIDs and GIDs will be 'squashed'**
  It is also possible to selectively set the UID that will be squashed and let the oters be written as tehy are on the client's hosts. The mounts options are: `squash_uids` and `squash_gids`.
  eg.
  ```
   /public *(rw,squash_uids=0-499,squash_gids=0-100)
  ```
  Makes sure all the files and directories written into this share which originate from users having a UID from 0 to 500 will be written with the UID of user nobody. The same is true for the GIDs from 0 to 100.

- **Mounting share as `root` user**
  A special issue concerning this above NFS UID phenomenon is that if the root user on a client's host writes files or directories in the NFS shares, the effective file's UID and GID will be of the user `nobody` instead of the user `root` (0,0) for obvious security reasons. To turn this security feature OFF (NOT recommended) a special mount option(`no_root_squash`) can be included in the `/etc/export`.
  eg.
  ```
  /public *.myfirma.com(rw) admin.myfirma.com(rw,no_root_squash)
  ```
  would result that only files or directories written by `root` on the client host `admin.myfirma.com` would result in keeping the UID and GID of the user `root` on the NFS server's host': UID=0 and GID=0.

-

- **1.113.5 Setup and configure basic DNS services**
  Weight: 4

  Description: Candidate should be able to configure hostname lookups and troubleshoot problems with local caching-only name server. Requires an understanding of the domain registration and DNS translation process. Requires understanding key differences in configuration files for bind 4 and bind 8.

  Key files, *terms*, and utilities include:

  ```
  /etc/hosts                /etc/nsswitch.conf
  /etc/named.boot (v.4) or  /etc/named.conf (v.8)
  /etc/resolv.conf          named
  ```

- **1.113.7 Set up secure shell (OpenSSH)**
  Weight: 4

  Description: The candidate should be able to obtain and configure OpenSSH. This objective includes basic OpenSSH installation and troubleshooting, as well as configuring **sshd** to start at system boot..

  Key files, *terms*, and utilities include:

  ```
  /etc/hosts.allow    /etc/ssh/sshd_config
  /etc/hosts.deny     /etc/ssh_known_hosts
  /etc/nologin        /etc/sshrc
  sshd                ssh-keygen
  ```

Topic 114: Security

- **1.114.1 Perform security administration tasks**
  Weight: 4

  Description: Candidates should know how to review system configuration to ensure host security in accordance with local security policies. This objective includes how to configure TCP wrappers, find files with SUID/SGID bit set, verify packages, set or change user passwords and password aging information, update binaries as recommended by CERT, BUGTRAQ, and/or distribution's security alerts. Includes basic knowledge of **`ipchains`** and **`iptables`**.

  Key files, *terms*, and utilities include:

  ```
  /proc/net/ip_fwchains         find          socket
  /proc/net/ip_fwnames          ipchains      iptables
  /proc/net/ip_masquerade       passwd
  ```

- **1.114.2 Setup host security**
  Weight: 3

  Description: Candidate should know how to set up a basic level of host security. Tasks include syslog configuration, shadowed passwords, set up of a mail alias for root's mail and turning of all network services not in use.

  Key files, *terms*, and utilities include:

  ```
  /etc/inetd.conf     or   /etc/inet.d/*
  /etc/nologin             /etc/passwd
  /etc/shadow              /etc/syslog.conf
  ```

- **1.114.3 Setup user level security**
  Weight: 1

  Description: Candidate should be able to configure user level security. Tasks include limits on user logins, processes, and memory usage.

  Key files, *terms*, and utilities include:

  ```
  quota          usermod
  ```