

# **Study Guide for**

# **Linux System Administration II**

**Lab work for LPI 102**

**released under the GFDL by LinuxIT**



Copyright (c) 2005 LinuxIT.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being History, Acknowledgements, with the Front-Cover Texts being "released under the GFDL by LinuxIT".

**see full GFDL license agreement on p.137**

## ***Introduction:***

## **Acknowledgements**

The original material was made available by LinuxIT's technical training centre [www.linuxit.com](http://www.linuxit.com). Many thanks to Andrew Meredith for suggesting the idea in the first place. A special thanks to all the students who have helped dilute the technical aspects of Linux administration through their many questions, this has led to the inclusion of more illustrations attempting to introduce concepts in a userfriendly way. Finally, many thanks to Paul McEnery for the technical advice and for starting off some of the most difficult chapters such as the ones covering the X server (101), modems (102), security (102) and the Linux kernel (102).

The manual is available online at <http://savannah.nongnu.org/projects/lpi-manuals/>. Thank you to the Savannah Volunteers for assessing the project and providing us with the Web space.

## **History**

First release (version 0.0) October 2003. Reviewed by Adrian Thomasset.

Second release (revision1) January 2003. Reviewed by Andrew Meredith

Release (version 1.1-test) March 2004. Reviewed by Adrian Thomasset.

Reviewed in January-June 2005 by Adrian Thomasset

## **Audience**

This course is designed as a 3 to 4 days practical course preparing for the LPI 102 exam. It is recommended that candidates have at least one year experience doing Linux administration professionally. However for those who are ready for a challenge the training is designed to provide as much insight and examples as possible to help non specialists understand the basic concepts and command sets which form the core of Linux computing.

## **The LPI Certification Program**

There are currently two LPI certification levels. The first level LPIC-1 is granted after passing both exams LPI 101 and LPI 102. Similarly passing the LPI 201 and LPI 202 exams will grant the second level certification LPIC-2.

There are no pre-requisites for LPI 101 and 102. However the exams for LPIC-2 can only be attempted once LPIC-1 has been obtained.

## **Exam Registration**

## Introduction



In order to register for an LPI exam you first need to get a unique LPI at [www.lpi.org](http://www.lpi.org). You will also need to register with one of the testing organisations such as [www.vue.com](http://www.vue.com) or [www.prometric.com](http://www.prometric.com)

## No Guarantee

The manual comes with no guarantee at all.

## Resources

[www.lpi.org](http://www.lpi.org)  
[www.linux-praxis.de](http://www.linux-praxis.de)  
[www.lpiforums.com](http://www.lpiforums.com)  
[www.tldp.org](http://www.tldp.org)  
[www.fsf.org](http://www.fsf.org)  
[www.linuxit.com](http://www.linuxit.com)

## Notations

Commands and filenames will appear in the text in **bold**.

The <> symbols are used to indicate a non optional argument.

The [ ] symbols are used to indicate an optional argument

Commands that can be typed directly in the shell are highlighted as below

*command*

or



*command*

<b>The Linux Kernel.....</b>	<b>1</b>
1. Kernel Concepts .....	2
2. The Modular Kernel.....	3
3. Routine Kernel Recompilation.....	5
4. Exercises and Summary.....	11
<b>Booting Linux.....</b>	<b>14</b>
1. Understanding Runlevels.....	15
2. Services and Runtime Control Scripts .....	16
3. The joys of inittab.....	18
4 LILO and GRUB.....	19
5. From boot to bash.....	22
6. Exercises and Summary.....	24
<b>Managing Groups and Users.....</b>	<b>26</b>
1. Creating new users.....	27
2. Working with groups.....	28
3. Configuration files.....	30
4. Command options.....	32
5. Modifying accounts and default settings.....	32
6. Exercises and Summary.....	34
<b>Network Configuration.....</b>	<b>36</b>
1. The Network Interface.....	37
2. Host Information.....	38
3. Stop and Start Networking.....	39
4. Routing.....	40
5. Common Network Tools.....	42
6. Exercises and Summary.....	45
<b>TCP/IP Networks.....</b>	<b>48</b>
1. Binary Numbers and the Dotted Quad.....	49
2. Broadcast Address, Network Address and Netmask.....	49
3. Network Classes.....	51
4. Classless Subnets.....	52
5. The TCP/IP Suite.....	53
6. TCP/IP Services and Ports.....	54
7. Exercises and Summary.....	56
<b>Network Services.....</b>	<b>57</b>
1. The inetd daemon (old).....	58
2. The xinetd Daemon.....	59
3. Telnet and FTP.....	60
3. TCP wrappers .....	61
4. Setting up NFS.....	62
5. SMB and NMB.....	64
6. DNS services.....	66
7. Sendmail main Configuration.....	71
8. The Apache server.....	73
9. Exercises and Summary.....	74

---

<b>Bash Scripting.....</b>	<b>78</b>
1. The bash environment.....	79
2. Scripting Essentials.....	81
3. Logical evaluations.....	82
4. Flow Control and Loops.....	83
5. Expecting user input.....	85
6. Working with Numbers.....	85
7. Exercises and Summary.....	86
 <b>Basic Security.....</b>	 <b>88</b>
1. Local Security.....	89
2. Network Security.....	91
3. The Secure Shell.....	95
4. Time Configuration.....	97
5. Exercises and Summary.....	100
 <b>Linux System Administration.....</b>	 <b>102</b>
1. Logfiles and configuration files.....	103
2. Log Utilities.....	105
3. Automatic Tasks.....	106
4. Backups and Compressions.....	108
5. Documentation.....	110
6. Exercises and Summary.....	114

# *The Linux Kernel*

## Prerequisites

- ◆ Understand shell tools and commands (see LPI 101)
- ◆ Experience compiling and installing software from source (see LPI 101)

## Goals

- ◆ Manage Linux kernel modules
- ◆ Configure the kernel source
- ◆ Compile and install a kernel

## Contents

<b>The Linux Kernel.....</b>	<b>1</b>
1. Kernel Concepts .....	2
2. The Modular Kernel.....	3
3. Routine Kernel Recompilation.....	5
3.1 Source extraction.....	5
3.2 Kernel Configuration.....	6
3.3 Kernel Compilation.....	7
3.4 Installing a New Kernel.....	8
3.5 The full kernel version .....	9
3.5 Initial Ramdisks.....	9
3.6 Optional.....	10
3.7 Re-installing LILO.....	10
4. Exercises and Summary.....	11

## 1. Kernel Concepts

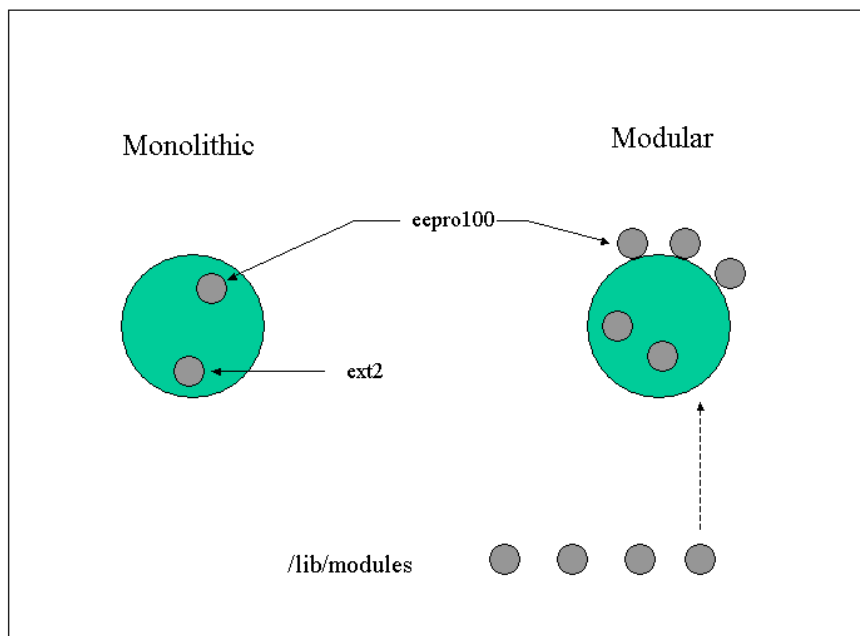
The two different types of Linux kernel are:

**A: Monolithic**

A monolithic kernel is one which has support for all hardware, network, and filesystem compiled into a single image file.

**B: Modular**

A modular kernel is one which has some drivers compiled as object files, which the kernel can load and remove on demand. Loadable modules are kept in **/lib/modules**.



The advantage of a modular kernel is that it doesn't always need to be recompiled when hardware is added or replaced on the system. Monolithic kernels boot slightly faster than modular kernels, but do not outperform the modular kernel



## 2. The Modular Kernel

Many components of the Linux kernel may be compiled as modules which the kernel can dynamically load and remove as required.

- The modules for a particular kernel are stored in **/lib/modules/<kernel-version>**.
- The best components to modularise are ones not required at boot time, for example peripheral devices and supplementary file systems.
- Kernel modules are controlled by utilities supplied by the **modutils** package:
  - **lsmod** list currently loaded modules
  - **rmmmod** remove a single module
  - **insmod** insert a single module
  - **modprobe** insert a module and dependencies listed in **modules.dep**
  - **modinfo** list information about the author, license type and module parameters

Many modules are dependant on the presence of other modules. A flat file database of module dependencies **/lib/modules/<kernel-version>/modules.dep** is generated by the **depmod** command. This command is run at boot time (for example by the **rc.sysinit** script).

-- **modprobe** will load any module and dependent modules listed in **modules.dep** (or **conf.modules**)

Search for example for modules that will be loaded at the same time as **tvaudio**.



```
grep tvaudio /lib/modules/kernel-version/modules.dep
/lib/modules/kernel-version/kernel/drivers/media/video/tvaudio.o: \
/lib/modules/kernel-version/kernel/drivers/i2c/i2c-core.o
```

This means that the module **i2c-core.o** will also be loaded when using **modprobe**. This dependency is also apparent when listing the module with **lsmod**:



```
lsmod
```

Module	Size	Used by	Not tainted
tvaudio	16796	0 (unused)	
i2c-core	19236	0 [tvaudio]	

-- **/etc/modules.conf** is consulted for module parameters (IRQ and IO ports) but most often contains a list of aliases. These aliases allow applications to refer to a device using a common name. For example the first ethernet device is always referred to as **eth0** and not by the name of the particular driver.

Sample `/etc/modules.conf` file

```
alias eth0 e100
alias usb-core usb-uhc
alias sound-slot-0 i810_audio
alias char-major-108 ppp_generic
alias ppp-compress-18 ppp_mppe

# 100Mbps full duplex
options eth0 e100_speed_duplex=4
```

`--modinfo` will give information about modules.



`modinfo tvaudio`

```
filename:      /lib/modules/kernel-version/kernel/drivers/media/video/tvaudio.o
description:   "device driver for various i2c TV sound decoder / audiomux chips"
author:        "Eric Sandeen, Steve VanDeBogart, Greg Alexander, Gerd Knorr"
license:       "GPL"
parm:          debug int
parm:          probe short array (min = 1, max = 48), description "List of
adapter,address pairs to scan additionally"
parm:          probe_range short array (min = 1, max = 48), description "List of
adapter,start-addr,end-addr triples to scan additionally"
parm:          ignore short array (min = 1, max = 48), description "List of
adapter,address pairs not to scan"
parm:          ignore_range short array (min = 1, max = 48), description "List
of adapter,start-addr,end-addr triples not to scan"
parm:          force short array (min = 1, max = 48), description "List of
adapter,address pairs to boldly assume to be present"
parm:          tda9874a_SIF int
parm:          tda9874a_AMSEL int
parm:          tda9874a_STD int
parm:          tda8425 int
parm:          tda9840 int
```

To get information only about parameter option use `modinfo -p`, to get information about the license type use `modinfo -l`, etc.

-- **kmod** is a mechanism that allows the kernel to automatically load modules as needed (one seldom needs to insert modules manually). This is in fact a statically compiled (resident) module that needs to be configured before compiling the kernel. The command used by the kernel to load the modules is defined in `/proc/sys/kernel/modprobe`.

## 3. Routine Kernel Recompilation

### 3.1 Source extraction

The kernel source is stored in the `/usr/src/linux` directory tree, which is a symbolic link to the `/usr/src/(kernel-version)` directory. When extracting a new kernel source archive it is recommended to:

- remove the symbolic link to the old kernel source directory tree



```
rm linux
```

Kernel sources which have been packaged as an RPM often create a link called **linux-2-4**

- extract the new source archive (e.g `linux-2.4.20.tar.bz2`)



```
tar xjf linux-2.4.29.tar.bz2
```

**Note:** The archived 2.2 series kernels create a directory called `linux` instead of `linux-version`. This is why the first step is important, otherwise you may overwrite an old source tree with the new one. Since kernel 2.4 the name of the directory is `linux-version`.

- create a symbolic link called **linux** from the newly created directory



```
ln -s linux-2.4.20 linux
```

- The kernel is almost ready to be configured now, but first we need to make sure that all old binary files are cleared out of the source tree, and this is done with the ***make mrproper*** command.

**Warning:** this command will also delete the kernel configuration file `.config` discussed later.



```
cd /usr/src/linux  
make mrproper
```

**Note:** `mrproper` is a Scandinavian brand of cleaner that gets things “cleaner than clean”, it is one step beyond “make clean”.

### 3.2 Kernel Configuration

First edit the **Makefile** and make sure that the “EXTRAVERSION” variable is different from the existing version:

```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 20
EXTRAVERSION = -test
```

The kernel is now ready to be configured. This essentially means creating a configuration file called **.config**. This is done from the kernel source tree directory **/usr/src/linux** with any of the following

```
make menuconfig
make xconfig
make config
```

All these methods will save the configuration file as **/usr/src/linux/.config**

It is often easier to configure a new kernel using an older **.config** file by using the **make oldconfig** command. This will prompt the user only for new features in the kernel source tree (if the kernel is newer or has been patched).

**Notice:** Some distributions such as RedHat have a **configs** subdirectory containing files to be used as **.config** files with predefined configurations.

To enable kernel features (with **make menuconfig**) you will enter the top level category by moving with the arrow keys and pressing enter to access the desired category. Once in the particular category, pressing the space bar will change the kernel support for a feature or driver.

Possible support types are

- supported (statically compiled) **[\*]**
- modular (dynamically compiled) **[M]**
- not supported **[ ]**

The same choices are available with the other menu editors **config** and **xconfig**.

**Troubleshooting:** The **make menuconfig** target needs the **ncurses** header files. These are provided by the **ncurses-devel** package and must be installed for this target to work.

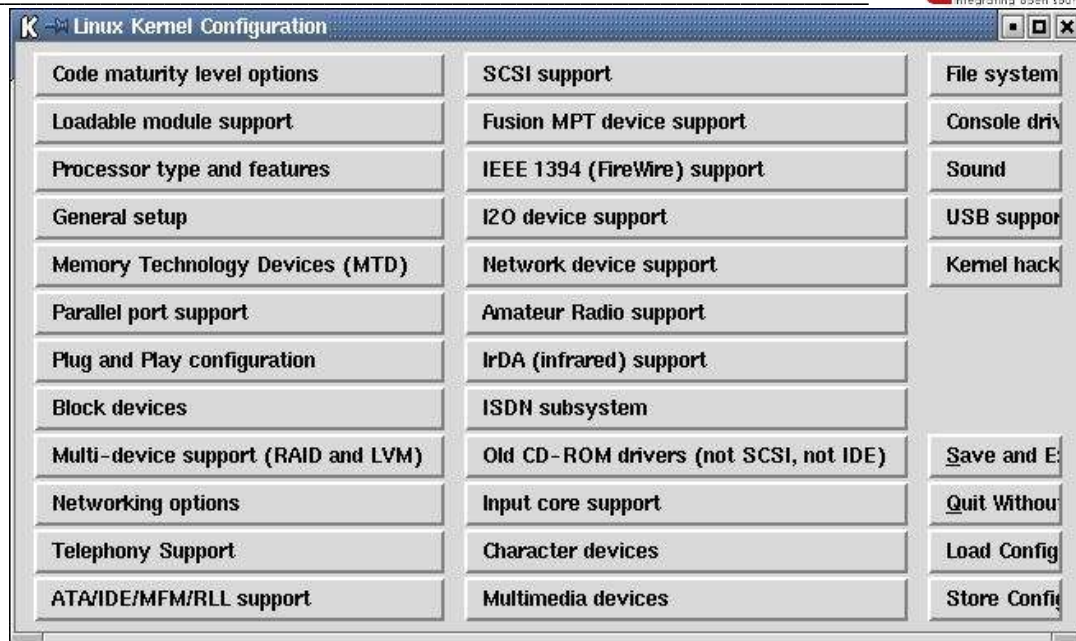


Fig 2: The *make xconfig* top level menu:

### 3.3 Kernel Compilation

#### make clean

The **make** command gets instructions from the **Makefile** and will build what is needed. If some files are already present **make** will use them as is. In particular files with \*.o extensions. To make sure that all the configuration options in **.config** are used to rebuild the files needed one has to run **make clean** (this deletes \*.o files)

**Notice:** you do not need to do “make clean” at this stage if you already prepared the source directory with “make mrproper”

#### make dep

Once the kernel configuration is complete, it is necessary to reflect these choices in all the subdirectories of the kernel source tree. This is done with the **make dep** command. The files named **.depend** containing paths to header files present in the kernel source tree (**/usr/src/linux/include**) are generated this way.

The kernel itself is compiled with one of the commands:

**make zImage**  
**make bzImage**

When the command exits without any errors, there will be a file in the **/usr/src/linux/** directory called **vmlinux**. This is the uncompressed kernel.

The two other commands will write an additional file in `/usr/src/linux/arch/i386/boot/` called **zImage** and **bzImage** respectively. These are compressed kernels using gzip and bzip2. See the next section **Installing the New Kernel** to find out how to proceed with these files.

### make modules

The modules are compiled with **make modules**.

### make modules\_install

Once the modules are compiled they need to be copied to the corresponding subdirectory in `/lib/modules`. The **make modules\_install** command will do that.

The sequence of commands are depicted in Fig 3.

### Kernel compilation commands:

```
make dep
make clean
make bzImage
make modules
make modules_install
```

## 3.4 Installing a New Kernel

The new kernel can be found in `/usr/src/linux/arch/i386/boot/bzImage`, depending on your architecture of your system. This file must be copied to the `/boot` directory, and named **vmlinuz-`<full-kernel-version>`**



```
cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-<full-kernel-version>
```

Next the `/etc/lilo.conf` or `/boot/grub/grub.conf` file needs to be edited to add our newly compiled kernel to the boot menu. Copy the “image” section from your existing kernel and add a new image section at the bottom of the file, as shown below:

### Editing the `/etc/lilo.conf` file

```
prompt
timeout=50
message=/boot/message

image=/boot/vmlinuz
    label=linux
    root=/dev/hda6
    read-only
```

**Existing section**


---

```
image=/boot/vmlinuz-<full-kernel-version>
    label=linux-new
    root=/dev/hda6
    read-only
```

**Added section**

```
-----snip-----
```

The symbol table for the various kernel procedures can be copied to the /boot directory:



```
cp /usr/src/linux/System.map /boot/System.map-<full-kernel-version>
```

### 3.5 The full kernel version

On a system, the version of the running kernel can be printed out with

**uname -r**


This kernel version is also displayed on the virtual terminals if the **Vk** option is present in **/etc/issue**.

### 3.5 Initial Ramdisks

If any dynamically compiled kernel modules are required at boot time (e.g a scsi driver, or the filesystem module for the root partition) they will be loaded using an initial ramdisk.

The initial ramdisk is created with the **mkinitrd** command which only takes two parameters: the filename, and the kernel version number.

If you use an initial ramdisk then you will need to add an **initrd=** line in your **/etc/lilo.conf**



```
mkinitrd /boot/initrd-full-version.img full-version
```

### 3.6 Optional

It is recommended to copy the `/usr/src/linux/.config` file to `/boot/config-<full-kernel-version>`, just to keep track of the capabilities for the different kernels that have been compiled.

### 3.7 Re-installing LILO

Finally lilo needs to be run in order to update the boot loader . First ***lilo*** can be run in test mode to see if there are any errors in the configuration file:

NOTICE
The LILO bootloader needs to be updated using <b><i>lilo</i></b> every time a changed is made in <code>/etc/lilo.conf</code>



## 4. Exercises and Summary

Files	Description
/etc/modules.conf	used by <b>modprobe</b> before inserting a module
/lib/modules/<kernel-version>/	directory where the modules for given kernel version are stored
/lib/modules/<kernel-version>/modules.dep	list of module dependencies created by <b>depmod</b>

Command	Description
depmod	<b>depmod(8)</b> – kernel modules can provide services (called "symbols") for other modules to use (using EXPORT_SYMBOL in the code). If a second module uses this symbol, that second module clearly depends on the first module. <b>Depmod</b> creates a list of module dependencies, by reading each module under /lib/modules/version and determining what symbols it exports, and what symbols it needs. By default this list is written to <b>modules.dep</b> in the same directory
insmod	<b>insmod(8)</b> – a trivial program to insert a module into the kernel: if the filename is a hyphen, the module is taken from standard input. Most users will want to use <b>modprobe(8)</b> instead, which is cleverer
make clean	delete all object files in the source tree
make config	configure the Linux kernel
make dep	creates a list of extra headers in files called .depend needed to satisfy module dependencies
make menuconfig	configure the Linux kernel using a menu
make modules	compile all the external/dynamic modules for this kernel
make modules_install	install the compiled modules in /lib/module/kernel-version
make oldconfig	create a default <b>.config</b> if it doesn't exist. If a .config file already exists the chosen configuration is unchanged. If the source tree has changed, for example after a patch (see LPI 201) or the .config file corresponds to an older kernel, then extra configuration options must be supplied
make xconfig	configure a Linux kernel using a menu
lsmod	list all dynamically loaded modules
modinfo	print information about a kernel module such as the author ( <b>-a</b> ), the description ( <b>-d</b> ), the license ( <b>-l</b> ) or parameters ( <b>-p</b> )
modprobe	<b>modprobe(8)</b> - will automatically load all base modules needed in a module stack, as described by the dependency file <i>modules.dep</i> . If the loading of one of these modules fails, the whole current stack of modules loaded in the current session will be unloaded automatically
rmmod	<b>rmmod(8)</b> – tries to unload a set of modules from the kernel, with the restriction that they are not in use and that they are not referred to by other modules

Before starting with the exercises make sure you don't have an existing kernel tree in `/usr/src/`. If you do, pay attention to the `/usr/src/linux` symbolic link.

1. Manually recompile the kernel following the compilation steps.

- Get the **kernel-version.src.rpm** package from an FTP mirror site or a CD. Installing this package will also give you a list of dependencies, such as the **gcc** compiler or **binutils** package if they haven't yet been met.

- Install the package with **-i** (this will put all the code in `/usr/src/` )

- Go into the `/usr/src/linux-version` directory and list the **configs** directory

- Copy the kernel config file that matches your architecture into the current directory and call it `.config`

- Run

```
make oldconfig
```

at the command line to take into account this new `.config` file.

- Edit the Makefile and make sure the version is not the same as your existing kernel. You can get information on your current kernel by running **uname -a** at the command line or list the `/lib/modules` directory.

- Run

```
make menuconfig (or menu or xconfig)
```

and remove ISDN support from the kernel.

- When you exit the above program the `.config` file is altered but the changes have not yet taken place in the rest of the source tree. You next need to run

```
make dep
```

- Finally to force new object files (`.o`) to be compiled with these changes you delete all previously compiled code with

```
make clean
```

- You can now build the kernel the modules and install the modules with:

```
make bzImage modules modules_install
```

- The modules are now installed in the `/lib/modules/version` directory. The kernel is called **bzImage** and is in the following directory:

`/usr/src/linux/arch/i386/boot/`

We need to manually install this kernel (2 steps):

(i)

```
cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-<full-kernel-version>
```

(ii) That was easy! We next edit the bootloader configuration file:

- if you are using LILO, edit `/etc/lilo.conf` and add an 'image' paragraph that will tell LILO where to find this kernel and the root filesystem. Run `/sbin/lilo` and reboot
- if your are using GRUB, edit `/boot/grub/grub.conf` or `/boot/grub/menu.lst`

2. Since we downloaded the `kernel-version.src.rpm` package we can now use this package to recompile a 'RedHat preconfigured' kernel. Notice that although no intervention is needed you won't be able to change the `.config` menu.

- First rebuild the compiled binary package with

```
rpm --rebuild kernel-version.src.rpm (...wait!)
```

- This will eventually generate the `kernel-version.i386.rpm` in `/usr/src/redhat/RPMS/i386/`.
- Next, upgrade you kernel with the RPM manager using the `-U` option.

# Booting Linux

## Prerequisites

None

## Goals

- ◆ Manage services (e.g mail, webserver, etc) using runlevels
- ◆ Understand the role of the **init** process and its configuration file **/etc/inittab**
- ◆ Recognise the three phases of the booting process: Bootloader, Kernel and Init

## Contents

<b>Booting Linux.....</b>	<b>14</b>
1. Understanding Runlevels.....	15
2. Services and Runtime Control Scripts .....	16
3. The joys of inittab.....	18
4 LILO and GRUB.....	19
5. From boot to bash.....	23
6. Exercises and Summary.....	24

## Overview

Taking a closer look at the booting process helps troubleshooting when dealing with both hardware and software problems.

We first focus on the role of the **init** program and its' associated configuration file **/etc/inittab**. The role of LILO or GRUB is investigated in greater depth. Finally we summarise the booting process. The document "From Power to Bash Prompt" written by Greg O'Keefe as well as the boot(7) manpage are both good references for this module.

## 1. Understanding Runlevels

Unlike most non-UNIX operating systems which only have 2 modes of functionality (on and off), UNIX operating systems, including Linux, have different runlevels such as "maintenance" runlevel or "multi-user" runlevel, etc.

Runlevels are numbered from 0 to 6 and will vary from one Linux distribution to another. The description for each runlevel functionality is sometimes documented in **/etc/inittab**.

Example Linux runlevels
Runlevel 0 <b>shuts down</b> the machine safely the operating system will also attempt to poweroff the system if possible
Runlevel 1 is <b>single user</b> mode only one terminal is available for the (single) user <b>root</b> all other users are logged out
Runlevel 2 is <b>multi-user</b> mode, but <i>does not start</i> NFS most network services like email or web services are also stopped
Runlevel 3 is <b>full multi-user</b> mode. Selected network services are all on
Runlevel 4 is not defined and generally unused
Runlevel 5 is like runlevel 3 but <i>runs a Display Manager as well</i>
Runlevel 6 <b>restarts</b> the machine safely

Highlighted runlevels **0**, **1** and **6** offer to the same functionalities for all Linux flavours.

### INIT Controls Runlevels

Both **init** and **telinit** are used to switch from one runlevel to another. Remember that **init** is the first program launched after the kernel has accessed the root device.

At boot time **init** is instructed which runlevel to reach in **/etc/inittab** with the line:


```
id:5:initdefault:
```

When the system is started it is possible to change runlevels by invoking **init** (or **telinit** which is a symbolic link pointing at **init**).

For example we switch to runlevel 4 with either of the next commands:



```
init 4
```



```
telinit 4
```

The PID for **init** is always '1'. It is possible to find out which runlevel the system is currently in with the command **runlevel**



```
runlevel
```


N 5

The first number is the previous runlevel (or N if not applicable) and the second number is the current runlevel.

## 2. Services and Runtime Control Scripts

Each runlevel is characterised by a set of services that are either started or stopped. The services are controlled by runtime control scripts kept in **/etc/rc.d/init.d** or **/etc/init.d**. Each rc-script will control the daemon associated with the service using an argument.


Example: restarting the **apache** server:



```
/etc/rc.d/init.d/httpd restart
```

Expected arguments	
restart	do stop the start
stop	stop the daemon associated with the service
start	start the service
status	return the status of the services (running or stopped)

*Typical services in /etc/rc.d/init.d/*



```
ls /etc/rc.d/init.d/
anacron cups identd kadmin krb5kdc mcscv nscd random smb xfs
apmd dhcpd innd kdcrotate kudzu named ntpd rawdevices snmpd xinetd
arpwatch functions ipchains keytable ldap netfs pcmcia rhnsd squid
atd gpm iptables killall linuxconf network portmp rhod sshd
autofs halt irda kprop lpd nfs pgsql sendmail syslog
crond httpd isdn krb524 marsrv nfslock pppoe single tux
```

Once a service is started it will run until a new runlevel is started.

### Selecting Services per Runlevel

We will follow what happens when we switch from one runlevel to another.

Say you want to be in runlevel 2, you would type:




```
/sbin/init 2
```

This in turn forces **init** to read its configuration file **/etc/inittab**. We will look at this file in detail in the next section. For now we are concerned with the single line in **/etc/inittab** that will start all the services::

```
L2:2:wait:/etc/rc.d/rc 2
```

The "**/etc/rc.d/rc 2**" command will start scripts in **/etc/rc.d/rc2.d** starting with an **S** and will stop of services starting with a **K**. The next sample listing shows that the **httpd** daemon will be stopped, while the **syslogd** daemon



```
ls /etc/rc.d/rc2.d/ -l | egrep "httpd|syslog"
lrwxrwxrwx 1 root root 15 Mar 23 21:01 /etc/rc.d/rc2.d/K15httpd -> ../init.d/httpd
lrwxrwxrwx 1 root root 16 Mar 20 20:03 /etc/rc.d/rc2.d/S12syslog -> ../init.d/syslog
```

One can also see that the scripts are *symbolic links* pointing to the rc-scripts in **/etc/rc.d/init.d**.

Therefore, if you don't want a process to run in a given runlevel N you can delete the corresponding symlink in **/etc/rc.d/rN.d** beginning with a S and add one beginning with a K.

### Runtime Editors (not an LPI objective)

A runtime editor will automatically manage these symbolic links allowing a system administrator to switch a service *on* or *off* per runlevel as needed. Once again different distributions use different tools. Since the LPI certification is vendor independent none of these tools are examinable.

### 3. The joys of inittab

As promised we next take a closer look at **/etc/inittab**.

The file has the following structure:

id : runlevel : action : command
----------------------------------

#### The /etc/inittab file

```
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
-----snip-----
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
-----snip-----
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```

The **id** field can be anything. If a **runlevel** is specified then the **command** and the required **action** will be performed only at that specific runlevel. If no number is specified then the line is executed at *any* run level.

Recognisable features in the /etc/inittab file:

**The default runlevel:** this is set at the beginning of the file with the id **id** and the action **initdefault**. Notice that no command is given. This line simply tells **init** what the default runlevel is.

**First program called by init:** /etc/rc.d/rc.sysinit. This script sets system defaults such as the PATH variable, determines if networking is allowed, the hostname, etc ...



## Booting Linux



**Default runlevel services:** If the default runlevel is 3 then only the line "l3" will be executed. The action is "wait", no other program is launched until all services in run level 3 are running.

**The getty terminals:** The lines with id's 1-to-6 launch the virtual terminals. This is where you can alter the number of virtual terminals.

**Runlevel 5:** The final line in **inittab** launches the Xwindow manager if runlevel 5 is reached.

### Remarks:

1. You can set a modem to listen for connections in **inittab**. If your modem is linked to /dev/ttyS1 then the following line will allow data connections (no fax) after 2 rings:

```
S1:12345:respawn:/sbin/mgetty -D -x 2 /dev/ttyS1
```

2. When making changes to **/etc/inittab** you need to force **init** to reread this configuration file. This is most easily done using:



```
/sbin/init q
```

## 4 LILO and GRUB

During boot-up, boot loaders need to know where the kernel is (usually in /boot) and which device is the root-device.

```
BOOTLOADER ----> KERNEL ----> / ----> /sbin/init
```

Alternatively, a boot loader can load a RAM disk into memory containing scripts and kernel modules needed to access the root device. This will be the case when the root-device is handled by non-resident (also called dynamic) modules.

```
BOOTLOADER ----> INITRD ----> KERNEL ----> / ----> /sbin/init
```

### Common dynamic modules

<b>ext3</b>	Third extended filesystem type
-------------	--------------------------------

## Booting Linux



<b>lvm</b>	Logical volume support
<b>raidx</b>	software raid level x support
<b>scsi</b>	SCSI support

### ● Installing LILO

The bootloader LILO is installed by **/sbin/lilo** (the bootloader mapper or installer) which in turn reads configuration options from the file **/etc/lilo.conf**.

LILO cannot read filesystems, only offsets on the physical disks. Therefore the mapper will read information from the **/etc/lilo.conf** file (e.g which second stage bootloader to use, which kernel or which initial ram disk) and will translate this information using a system of maps for LILO to read at boot time.

The main options in **/etc/lilo.conf** are specified here

<b>boot*</b>	where LILO should be installed (/dev/hda is the MBR)
<b>install</b>	which second stage to install ( <b>boot.b</b> is the default)
<b>prompt</b>	give the user a chance to choose an OS to boot
<b>default</b>	name of the image that will be booted by default
<b>timeout</b>	used with prompt, causes LILO to pause (units are 1/10 of a sec)
<b>image*</b>	path to the kernel to boot (one can use 'other' to chain load)
<b>label*</b>	name of the image. This is the name a user can type at the boot prompt
<b>root*</b>	the name of the disk device which contains the root filesystem /
<b>read-only*</b>	mount the root filesystem read-only for <b>fsck</b> to work properly
<b>append</b>	give kernel parameters for modules that are statically compiled.
<b>linear/lba3</b>	these options are mutually exclusive. Both ask LILO to read the disk using Linear Block Addressing. <b>linear</b> is typically used for very large disks. <b>lba32</b> is used to allow boot time access to data beyond the first 1024 cylinders (also see p.Error: Reference source not found)

### ● Installing GRUB

The GRUB boot loader is installed with the command **grub-install**. Configuration options are stored in the file **/boot/grub/menu.lst** or **/boot/grub/grub.conf**. Unlike LILO, GRUB is a small shell that can read certain filesystem. This allows GRUB to read information in the **grub.conf** or **menu.lst** files.

Main sections in **/boot/grub/grub.conf** or **menu.lst**

#### 1. General/Global

<b>default</b>	image that will boot by default (the first entry is 0)
<b>timeout</b>	prompt timeout in seconds

#### 2. Image

## Booting Linux



<b>title</b>	name of the image
<b>root</b>	where the 2 <sup>nd</sup> stage bootloader and kernel are e.g (hd0,0) is /dev/hda
<b>kernel</b>	path for the kernel starting from the previous root e.g /vmlinuz
<b>ro</b>	read-only
<b>root</b>	the filesystem root
<b>initrd</b>	path to the initial root disk

### ● Bootloader Options

It is possible to give parameters at boot time to both LILO and GRUB. Both loaders have a limited interface which can read user input.

#### Passing parameters at the LILO prompt:

```
boot: linux s
```

#### Passing parameters at the GRUB prompt:

Once the GRUB boot loader has successfully started you will see the main menu screen with a list of menu titles.

Do the following:

1. press 'e' to edit a given menu title
2. scroll down to the line containing 'kernel' and press 'e' again
3. you can add any options here
4. to boot with the current options type 'b' – Otherwise just press return to get the unaltered line back

Notice that pressing the ESC key will bring you back to a previous stage. You can navigate back to the main menu this way.

Alternatively the boot loader configuration files (**lilo.conf** or **grub.conf**) can be used to save these option

### ● Passing init parameters:

Boot loaders can pass the runlevel parameter to **init**. Once the kernel is loaded, it will start **/sbin/init** by default which then takes over the booting process.

Common runlevels are **s,single,S,1,2,3,4,5**

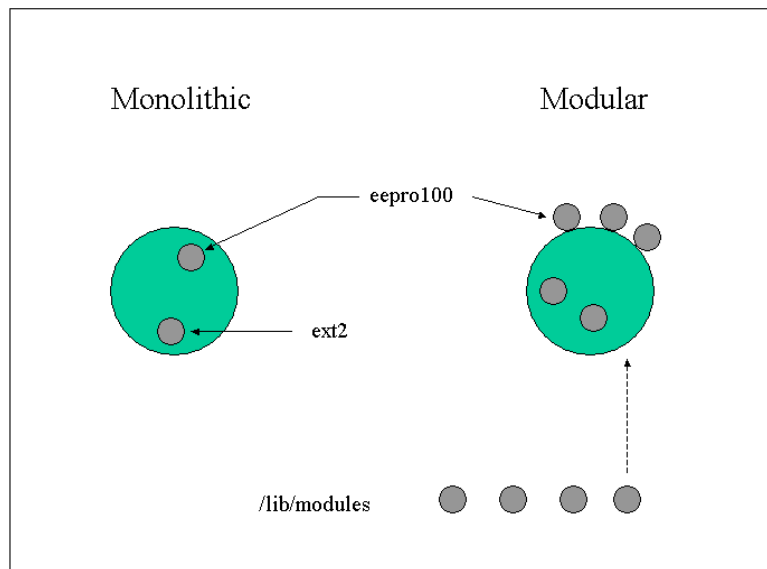
If no parameters are given, **init** will launch the default runlevel specified in **/etc/inittab**.

### ● Passing Kernel parameters:

Kernel options are of the form *item=value*.

Common kernel parameters	
<b>acpi=</b>	enable/disable ACPI
<b>init=</b>	tell the kernel which program to start from the root device

<b>mem=</b>	specify amount of RAM to use
<b>root=</b>	specify the root device



**Warning!** The boot loader kernel parameters are passed to the resident kernel modules only.

In `/etc/lilo.conf` kernel parameters are declared with the **append** option.

#### Examples

```
append= "pci=bisoirq"
append="ram=16M"
append="/dev/hdc=ide-scsi"    (for CD writers)
```

During bootup all kernel messages are logged to `/var/log/dmesg` by default. This file can either be read or flushed to stdout with the `/bin/dmesg` utility.

## 5. From boot to bash

We can now attempt to go through each stage of the booting process.

### 1. Boot Loader stage:

If the bootloader is successful it will start its second stage which displays a prompt or a splash image with a list of operating systems or kernels to boot

If an initial ram disk is specified it is loaded here.  
The kernel is loaded into memory

### 2. Kernel Stage

The kernel is loaded from the medium, specified in the **lilo.conf/grub.conf** configuration file. As it loads it is decompressed. If an initial ramdisk is loaded, extra modules are loaded here

The kernel will scan the hardware in the system: CPU, RAM, PCI bus, etc  
The kernel then mounts the root device as read-only.  
From here on programs in **/bin** and **/sbin** are made available.  
The kernel then loads **/sbin/init** - the first 'userspace' process.

### 3. The INIT stage

Init reads **/etc/inittab** and follows the instructions  
the default runlevel is read  
the **rc.sysinit** is run:

- all local filesystems are mounted or, if needed, an integrity check (**fsck**) is performed in accordance with entries in **/etc/fstab**
- quotas are started, etc ...

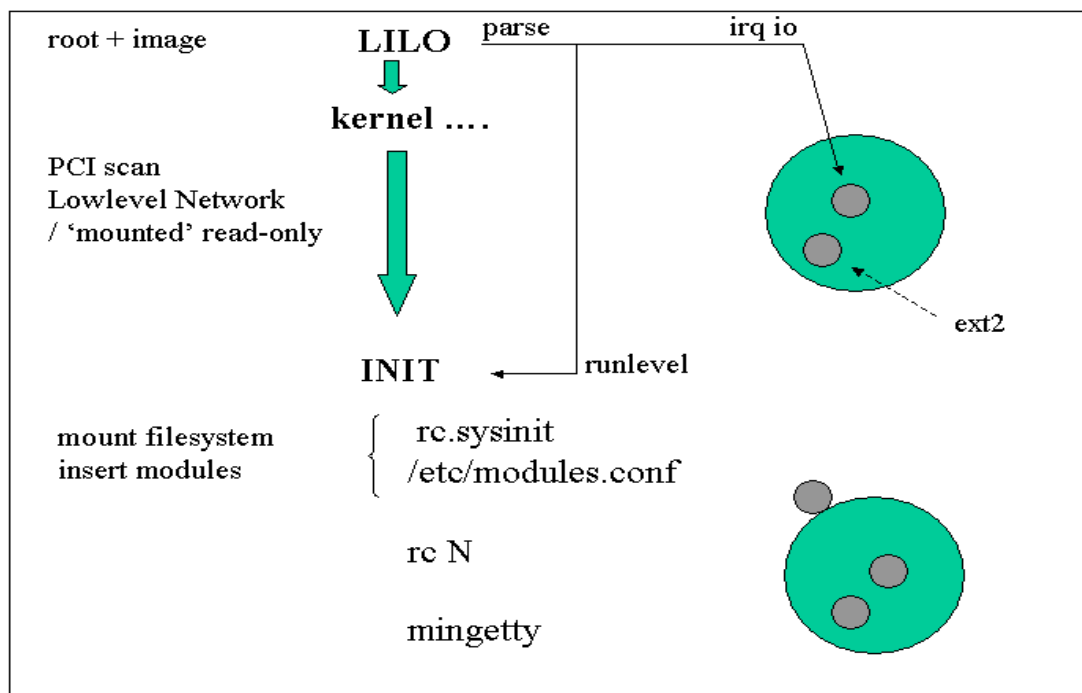
next **init** goes into the default runlevel **/etc/rc.d/rc N**  
the **gettys** start and the boot process is over

The prompt to login is now managed by the gettys on the ttys. After the user has typed in their username and pressed return;

`/bin/login` is started.

The user is prompted by `/bin/login` for the password. The user enters a password and presses return.

The password the user is compared to the password in `/etc/passwd` or `/etc/shadow`.



## 6. Exercises and Summary

### Files

Files	Description
/etc/init.d	directory containing all the scripts used to stop and start services at boot time
/etc/inittab	<b>inittab(5)</b> - The inittab file describes which processes are started at boot-up and during normal operation. Init distinguishes multiple runlevels, each of which can have its own set of processes that are started

### Commands

Commands	Description
init	<b>init(8)</b> – is the parent of all processes. Its primary role is to create processes from a script stored in the file <b>/etc/inittab</b>
shutdown	<b>shutdown(8)</b> – brings the system down in a secure way. All logged-in users are notified that the system is going down, and login(1) is blocked. It is possible to shut the system down immediately or after a specified delay. All processes are first notified that the system is going down by the signal SIGTERM. This gives programs like vi(1) the time to save the file being edited, mail and news processing programs a chance to exit cleanly, etc. shutdown does its job by signalling the init process, asking it to change the runlevel

### References

Take a look at the **boot(7)** manpage, it covers most of what we did in this module

### Exercices

1. Use **init** to change you current runlevel (e.g switch between runlevel 3 and 5).  
How do you know what your current runlevel is?
2. Enable the Ctrl+Alt+Del in runlevel 3 only.  
How can you force **init** to read its' configuration file?
3. Add a new login prompt on tty7.
4. Use **dmesg** to read the chipset of your ethernet card.
5. Investigate differences between **shutdown**, **halt** and **reboot**.  
Which option to **shutdown** will force an **fsck** at the next boot?
6. Use the tools **chkconfig** or **ntsysv** to disable the **sshd** daemon in runlevel 2,3,4, and 5  
Verify that the symbolic links in the rc2.d, rc3.d, rc4.d and rc5.d directories have changed.
7. Reboot the system. At the boot prompt give the appropriate **init=** parameter to skip **/sbin/init** and start a simple bash session.

---

## *Managing Groups and Users*

### Prerequisites

- ◆ None

### Goals

- ◆ Manage user accounts
- ◆ Manage group accounts
- ◆ Modify accounts settings

### Contents

<b>Managing Groups and Users.....</b>	<b>26</b>
1. Creating new users.....	27
2. Working with groups.....	28
3. Configuration files.....	30
4. Command options.....	32
5. Modifying accounts and default settings.....	32
6. Exercises and Summary.....	34



## 1. Creating new users

### Step 1: Create an account

The **/usr/sbin/useradd** command adds new users to the system and the symbolic link **adduser** points to it.

Syntax:

```
useradd [options] login-name
```

Example: add a user with login-name *rufus*



```
useradd rufus
```

Default values will be used when no options are specified. You can list these values with **useradd -D**.

Default options listed with **useradd -D**

```
GROUP=100  
HOME=/home  
INACTIVE=-1  
EXPIRE=  
SHELL=/bin/bash  
SKEL=/etc/skel
```

Notice that this information is also available in the file **/etc/default/useradd**

### Step 2: Activate the account with a new password

To allow a user to access his or her account the administrator must allocate a password to the user using the **passwd** tool.

Syntax:

```
passwd login-name
```

These steps create a new user. This has also defined the user's environment such as a *home directory* and a *default shell*. The user has also been assigned to a group, his *primary* group.

## 2. Working with groups

Every new user is assigned to an initial (or *primary*) group. Two conventions exist.

Traditionally this *primary* group is the same for all users and is called **users** with a group id (GID) of **100**. Many Linux distributions adhere to this convention such as Suse and Debian.

The User Private Group scheme (UPG) was introduced by RedHat and changes this convention without changing the way in which UNIX groups work. With UPG each new user belongs to their own *primary* group. The group has the same name as the login-name (default), and the GID is in the 500 to 60000 range (same as UIDs).

As a consequence, when using the traditional scheme for groups the user's **umask** (see LPI 101) is set to **022**, whereas in the UPG scheme the **umask** is set to **002**.

### Belonging to groups

A user can belong to any number of groups. However at any one time (when creating a file for example) only one group is the *effective* group.

The list of all groups a user belongs to is obtained with either the **groups** or **id** commands.

Example for user root:

### List all ID's:



`id`

```
→ ► uid=0(root) gid=0(root) groups=0(root), 1(bin), 2(daemon), 3(sys),
    4(adm), 6(disk), 10(wheel), 600(sales)
```

### List all groups:



`groups`

```
→ ► root bin daemon sys adm disk wheel sales
```

### Joining a group

Joining a group changes the user's *effective* group and starts a new session from which the user can then logout. This is done with the **newgrp** command.

Example: joining the *sales* group



```
newgrp sales
```

If the **groups** command is issued, the first group on the list would no longer be *root* but *sales*.

### Creating and deleting groups

The **groupadd** tool is used to add new groups. It will add an entry in the */etc/group* file.

Example: Create the group *devel*



```
groupadd devel
```

The **groupdel** tool is used to delete groups. This will remove relevant entries in the */etc/group* file.

Example: Delete the group *devel*



```
groupdel devel
```

### Adding a user to a group

Administration tasks can be carried out with the **gpasswd** tool. One can add (**-a**) or remove (**-d**) users from a group and assign an administrator (**-A**). The tool was originally designed to set a single password on a group, allowing members of the same group to login with the same password. For security reasons this feature no longer works.

Example: Add *rufus* to the group *devel*



```
gpasswd -a rufus devel
```

### 3. Configuration files

#### The /etc/passwd and /etc/shadow files:

The names of all the users on the system are kept in **/etc/passwd**. This file has the following structure:

1. Login name
2. Password (or x if using a shadow file)
3. The UID
4. The GID
5. Text description for the user
6. The user's home directory
7. The user's shell

These 7 fields are separated by colons. As in the example below.

#### /etc/passwd entry with encrypted passwd:

```
george:$1$K05gMbOv$b7ryoKGTd2hDrW2sT.h:Dr G Micheal:/home/georges:/bin/bash
```

In order to hide the encrypted passwords from ordinary users you should use a shadow file. The **/etc/shadow** file then holds the user names and encrypted passwords and is readable only by root.

If you don't have a shadow file in /etc then you should issue the following command:

A small icon of a computer monitor with a terminal window.

```
/usr/sbin/pwconv (passwd -> shadow)
```

This will leave an 'x' in the 2<sup>nd</sup> field of /etc/passwd and create the /etc/shadow file. If you don't wish to use shadow passwords you can do so using

A small icon of a computer monitor with a terminal window.

```
/usr/sbin/pwunconv (shadow -> passwd)
```

**Caution:** When using a shadow password file the **/etc/passwd** file may be world readable (644) and the **/etc/shadow** file must be more restricted (600 or even 400). However when using **pwunconv** make sure to change the permissions on **/etc/passwd** (600 or 400).

#### The /etc/group and gshadow files:

## Managing Groups and Users



In the same way, information about groups is kept in **/etc/group**. This file has 4 fields separated by colons.

1. Group name
2. The group password (or x if gshadow file exists)
3. The GID
4. A comma separated list of members

Example **/etc/group** entry:

```
java:x:550:jade, eric, rufus
```

As for users there is a **/etc/gshadow** file that is created when using shadow group passwords. The utilities used to switch backwards and forward from shadow to non-shadow files are as follow



**/usr/sbin/grpconv**

creates the **/etc/gshadow** file



**/usr/sbin/grpunconv**

deletes the gshadow file

### The **/etc/login.defs** and **/etc/skel/** files

The **/etc/login.defs** file contains the following information:

- the mail spool directory:  
MAIL\_DIR
- password aging controls:  
PASS\_MAX\_DAYS, PASS\_MIN\_DAYS, PASS\_MAX\_LEN, PASS\_WARN\_AGE
- max/min values for automatic UID selection in **useradd**:  
UID\_MIN, UID\_MAX
- max/min values for automatic GID selection in **groupadd**:  
GID\_MIN, GID\_MAX
- automatically create a home directory with **useradd**:  
CREATE\_HOME

The **/etc/skel** directory contains default files that will be copied to the home directory of newly created users:  
**.bashrc**, **.bash\_profiles**, ...

## 4. Command options

**useradd (options)**

<b>-c</b>	comment (Full Name)
<b>-d</b>	path to home directory
<b>-g</b>	initial group (GID). The GID must already exist
<b>-G</b>	comma separated list of supplementary groups
<b>-u</b>	user's UID
<b>-s</b>	user's default shell
<b>-p</b>	password (md5 encrypted, use quotes!)
<b>-e</b>	account expiry date
<b>-k</b>	the skel directory
<b>-n</b>	switch off the UPG group scheme

**groupadd (options)**

<b>-g</b>	assign a GID
-----------	--------------

**5. Modifying accounts and default settings**

All available options while creating a user or a group can be modified. The **usermod** utility has the following main options:

**usermod (options)**

<b>-d</b>	the users directory
<b>-g</b>	the users initial GID
<b>-l</b>	the user's login name
<b>-u</b>	the user's UID
<b>-s</b>	the default shell.

Notice these options are the same as for **useradd**.

Likewise, you can change details about a group with the **groupmod** utility. There are mainly two options:

**groupmod (options)**

<b>-g</b>	the GID
<b>-n</b>	the group name.

Locking an account

- A user's account can be locked by prefixing an exclamation mark to the user's password. This can also be done with the following command line tools:

Lock	Unlock
<b>passwd -l</b>	<b>passwd -u</b>

---

`usermod -L``usermod -U`

---

- When using shadow passwords, replace the **x** with a **\***
- A less useful option is to remove the password entirely with **passwd -d**.
- Finally, one can also assign **/bin/false** to the user's default shell in **/etc/passwd**.

#### Changing the password expiry dates:

By default a user's password is valid for 99999 days, that is 273,9 years (default `PASS_MAX_DAYS`). The user is warned for 7 days that his password will expire (default `PASS_WARN_AGE`) with the following message as he logs in:

```
Warning: your password will expire in 6 days
```

There is another password aging policy number that is called `PASS_MIN_DAYS`. This is the minimum number of days before a user can change his password; it is set to zero by default.

The **chage** tool allows an administrator to change all these options.

```
Usage: chage [ -l ] [ -m min_days ] [ -M max_days ] [ -W warn ]  
[ -I inactive ] [ -E expire ] [ -d last_day ] user
```

The first option **-l** lists the current policy values for a user. We will only discuss the **-E** option. This locks an account at a given date. The date is either in UNIX days or in YYYY/MM/DD format.

Notice that all these values are stored in the **/etc/shadow** file, and can be edited directly.

#### Removing an account:

A user's account may be removed with the **userdel** command line. To make sure that the user's home directory is also deleted use the **-r** option.



```
userdel -r jade
```

## 6. Exercises and Summary

### Files

File	Description
/etc/group	contains the names of all the groups on the system
/etc/gshadow	contains (optionally) passwords associated to a group
/etc/login.defs	contains predefined values needed when adding a new user such as the minimum and maximum UID and GID, the minimum password length, etc
/etc/passwd	<b>passwd(5)</b> – text file that contains a list of the system's accounts, giving for each account some useful information like user ID, group ID, home directory, shell, etc. Often, it also contains the encrypted passwords for each account. It should have general read permission (many utilities, like ls(1) use it to map user IDs to user names), but write access only for the superuser
/etc/shadow	<b>shadow(5)</b> – contains the encrypted password information for user's accounts and optional the password aging information
/etc/skel/	directory containing files and directories to be copied into the home directory of every newly created user

### Commands

Commands	Description
chage	<b>chage(1)</b> – changes a user's password expiry information
gpasswd	<b>gpasswd(1)</b> – administer the /etc/group file
groupadd	add a new group to the system
groupmod	modify an existing group
groups	print out all the groups a user belongs to
id	print out the UID as well as the GIDs of all the groups a user belongs to
passwd	change the password for an account
useradd	add a new user to the system
usermod	modify an existing user account

#### 1. Creating users

Use **adduser** to create a user called *tux* with user ID 600 and group ID 550

Use **usermod** to change this user's home directory.

Does the new directory need to be created? (Hint: check the effect of the **-m** flag)

Is the content of */etc/skel* copied to the new directory?

Use **usermod** to add *tux* to the group wheel.

#### 2. Working with groups



Create a group called sales using **groupadd**.

Add tux to this group using **gpasswd**.

Login as tux and join the group sales using **newgrp**.

### 3. Configuration files

Add a user to the system by editing `/etc/passwd` and `/etc/group`

Create a group called share and add user tux to this group by manually editing `/etc/group`

### 4. Modifying an Account

Change the expiry date for user tux's account using **usermod**.

Lock the user's account. (Use `tools` or edit `/etc/shadow` ...)

Prevent the user from login in by changing the user's default shell to `/bin/false`

Change the `PASS_MAX_DAYS` for user tux to 1 in `/etc/shadow`

### 5. Changing default settings

Use **useradd -D** to change the system's default settings such that every new user will be assigned `/bin/sh` instead of `/bin/bash`. (Notice that this will change the file in `/etc/defaults/`)

Edit `/etc/login.defs` and change the default `PASS_MAX_DAYS` so that new users need to change their password every 5 days

# Network Configuration

## Prerequisites

- ◆ Hardware configuration (see LPI 101)

## Goals

- ◆ Configure a Linux system for networking
- ◆ Understand routing
- ◆ Use network troubleshooting tools

## Contents

<b>Network Configuration.....</b>	<b>36</b>
1. The Network Interface.....	37
2. Host Information.....	38
3. Stop and Start Networking.....	39
4. Routing.....	40
5. Common Network Tools.....	42
6. Exercises and Summary.....	45

## 1. The Network Interface

The network interface card (NIC) must be supported by the kernel. To determine which card you are using you can get information from **dmesg**, **/proc/interrupts**, **/sbin/lsmmod** or **/etc/modules.conf**

Example:



*dmesg*

```
► Linux Tulip driver version 0.9.14 (February 20, 2001)
   PCI: Enabling device 00:0f.0 (0004 -> 0007)
   PCI: Found IRQ 10 for device 00:0f.0
   eth0: Lite-On 82c168 PNIC rev 32 at 0xf800, 00:A0:CC:D3:6E:0F, IRQ 10.
   eth0: MII transceiver #1 config 3000 status 7829 advertising 01e1.
```



*cat /proc/interrupts*

```
►  0:   8729602          XT-PIC  timer
   1:         4          XT-PIC  keyboard
   2:         0          XT-PIC  cascade
   7:         0          XT-PIC  parport0
   8:         1          XT-PIC  rtc
  10:   622417          XT-PIC  eth0
  11:         0          XT-PIC  usb-uhci
  14:   143040          XT-PIC  ide0
  15:    180          XT-PIC  ide1
```



*/sbin/lsmmod*

```
►  Module              Size  Used by
   tulip                37360   1 (autoclean)
```

From the example above we see that the Ethernet card's chipset is Tulip, the i/o address is 0xf800 and the IRQ is 10. This information can be used either if the wrong module is being used or if the resources (i/o or IRQ) are not available.

This information can either be used to insert a module with a different i/o address (using the **modprobe** or **insmod** utilities) or can be saved in **/etc/modules.conf** (this will save the settings for the next system boot).

## 2. Host Information

The following files are used to store networking information.

- **/etc/resolv.conf** contains a list of DNS servers

```
nameserver 192.168.1.108
nameserver 192.168.1.1
search linuxit.org
```

- **/etc/HOSTNAME** or **/etc/hostname** is used to give a name to the PC
- One can also associate a name to a network interface. This is done in differently across distributions.
- **/etc/hosts** contains your machine's IP number as well as a list of known hosts

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1    localhost localhost.localdomain
# other hosts
192.168.1.108 mesa mesa.domain.org
192.168.1.119 pico
```

- **/etc/sysconfig/network** defines if networking must be started. (can also contain the HOSTNAME variable)

```
NETWORKING=yes
HOSTNAME=mesa.domain.org
GATEWAY=192.168.1.1
GATEWAYDEV=
```

- **/etc/sysconfig/network-scripts/ifcfg-eth0** The configuration parameters for eth0

```
DEVICE=eth0
BOOTPROTO=none
BROADCAST=192.168.1.255
IPADDR=192.168.1.108
```

```
NETWORK=192.168.1.0
```

```
ONBOOT=yes
```

```
USERCTL=no
```


### 3. Stop and Start Networking

- From the command line


The main tool used to bring up the network interface is **/sbin/ifconfig**. Once initialised the kernel module aliased to **eth0** in **/etc/modules.conf** (e.g **tulip.o**) is loaded and assigned an IP and netmask value.

As a result the interface can be switched on and off without losing this information as long as the kernel module is inserted.


Examples: Using **ifconfig**.



```
/sbin/ifconfig eth0 192.168.10.1 netmask 255.255.128.0
```



```
/sbin/ifconfig eth0 down
```




```
/sbin/ifconfig eth0 up
```

Another tool is **/sbin/ifup**. This utility reads the system's configuration files in **/etc/sysconfig/** and assigns the stored values for a given interface. The script for **eth0** is called **ifcfg-eth0** and has to be configured. If a boot protocol such as DHCP is defined then **ifup** will start the interface with that protocol.


Examples: Using **ifup**.



```
/sbin/ifup eth0
```



```
/sbin/ifup ppp0
```



```
/sbin/ifdown eth0
```

- Using the network script

At boot time the ethernet card is initialised with the **/etc/rc.d/init.d/network** script. All the relevant networking files are sourced in the **/etc/sysconfig/** directory.

In addition the script also reads the **sysctl** options in **/etc/sysctl.conf**, this is where you can configure the system as a router (allow IP forwarding in the kernel). For example the line:

```
net.ipv4.ip_forward = 1
```

will enable ip forwarding and the file **/proc/sys/net/ipv4/ip\_forward** will contain a one.

The **network** script is started with the following command

```
/etc/rc.d/init.d/network restart
```

- Renewing a DHCP lease

The following tools can query the DHCP server for a new IP:

**pump**

**dhcpcclient**

A client daemon exists called **dhcpcd** (do not confuse this with the DHCP server daemon **dhcpcd**)

## 4. Routing

A noticeable difference when using a system script such as **ifup** rather than **ifconfig** on its own, is that the system's routing tables are set in one case and not in the other.

This is because either the **/etc/sysconfig/network** file is read, where a **default gateway** is stored, or the DHCP server has sent this information together with the IP number. The routing tables are configured, checked and changed with the **/sbin/route** tool.

Routing examples:

Add a static route to the network 10.0.0.0 through the device eth1 and use 192.168.1.108 as the gateway for that network:



```
/sbin/route add -net 10.0.0.0 gw 192.168.1.108 dev eth1
```

Add a default gateway:



```
/sbin/route add default gw 192.168.1.1 eth0
```

Listing the kernel routing table:



```
/sbin/route -n
```

► Kernel IP routing table

Destination	Gateway	Genmask	Iface
192.168.1.0	0.0.0.0	255.255.255.0	eth0
10.1.8.0	192.168.1.108	255.0.0.0	eth1
127.0.0.0	0.0.0.0	255.0.0.0	lo
0.0.0.0	192.168.1.1	0.0.0.0	eth0

**Default Gateway:**

In the last listing, the Destination field is a list of networks. In particular, 0.0.0.0 means 'anywhere'. With this in mind, there are two IP's in the Gateway field. Which one is the default gateway ?

☞ To avoid having to enter static routes by hand special daemons **gated** or **routed** are run to dynamically update routing tables across a network

If you belong to the 192.168.10.0 network and you add a route to the 192.168.1.0 network you may find that machines in the latter network are not responding. This is because no route has been set from the 192.168.1.0 network back to your host!! This problem is solved using dynamic routing.

### Permanent Static Routes

If you have several networks with more than one gateway you can use the `/etc/sysconfig/static-routes` (instead of routing daemons). These routes will be added at boot time by the **network** script.

### Naming Networks

Using the `/etc/networks` file it is possible to assign names to network numbers (for network numbers see TCP/IP Networks on p. 48).

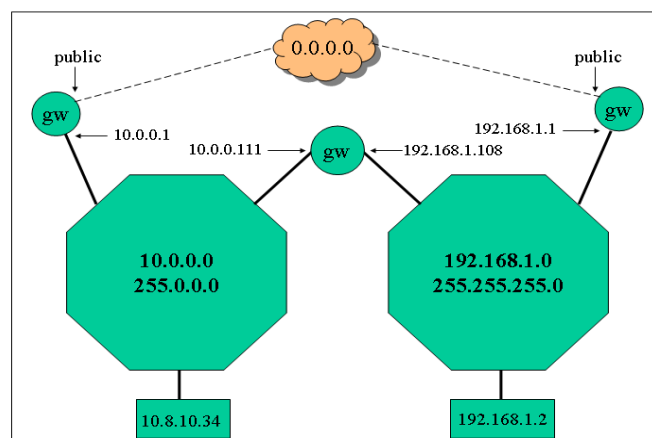
<code>/etc/networks</code> format	
network-name	network-number aliases

For example, the network number 10.0.0.0 can be called `office.org`, following the above format. It is then possible to use network names with tools like **route** as below:



```
route add -net office.org netmask 255.0.0.0
```

*A routing scenario:*



10.8.10.34		192.168.1.2	
network	gw	network	gw
10.0.0.0	0.0.0.0	192.168.1.0	0.0.0.0
0.0.0.0	10.0.0.1	0.0.0.0	192.168.1.1
route add 192.168.1.0 \ netmask 255.255.255.0 \ gw 10.0.0.111		route add 10.0.0.0 \ netmask 255.0.0.0 \ gw 192.168.1.108	
network	gw	network	gw
10.0.0.0	0.0.0.0	192.168.1.0	0.0.0.0
192.168.1.0	10.0.0.111	10.0.0.0	192.168.1.108
0.0.0.0	10.0.0.1	0.0.0.0	192.168.1.1

## 5. Common Network Tools

Here is a short list of tools helpful when trouble shooting network connections.

### **ping:**

This tool sends an ICMP `ECHO_REQUEST` datagram to a host and expects an ICMP `ECHO_RESPONSE`.

Options for ping	
<b>-b</b>	ping a broadcast address
<b>-c N</b>	send N packets
<b>-q</b>	quiet mode: display only start and end messages

### **tcpdump:**

This is a tool used to analyse network traffic by capturing network packets. The following commands illustrate some options:

Let tcpdump autodetect network interface
<code>tcpdump</code>
Specify a network interface to capture packets from
<code>tcpdump -i wlan0</code>
Give an expression to match
<code>tcpdump host 192.168.10.1 and port 80</code>



Notice that in a switched environment the switch may be configured to send packets to a given network interface only if those packets were addressed to that interface. In that case it is not possible to monitor the whole network.

### **netstat:**

You may get information on current network connections, the routing table or interface statistics depending on the options used.

Options for <b>netstat</b> :	
<b>-r</b>	same as /sbin/route
<b>-l</b>	display list of interfaces
<b>-n</b>	don't resolve IP addresses
<b>-p</b>	returns the PID and names of programs (only for root)
<b>-v</b>	verbose
<b>-c</b>	continuous update

Example: Output of netstat --inet -n :

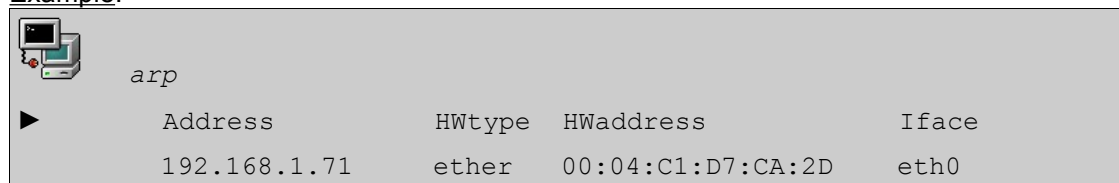
```
► Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address      Foreign Address    State
tcp        0      0 192.168.1.10:139   192.168.1.153:1992 ESTABLISHED
tcp        0      0 192.168.1.10:22    192.168.1.138:1114 ESTABLISHED
tcp        0      0 192.168.1.10:80    192.168.1.71:18858 TIME_WAIT
```

In the above listing you can see that the local host has established connections on ports 139, 22 and 80.

### **arp:**

Display the kernel address resolution cache.

Example:



```
arp
► Address          HWtype  HWaddress          Iface
  192.168.1.71     ether   00:04:C1:D7:CA:2D  eth0
```

---

**traceroute:**

Displays the route taken from the local host to the destination host. Traceroute forces intermediate routers to send back error messages (ICMP `TIME_EXCEEDED`) by deliberately setting the `ttl` (time to live) value too low. After each `TIME_EXCEEDED` notification **traceroute** increments the `ttl` value, forcing the next packet to travel further, until it reaches its' destination.

Options for <b>traceroute</b> :	
<b>-f</b> <code>ttl</code>	change the initial time to live value to <code>ttl</code> instead of 1
<b>-n</b>	do not resolve IP numbers
<b>-v</b>	verbose
<b>-w</b> <code>sec</code>	set the timeout on returned packets to <code>sec</code>

## 6. Exercises and Summary

### Files

File	Description
/etc/resolv.conf	file containing a list of DNS servers used to resolve computer host names
/etc/sysctl.conf	configuration file for the sysctl tool used to customise kernel settings in <b>/proc/sys/</b>
/proc/sys/net/ipv4/ip_forward	file containing information about the kernel forwarding status. The kernel will either forward or not packets that are addressed to a different host depending if the file contains a 1 or a 0

### Commands

Command	Description
arp	print the kernel ARP cache
dhcpcd	a DHCP client daemon
dhcpcclient	a DHCP client daemon
ifconfig	<b>ifconfig(8)</b> – is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary
netstat	<b>netstat(8)</b> – print information about network connections, routing tables, interface statistics, etc
ping	<b>ping(8)</b> – uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of "pad" bytes used to fill out the packet
pump	<b>pump(8)</b> – is a daemon that manages network interfaces that are controlled by either the DHCP or BOOTP protocol. While pump may be started manually, it is normally started automatically by the /sbin/ifup script for devices configured via BOOTP or DHCP
route	<b>route(8)</b> – manipulates the kernel's IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the ifconfig(8) program. When the add or del options are used, route modifies the routing tables. Without these options, route displays the current contents of the routing tables
sysctl	<b>sysctl(8)</b> – is used to modify kernel parameters at runtime. The parameters available are those listed under /proc/sys/
traceroute	<b>traceroute(8)</b> - utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host

1. In the **Routing Scenario** section of this chapter give the routing table for the LAN's gateway.

2. Start your network interface manually

```
ifconfig eth0 192.168.0.x
```

List the kernel modules. Make sure that the eth0 module is loaded (check /etc/modules.conf).

3. Stop the network interface with:

---

(i) `ifconfig eth0 down`

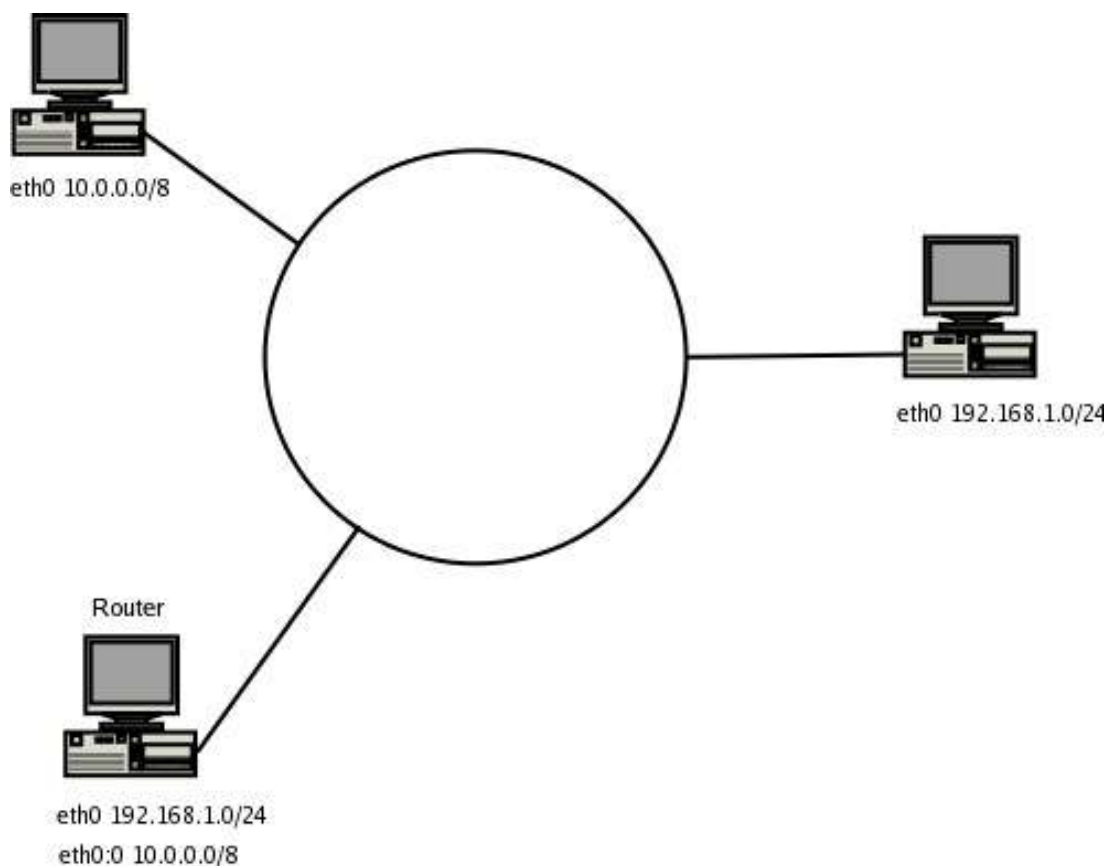
Verify that you can bring the interface back up without entering new information:

(ii) `ifconfig eth0 up`

4. Stop the interface and remove the kernel module (`rmmod module`). What happens if you repeat step 3(ii)?

5. Divide the class into two networks A (192.168.1.0) and B (10.0.0.0).

#### First Senario – at least 3 hosts



- Try accessing machines across networks (this shouldn't work!)
- Choose an existing machine to be the gateway (on either network)

If you choose the router to be on the existing 192.168.1.0 network then do the following on that router:

-- create an aliased interface on the 10.0.0.0 network (x is any available number)

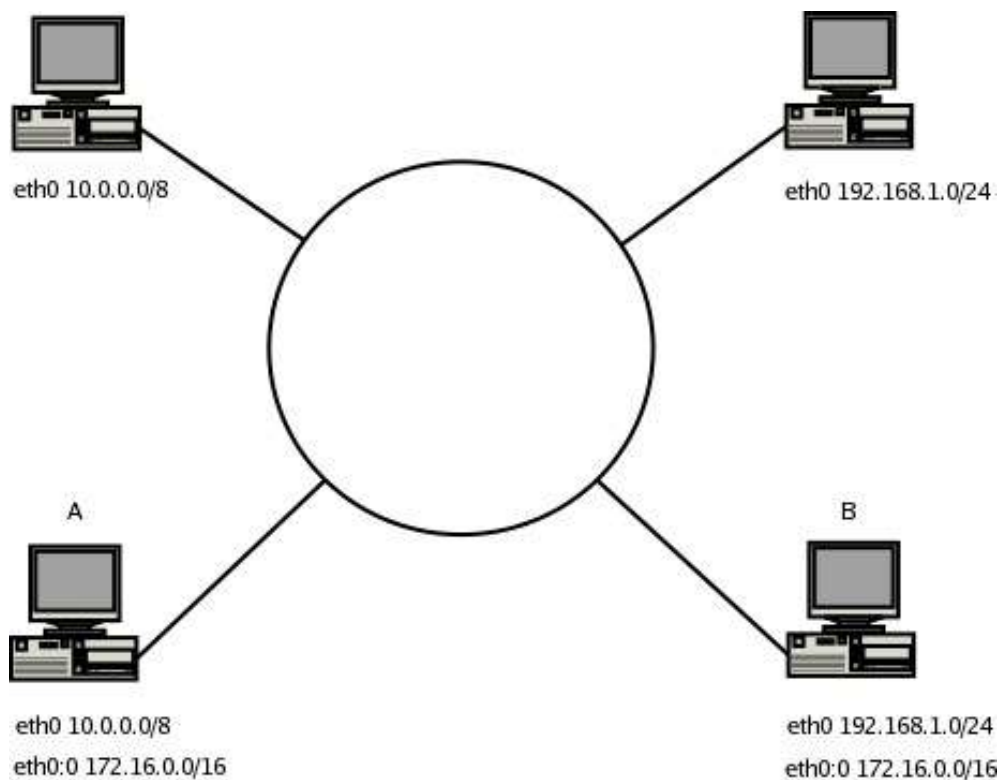
```
ifup eth0:1 10.0.0.x
```

-- allow IP forwarding

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

-- add a route to the other network using the gateway machine (you will need to know either the eth0 or eth0:1 setting of the gateway depending on which network you are on).

### Second scenario – at least 4 hosts



Make sure the routers force routing through the aliased interface. For example on router A;

```
route add -net 192.168.1.0/24 gw 172.16.0.10 dev eth0:0
```

---

# ***TCP/IP Networks***

## **Prerequisites**

- ◆ Network configuration (p. 36)

## **Goals**

- ◆ Understand formal TCP/IP network concepts
- ◆ Manage subnets
- ◆ Understand the four layer TCP/IP model
- ◆ Introduce service port numbers

## **Contents**

<b>TCP/IP Networks.....</b>	<b>48</b>
1. Binary Numbers and the Dotted Quad.....	49
2. Broadcast Address, Network Address and Netmask.....	49
3. Network Classes.....	51
4. Classless Subnets.....	52
5. The TCP/IP Suite.....	53
6. TCP/IP Services and Ports.....	55
7. Exercices and Summary.....	56

## 1. Binary Numbers and the Dotted Quad

### Binary numbers

$10 = 2^1$	$100 = 2^2$	$101 = 2^2 + 1$	$111 = 100 + 010 + 001$
------------	-------------	-----------------	-------------------------

This means that a binary number can easily be converted into a decimal as follows:

10000000	=	$2^7$	=	128
01000000	=	$2^6$	=	64
00100000	=	$2^5$	=	32
00010000	=	$2^4$	=	16
00001000	=	$2^3$	=	8
00000100	=	$2^2$	=	4
00000010	=	$2^1$	=	2
00000001	=	$2^0$	=	1

### The Dotted Quad:

The familiar IP address assigned to an interface is called a dotted quad. In the case of an ipv.4 address this is 4 bytes (4 times 8 bits) separated by dots.

Decimal	Binary
192.168.1.1	11000000.10101000.00000001.00000001

## 2. Broadcast Address, Network Address and Netmask

An IP number contains information about both the host address (or interface) and network address.

### ● The Netmask

A netmask is used to define which part of the IP address is used for the network, it is also called a subnet mask.

A 16 bit and 17 bit netmask:

255.255.0.0	16-bit	1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . 0 0 0 0 0 0 0 0 . 0
255.255.128.0	17-bit	1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . 1 0 0 0 0 0 0 0 . 0

The netmask is usually given in decimal.

Example: with a 16-bit netmask the following IPs are on the same networks:

00100000	.	10000000	.	00000001	.	00000001
00100000	.	10000000	.	00000000	.	00000011

This means that any bits that are changed **inside** the box (8+8=16 bits) will change the network address and the interfaces will need a gateway to connect to each other.

In the same way, any bits that are changed **outside** the box will change the interface address without changing networks.

For example with a 24-bit netmask the above two IPs would be on different networks:

00100000	.	10000000	.	00000001	.	00000001
00100000	.	10000000	.	00000000	.	00000011

### ● The Network Address

Every network has a number which is needed when setting up routing. The network number is a portion of the dotted quad. The host address portion is replaced by zero's.

Typical network address: 192.168.1.0

### ● The Broadcast Address

A machine's broadcast address is a range of hosts/interfaces that can be accessed on the same network. For example a host with the broadcast address 10.1.255.255 will access any machine with an IP address of the form 10.1.x.x. Typical broadcast: 192.168.1.255

The dotted quad revisited

Simple logical operations can be applied to the broadcast, netmask and network numbers.

To retrieve the network address from an IP number simply AND the IP with the netmask..

Network Address	=	IP	AND	Netmask
-----------------	---	----	-----	---------

Similarly the broadcast address is found with the network address OR 'not MASK'.

Broadcast Address	=	Network	OR	not[Netmask]
-------------------	---	---------	----	--------------

Here AND and OR are logical operations on the binary form of these addresses

### Example:

Take the IP **192.168.3.5** with a net mask **255.255.255.0**. We can do the following operations:

Network address	=	IP	AND	MASK
		11000000. 10101000.00000011.00000101		(192.168.3.5)



AND

---

 11111111.11111111.11111111.00000000 (255.255.255.000)
 

---

 11000000.10101000.00000011.00000000 (**192.168.3.0**)
Broadcast Address

=

IP

OR

NOT-MASK

11000000. 10101000.00000011.00000101 (192.168.3.5)

OR

 00000000.00000000.00000000.11111111 (000.000.000.255)
 

---

 11000000.10101000.00000011.11111111 (**192.168.3.255**)

It is clear from the above example that an IP number together with a netmask is enough to retrieve all the information relative to the network and the host.

### 3. Network Classes

- Reserved IP addresses

For private networks a certain number of IP addresses are allocated which are never used on the Internet. These reserved IP's are typically used for LAN's.

The following table displays the various private/reserved classes.

Table1: Reserved addresses

<b>1</b>	Class A	10.x.x.x
<b>16</b>	Class B	172.16.x.x -- 172.31.x.x
<b>255</b>	Class C	192.168.o.x

- IP classes

**Class A:** 8-bit network address and 24-bit host address

The first byte of the IP number is reserved for the network address. So the default subnet mask would be **255.0.0.0**. The 3 remaining bytes are available to set host interfaces.

Since 255.255.255 and 0.0.0 are invalid host numbers there are  $2^{24} - 2 = 16\,777\,214$  possible hosts. IP numbers have the first byte ranging from **1** to **127**. This corresponds to a binary range of 00000001 to 01111111. The first two bits of a class A address can be set to "00" or "01".

**Class B:** 16-bit network address and 16-bit host address

The two first bytes of the IP number are reserved for the network address. The default subnet mask is **255.255.0.0**. There are  $2^{16} - 2 = 65\,534$  possible hosts.

The first byte ranges from **128** to **191**. Notice that the binary range of the first byte is 10000000 to 10111111. That is the first two bits of a class B address are always set to "10".

**Class C:** 24-bit network address and 8-bit host address

The three first bytes are reserved for the network address. The default subnet mask is **255.255.255.0**. There are  $2^8 - 2 = 254$  possible hosts.

The first byte ranges from **192** to **223**. This corresponds to a binary range from 11000000 to 11011111.

From this we conclude that the first two bits of a class C address is always set to "11".

## 4. Classless Subnets

Subnetting occurs when bits reserved for hosts are used for the network. This is determined by the netmask and results in networks being split.

For example a regular class A netmask 255.0.0.0 can be altered to allow the first 1-bit of the second byte to be part of the network. This results in a 9-bit network address and a 23-bit host address IP.

The binary netmask looks like

11111111.10000000.00000000.00000000 or 255.128.0.0

**Slash Notation**

A network can be described using a slash notation. The following notations are equivalent:

10.0.0.0/9

network 10.0.0.0, netmask 255.128.0.0

We will take the example of a class C address **192.168.1.0**. We investigate a 25-bit then a 26-bit network.

**25-bit network**

Netmask: 11111111.11111111.11111111.**10000000** or 255.255.255.128

Since Network = IP AND Netmask, we see from the netmask that two network addresses can be formed depending on the hosts range:

1. Host addresses in the 192.168.1.**0xxxxxxx** range result in a 192.168.1.**0** network. We say the network number is 0
2. Host addresses in the 192.168.1.**1xxxxxxx** range result in a 192.168.1.**128** network. We say the network number is 128

*In both cases substitution of the x's by zeros or ones have a special meaning*

Network address	Substitute with 0's	Substitute with 1's
<b>0</b>	Network: 0	Broadcast: 127
<b>128</b>	Network: 128	Broadcast: 255

We are left with the task of counting the number of hosts on each network. Since the host address is 7-bit long and we exclude 2 values (all 1's and all 0's) we have  $2^7 - 2 = 126$  hosts on each network or a total of 252 hosts.

Notice that if the default subnet mask 255.255.255.0 is used we have 254 available host addresses. In the above example 192.168.1.127 and 192.168.1.128 are taken for the first broadcast and second network respectively, this is why only 252 host addresses can be used.

### 26-bit network

Netmask: 11111111.11111111.11111111.**11000000** or 255.255.255.192

Here again depending on the host's address 4 different network addresses can be determined with the AND rule.

1. Host addresses in the 192.168.1.**00xxxxxx** range result in a 192.168.1.**0** network.
2. Host addresses in the 192.168.1.**01xxxxxx** range result in a 192.168.1.**64** network.
3. Host addresses in the 192.168.1.**10xxxxxx** range result in a 192.168.1.**128** network.
4. Host addresses in the 192.168.1.**11xxxxxx** range result in a 192.168.1.**192** network.

Substituting the x's with 1's in the numbers above give us the corresponding broadcast addresses:

192.168.1.63, 192.168.1.127, 192.168.1.191, 192.168.1.255

Each subnet has  $2^6 - 2 = 62$  possible hosts or a total of 248.

## 5. The TCP/IP Suite

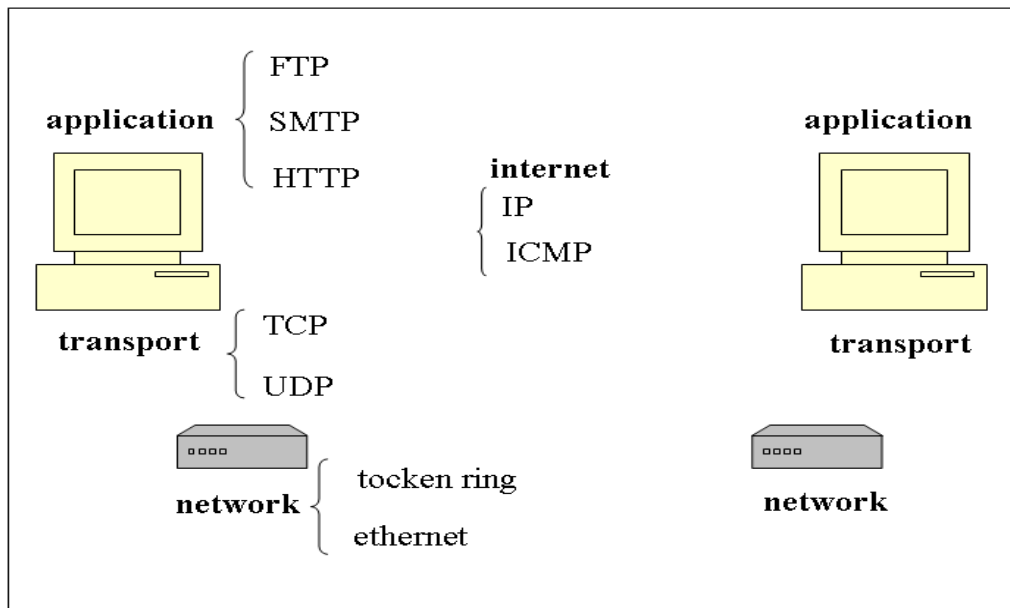
TCP/IP is a suite of protocols used on the Internet. The name is meant to describe that several protocols are needed in order to carry data and programs across a network. The main two protocols are TCP **Transmission Control Protocol** and IP **Internet Protocol**.

To simplify, IP handles packets or datagrams only (destination address, size...) whereas TCP handles the connection between two hosts. The idea is that protocols relay each other, each one doing its' specialised task. In this context one speaks of the TCP/IP stack.

The protocols intervene therefore at various layers of the networking process.

The 4 layer TCP/IP model:

Application	application level (FTP, SMTP, SNMP)
Transport	handles hosts (TCP, UDP)
Internet	routing (IP, ICMP, IGMP, ARP)
Network Access	network cards, e.g Ethernet, token ring ...



#### ● Protocol Overview

<b>IP</b>	The Internet Protocol (IP) is the transport for TCP, UDP, and ICMP data. IP Provides an unreliable connectionless service, allowing all integrity to be handled by one of the upper layer protocols, i.e. TCP, or some application-specific devices. There is no guarantee that a datagram will reach the host using IP alone. The IP protocol handles the addressing and the routing between networks. IP is the datagram delivery service.
<b>TCP</b>	Transmission Control Protocol (TCP) provides a reliable connection orientated service to applications that use it. TCP is connection orientated and checks on each host the order in which the packets are sent/received and also verifies that all the packets are transmitted. Applications such as telnet or ftp use the TCP protocol and don't need to handle issues over data loss etc ...
<b>UDP</b>	The User Datagram Protocol provides direct access to IP for application programs but unlike TCP, is connectionless and unreliable. This provides less overhead for applications concentrated on speed. If some form of packet accounting is needed this has to be provided by the application.
<b>ICMP</b>	The Internet Control Message Protocol is used by routers and hosts to report on the status of the network. It uses IP datagrams and is itself connectionless
<b>PPP</b>	The Point to Point Protocol establishes a TCP/IP connection over phone lines. It can also be used inside encrypted connections such as pptp.

## 6. TCP/IP Services and Ports

The list of known services and their relative ports is generally found in **/etc/services**. The official list of services and associated ports is managed by the IANA (Internet Assigned Numbers Authority).

Since the port field is a 16-bit digit there are 65535 available numbers. Numbers from 1 to 1023 are privileged ports and are reserved for services run by root. Most known applications will listen on one of these ports.

We will look at the output of portscans. Beware that unauthorised portscanning is illegal although many people use them.

Here is the output of a portscan:

Port	State	Service
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
25/tcp	open	smtp
70/tcp	open	gopher
79/tcp	open	finger
80/tcp	open	http

This shows open ports, these are ports being used by an application.

The /etc/services main ports:

<b>ftp-data</b>	<b>20/tcp</b>		
<b>ftp</b>	<b>21/tcp</b>		
<b>ssh</b>	<b>22/udp</b>		
<b>ssh</b>	<b>22/tcp</b>		
<b>telnet</b>	<b>23/tcp</b>		
<b>smtp</b>	<b>25/tcp</b>	<b>mail</b>	
<b>domain</b>	<b>53/tcp</b>		
<b>domain</b>	<b>53/udp</b>		
<b>http</b>	<b>80/tcp</b>		# www is used by some broken
<b>pop-3</b>	<b>110/tcp</b>		# PostOffice V.3
<b>sunrpc</b>	<b>111/tcp</b>		
<b>sftp</b>	<b>115/tcp</b>		
<b>uucp-path</b>	<b>117/tcp</b>		
<b>nnrp</b>	<b>119/tcp</b>	<b>usenet</b>	# Network News Transfer
<b>ntp</b>	<b>123/tcp</b>		# Network Time Protocol
<b>netbios-ns</b>	<b>137/tcp</b>	<b>nbns</b>	
<b>netbios-ns</b>	<b>137/udp</b>	<b>nbns</b>	
<b>netbios-dgm</b>	<b>138/tcp</b>	<b>nbdgm</b>	
<b>netbios-dgm</b>	<b>138/udp</b>	<b>nbdgm</b>	
<b>netbios-ssn</b>	<b>139/tcp</b>	<b>nbssn</b>	
<b>imap</b>	<b>143/tcp</b>		# imap network mail protocol
<b>NeWS</b>	<b>144/tcp</b>	<b>news</b>	# Window System
<b>snmp</b>	<b>161/udp</b>		
<b>snmp-trap</b>	<b>162/udp</b>		

## 7. Exercices and Summary

---

**Registering a service with xinetd**

1. Write a bash script that echo's "Welcome" to stdout. Save it in **/usr/sbin/hi**

```
#!/bin/bash
echo Welcome
```

Change the permission on the script to make it executable.

2. In **/etc/xinetd.d** create a new file called **fudge** with the following:

```
service fudge
{
    socket_type      = stream
    server           = /usr/sbin/hi
    user             = root
    wait             = no
    disable          = no
}
```

3. Add a service called **fudge** in **/etc/services** that will use port 60000.  
4. Restart **xinetd** and telnet to port 60000  
5. You have been assigned a range of IPs on the 83.10.11.0/27 network.  
How many networks have the same first 3 bytes as yours?  
How many hosts are on your network?  
What is the broadcast address for this first network?

# Network Services

## Prerequisite

- ◆ Booting Linux (p.14)
- ◆ Network Configuration (p. 36)

## Goals

- ◆ Understand the difference between **inetd** and **xinetd**
- ◆ Use the **libwrap** or “TCP wrapper” mechanism to secure services
- ◆ Configure NFS and SMB shares
- ◆ Configure network services: DNS (BIND), Sendmail and Apache

## Contents

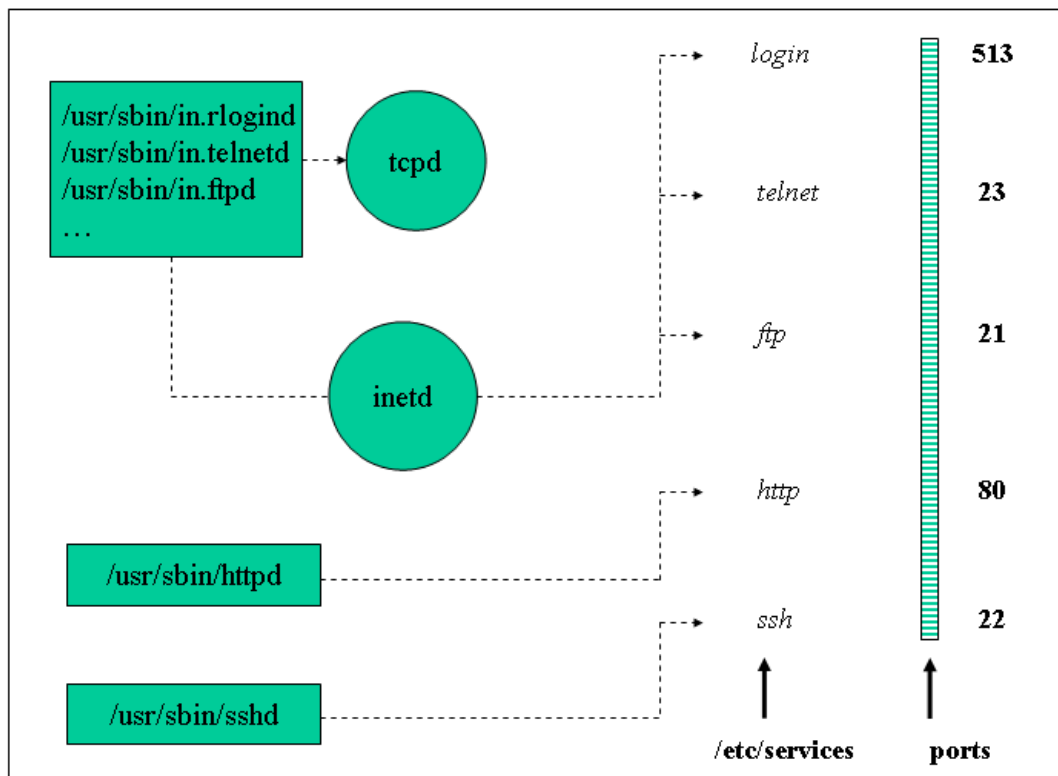
<b>Network Services.....</b>	<b>57</b>
1. The inetd daemon (old).....	58
2. The xinetd Daemon.....	59
3. Telnet and FTP.....	60
3. TCP wrappers .....	62
4. Setting up NFS.....	63
5. SMB and NMB.....	64
6. DNS services.....	66
7. Sendmail main Configuration.....	71
8. The Apache server.....	73
9. Exercises and Summary.....	74

Network services can either continuously run as standalone applications which listen for connections and handle clients directly or they can be called by the network daemon **inetd** (old) or **xinetd**.

## 1. The **inetd** daemon (old)

This daemon is started at boot time and listens for connections on specific ports. This allows the server to run a specific network daemon only when needed.

For example, the **telnet** service has a daemon **/usr/sbin/in.telnetd** which handles telnet sessions. Instead of running this daemon all the time **inetd** is instructed to listen on port 23. These instructions are set in **/etc/inetd.conf**.



*The **inetd** daemon*

The fields of **/etc/inetd.conf** contain the following:



service-name	valid name from <b>/etc/services</b>
socket type	stream for TCP and dgram for UDP
protocol	valid protocol from <b>/etc/protocols</b>
flag	nowait if multithreaded and wait if single-threaded
user/group	run application as user or group.
program	usually tcpd
argument	the name of the program to be run for this service

**Example:**

```
pop-3 stream tcp nowait root /usr/sbin/tcpd ipop3d
```

**Notice:** The **/etc/services** file is used to make the correspondence between service names and socket port numbers. The fields in services are as follows:

<b>service-name</b>	<b>port/protocol</b>	<b>[aliases]</b>
---------------------	----------------------	------------------

## 2. The xinetd Daemon

This is the most recent version of **inetd**. The **tcpd** daemon is no longer used, instead **xinetd** does everything. Configuration is done either through a single file **/etc/xinetd.conf** or by editing individual files in **/etc/xinetd.d/** corresponding to the services being monitored by **xinetd**. It is possible to migrate from the old **inetd** configuration file to the configuration files for the modern **xinetd**. Nothing else needs to be done.

### Structure of service file in xinetd.d

```
Service-name {
    disable = yes/no
    socket_type = stream for TCP and dgram for UDP
    protocol = valid protocol from /etc/protocols
    wait = <yes or no>
    user= the user the application runs as
    group= the group the application runs as
    server= the name of the program to be run for this service
}
```

## 3. Telnet and FTP

Telnet and ftp are common examples of services using the inetd/xinetd mechanism to listen for incoming connections.

**TELNET** is the name of the application layer protocol used to establish a "bi-directional communication facility" (RFC854). "Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other".

The server runs a telnet daemon (usually **in.telnetd**) and communications are initiated from the client using a telnet client (called **telnet** too). For RPM based machines the server package is called **telnet-server** and the client package is called **telnet**.

Once the **telnet-server** package is installed the configuration files **/etc/inetd.conf** or **/etc/xinetd.conf** need the following options:

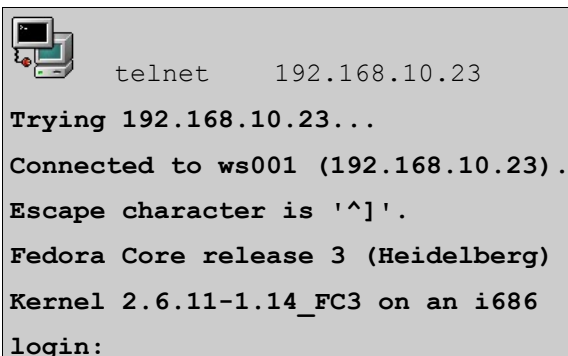
**/etc/inetd.conf** (for the **inetd** daemon)

```
telnet  stream  tcp    nowait  root    /usr/sbin/tcpd  in.telnetd
```

**/etc/xinetd.conf** (for the **xinetd** daemon)

```
service telnet
{
    disable          = no
    flags             = REUSE
    socket_type       = stream
    wait              = no
    user              = root
    server             = /usr/sbin/in.telnetd
    log_on_failure    += USERID
}
```

The next command attempts to connect to the host 192.168.10.23. Notice that the content of **/etc/issue.net** is also displayed:


 A terminal window showing a telnet session. The user enters 'telnet 192.168.10.23'. The output shows the connection attempt, successful connection to 'ws001 (192.168.10.23)', the escape character '^]', and the system banner for Fedora Core release 3 (Heidelberg) with kernel 2.6.11-1.14\_FC3 on an i686. The prompt 'login:' is shown at the end.
 

```
telnet 192.168.10.23
Trying 192.168.10.23...
Connected to ws001 (192.168.10.23).
Escape character is '^]'.
Fedora Core release 3 (Heidelberg)
Kernel 2.6.11-1.14_FC3 on an i686
login:
```

**FTP** is the "files transfer protocol". The objectives of this application layer protocol stated in RFC959 are "1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user

from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently”

There are several ftp servers available for Linux. In these notes we choose to configure **vsftpd** (very safe FTP server) which is available as a package of the same name. There are many FTP clients provided by the packages **ftp**, **ncftp**, **lftp** or **gftp** (graphical).

The **vsftpd** can be started as a stand alone server (recommended) but can also use **inetd** or **xinetd** to handle incoming connections with the following options

```
/etc/vsftpd/vsftpd.conf
```

```
listen=NO
```

```
/etc/inetd.conf
```

```
ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/vsftpd
```

```
/etc/xinetd.conf
```

```
service ftp
{
    socket_type          = stream
    wait                 = no
    user                 = root
    server                = /usr/sbin/vsftpd
    nice                  = 10
    disable               = yes
}
```

It is possible to log onto an FTP server either as an anonymous user or as a regular system user (e.g a user with an entry in **/etc/passwd**). Anonymous FTP allows a user to login with the username-password pair **anonymous** and **email-address**. A regular user will initially have access to his or her home directory where as anonymous users can only browse the contents of **/var/ftp/**.



```
ftp 192.168.10.23
```

```
Connected to 192.168.10.23.
```

```
220 (vsFTPD 2.0.1)
```

```
530 Please login with USER and PASS.
```

```
KERBEROS_V4 rejected as an authentication type
```

```
Name (192.168.10.23:tux)
```

### 3. TCP wrappers

If programs have been compiled with the libwrap library then they can be listed in the files `/etc/hosts.allow` and `/etc/hosts.deny`. The **libwrap** library will verify these files for matching hosts.

Default format for `/etc/hosts.{allow,deny}` :

```
DAEMON :      hosts [EXCEPT hosts ] [: spawn command]
```

One can also use these files to log unauthorised services. This can also help as an early warning system. Here are a few examples.

Getting information about a host:

- `/etc/hosts.allow`  
in.telnetd: LOCAL, .my.domain
- `/etc/hosts.deny`  
in.telnetd: ALL : spawn (/usr/sbin/safe\_finger -l @%h | mail root)

Redirect to a bogus service or "honey pot" :

- `/etc/hosts.allow`  
in.telnetd: ALL : twist /dtk/Telnetd.pl

The last example comes from the dtk (Deception Tool Kit) that can be downloaded from <http://all.net/dtk/download.html>

The **inetd** and **xinetd** daemons as well as some stand alone servers such as **sshd** and **vsftpd** have been dynamically compiled with libwrap:



```
ldd /usr/sbin/xinetd | grep libwrap
libwrap.so.0 => /usr/lib/libwrap.so.0 (0x003da000)
```



```
ldd /usr/sbin/xinetd | grep libwrap
libwrap.so.0 => /usr/lib/libwrap.so.0 (0x003da000)
```



```
ldd /usr/sbin/vsftpd | grep libwrap
libwrap.so.0 => /usr/lib/libwrap.so.0 (0x00204000)
```

## 4. Setting up NFS

### • Client settings

For a Linux client to mount remote file systems

1. the **nfs** file system must be supported by the kernel
2. the **portmap** daemon must be running.

The portmapper is started by the **/etc/rc.d/init.d/portmap** script. The **mount** utility will mount the filesystem.

For example we can create a new directory called **/mnt/nfs** and mount a shared directory from the server nfs-server called **/shared/dir**. This can be done by adding the following line to **/etc/fstab**

/etc/fstab					
nfs-server:/shared/dir	/mnt/nfs	nfs	defaults	0	0

If no entry is set in **/etc/fstab** then the complete command would be:



### • Server settings

A NFS server needs **portmap** to be running before starting the nfs server. The nfs server should be started or stopped with the **/etc/rc.d/init.d/nfs** script.

The main configuration file is **/etc/exports**.

Sample /etc/exports file	
/usr/local/docs	*.local.org(rw, no_root_squash) *(ro)

The **/usr/local/docs** directory is exported to all hosts as read-only, and read-write to all hosts in the **.local.org** domain.

The default **root\_squash** option which avoids the root user (uid = 0) on the client to access the share on the server can be changed with the **no\_root\_squash** option.

The **/etc/exports** file matches hosts such as **\*.machine.com** where as **/etc/hosts.allow/deny** match hosts such as **.machine.com**

If the **/etc/exports** file has been changed then the **exportfs** utility should be run. If existing directories in **/etc/exports** are modified then it may be necessary to unmount all nfs shares before remounting them all. Individual directories are made available for mounting with **exportfs**.

---

Unexporting and exporting all directories in /etc/exports:



```
exportfs -ua ; exportfs -a
```

## 5. SMB and NMB

Linux machines can access and provide Windows shared resources (directories and printers). The protocol used for this is the MS Windows Server Message Block **SMB**. Samba is the most common Linux tool which provides client and server software.

### From the Command Line

The **smbclient** utility is used to list shared resources. Remote directories are typically mounted with **smbmount** although 'mount -t smbfs' can also be used.

Examples:

Send a pop up message to the win98desk computer



```
smbclient -M win98desk
```

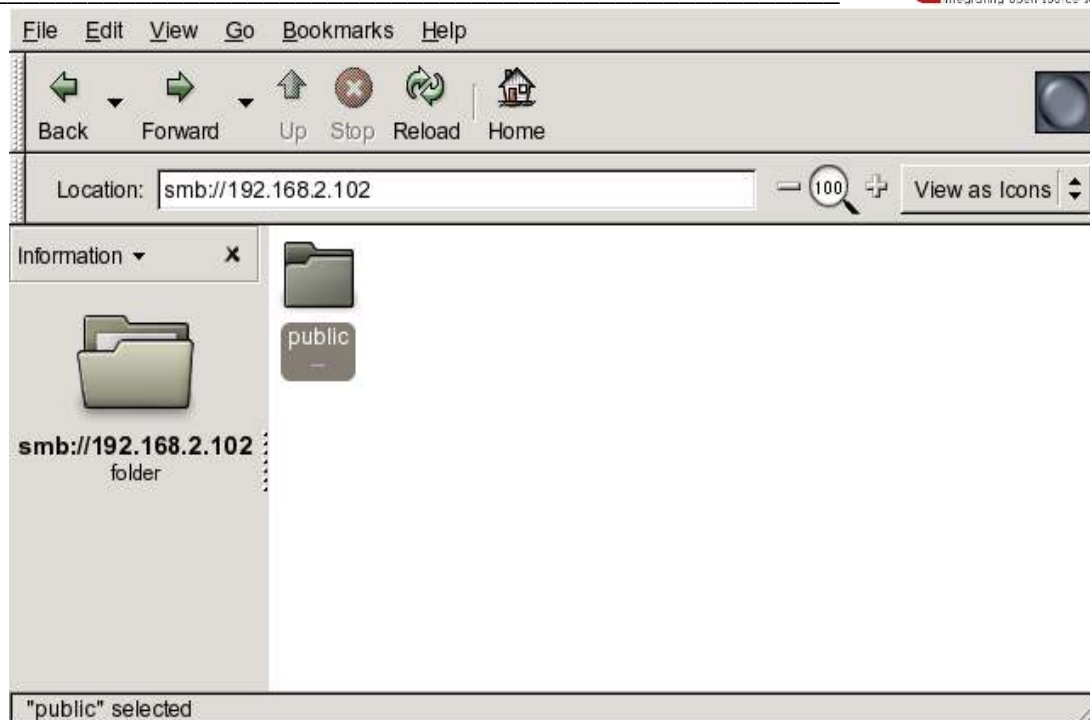
Mount the shared directory of the winserv computer



```
smbmount //winserver/shared /mnt/winserver/shared
```

The Samba server is configured with the **/etc/smb.conf** file. The server is stopped and started with the **/etc/rc.d/init.d/smb** script. Notice that **smb** will also start the **NMB** services. This is the NetBIOS Message Block which enables name resolution in the Windows realm.

Figure1: Nautilus Browsing SMB shares:

**Main entries in /etc/smb.conf:**

```
[global]
    workgroup = LINUXIT
    os level = 2
    kernel oplocks = No
    security = user
    encrypt passwords = Yes
    guest account = nobody
    map to guest = Bad User

[homes]
    comment = Home Directories
    read only = No
    create mask = 0640
    directory mask = 0750
    browseable = No

[printers]
    comment = All Printers
    path = /var/tmp
    create mask = 0600
    printable = Yes
    browseable = No
```

**SWAT and Webmin GUI Configuration**

If you install the swat package then you can administrate a samba server via a web-based GUI on port 901.

Another popular general administration tool is **webmin**. It can be downloaded at [www.webmin.com](http://www.webmin.com)

#### NOTICE

The configuration file `/etc/samba/smb.conf` is a good source of documentation. All options are explained and can be switch on by deleting the comment character `;`. Also read the **smb.conf(5)** manpage

## 6. DNS services

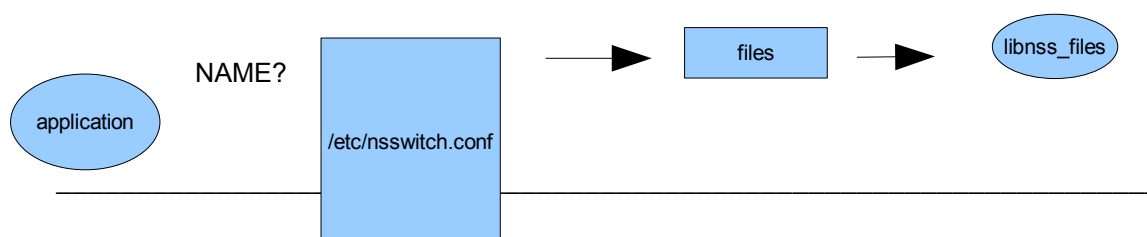
### 🔍 Finding a Name with `/etc/nsswitch.conf`

The file `/etc/nsswitch.conf` (previously `/etc/host.conf`) holds all the information needed by an application to find a name. The types of names are designated by a keyword.

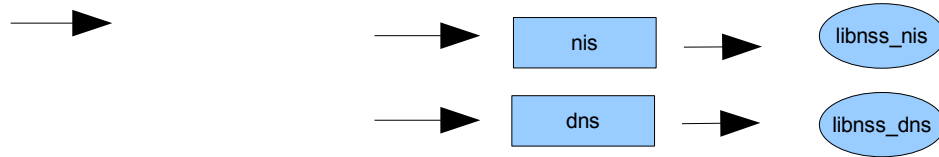
Common Names	
keyword	description
passwd	user names
group	group names
hosts	host names
networks	network names

Names are searched in a number of 'databases'. Each database can be accessed by a specialised library. For example there will be libraries called `libnss_files`, `libnss_nis` and `libnss_dns` to deal with each databases listed below.

Common databases	
keyword	description
files	flat files, generally in <code>/etc</code>
nis	a map from a NIS server
dns	a DNS server







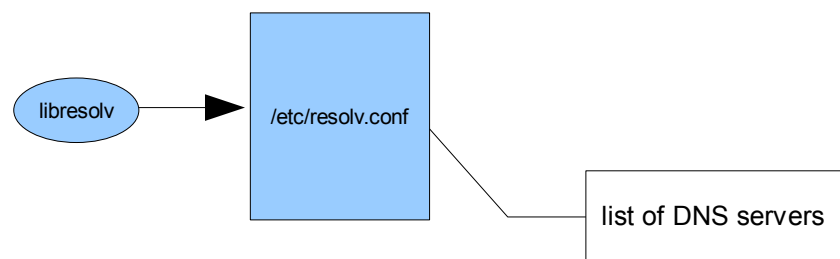
#### Sample /etc/nsswitch.conf

```
hosts:      files dns
networks:   files nis ldap
```

The first line indicates that files (here **/etc/hosts**) should be queried first and then a DNS server if this fails. The second line instructs to use the **/etc/networks** file for network information.

### ● The Resolver

When a program needs to resolve a host name using a DNS server it uses a library called a resolver. The resolver will first consult the **/etc/resolv.conf** file and determine which DNS server to contact.



#### Sample /etc/resolv.conf

```
search example.com
nameserver 192.168.123.1
```

If the resolver needs to use a domain name server (DNS) then it will consult the **/etc/resolv.conf** file for a list of available servers to query from.

### The **/etc/hosts** file

With a small number of networked computers it is possible to convert decimal IP numbers into names using the **/etc/hosts** file. The fields are as follows:

IP	machine	machine.domain	alias
----	---------	----------------	-------

Example `/etc/hosts` file:

192.168.1.233	io	io.my.domain
61.20.187.42	callisto	callisto.physics.edu

### ● Hierarchical structure

Name servers have a hierarchical structure. Depending on the location in the fully qualified domain name (FQDN) a domain is called top-level, second-level or third-level.

Example top-level domains

<b>com</b>	Commercial organisations
<b>edu</b>	US educational institutions
<b>gov</b>	US government institutions
<b>mil</b>	US military institutions
<b>net</b>	Gateways and network providers
<b>org</b>	Non commercial sites
<b>uk</b>	UK sites

### ● Types of DNS servers

Domains can be further divided into sub-domains. This limits the amount of information needed to administer a domain. Zones have a **master** domain name server (previously called a **primary** DNS) and one or several **slave** domain name servers (previously called **secondary**). Administration of a name server consists of updating the information about a particular zone. The **master** servers are said to be authoritative.

### ● DNS Configuration Files

In old versions of BIND (prior to BIND version 8) the configuration file was `/etc/named.boot`. With BIND version 8 the `/etc/named.conf` file is used instead. One can use the `named-bootconf.pl` utility to convert old configuration files.

The `/etc/named.boot` file:

directory		/var/named
cache	.	named.ca
primary	myco.org	named.myco
primary	0.0.127.in-addr.arpa	named.local
primary	1.168.192.in-addr.arpa	named.rev

The first line defines the base directory to be used. The `name.ca` file will contain a list of DNS IP addresses for querying external addresses. The third line is optional and contains records for the local LAN. The two next entries are for reverse lookups.

In `/etc/named.conf`:

`cache` is replaced by *hint*  
`secondary` is replaced by *slave*  
`primary` is replaced by *master*.

Applying these changes to BIND4 configuration files will generate BIND8 and BIND9 files such as the following.

The `/etc/named.conf` file:

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

zone "." {
    type hint;
    file "named.ca";
};

zone "myco.org" {
    type master;
    file "named.myco";
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "named.rev";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};
```

## ● DNS zone files

In this example the server is set as a caching-only server. All the zone files contain resource records.

Sample `named.local` zone file:

```
@    IN    SOA    localhost. root.localhost. (
                        2001022700 ; Serial
                        28800      ; Refresh
                        14400      ; Retry
                        3600000     ; Expire
```

```

      86400 ) ; Minimum
1      IN      NS      localhost.
      IN      PTR     localhost.

```

This is a very simple zone file but it gives us enough information to understand the basic mechanism of a name server.

The @ sign will resolve to the related zone declared in **/etc/named.conf**. This allows any zone file to be used as a template for further zones (see the exercises).

Table1: Common Record Types

NS	Specify the zones primary name server
PTR	Reverse mapping of IP numbers to hostnames
MX	Mail exchange record
A	Associate an IP address with a hostname
CNAME	Associate an alias with the host's main name

Table2: Zone parameters

@	IN	SOA	Start Of Authority. Identifies the zone followed by options enclosed in brackets.
serial			Is manually incremented when data is changed. Secondary servers query the master server's serial number. If it has changed, the entire zone file is downloaded
refresh			Time in seconds before the secondary server should query the SOA record of the primary domain. This should be at least a day.
retry			Time interval in seconds before attempting a new zone transfer if the previous download failed
expire			Time after which the secondary server discards all zone data if it contact the primary server. Should be a week at least
minimum			This is the ttl for the cached data. The default is one day (86400 seconds) but should be longer on stable LANs

## Testing

Here we only check the records of type **MX**. Other types are **ANY**, **A** or **NS**.

- Check local domain: **dig** and **host** do the same thing except that **dig** will printout results that can be used in a zone file:

```

 dig @127.0.0.1 gogo.com MX
 host -t mx gogo.com 127.0.0.1

```

- Use local caching server to query any domain: replace the domain gogo.com in the commands above with any other domain you wish to query.

## 7. Sendmail main Configuration

Sendmail is the most popular mail transfer agent (MTA) on the Internet. It uses the Simple Mail Transfer Protocol (SMTP) and runs as a daemon listening for connections on port 25.

The **sendmail** script which stops or starts the sendmail daemon is usually located in the **/etc/rc.d/init.d/** directory.

### Configuration Features

The main configuration file is **/etc/mail/sendmail.cf** (or **/etc/sendmail.cf**). Here you can specify the name of the server as well as the names of the hosts from which and to which mail relay is allowed.

#### WARNING

You do not need to know how to write sendmail.cf rules. In fact all the rules can be generated using the **sendmail.m4** or **sendmail.mc** macro file to produce a **sendmail.cf** file by running the following

```
m4 sendmail.mc > sendmail.cf
```

This process is not part of the LPI objectives

#### sendmail.cf options

<b>Cw</b>	the mailer hostname. Can also contain a list of hostnames or domain names the mailer will assume but it is better to use Fw for this
<b>Fw</b>	path to the file containing domain names sendmail will receive mail for
<b>Ds</b>	address for 'smart host', this is a mailer that will relay our outgoing mail

#### Files in /etc/mail

<b>access</b>	list of hosts authorised to use the server to relay mail
<b>local-host-names</b>	list of domain names

### Aliases and mail forwarding

The **/etc/aliases** file contains two fields as follows:

```
alias: user
```

For example if the mail server has a regular UNIX account for user *foo* then mail addressed to *mr.foo* will reach this user only if the following line is included in **/etc/aliases**:

```
mr.foo: foo
```

Or if you want to forward all mail to an external address:

```
mr.foo: foo@someisp.net
```

For other options see the manpage **aliases(5)**.

When changes to the **/etc/aliases** file are made the **newaliases** command must be run to rebuild the database **/etc/aliases.db**.

When mail is addressed to a local user (say *foo*) then this user can choose to forward this mail to a list of other users using a local file **~/.forward** (one address per line).

In LPI 202 we will see that mail can also be forwarded to a file, a pipe or an include file.

### The Mail Queues

When mail is accepted by the server it is concatenated in a single file with the name of the user. These files are stored in **/var/spool/mail/**.

Depending on the Mail User Agent used (mutt, pine, elm ...), a user can either store these messages in his home directory or download them on another machine.

All outgoing mail is spooled in **/var/spool/mqueue**

If the network is down or very slow, or if many messages are being sent, then mail accumulates in the mail queue **/var/spool/mqueue**. You can query the queue with the **mailq** utility or **sendmail -bp**.

An administrator can flush the server's queue with **sendmail -q**.

### Registering a Mailer for a Domain

Finally in order to use a domain name as a valid email address an MX record needs to be added on an authoritative name server for your domain (usually your ISP).

For example if **mail.company.com** is a mail server, then in order for it to receive mail such as **joe@company.com** you should have the following configuration:

1. Add **company.com** to **/etc/mail/local-host-names**
2. **company.com**      **MX 10 mail.company.com**      in a DNS zone file

## 8. The Apache server

### ● Configuration Files

The **/etc/httpd/conf/httpd.conf** file contains all the configuration settings

---

Older releases of apache had two extra files, one called **access.conf** where restricted directories were declared, and another file called **srm.conf** specifying the server's root directory.

### Configuration Highlights:

**ServerType** standalone/inetd

**ServerRoot** "/etc/httpd"

**DocumentRoot** "/var/www/html"

```
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>

<VirtualHost 122.234.32.12>
    DocumentRoot "/www/docs/server1"
    ServerName virtual.mydomain.org
</VirtualHost>
```

### ● Running Apache

To stop and start the server one can use the **/etc/rc.d/init.d/httpd** script. On a busy server it is preferable to use **apachectl** especially with the **graceful** option which will restart the server only when current connections have been dealt with.

The main log files are in **/var/log/httpd/**. It may be useful for security reasons to regularly check the **error\_log** and **access\_log** files.

## 9. Exercises and Summary

### Files

File	Description
/etc/hosts.allow /etc/hosts.deny	file used by the libwrap library to determine access to a service from a given host, network or domain
/etc/aliases	<b>aliases(5)</b> - file describes user ID aliases used by sendmail. Each line is of the form <code>name: addr_1, addr_2, ...</code> where <code>name</code> is a local username to alias and <code>addr_n</code> can be another alias, a local username, a local file name, a command, an include file, or an external address
/etc/exports	<b>exports(8)</b> – the file /etc/exports serves as the access control list for file systems which may be exported to NFS clients. It is used by <code>exportfs(8)</code> to give information to <code>mountd(8)</code> and to the kernel based NFS file server daemon <code>nfds(8)</code>
/etc/host.conf	main configuration file for the resolver
/etc/hosts	database of host IPs and names
/etc/inetd.conf	configuration file for the <code>inetd</code> daemon
/etc/mail/*	directory containing all the sendmail configuration files
/etc/named.boot	name of the BIND4 version of named
/etc/named.conf	name of the BIND8 and 9 versions of named
/etc/nsswitch.conf	<b>nsswitch.conf(5)</b> – System Databases and Name Service Switch configuration file.
/etc/resolv.conf	list of DNS servers used by the resolver to determine host names
/etc/sendmail.cf	the main configuration file for sendmail
Cw	option within <code>sendmail.cf</code> that specifies the name of the server (may be a domain name)
Ds	option to specify a smarthost in <code>sendmail.cf</code>
Fw	option setting the name of the file that contains all the names of the mail server
/etc/smb.conf	main configuration file for the samba server <b>smbd</b>
/etc/xinetd.conf	configuration file for the <b>xinetd</b> daemon
/var/spool/mail/	directory containing received mail for local users
/var/spool/mqueue	spool directory for outgoing mail
~/.forward	file containing a list of addresses where valid local account mail is forwarded to
/etc/httpd/conf/access.conf	configuration file containing web directories that need extra identification mechanisms such as <code>htaccess</code> (old)
/etc/httpd/conf/httpd.conf	main configuration file for web server daemon <b>httpd</b>
/etc/httpd/conf/srm.conf	configuration file defining the document root of the web server (old)



## Commands

Command	Description
apachectl	<b>apachectl(8)</b> – apache HTTP server control interface. On the command line the script will simply pass all the given arguments to the <b>httpd</b> server
dig	<b>dig(1)</b> – (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried
host	<b>host(1)</b> – a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa
exportfs	<b>exportfs(8)</b> – command is used to maintain the current table of exported file systems for NFS. This list is kept in a separate file named <code>/var/lib/nfs/xtab</code> which is read by mountd when a remote host requests access to mount a file tree, and parts of the list which are active are kept in the kernel's export table
inetd	see <b>xinetd</b>
mailq	<b>mailq(1)</b> – prints a summary of the mail messages queued for future delivery
portmap	<b>portmap(8)</b> – is a server that converts RPC program numbers into DARPA protocol port numbers. It must be running in order to make RPC calls. When an RPC server is started, it will tell portmap what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact portmap on the server machine to determine the port number where RPC packets should be sent. Portmap must be started before any RPC servers are invoked
smbclient	<b>smbclient(1)</b> – is a client that can 'talk' to an SMB/CIFS server. It offers an interface similar to that of the ftp program (see <b>ftp(1)</b> ). Operations include things like getting files from the server to the local machine, putting files from the local machine to the server, retrieving directory information from the server and so on
smbmount	<b>smbmount(8)</b> – mounts a Linux SMB filesystem. It is usually invoked as <code>mount.smbfs</code> by the <code>mount(8)</code> command when using the <code>"-t smbfs"</code> option. This command only works in Linux, and the kernel must support the smbfs filesystem
sendmail	<b>sendmail(8)</b> – sends a message to one or more recipients, routing the message over whatever networks are necessary. Sendmail does internetwork forwarding as necessary to deliver the message to the correct place
xinetd	<b>xinetd(8)</b> – performs the same function as <code>inetd</code> : it starts programs that provide Internet services. Instead of having such servers started at system initialization time, and be dormant until a connection request arrives, <code>xinetd</code> is the only daemon process started and it listens on all service ports for the services listed in its configuration file. When a request comes in, <code>xinetd</code> starts the appropriate server. Because of the way it operates, <code>xinetd</code> (as well as <code>inetd</code> ) is also referred to as a super-server

## Setting up a DNS master server

As an exercise we will install the BIND9 rpm package **bind9-9.1.3-252.i386.rpm** and configure a domain called `gogo.com`.

1. Carry out the following alterations in `/etc/named.conf`:  
Copy/Paste the following paragraphs and alter as follows:

zone "localhost" in { type master; file "ZONEFILE-for-localhost"; }	<i>becomes</i>	zone "gogo.com" in { type master; file "gogo.zone"; }
zone "0.0.127.in-addr.arpa" in { type master; file "ZONEFILE-for-127.0.0"; };	<i>becomes</i>	zone "2.168.192.in-addr.arpa" in { type master; file "192.168.2.zone"; };

**2. In /var/named:**

```
cp ZONEFILE-for-127.0.0 192.168.2.zone
cp ZONEFILE-for-localhost gogo.zone
```

3. Change the appropriate fields in the new zone files. Add a host called *harissa*.

4. Add the line "nameserver 127.0.0.1" to **/etc/resolv.conf**.

5. Use **host** to resolve *harissa.gogo.com*

**Apache administration**

Basic configurations in **/etc/httpd/conf/httpd.conf**

1. Change the port directive **Port** from **80** to **8080**. (If you are using http2 then change the **Listen** directive).
2. Check that apache is responding with **telnet localhost 8080**. You should get:

```
Trying 127.0.0.1...
Connected to localhost.linuxit.org.
Escape character is '^['.
```

Next type **'GET /'** to download the index file.

3. Set "**StartServer**" to 15. Restart the **httpd** and check that 15 processes are started (instead of the default 8)

IP based virtual server

Your ethernet card must be aliased to a new IP (say *new-IP*)

```
ifconfig eth0:0 new-IP
```

Add the following paragraph to **/etc/httpd/conf/httpd.conf**:

```
<VirtualHost new-IP>
DocumentRoot /var/www/html/virtual
ServerName www1
</VirtualHost>
```

### ***Setting up a shared SMB directory***

In most cases you won't need to add smbusers to the system to do this. Simply edit **smb.conf** and add the following:

```
[public]
    comment = Example Shared Directory
    path = /home/samba
    guest ok = yes
    writeable = yes
```

### **Setting up a shared printer:**

```
[global]
--- snip ---
printcap name = /etc/printcap
    load printers = yes

[printers]
    comment = All Printers
    path = /var/spool/samba
    browseable = no
# Set public = yes to allow user 'guest account' to print
    guest ok = yes
    writable = no
    printable = yes
```

# ***Bash Scripting***

## **Prerequisite**

- ◆ None

## **Goals**

- ◆ Review the main configuration files associated with the **bash** shell
- ◆ Write and execute shell scripts
- ◆ Syntax for logical evaluations, flow controls and loops
- ◆ Miscellaneous features (Not part of the LPI 102 objectives)

## **Contents**

<b>Bash Scripting.....</b>	<b>78</b>
1. The bash environment.....	79
2. Scripting Essentials.....	81
3. Logical evaluations.....	82
4. Flow Control and Loops.....	83
5. Expecting user input.....	85
6. Working with Numbers.....	85
7. Exercises and Summary.....	86

## 1. The bash environment

### Variables

When you type a command at the prompt the bash shell will use the **PATH** variable to find which executable on the system you want to run. You can check the value of path using the echo command:



```
echo $PATH
```

```
/usr/bin:/bin:/usr/sbin:/usr/X11R6/bin:/usr/local/bin:/sbin:/usr/local/sbin/
```

In fact many variables are needed by the shell to accommodate for each user's environment. For example **PWD**, **HOME**, **TERM** and **DISPLAY** are such variables.

To initialise and declare a variable the syntax is as follows:

```
VARIABLE=VALUE
```

Remember not to put any spaces around the '=' sign. Once a variable is declared and initialised it can be referenced by using the dollar symbol in front as here:



```
echo $VARIABLE
```

This declares a *local* variable (only available for the current process) that can be listed with **set**. It is possible to get an *exported* variable (available to all child processes spawned after the variable has been defined) using **export**. Exported variables are listed with the **env** command.

When a shell session is started a number of configuration files are read and most of the variables are set.

To free a variable from its current value use **unset**.

### Configuration files

One can distinguish configuration files which are read at login time and configuration files which are read for each new bash session.

#### The profiles

The first file to be read at login is **/etc/profile**, after that the shell will search for the files **~/.bash\_profile**, **~/.bash\_login** and **~/.profile** and execute the commands from the first available one. For every new shell (for example if an xterm emulator is started) these profiles are not read again.

**Contents:** the profiles are used to define exported variable (e.g PATH) that will be available for every subsequent program.

#### The bashrc files

The runtime control files `~/.bashrc` and `/etc/bashrc` are sourced every time a shell is started

**Contents:** the runtime control files will store aliases and functions.

Notice that non-interactive shells read neither of these files. Instead a `BASH_ENV` variable pointing to the file to be sourced is declared in the script.

### **Function syntax**

```
function-name ()
{
  command1;
  command2;
}
```

You can test which files are being read by adding an `echo Profile` line in `/etc/profile`, the type:

```
bash          No profile is read, you shouldn't see anything
bash -login    This forces bash to act as a login bash, the word
               Profile should show up.
```

The following commands control the way bash starts:


```
bash -norc
bash -noprofile
```

**Notice** that any new bash session will inherit the parent's global variables defined in `/etc/profile` and `~/.bash_profile`.

### **Controlling readline**

The GNU library *readline* is used by programs that expect user input. It also offers extensive vi and emacs style editing functionality.

Example: the readline default editor setting for bash is *emacs*. One can for example use **^E** to go to the end of a line. What happens when we next start, as below, a shell without editing support?



```
bash --noediting
```

The files `/etc/inputrc` or `~/.inputrc` are used to control the readline library. One can for example link a keyboard combination to an action.

Example options for <b>inputrc</b> :	
set editing-mode vi	change the initial editor style (default is <i>emacs</i> )
Control-o: "> output"	bind the sequence Ctrl+o will cause the string "> output " to be printed
TAB: complete	automatically complete commands and file names (is set by default)
set bell-style none	input errors are not audible (other option is <i>audible</i> )

Finally, when a user logs out, the shell will read commands from `~/.bash_logout` if it exists. This file usually contains the *clear* command which clears the screen once the shell exits.

## 2. Scripting Essentials

### The script file

A shell script is a list of instructions saved in a flat file. Only two things are necessary.

1. The script's first line must be **#!/bin/bash** (for a bash script)
2. The file must be readable and executable (with 755 permission for example)

Assuming the script is in your current directory it can be started with



```
./script-name
```

### NOTICE

The interpreter specified after the **#!** sign (pronounce she-bang!) is used to read the commands in the script. If no interpreter is specified then the shell will attempt to interpret the commands itself.

### Alternative methods

<code>bash script-name</code>	start a new interactive <b>bash</b> which will run the script then exit
<code>source script-name</code>	force your current shell to run the script
<code>. script-name</code>	same as <b>source</b>
<code>exec ./script-name</code>	same as <b>./script-name</b> except that the current shell will exit once the script has run

### Passing variables to the script

Variables entered at the command line are referenced inside the script as \$1 for the first argument, \$2 for the second, etc ...

Example script, mycat:

```
#!/bin/bash
cat $1
```

This script is expecting one argument, a file, and will display the content of the file using **cat**. To run this script on the lilo.conf file, you would run:



```
./mycat /etc/lilo.conf
```

## Bash Scripting



Another way of passing variables to a script is by letting the script prompt the user for input interactively. This is achieved using the **read** command. The default name of the read variable is **REPLY**. Here is the modified script:

Interactively passing:

```
#!/bin/bash
echo -n "Which file shall I display ?"
read
cat $REPLY
```

or

```
read -p "File to display: " FILENAME
cat $FILENAME
```

### Special Variables

Special variables can only be referenced and are automatically set by bash. These are the most common special variables you will encounter:

<b>\$*</b>	List of all variables entered at the command line
<b>\$#</b>	Number of arguments entered at the command line
<b>\$0</b>	The name of the script
<b>\$_</b>	PID of the most recent background command
<b>\$\$</b>	PID of the current shell
<b>\$?</b>	Exit code of the last command

For the positional parameters \$1, \$2 etc ... there is a **shift** operator which renames each parameter in a cyclic way as follows.

\$2 becomes \$1

\$3 becomes \$2 ... etc

This can be summarised as **\$(n+1) → \$n**

## 3. Logical evaluations

Logical statements are evaluated with the **test** command or the brackets **[ ]**. In both case the result is stored in the **\$?** variable such that:

if the statement is true then **\$?** is 0


if the statement is false then **\$?** is not 0

Here are some examples to illustrate:


using <b>test</b>	using <b>[ ]</b>	meaning
test -f /bin/bash	[ -f /bin/bash ]	test if /bin/bash is a file
test -x /etc/passwd	[ -x /bin/passwd ]	test if /etc/passwd is executable



One can evaluate more than one statement at a time using the **||** (OR) and **&&** (AND) logical operators on the command line. For example we could test if **/bin/bash** is executable and in **/etc/inittab** exists:




```
test -x /bin/bash && test -f /etc/inittab
```




```
[ -e /bin/kbash ] || [ -f /etc/passwd ]
```

This is the same as using the flags **-o** and **-a** within the **test** operator for example



```
test -x /bin/bash -a -f /etc/inittab
```



```
[ -e /bin/kbash -o -f /etc/passwd ]
```

## 4. Flow Control and Loops

### **if then**

Syntax:

```
if      CONDITION ; then
    command1
    command2
fi
```

```
#!/bin/bash
```

```
if [ -x /bin/bash ] ; then
echo "The file /bin/bash is executable"
```

### **while loop**

Syntax: while CONDITION is **true**; do

```
    command
done
```

Example: Align 10 hashes (#) then exit

```
#!/bin/bash
COUNTER=0
```

```
while [ $COUNTER -lt 100 ]; do
    echo -n "#"
    sleep 1
    let COUNTER=COUNTER+1
done
```

### ***Until loop***

Syntax: until CONDITION is **false**; do  
                                  command  
                  done

Example: Same as above, notice the C style increment for COUNTER

```
#!/bin/bash
COUNTER=20
until [ $COUNTER -lt 10 ]; do
    echo -n "#"
    sleep 1
    let COUNTER-=1
done
```

### ***for loop***

Syntax for VARIABLE in SET; do  
                                  command  
                  done

Example: the set 'SET' can be the lines of a file

```
#!/bin/bash
for line in `cat /etc/lilo.conf`; do
    IMAGE=$(echo $line | grep image)
    if [ "$IMAGE" != "" ]; then
        echo Kernel configured to boot: $line
    fi
done
```

## **5. Expecting user input**

We assume that the script is waiting for user input, depending on the answer, the rest of the program will execute something accordingly. There are two possible ways to achieve this: **select** and **case**.

**Using case**

Syntax: case \$VARIABLE in  
           CHOICE1) command1 ;;  
           CHOICE2) command2 ;;  
           esac

**Using select**

Syntax: select VARIABLE in SET; do  
           if [ \$VARIABLE = CHOICE ]; then  
               command  
           fi  
           if [ \$VARIABLE = CHOICE ]; then  
               command  
           fi  
       done

**6. Working with Numbers**

While shell scripts seamlessly handle character strings, a little effort is needed to perform very basic arithmetic operations.

**Binary operations**

Adding or multiplying numbers together can be achieved using either **expr** or the **\$(( ))** construct.

**Example:**

```
expr 7 + 3; expr 2 \* 10; expr 40 / 4; expr 30 - 11
$((7+3)); $((2*10)); $((40/4)); $((30-11))
```

**Comparing values**

Test operators:

Numbers	Strings
-lt	< (sort strings lexicographically)
-gt	> (sort strings lexicographically)
-le	N/a
-ge	N/a
-eq	==
-ne	!=

**7. Exercises and Summary****Files**

## Bash Scripting



Files	Description
/etc/bashrc	a system wide startup file for interactive bash sessions (used for setting up the PS1 prompt)
/etc/inputrc	startup file for the readline library used by the shell to read and edit user input. This file combines keyboard combinations with editing commands but can also be used to associate keyboard combinations to any command (macro)
/etc/profile	system wide configuration file for <b>bash</b> . It contains exported variables such as the PATH and is always read at login
~/.bash_profile	the user's customised configuration file for <b>bash</b> . It contains exported variables and is always read at login
~/.bashrc	the user's customised startup file for <b>bash</b> . It is read every time a new interactive shell is started unless the <b>-norc</b> option is given
~/.inputrc	the user's customised startup file for the readline library

### Scripting items

Item	Description
\$(( ))	operator used to substitute the result of a numerical evaluation in a command line
expr	perform a numerical evaluation
for loop	see p.72
if then	see p.71
until loop	see p.72
while loop	see p.71

### Commands

Command	Description
test	<b>test(1)</b> – check file types and compare values
unset	(bash built-in) command that removes a variable value or a function
env	print all exported (global) variables defined in the current shell
export	(bash built-in) command that makes a variable part of the environment of subsequent processes
set	(bash built-in) command that when started with no arguments prints the value of all shell variables defined

1. On the command line export the variable TEST



```
export TEST=old
```

2. Write the script

```
#!/bin/bash
```

```
echo old variable: $TEST
export TEST=new
echo exported variable: $TEST
```

3. What is the value of `$TEST` once the script has run?

4. The following script called `test_shell` will print the PID of the shell that is interpreting it

```
test_shell
#!/bin/bash
if [ -n $(echo $0 |grep test) ]; then
echo The PID of the script is: $$
else
echo The PID of the interpreter is: $$
fi
```

5) Set the permissions to 755 and test the following commands

```
test_shell
./test_shell
bash test_shell
. test_shell
source test_shell
exec ./test_shell
```

# ***Basic Security***

## **Prerequisites**

- ◆ None

## **Goals**

- ◆ Overview of local and network security issues
- ◆ Understand the secure shell
- ◆ Configure a NTP server

## **Contents**

<b>Basic Security.....</b>	<b>88</b>
1. Local Security.....	89
2. Network Security.....	91
3. The Secure Shell.....	95
4. Time Configuration.....	97
5. Exercises and Summary.....	99

## 1. Local Security

### The BIOS

If anyone has access to a rescue disks or a linux disk that boots from a floppy or a CDROM it is extremely easy to gain read access to any files on the system. To prevent this the BIOS should be set to boot only off the hard drive. Once this is done set a password on the BIOS.

### LILO

LILO can be given options at boot time. In particular some Linux distributions will not ask for a password when starting the system in *single user* mode or runlevel 1.

There are two options that should be added to the `/etc/lilo.conf`:  
 the **restricted** option prompts the user for a password  
 the **password=""** option, set the password string.

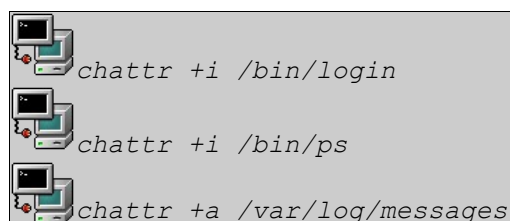
Restricted means that LILO cannot be given any parameters without the "password" specified in `lilo.conf`.

```
boot=/dev/hda
install=/boot/boot.b
prompt
timeout=50
password="password"
restricted
```

### File permissions

To prevent attackers causing too much damage it is recommended to take the following steps.

1) Make vital system tools immutable, or logfiles append-only:


 A terminal window showing three commands being executed: `chattr +i /bin/login`, `chattr +i /bin/ps`, and `chattr +a /var/log/messages`. Each command is preceded by a small icon of a computer monitor.

2) Make directories `/tmp` and `/home` nosuid or noexec:

Lines to be changed in <code>/etc/fstab</code>					
<code>/tmp</code>	<code>/tmp</code>	<code>ext2</code>	<code>nosuid</code>	<code>1</code>	<code>2</code>
<code>/home</code>	<code>/home</code>	<code>ext2</code>	<code>noexec</code>	<code>1</code>	<code>2</code>

3) Find all files on the system that don't belong to a user or a group:



```
find / -nouser -o -nogroup
```

```
find / -perm +4000
```

## Log Files

The main logs are

**/var/log/messages** : contains information logged by the **syslogd** daemon

**/var/log/secure** : contains information on failed logins, added users, etc.

The **last** tool lists all successful logins and reboots. The information is read from the **/var/log/wtmp** file.

The **who** and **w** tools list all users currently logged onto the system using the **/var/run/utmp** file.

## User Limits

When the **/etc/nologin** file is present (can be empty) it will prevent all users from login in to the system (except user root). If the **nologin** file contains a message this will be displayed after a successful authentication.

In the **/etc/security/** directory are a collection of files that allow administrators to limit user CPU time, maximum file size, maximum number of connections, etc

**/etc/security/access.conf** : disallow logins for groups and users from specific locations.

**/etc/security/limits.conf**

The format of this file is

**<domain>      <type> <item> <value>**

<b>domain</b>	a user name, a group name (with @group)		
<b>type</b>	hard or soft		
<b>item</b>	<i>core</i>	- limits the core file size (KB)	
	<i>data</i>	- max data size (KB)	
	<i>fsize</i>	- maximum filesize (KB)	
	<i>memlock</i>	- max locked-in-memory address space (KB)	
	<i>nofile</i>	- max number of open files	
	<i>cpu</i>	- max CPU time (MIN)	
	<i>proc</i>	- max number of processes	
	<i>as</i>	- address space limit	
	<i>maxlogins</i>	- max number of simultaneous logins for this user	
	<i>priority</i> -	the priority to run user process with	
	<i>locks</i>	- max number of file locks the user can hold	



## 2. Network Security

In this section we breakdown the network security into host based security and port based security.

### Host Based Security

Access to resources can be granted based on the host requesting the service. This is handled by `tcp_wrappers`. The **libwrap** library also known as `tcp_wrappers` provides host based access control lists for a variety of network services. Many services, such as **xinetd**, **sshd**, and **portmap**, are compiled against the `libwrap` library thereby enabling **tcp\_wrapper** support for these services.

When a client connects to a service with `tcp_wrapper` support, the `/etc/hosts.allow` and `/etc/hosts.deny` files are parsed to challenge the host requesting the service. Based on the outcome the service will either be granted or denied.

The `hosts_access` files have 2, optionally 3 colon separated fields. The first field is the name of the process, the second is the fully qualified host name or domain name with a "leading dot", IP address or subnet with a "trailing dot". Wildcards like `ALL` and `EXCEPT` are also accepted.

The syntax for the `/etc/hosts.{allow | deny}` file is as follows:

```
service : hosts [EXCEPT] hosts
```

Example:

```
/etc/hosts.deny
ALL:          ALL      EXCEPT      .example.com

/etc/hosts.allow
ALL:          LOCAL 192.168.0.
in.ftpd:      ALL
sshd:         .example.com
```

`Tcp_wrappers` can run a command locally upon a host match in the `host_access` files.

This is accomplished with the **spawn** command. With the use of the `%` character, substitutions can be made for the host name and the service.

Example:

```
/etc/hosts.deny
ALL:          ALL : spawn (/bin/echo `date` from %c for %d >> /var/log/tcpwrap.log)
```

For more information on the use of `%` substitutions see the **hosts\_access (5)** man page.

### Port Based Security

## Basic Security



With packet filtering functionality built into the Linux kernel, it is possible to limit access to resources by creating rulesets with utilities such as **ipchains** and **iptables**, which are able to evaluate a packet entering any of its network interfaces. The rules determine what happens to each packet.

We will cover **ipchains** and **iptables** separately. However **ipchains** and **iptables** share the following options

- A Append rule to a chain
- D Delete a rule
- P Change the default Policy for a chain
- I Insert
- F Flush the rules(s) in a chain
- N Create a user defined chain
- X Delete a user defined chain
- L List

### -- ipchains

There are three built in chains in **ipchains**:

input, forward and output

These chains, respectively are evaluated when the packets

- 1) enter the network interface
- 2) transit to another interface or host
- 3) exit the network interface and have been either generated by the local host or forwarded

#### TARGETS:

The possible actions (or TARGETS) are ACCEPT, DENY, REJECT, MASQ, REDIRECT and RETURN, or can possibly point to another user defined chain. Targets are specified with the **-j** flag.

Example: *All packets from 192.168.0.254 will be logged and denied*



```
ipchains -A input -s 192.168.0.254 --log -j DENY
```

**POLICY:** If a packet has gone through all the rules in a specific chain unaltered then it will be dealt with by the default policy rule for that chain. Valid policy targets are DENY (silently drop the packets) or ACCEPT.

Example: *Set the policy for all chains to DENY*



```
ipchains -P input DENY
```



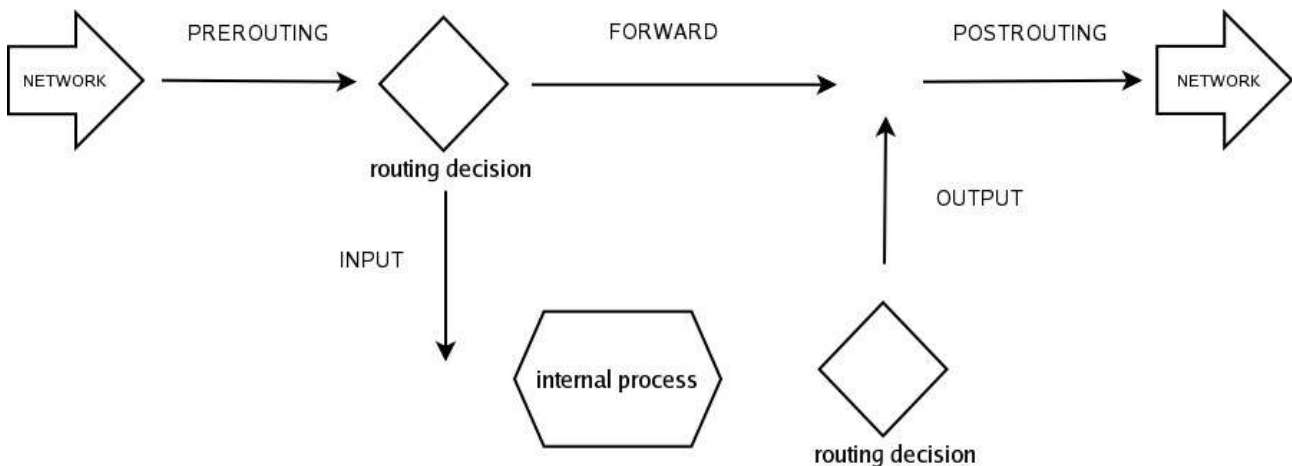
```
ipchains -P forward DENY
```



```
ipchains -P output DENY
```

### -- iptables

One of the main differences with **ipchains** is that the filtering rules (decisions to allow or deny a packet, etc..) have been separated from packet alteration operations (network address translation (NAT), etc). This has been achieved by introducing independent tables, each table is assigned a specific role and each table contains its own built-in chains.



**Figure:** The Netfilter kernel framework for **iptables**

**iptables** has three tables each containing the following built-in chains:

**filter:** this table is the default and deals with filtering rules using its built-in chains INPUT, OUTPUT and FORWARD

**nat:** only network address translation (NAT) operations are defined in this table. The built-in chains are PREROUTING, POSTROUTING and INPUT

**mangle:** this table handles packet alterations other than natting. There are two built-in chains PREROUTING and OUTPUT.


**NOTICE:** the built-in chains for **iptables** are all in UPPERCASE!!

**TARGETS:** Different targets are valid depending on the table.


Valid targets for the **filter** table are DROP, REJECT, ACCEPT or MIRROR.

Valid targets for the **nat** table are REDIRECT (in the PREROUTING and OUTPUT chains), MASQUERADE (in the POSTROUTING chain), DNAT (in the PREROUTING and OUTPUT chains) and SNAT (in the POSTROUTING and OUTPUT chains).

**Example:** All packets from 192.168.0.254 will be logged and denied




```
iptables -A INPUT -s 192.168.0.254 -j LOG
```




```
iptables -A INPUT -s 192.168.0.254 -j DROP
```

POLICY: The iptables chain policy can be set to either DROP, ACCEPT or MIRROR


Example: *The default policy is set to drop all packets*



```
iptables -P INPUT DROP
```



```
iptables -P FORWARD DROP
```




```
iptables -P OUTPUT DROP
```

### **-- more background**


With the development of the 2.4 Linux kernel came the development of the Netfilter project, which uses the iptables utility to manage firewall rules. Another major difference between iptables and ipchains is that iptables has support for evaluating the packets based on their state in terms of other packets that have passed through the kernel. It is this stateful packet evaluation that makes iptables far superior.

Example:      1) *Deny all packets on the INPUT chain:*



```
ipchains -P INPUT DENY
```

2) *Accept established connections that have been initiated by the host:*



```
ipchains -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

Example: *A Basic script that will work as a gateway. Here are the highlights:*

- allow IP forwarding:  
    echo "1" > /proc/sys/net/ipv4/ip\_forward

```
- masquerade:
    $IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE

- allow connections to port 80 ONLY:
    $IPTABLES -A INPUT -p TCP -i $INET_IFACE -m state --state NEW --dport http -j ACCEPT
```

```
#!/bin/sh
# Variables
IPTABLES="/sbin/iptables"
LAN_IFACE="eth0"
INET_IFACE="eth1"
INET_IP="1.2.3.4"
LOCALHOST_IP="127.0.0.1/32"
LAN_IP="192.168.0.1/32"
LAN_BCAST="192.168.0.0/24"

# Setup IP Masquerading

echo "1" > /proc/sys/net/ipv4/ip_forward
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE

# Specify the default policy for the built in chains
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP

# Specify INPUT Rules
$IPTABLES -A INPUT -i !$INET_IFACE -j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -m state --state NEW --dport http -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Specify FORWARD Rules
$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Specify OUTPUT RULES
$IPTABLES -A OUTPUT -p ALL -s $LOCALHOST_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
```

### 3. The Secure Shell

---

The secure shell is a secure replacement for **telnet** and remote tools like **rlogin**, **rsh** and **rcp**. The daemon **sshd** is started on the server using the rc-script **/etc/init.d/sshd**. The ssh service uses port 22 and clients connect using the **ssh** tool.

### • Host Authentication

With ssh both the host and the user authenticate. The host authentication is done by swapping keys. The host's public and private keys are usually kept in **/etc/ssh** if you are using OpenSSH. Depending on the protocol used the host key file will be called **ssh\_host\_key** for Protocol 1 and **ssh\_host\_rsa\_key** or **ssh\_host\_dsa\_key** for Protocol 2. Each of these keys have their corresponding public key, for example **ssh\_host\_key.pub**.

When an ssh client connects to a server, the server will give the hosts public key. At this stage the user will be prompted with something like this:

```
The authenticity of host 'neptune (10.0.0.8)' can't be established.  
RSA key fingerprint is 8f:29:c2:b8:b5:b2:e3:e7:ec:89:80:b3:db:42:07:f4.  
Are you sure you want to continue connecting (yes/no)?
```


If you accept to continue the connection the server's public key will be added to the local **\$HOME/.ssh/known\_hosts** file.

### • User Authentication (using passwords)

Then the user is prompted for the password for his account on the remote server and logs in.

### • User Authentication (using keys)

The user authentication can also involve swapping keys. For this the user will need to generate a pair of private/public keys. For example:

A small icon of a computer monitor with a terminal window.

```
ssh-keygen -t dsa -b 1024
```

will generate a 1024 bit DSA key. By default these keys will be saved in **\$HOME/.ssh** and in this example are called **id\_dsa** and **id\_dsa.pub**.

If we assume we have a **id\_dsa.pub** key we can 'plant' this key on a remote account and avoid typing passwords for further connections. To do this we need to copy the content of the file **id\_dsa.pub** into a file called **authorized\_keys2** kept in the remote **\$HOME/.ssh** directory.

#### WARNING

All private keys in **/etc/ssh/\*** and **~/.ssh/\*** should have a permission of 600

### • sshd configuration file

Sample /etc/ssh/sshd\_config file:

```
#Port 22
#Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::

# HostKey for protocol version 1
#HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key
```

### •ssh configuration file

Sample /etc/ssh/ssh\_config or \$HOME/.ssh/config file:

```
# Host *
#   ForwardX11 no
#   RhostsAuthentication no
#   RhostsRSAAuthentication no
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   HostbasedAuthentication no
#   CheckHostIP yes
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
```

#### NOTICE

The **sshd** daemon has been compiled with libwrap. We can see this with the following:

```
ldd /usr/sbin/sshd | grep wrap
libwrap.so.0 => /usr/lib/libwrap.so.0 (0x0075f000)
```

This means that **sshd** is a valid entry for **/etc/hosts.allow** or **/etc/hosts.deny**.

## 4. Time Configuration

### The System date

The system date can be changed with the **date** command. The syntax is:

```
date MMDDhhmmCCYY[.ss]
```

### The Hardware Clock

The hardware clock can be directly changed with the **hwclock** utility. The main options are:

- r or **—show** prints the current times
- w or **—systohc** set the hardware clock to the current system time
- s or **—hctosys** set the system time to the current hardware clock time

### Time Zones

In addition to UCT time some countries apply “day light saving” policies which add or remove an hour at a given date every year. These policies are available on a linux system in **/usr/share/zoneinfo/**. By copying the appropriate zone file to **/etc/localtime** one can enforce a particular zone policy.

For example if we copy **/usr/share/zoneinfo/Hongkong** to **/etc/localtime** the next time we run **date** this will give us the time in Hongkong. This is because **date** will read **/etc/localtime** each time it is run.

### Using NTP

The Coordinated Universal Time (UTC) is a standard used to keep track of time based on the Earth's rotation about its axis. However because of the slight irregularities of the rotation leap seconds need to be inserted into the UTC scale using atomic clocks.

Since computers are not equipped with atomic clocks the idea is to use a protocol to synchronize computer clocks across the Internet. NTP stands for **Network Time Protocol** and is one such protocol.

Computers that are directly updated by an atomic clock are called primary time servers and are used to update a larger number of secondary time servers. This forms a tree structure similar to the DNS structure. The root servers are on the first level or stratum, the secondary server on the second and so on.

#### Configuring a client to query an NTP server:

An NTP daemon called **ntpd** is used to regularly query a remote time server. All that is needed is a **server** entry in **/etc/ntp.conf** pointing to a public or corporate NTP server. Public NTP servers can be found online.

The NTP protocol can also estimate the frequency errors of the hardware clock from a sequence of queries, this estimate is written to a file referred to by the **driftfile** tag.



**Minimal /etc/ntp.conf file**

```
server ntp2.somewhere.com
driftfile /var/lib/ntp/drift
```

NOTICE: on some systems the **driftfile** tag is pointing to **/etc/ntp.drift** or **/etc/ntp/drift**.

Once **ntpd** is started it will itself be an NTP server providing services on port 123 using UDP.

One off queries:

The **ntp** package also provides the **ntpdate** tool which can be used to set the time from the command line:

```
ntpdate ntp2.somewhere.com
```

## 5. Exercises and Summary

### Files

Files	Description
/etc/fstab	noexec – mount option which prevents any executables to execute from the device nosuid – mount option which prevents the SUID and SGID bits to take effect (see LPI101)
/etc/localtime	contains the time zone policy used to determine the system time (with <b>date</b> )
/etc/ntp.conf	configuration file for the NTP daemon <b>ntpd</b>
/etc/ntp.drift or /etc/ntp.drift	file used by <b>ntpd</b> to keep track of the hardware clock drift
/etc/security/access.conf	file used to grant or deny access based on the user's name and the origin (local tty or remote host). One can also specify a NIS group using <i>@group</i> notation
/etc/security/limits.conf	file used to impose resource limits on login (see the file itself for details)
/etc/ssh	directory containing configuration files for both the <b>ssh</b> client and the <b>sshd</b> server
/usr/share/zoneinfo/	collection of time zone files. Depending on the user's location one of these files is copied to <b>/etc/localtime</b>
/var/log/messages	the main system log file
/var/log/secure	log file containing information about failed logins or user accounts
/var/log/wtmp	the <b>wtmp</b> file records all logins and logouts.
/var/run/utmp	<b>utmp(5)</b> – the <b>utmp</b> file allows one to discover information about who is currently using the system. There may be more users currently using the user's private key used during the user authentication process of an ssh sessionhe system, because not all programs use <b>utmp</b> logging
\$HOME/.ssh	directory containing <b>knownhosts</b> , <b>authorized_keys2</b> , <b>id_dsa</b> and <b>id_dsa.pub</b>
authorized_keys2	contains a list a public id keys from remote users that are authorised to use this account (via ssh)
id_dsa	the user's private key used during the user authentication process of an ssh session
id_dsa.pub	the user's public key used during the user authentication process of an ssh session – this key must be present in the <b>authorized_keys2</b> file of the account one is attempting to ssh to
known_hosts	list of server public keys used for host authentication
ssh_config	configuration file for <b>ssh</b>
sshd_config	configuration file for <b>sshd</b>

### Commands

Command	Description
chattr	change file attributes on an ext2/3 filesystem (see <b>chattr(1)</b> for details)
date	print or set the system time
hwclock	query or set the hardware clock
ipchains	
iptables	<b>iptables(8)</b> – administration tool for IPv4 packet filtering and NAT
last	<b>last(1)</b> – searches back through the file /var/log/wtmp and displays a list of all users logged in (and out) since that file was created. The pseudo user reboot logs in each time the system is rebooted. Thus last reboot will show a log of all reboots since the log file was created
ntpd	the NTP daemon
ntpdate	<b>ntpdate(1)</b> – sets the local date and time by polling the Network Time Protocol (NTP) server(s) given as the server arguments to determine the correct time. It must be run as root on the local host
ssh	<b>ssh(1)</b> – program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel
ssh-keygen	<b>ssh-keygen(1)</b> – generates, manages and converts authentication keys for ssh(1). ssh-keygen can create RSA keys for use by SSH protocol version 1 and RSA or DSA keys for use by SSH protocol version 2.83
sshd	<b>sshd(8)</b> – daemon program that listens for ssh connections from clients. It is normally started at boot from /etc/rc. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange
who	<b>who(1)</b> – show who is logged on

1. Use **/etc/hosts.deny** to disable **sshd** service from everywhere
2. Find all files in **/usr/** that have the SUID bit set
3. Log onto a remote host using ssh and authenticate using a pair of public/private keys
4. Use **ipchains** (resp. **iptables**) to deny access for all incoming, outgoing and forward traffic by default
5. Configure **ntpd**

# *Linux System Administration*

## Prerequisites

- ◆ None

## Goals

- ◆ Customise system logging system
- ◆ Configure **cron** and **at**
- ◆ Understand backup tools and strategies
- ◆ Finding documentation

## Contents

<b>Linux System Administration.....</b>	<b>101</b>
1. Logfiles and configuration files.....	102
2. Log Utilities.....	104
3. Automatic Tasks.....	105
4. Backups and Compressions.....	107
5. Documentation.....	110
6. Exercises and Summary.....	113

## Overview

We will concentrate on the main tasks of system administration such as monitoring log files, scheduling jobs using **at** and **cron**. This also includes an overview of the documentation available (**manpages** and online resources) as well as some backup concepts.

## 1. Logfiles and configuration files

### *The /var/log/ directory*

This is the directory where most logfiles are kept. Some applications generate their own log files (such as squid or samba). Most of the system logs are managed by the **syslogd** daemon. Common system files are :

cron	keeps track of messages generated when <b>cron</b> executes
mail	messages relating to <b>mail</b>
messages	logs all messages except private authentication authpriv, cron, mail and news
secure	logs all failed authentications, users added/deleted etc

The most important log file is **messages** where most activities are logged.

### *The /etc/syslog.conf file*

When **syslogd** is started it reads the **/etc/syslog.conf** configuration file by default. One can also start **syslogd** with **-f** and the path to an alternative config file. This file must contain a list of items followed by a priority, followed by the path to the log-file:

```
item1.priority1 ; item2.priority2           /path-to-log-file
```

Valid items are :

<b>auth</b> and <b>authpriv</b>	user general and private authentication
<b>cron</b>	cron daemon messages
<b>kern</b>	kernel messages
<b>mail</b>	
<b>news</b>	
<b>user</b>	user processes
<b>uucp</b>	

Valid priorities are: (from highest to lowest)

**emerg**  
**alert**

**crit**  
**err**  
**warning**  
**notice**  
**info**  
**debug**  
**\***  
**none**

Priorities are *minimal*! All higher priorities will be logged too. To force a priority to be **info** only you need to use an '=' sign as in:

user.=info /var/log/user\_activity

#### Listing of /etc/syslog.conf

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.* /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none /var/log/messages

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* /var/log/maillog

# Log cron stuff
cron.* /var/log/cron

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg *
*.emerg @10.1.1.254

# Save boot messages also to boot.log
local7.* /var/log/boot.log
#
news.=crit /var/log/news/news.crit
news.=err /var/log/news/news.err
news.notice /var/log/news/news.notice
```

## 2. Log Utilities

### *The logger command*

The first utility **logger** conveniently logs messages to the `/var/log/messages` file:  
If you type the following:



```
logger program myscript ERR
```

The end of `/var/log/messages` should now have a message similar to this:

```
Jul 17 19:31:00 localhost penguin: program myscript ERR
```

### **local settings**

The **logger** utility logs messages to `/var/log/messages` by default. There are local items defined that can help you create your own logfiles as follows. **local0** to **local7** are available items for administrators to use. The availability depends on the system (RedHat **local7** logs boot-time information in `/var/log/boot.log`). Add the following line to `/etc/syslog.conf`:

```
local4.*                /dev/tty9
```

Restart the **syslogd** or force it to re-read its' configuration file as follows:



```
killall -HUP syslogd
```

The next command will be logged on the `/dev/tty9`



```
logger -p local4.notice "This script is writing to /dev/tty9"
```

An interesting device is the `/dev/speech` this is installed with the Festival tools.

### **logrotate**

The log files are updated using **logrotate**. Usually **logrotate** is run daily as a cron job. The configuration file `/etc/logrotate.conf` contains commands to create or compress files.

#### Listing of logrotate.conf

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
```

```
# send errors to root
errors root
# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own lastlog or wtmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
```

### 3. Automatic Tasks

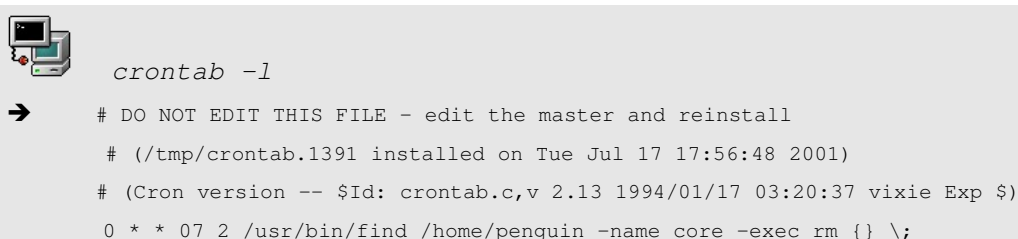
#### Using cron

The program responsible for running crons is called **crond**. Every minute the **crond** will read specific files containing command to be executed. These files are called **crontabs**.

User crontabs are in **/var/spool/cron/<username>**. These files should not be edited directly by non-root users and need to be edited using the **crontab** tool (see below).

The system crontab is **/etc/crontab**. This file will periodically execute all the scripts in **/etc/cron.\*** this includes any symbolic link pointing to scripts or binaries on the system.

To manipulate **cron** entries one uses the **crontab** utility. Scheduled tasks are view with the **-l** option as seen below:



```
crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall
# (/tmp/crontab.1391 installed on Tue Jul 17 17:56:48 2001)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
0 * * 07 2 /usr/bin/find /home/penguin -name core -exec rm {} \;
```

Does the user **root** have any crontabs?

Similarly the **-e** option will open your default editor and lets you enter a cron entry. User **root** can use the **-u** to view and change any user's cron entries. To delete your crontab file use **crontab -r**.

This is the format for crontabs :



Minutes(0-59) Hours(0-23) Day of Month(1-31) Month(1-12) Day of Week(0-6) **command**

#### Permissions:

By default only the root user can use **crontab**. The files **/etc/cron.deny** and **/etc/cron.allow** are available to allow or disallow the creation of crontabs for users listed in **/etc/passwd**.

#### **Scheduling with “at”**

The **at** jobs are run by the **atd** daemon. At jobs are spooled in **/var/spool/at/**

The **at** command is used to schedule a one off task with the syntax


```
at [time]
```

Where time can be expressed as:

```
now
3am + 2days
midnight
10:15 Apr 12
teatime
```

For a complete list of valid time formats see **/usr/share/doc/at-xxx/timespec**.

You can list commands that are scheduled with **atq** or **at -l**. The **at** jobs are saved in **/var/spool/at/**:



```
ls /var/spool/at/
→ a0000100fd244d  spool
```

When using **atq** you should have a list of jobs preceded by a number. You can use this number to dequeue it:



```
atq
→ 1      2001-07-17 18:21 a root
```

From the **atq** listing we see that the job number is **1**, so we can remove the job from the spool as follows:



```
atrm 1
```

#### Permissions:

By default **at** is restricted to the root user. To override this you must either have an empty **/etc/at.deny** or have a **/etc/at.allow** with the appropriate names.

## 4. Backups and Compressions

### Backup strategies


There are three main strategies to back up a system:

**Full:** copy all files  
**Incremental:** The first incremental copies all files added or changed since the last full backup, and subsequently copies all the files added or changed since the last incremental backup  
**Differential:** Copies all files added or changed since the last full backup

Example: If you made a full backup and 3 differential backups before a crash, how many tapes would you need to restore ?


### Creating archives with tar

The main option to create an archive with **tar** is **-c**. You can also specify the name of the archive as the first argument if you use the **-f** flag.



```
tar -cf home.tar /home/
```

If you don't specify the file as an argument **tar -c** will simply output the archive as standard output:



```
tar -c /home/ > home.tar
```

### Extracting archives with tar

Extracting is straight forward. Replace the **-c** flag with an **-x**. This will cause the archive file to create directories if necessary and copy the archived files in your current directory. To redirect the output of the extracted archive into the directory **/usr/share/doc**, for example, you can do:



```
tar xf backeddocs.tar -C /usr/share/doc
```

### Compressions

All archives can be compressed using different compression utilities. These flags are available when creating, testing or extracting an archive:

tar option	compression type
<b>Z</b>	compress
<b>z</b>	gzip

---

j	bzip2.
---	--------

### The *cpio* utility

The **cpio** utility is used to copy files to and from archives. List of files must be given to **cpio** either through a pipe (as when used with **find**) or via a file redirection such as with;

- Extract an archive on a tape:



```
cpio -i < /dev/tape
```

- Create an archive for the */etc* directory:



```
find /etc | cpio -o > etc.cpio
```

### The *dump* and *restore* utilities

Finally, it is also possible to perform backups using **dump**. Remember that the field after the *options* in */etc/fstab* is used to specify if a device should be backed up or not using **dump**. An entire device can be backed up this way. However **dump** can also back directories

When backing up an entire device (not a directory) Information about the previous full or incremental backups is stored in */etc/dumpdates*. **Dump** can automatically do up to 9 incremental backups.

By default **dump** will save the archive to */dev/st0*. Backups are recovered with the **restore** utility.



```
dump -0 -f /tmp/etc.dump /etc
```

You can test this archive with



```
restore -t -a -f /tmp/etc.dump
```

Extract all the files with



```
restore -x -a -f /tmp/etc.dump
```

or you can interactively extract a list of files (that gets interactively created too):

```
restore -i -a -f /tmp/etc.dump
restore > add etc/passwd etc/group
restore > extract
restoring ./etc/group
restoring ./etc/passwd
set owner/mode for '.'? [yn] y
restore > ^ D
```

### Backing up with dd

Remember from LPI 101 that the **dd** tool can make an image of a device preserving everything including:

- the underlying filesystem
- the boot sector (first 512 kB)

The image can be saved to a file or a device. The same is true retrieving the image.

Syntax:

**dd if=FILE/DEVICE of=FILE/DEVICE**

### What to backup

The following table extracted from the FHS document is used to determine how often specific directories need to be backed up:

	<i>shareable</i>	<i>unshareable</i>
<i>static</i>	/usr, /opt	/etc, /boot
<i>variable</i>	/var/mail	/var/run, /var/spool/mail

## 5. Documentation

### Manpages and the whatis database

The manpages are organised in sections	
NAME	the name of the item followed by a short one line description.
SYNOPSIS	the syntax for the command
DESCRIPTION	a longer description
OPTIONS	a review of all possible options and their function
FILES	files that are related to the current item (configuration files etc)
SEE ALSO	other manpages related to the current topic

These are the main sections one can expect to find in a manpage.

The **whatis** database stores the NAME section of all the manpages on the system. This is done through a daily **cron**. The **whatis** database has the following two entries:

<b>name (key) - one line description</b>
--

The syntax for **whatis** is:  
**whatis** <string>

The output is the full NAME section of the manpages where *string* matched *named(key)*

One can also use the **man** command to query the **whatis** database. The syntax is

**man -k** <string>

This command is similar to **apropos**. Unlike **whatis** this will query both the “name” and the “one line description” entries of the database. If the string matches a word in any of these fields the above query will return the full NAME section.

Example: (the matching string has been highlighted)

```
whatis lilo
lilo                (8) - install boot loader
lilo.conf [lilo]    (5) - configuration file for lilo
```

```
man -k lilo
grubby              (8) - command line tool for configuring grub, lilo, and elilo
lilo                (8) - install boot loader
lilo.conf [lilo]    (5) - configuration file for lilo
```

The FHS recommends manpages to be kept in **/usr/share/man**. However additional locations can be searched using the **MANPATH** environment variable set in **/etc/man.config**. Each directory is further divided into subdirectories corresponding to manpage sections.

Manpage Sections	
Section 1	Information on executables
Section 2	System calls, e.g mkdir(2)
Section 3	Library calls, e.g stdio(3)
Section 4	Devices (files in /dev)
Section 5	Configuration files and formats
Section 6	Games
Section 7	Macro packages
Section 8	Administration commands

To access a specific section *N* one has to enter:

**man *N* command**

Examples:

```
man mkdir  
man 2 mkdir
```

```
man crontab  
man 5 crontab
```

### Info pages

The FHS recommends info pages be kept in **/usr/share/info**. These pages are compressed files that can be read with the **info** tool.

The original GNU tools used info pages rather than manpages. Since then most info pages have been rewritten as manpages. However information about GNU projects such as **gcc** or **glibc** is still more extensive in the info pages compared to the manpages.

### Installed documents

GNU projects include documents such as a FAQ, README, CHANGELOG and sometimes user/admin guides. The formats can either be ASCII text, HTML, LaTeX or postscript.

These documents are kept in the **/usr/share/doc/** directory.

### HOWTOs and The Linux Documentation Project

The Linux Documentation Project provides many detailed documents on specific topics. These are structured guides explaining concepts and implementations. The website URL is [www.tldp.org](http://www.tldp.org). The LDP documents are freely redistributable and can be contributed too using a GPL type licence.

### Usenet News Groups

The main newsgroups for Linux are the **comp.os.linux.\*** groups (e.g comp.os.linux.networking, comp.os.linux.security ...). Once you have setup a news reader to connect to a news server (usually available through an ISP or a University campus) one downloads a list of all existing discussion groups and subscribes/unsubscribes to a given group.

There are many experienced as well as new users which rely on the newsgroups to get information on specific tasks or projects. Take the time to answer some of these questions if you feel you have the relevant experience.

### Notifying Users about the System

It is possible to print information for users login onto the system such as the sysadmin's contact details or the state of the system using either **/etc/issue** (**/etc/issue.net** for telnet users) or **/etc/motd**.

The *issue* file is printed on the login terminals (ttys) by **mingetty** and can be used to publish the companies warning regarding the usage of the computer equipment, contact details or even some ASCII art. The same information can be made available through a display manager (see LPI 101). The *issue.net* file is visible at a telnet login prompt, it should generally not contain information about the system (OS type, kernel version, etc)

The filename **motd** stand for "message of the day" and is only visible after a successful login.

## 6. Exercises and Summary

### Files

File	Description
/etc/at.allow, at.deny	<b>at.allow(5)</b> – determine which user can submit commands for later execution via <b>at(1)</b> or <b>batch(1)</b> . The format of the files is a list of usernames, one on each line. Whitespace is not permitted. The superuser may always use <b>at</b> . If the file <b>/etc/at.allow</b> exists, only usernames mentioned in it are allowed to use <b>at</b> . If <b>/etc/at.allow</b> does not exist, <b>/etc/at.deny</b> is checked
/etc/cron.allow, cron.deny	<b>crontab(1)</b> – If the <b>cron.allow</b> file exists, then you must be listed therein in order to be allowed to use this command. If the <b>cron.allow</b> file does not exist but the <b>cron.deny</b> file does exist, then you must <b>not</b> be listed in the <b>cron.deny</b> file in order to use this command. If neither of these files exists, only the super user will be allowed to use this command
/etc/crontab	System crontab file read by the <b>crond</b> daemon whenever its modified time is changed
/etc/dumpdates	Stores information about the last full or incremental dumps
/etc/issue	Message printed by the <b>mingetty</b> program at the login prompt on a <b>tty</b>
/etc/issue.net	Message printed by the telnet daemon at the login prompt
/etc/logrotate.conf	Configuration file for <b>logrotate</b>
/etc/motd	Message displayed by <b>login</b> after a successful login
/etc/syslog.conf	Configuration file for <b>syslogd</b>
/usr/share/info	Directory where info pages are stored
/usr/share/man	Directory where the various sections of the manpages are stored
/var/spool/at/	Directory containing spooled <b>at</b> and <b>batch</b> jobs
/var/spool/cron/	Directory containing user defined crontabs. The crontab file has the name of the user that created it and can only be edited with the <b>crontab -e</b> command

### Commands

Command	Description
apropos	<b>apropos(1)</b> – searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output
at	<b>at(1)</b> – read commands from standard input or a specified file which are to be executed at a later time
atd	<b>atd(8)</b> – run jobs queued by <b>at</b> for later execution
atq	<b>atq(1)</b> – lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed. The format of the output lines (one for each job) is: Job number, date, hour, job class
atrm	deletes jobs, identified by their job number
cron or crond	<b>cron(8)</b> – Cron searches <b>/var/spool/cron</b> for crontab files which are named after accounts in <b>/etc/passwd</b> ; crontabs found are loaded into memory. Cron also searches for <b>/etc/crontab</b> and the files in the <b>/etc/cron.d</b> directory, which are in a different format



Command	Description
crontab	file loaded by <b>crond</b> . It is also the name of the program used to edit crontabs created by users in <b>/var/spool/cron</b>
dd	copy files and devices with optional modifications such as block size ( see <b>info coreutils dd</b> )
dump	<b>dump(8)</b> – examines files on an ext2/3 filesystem and determines which files need to be backed up
info	read info documentation stored in <b>/usr/share/info</b>
logger	allows shell scripts to log messages with <b>syslogd</b>
logrotate	<b>logrotate(8)</b> – is designed to ease administration of systems that generate large numbers of log files. It allows automatic rotation, compression, removal, and mailing of log files. Each log file may be handled daily, weekly, monthly, or when it grows too large
man -k	same as <b>apropos</b>
restore	restore files or file systems from backups made with dump
syslogd	The system logger. Programs can send messages to <b>syslogd</b> which include information such as the date and the host name. The configuration file <b>/etc/syslog.conf</b> is used to customise where messages are logged (e.g file, device or remote logger)
tar	<b>tar(1)</b> – an archiving program designed to store and extract files from an archive file known as a tarfile. A tarfile may be made on a tape drive, however, it is also common to write a tarfile to a normal file
whatis	<b>whatis(1)</b> – search the whatis database for complete words

### Logging

1. Change **/etc/syslog.conf** to output some of the logs to **/dev/tty9** (make sure you restart **syslogd** and that the output is properly redirected)
2. Add a custom local5 item with critical priority to **/etc/syslog.conf** and direct the output to **/dev/tty10**. Restart **syslogd** and use **logger** to write information via local5.
3. Read the **/etc/rc.d/init.d/syslog** script and change **/etc/sysconfig/syslog** to allow remote hosts to send log outputs.

### Scheduling

4. Create a cron entry which starts **xclock** every 2 minutes. Remember that **cron** is unaware of system variables such as **PATH** and **DISPLAY**.
5. Use **at** to start **xclock** in the next five minutes.

### Archiving

6. Use **find** to list all files that have been modified during the past 24 hours.

(hint: Redirect the output of `find -mtime -1` to a file)

7. Use **cpio** to create an archive called `Incremental.cpio`.

(ans: Use the file created above and do `cat FILE | cpio -ov > Incremental.cpio`)

8 Use **tar** to create an archive of all files last accessed or changed 5 mins ago. (HINT: use **find** to create a list of files, then save the list to a file. The **tar** tool has a switch to take input from a file.

9. Test the archive before extracting it.

10. Extract the archive you have just created.

## Setting up PPP

### Prerequisites

- ◆ Hardware Configuration (see LPI 101)

### Goals

- ◆ Configure a modem for dial up
- ◆ Understand the roles of the **pppd** daemon and the **chat** script
- ◆ Configure options in **/etc/ppp/options** such as hardware flow control or persistent connections

### Contents

<b>Setting up PPP.....</b>	<b>116</b>
1. Detecting Modems.....	117
2. Dialup Configuration .....	118
3. pppd and chat .....	119
4. PPPD peers.....	120
5. Wvdial.....	121
6. Exercises and Summary.....	122

## 1. Detecting Modems

Linux assumes in general that serial modems are connected to a serial port (one of the `/dev/ttySN` devices). So you first need to find out which serial port the modem is connected to.

The **setserial -g** command will query the serial ports. If the resource is not available then the UART value will be unknown.

### Sample output for setserial:

```
setserial -g /dev/ttyS[0-3]
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
/dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
/dev/ttyS2, UART: unknown, Port: 0x03e8, IRQ: 4
/dev/ttyS3, UART: unknown, Port: 0x02e8, IRQ: 3
```

For non-serial modems it is possible to get information about available resources in `/proc/pci`. Here the i/o and IRQ settings can be transferred to a free `/dev/ttyS?` device. This is achieved with the following 2 lines:

```
setserial /dev/ttyS2 port 0x2000 irq 3
setserial /dev/ttyS2 autoconfig
```

The last line simply deals with setting up the proper UART settings.

These settings will be lost at the next boot and can be saved in `/etc/rc.serial`. This script is one of the last scripts executed by **rc.sysinit** at boot time.

### The rc.serial script:

```
#!/bin/bash

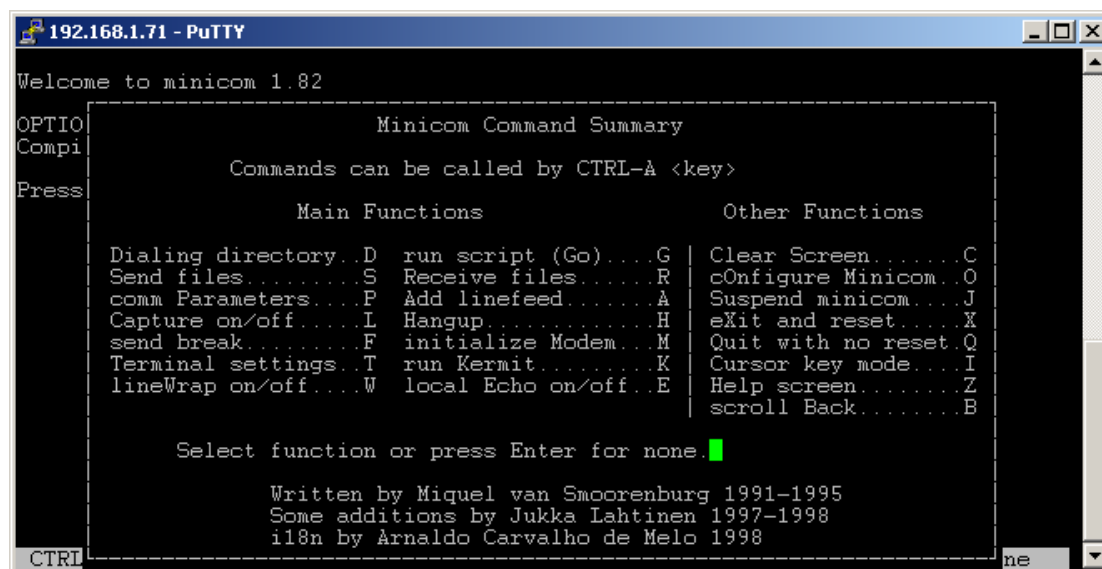
TTY=/dev/ttyS2
PORT=0x2000
IRQ=3

echo "Setting up Serial Card ..."
/bin/setserial $TTY port $PORT irq $IRQ 2>/dev/null
/bin/setserial $TTY autoconfig 2>/dev/null
```

## 2. Dialup Configuration

Once the modem is known to be connected to a serial device it is possible to send modem specific instruction such as **ATZ** or **ATDT**. One tool that will act as a terminal interface is **minicom**.

*minicom screenshot:*



Another common tool is **wvdialconf**. This tool will automatically scan for modems on the ttyS's and create a configuration file called **/etc/wvdial.conf**. The next command will create or update the configuration file

```
wvdialconf /etc/wvdial.conf
```

This file is used to handle password authentication and initialise the **pppd** daemon once the connection is established. If a dialer called **MYISP** is defined in **wvdial.conf** then the connection is started using

```
wvdial MYISP
```

### 3. pppd and chat

First of all the **chat** script is used to communicate with a remote host's modem. It is a series of expect/send strings. The format is:

'expected query' 'answer'
---------------------------

Expected queries from the modem are:

' ' 'OK' 'CONNECT' 'login' 'password' 'TIMEOUT' '>'

The script is read sequentially and starts with the empty query ' ' which is matched with the command **'ATZ'**. Once the modem is initialised it sends back the query **'OK'**. To this the script will answer with a **'ATDT'** dialing command. This conversation goes on and on until the '>' prompt is reached at which stage one can run **pppd**.

*Sample chat script:*

```
'ABORT' 'BUSY'
'ABORT' 'ERROR'
'ABORT' 'NO CARRIER'
'ABORT' 'NO DIALTONE'
'ABORT' 'Invalid Login'
'ABORT' 'Login incorrect'
' ' 'ATZ'
'OK' 'ATDT01172341212'
'CONNECT' ' '
'ogin:' 'adrian'
'ord:' 'adrianpasswd'
'TIMEOUT' '5'
'>' pppd
```

Of course this is one way of doing things. One can also start **pppd** manually and then invoke the chat script as follows:

```
pppd /dev/ttyS2 115200 \
nodetach \
lock \
debug \
crtsets \
asynmap 0000000 \
connect "/usr/sbin/chat -f /etc/sysconfig/network-scripts/chat-ppp0"
```

The lines below the **pppd** commands can be saved in **/etc/ppp/options**. This file contains most of the features which make the strength and flexibility of **pppd**.

The main options for **/etc/ppp/options** are listed in the next table.

Option	Description
crtcts	use hardware flow control using the RTS and CTS signals
noauth	do not require the peer to authenticate itself
persist	do not exit after a connection is terminated but try to reconnect
require-chap	use <b>/etc/ppp/chap-secrets</b> for authentication

Once a serial connection is established the **pppd** daemon will start the PPP protocol. At this point a network interface called *pppN* is assigned an IP address with the script **/etc/ppp/ip-up**.

When a connection is terminated the **pppd** daemon releases the IP with the **/etc/ppp/ip-down** script.

## 4. PPPD peers

There is a directory called **peers** in **/etc/ppp/**. In this directory one can create a file that contains all the necessary command line options for **pppd**. In this way peer connections can be started by all users.

Below is an example of a PPP peer file:

```
# This optionfile was generated by pppconfig 2.0.10.
hide-password
noauth
connect "/usr/sbin/chat -f /etc/sysconfig/network-scripts/chat-ppp0"
/dev/ttyS0
115200
defaultroute
noipdefault
user uk2
```

The previous peer file (called uk2) would be used as follows:

```
# pppd call uk2
```

This will dial the number specified in the “chat script” and authenticate as the user “uk2”. Please note that this requires a corresponding entry in the **/etc/ppp/chap-secrets**, and **/etc/ppp/pap-secrets**. The format for pap and chap secrets is as follows:

```
# Secrets for authentication using CHAP
# client      server      secret          IP addresses
uk2          *          "uk2"          *
```

This format allows different passwords to be used if you connect to different servers. It also allows you to specify an IP address. This is probably not going to work when connecting to an ISP, but when making private connections, you can specify IP addresses if there is a need. One example would be where you need to audit your network activity, and want to specify which users get a certain IP address.

## 5. Wvdial

This is the default method used by Red Hat to connect to a dial up network. To configure wvdial, it is easier to use one of the configuration tools provided with either Gnome or KDE. They configure the

## Setting up PPP

---



**/etc/wvdial.conf** file.

Below is a sample wvdial.conf file:

```
[Modem0]
Modem = /dev/ttyS0
Baud = 115200
Dial Command = ATDT
Init1 = ATZ
FlowControl = Hardware (CRTSCTS)
[Dialer UK2]
Username = uk2
Password = uk2
Phone = 08456091370
Inherits = Modem0
```

To use wvdial from the command line, you would execute it with the following syntax:

```
# wvdial <dialer-name>
```

In the example configuration file the following command would dial the connection called “uk2”

```
# wvdial uk2
```



## 6. Exercises and Summary

### Files

File	Description
/etc/ppp/options	options used by the <b>pppd</b> daemon (additional options can be passed on the command line)
/etc/ppp/chap-secrets	contains login information available when using the challenge handshake authentication protocol (CHAP)
/etc/ppp/pap-secrets	contains login information available when using the password authentication protocol (PAP)
/etc/ppp/peers/	contains files with connection information (user name, chat script) as well as <b>pppd</b> options
/etc/wvdial.conf	configuration file used by <b>wvdial</b>

### Commands

Command	Description
chat	<b>chat(8)</b> – The <code>chat</code> program defines a conversational exchange between the computer and the modem. Its primary purpose is to establish the connection between the Point-to-Point Protocol Daemon ( <code>pppd</code> ) and the remote <code>pppd</code> process
minicom	program used to communicate over a serial connection. Can be given a phone number, user name and password. Once the connection is established <b>minicom</b> acts as a terminal
pppd	<b>pppd(8)</b> – PPP is the protocol used for establishing internet links over dial-up modems, DSL connections, and many other types of point-to-point links. The <b>pppd</b> daemon works together with the kernel PPP driver to establish and maintain a PPP link with another system (called the peer) and to negotiate Internet Protocol (IP) addresses for each end of the link. <code>Pppd</code> can also authenticate the peer and/or supply authentication information to the peer.
wvdial	<b>wvdial(1)</b> – <code>wvdial</code> is an intelligent PPP dialer, which means that it dials a modem and starts PPP in order to connect to the Internet. It is something like the <code>chat(8)</code> program, except that it uses heuristics to guess how to dial and log into your server rather than forcing you to write a login script

# *Printing*

## Prerequisite

- ◆ None

## Goals

- ◆ Understand the GNU printing tools used to submit and administrate print jobs
- ◆ Configure a **LPRng** print spooler

## Contents

<b>Printing.....</b>	<b>123</b>
1. Filters and gs.....	124
2. Printers and print queues.....	124
3. Printing Tools.....	125
4. The configuration files.....	126
5. Exercises and Summary.....	129

## 1. Filters and gs

For non-text formats Linux and UNIX systems generally use filters. These filters translate `JPEG` or `troff` file formats into a postscript type format. This could directly be sent to a postscript printer, but since not all generic printers can handle postscript, an intermediate 'virtual postscript printer' is used called ghostscript or **gs** which translates the postscript into printer compatible language (PCL) or something that the printer understands.

The commercial version of ghostscript is Aladdin Ghostscript and the GNU version is derived from this.

The **gs** utility has a database of printer drivers it can handle (this list is usually up to date, for example many USB printers are supported) and converts the postscript directly into PCL for these known models. The **gs** utility plays a central role in Linux printing.

## 2. Printers and print queues

As seen above simple ascii text printing is not handled in the same way as image or postscript files. If you only have one printer and you would like to printout your mail for example, it may not be necessary to use a filter. You may want to define a queue without filters, which would print mail faster. You could also define a queue on the same printer, which would only handle postscript files.

All queues and printers are defined in `/etc/printcap`. Here is the full configuration of a remote printer 192.168.1.20 using the remote queue named 'lp':

```
lp:\
    :sd=/var/spool/lpd/lp:\
    :mx#0:\
    :sh:\
    :rm=192.168.1.20:\
    :rp=lp:
```

The essential options here are **rm** the remote host, **sd** the spool directory and **rp** the name of the remote queue. Notice that no filters are specified (you would use **if** for input filter). All the filtering is done on the remote host.

### 3. Printing Tools

#### lpr:

The **lpr** utility is used to submit jobs to a printer. This is a modern version of **lp** (line print). From a user's point of view it is helpful to understand that a printer can be associated with more than one queue. Here are two examples to print a file called LETTER.

*Send job to default printer:*

```
lpr LETTER
```

*Send job to the 'ljet' queue:*

```
lpr -Pljet LETTER
```

*Table1: Main Options for lpr*

<b>#</b> num	Print num copies
<b>-P</b> pq	Specify the print queue pq
<b>-s</b>	Make a symbolic link in the spool directory rather than copy the file in

#### lpq:

A user can monitor the status of print queues with the **lpq** utility. Here are a few examples.

*Show jobs in default queue:*

```
lpq
```

*Show jobs for all queues on the system:*

```
lpq -a
```

*Show jobs in the 'remote' queue:*

```
lpq -Premote
```

#### lprm:

Depending on the options in **/etc/lpd.perms** users may be allowed to delete queued jobs using **lprm**.

*Remove last job submitted:*

```
lprm
```

*Remove jobs submitted by user dhill:*

```
lprm dhill
```

*Remove all submitted jobs:*

```
lprm -a (or simply lprm -)
```

It is possible to remove a specific spooled job by referencing the job number; this number is given by **lpq**.

## lpc:

The Line Printer Control utility is used to control the print queues and the printers. The print queues can be disabled or enabled. Notice that **lprm** on the other hand can remove jobs from the queue but doesn't stop the queue.

One can either use **lpc** interactively (**lpc** has its own prompt), or on the command line.

Here is an output of **lpc --help**:

CMD: /usr/sbin/lpc help

► Commands may be abbreviated. Commands are:

abort	enable	disable	help	restart	status	topq	?
clean	exit	down	quit	start	stop	up	

The **enable/disable/topq/up/down** options relate to queues

The **start/stop** options relate to printers

## mpage:

This tool will format a document to print a fixed number of pages per sheet. The default is four pages per sheet. This is useful to have a quick overview of a document.

## 4. The configuration files

### /etc/printcap

As seen earlier in the chapter, this file defines all printers and queues that the system can use (remote and local).

The default printer can be specified with either variables LPDEST or PRINTER: PRINTER=lp  
If no environmental variable is set the default printer is the first printer defined in **/etc/printcap**.

The main definitions are:

<b>lp</b>	device name, usually /dev/lp0 for the parallel port
<b>mx</b>	maximum file size (zero=nolimit)
<b>sd</b>	spool directory (/var/spool/lpd/<queuename>/)
<b>if</b>	input filter
<b>rm</b>	remote host address or IP
<b>rp</b>	remote queue name

If this file is modified you will need to restart the **lpd** daemon.

### /etc/lpd.conf

This is a very lengthy file and by default all options are commented out. This file is used if an administrator wishes to have more control (i.e remote access authentication, user permissions ...) over the printing.

/etc/lpd.perms

This file controls permission for the **lpc**, **lpq** and **lprm** utilities. In particular you can grant users the right to dequeue their current job (using the **lprm** tool) with the line :

```
ACCEPT      SERVICE=M    SAMEHOST SAMEUSER
```

LPRng uses a system of keys to shorten the entries in **lpd.perms**. This is however not very to understand. For example the service 'M' corresponds to **lprm** in the above line.

Sample /etc/lpd.perms file:

```
## Permissions are checked by the use of 'keys' and matches. For each of
## the following LPR activities, the following keys have a value.
##
## Key          Match Connect Job   Job   LPQ  LPRM  LPC
##              Spool  Print
## SERVICE      S      'X'   'R'   'P'   'Q'   'M'   'C'
## USER         S      -     JUSR  JUSR  JUSR  JUSR  JUSR
## HOST          S      RH     JH    JH    JH    JH    JH
## GROUP         S      -     JUSR  JUSR  JUSR  JUSR  JUSR
## IP            IP     RIP    JIP   JIP   RIP   JIP   JIP
## PORT          N      PORT   -     -     PORT  PORT  PORT
## REMOTEUSER    S      -     JUSR  JUSR  JUSR  CUSR  CUSR
## REMOTEHOST    S      RH     RH    JH    RH    RH    RH
## REMOTEGROUP   S      -     JUSR  JUSR  JUSR  CUSR  CUSR
## REMOTEIP      IP     RIP    RIP   JIP   RIP   RIP   RIP
## CONTROLLINE   S      -     CL    CL    CL    CL    CL
## PRINTER       S      -     PR    PR    PR    PR    PR
## FORWARD       V      -     SA    -     -     SA    SA
## SAMEHOST      V      -     SA    -     SA    SA    SA
## SAMEUSER      V      -     -     -     SU    SU    SU
## SERVER        V      -     SV    -     SV    SV    SV
## LPC           S      -     -     -     -     -     LPC
## AUTH          V      -     AU    AU    AU    AU    AU
## AUTHTYPE      S      -     AU    AU    AU    AU    AU
## AUTHUSER      S      -     AU    AU    AU    AU    AU
## AUTHFROM      S      -     AU    AU    AU    AU    AU
## AUTHSAMEUSER  S      -     AU    AU    AU    AU    AU
##
## KEY:
## JH = HOST          host in control file
## RH = REMOTEHOST    connecting host name
## JUSR = USER        user in control file
## AUTH will match (true) if authenticated transfer
## AUTHTYPE will match authentication type
## AUTHUSER will match client authentication type
## AUTHFROM will match server authentication type and is NULL if not from server
## AUTHSAMEUSER will match client authentication to save authentication in job
##
## Example Permissions
##
## # All operations allowed except those specifically forbidden
## DEFAULT ACCEPT
##
## #Reject connections from hosts not on subnet 130.191.0.0
## # or Engineering pc's
```

```
## REJECT SERVICE=X NOT REMOTEIP=130.191.0.0/255.255.0.0
## REJECT SERVICE=X NOT REMOTEHOST=engpc*
##
## #Do not allow anybody but root or papowell on
## #astart1.astart.com or the server to use control
## #facilities.
## ACCEPT SERVICE=C SERVER REMOTEUSER=root
## ACCEPT SERVICE=C REMOTEHOST=astart1.astart.com REMOTEUSER=papowell
##
## #Allow root on talker.astart.com to control printer hpjet
## ACCEPT SERVICE=C HOST=talker.astart.com PRINTER=hpjet REMOTEUSER=root
## #Reject all others
## REJECT SERVICE=C
##
## #Do not allow forwarded jobs or requests
## REJECT SERVICE=R,C,M FORWARD
##
#
# allow root on server to control jobs
ACCEPT SERVICE=C SERVER REMOTEUSER=root
# allow anybody to get server, status, and printcap
ACCEPT SERVICE=C LPC=lpd,status,printcap
# reject all others
REJECT SERVICE=C
#
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M
# all other operations allowed
DEFAULT ACCEPT
```

### **/etc/hosts.{lpd,equiv}**

These files were used by the LPR printing suite and presented a security risk. When running a print server you needed to specify which hosts could access the printer in **/etc/hosts.lpd**. You also needed to add the hosts to **/etc/hosts.equiv**.

These files are now replaced in LPRng by the **/etc/lpd.perms** file

## 5. Exercises and Summary

Term	Definition
Filter	Scripts used to prepare a document before printing
Device	Type of connection used to access the printer (e.g parallel, USB or network)
Driver	Translates raw or postscript type formats into printer specific instructions such as PCL

### Files

File	Description
/etc/printcap	Read by the <b>lpd</b> daemon at start up and contains a list of configured printers
/etc/lpd.perms	Contains permissions applied to the <b>lpd</b> daemon such as remote access

### Commands

Command	Description
lpc	line printer control program
lpd	line printer daemon
lpq	print printer queue status
lpr	submit files for printing
lprm	remove a queued print job
mpage	print multiple pages of a document on one page

1. Start **printtool** and create a new local queue called **lp**.
2. Customise the device **/dev/tty10** as the printer device (remember to do **chmod 666 /dev/tty10** to allow printing on this device). You now have a virtual printer on your system!
3. Send jobs to the print queue using **lpr** and **pr** (pre-formatting tool)
4. With your system's print tool, define different remote queues:
  - a UNIX queue
  - a SMB queueIf you are the server, make sure the appropriate rules are defined in **/etc/lpd.perms**  
  
In each case
  - check the **/etc/printcap** file. Which filter is used? How is the remote host defined?
  - check the **/var/spool/lpd/** directory.
5. Stop the various printer queues and printers with **lpc**.



6. Check the contents of each queue with **lpq**
7. De-queue selected jobs with **lprm**

## ***LPI 102 Objectives***

### **1. Kernel**

#### Manage/Query kernel and kernel modules at runtime

Manage a kernel and kernel loadable modules. Use command-line utilities to get information about the kernel modules and the running kernel. Load modules with correct parameters and unload them. Load modules using aliases (p.3)

**Keywords:** `/lib/modules/kernel-version/modules.dep` (p.3), `/etc/modules.conf` or `/etc/conf.modules` (p.3)

**depmod (p.3), insmod (p.3), lsmod (p.3), rmmod (p.3), modinfo (p.4), modprobe (p.3), uname (p.9)**

#### Reconfigure, build, and install a custom kernel and kernel modules

Customise, build, and install a kernel and kernel loadable modules from source Customise the current kernel. Build a new kernel or new kernel modules as needed. Install the new kernel and reconfigure the boot loader.

**Keywords:** `/usr/src/linux/*` (p.5), `/usr/src/linux/.config` (p.6), `/lib/modules/kernel-version/*` (p.8), `/boot/*` (p.8)

**make, config (p.6), menuconfig (p.6), xconfig (p.6), oldconfig (p.6), modules (p.8), install, modules\_install (p.8), depmod (p.3)**

### **2. Boot, Initialisation, Shutdown and Runlevels**

#### Boot the system

Follow the system through the booting process. Parse parameters to the boot loader (runlevel and kernel options). Check events in the log files.

**Keywords:** **dmesg (p.22)**, `/var/log/messages`, `/etc/modules.conf` (p.3), LILO (p.19), GRUB (p.19)

#### Change runlevels and shutdown or reboot system

Manage the system's runlevels. The default runlevel. The single user mode. Shutdown and reboot. Alert users before switching runlevel.

**Keywords:** **shutdown (p.25), init (p.15)**, `/etc/inittab` (p.18)

### **3. Printing**

#### Manage printers and print queues

Manage print queues and print jobs. Monitor print server and user print queues. Troubleshoot general printing problems.

**Keywords:** **lpc (p.127), lpq (p.126), lprm (p.126), lpr (p.126)**, `/etc/printcap` (p.127)

#### Print files

Manage print queues and manipulate print jobs. Add and remove jobs from printer queues. Convert text files to postscript for printing.

**Keywords:** **lpr (p.126), lpq (p.126), mpage (p.127)**

#### Install and configure local and remote printers

Install a printer daemon. Install and configure a print filter (e.g.: `apsfilter`, `magicfilter`). Make local and remote printers accessible for a Linux system. SMB shared printers.

## LinuxIT Technical Education Centre

### Appendix

---

**Keywords:** **lpd**, `/etc/printcap` (p.127), `/etc/apsfilter/*`, `/var/lib/apsfilter/*`, `/etc/magicfilter/*`, `/var/spool/lpd/*`

#### 4. Documentation

##### Use and manage local system documentation

Use and administer the manpages and the material in `/usr/share/doc`. Find relevant man pages. Search vmware Error saving serial number: no matchman page sections. Find a command and all the documentation related to it. Configure access to man sources and the man system.

**Keywords:** **man** (p.112), **apropos** (p.111), **whatis** (p.111), `MANPATH` (p.111)

##### Find Linux documentation on the Internet

Find and use Linux documentation. Use Linux documentation from sources such as the *Linux Documentation Project* (LDP) (p.112), vendors and third-party websites. Linux specific newsgroups (p.112). Newsgroup archives. Mailing lists.

##### Notify users on system-related issues

Notify users about current issues related to the system. Logon messages.

**Keywords:** `/etc/issue` (p.9 and p.113), `/etc/issue.net` (p.113), `/etc/motd` (p.113)

#### 5. Shells, Scripting, Programming and Compiling

##### Customise and use the shell environment

Customise shell environments to meet users' needs. Set environment variables at login, or when spawning a new shell. Write bash functions for frequently used sequences of commands.

**Keywords:** `~/.bash_profile` (p.79), `~/.bash_login` (p.79), `~/.profile` (p.79), `~/.bashrc` (p.80), `~/.bash_logout` (p.80), `~/.inputrc` (p.80)

**function** (p.), **export** (p.79), **env** (p.79), **set** (p.79), **unset** (p.79)

##### Customise or write simple scripts

Customise existing scripts. Write simple new shell scripts. Use standard sh syntax (loops, tests). Use command substitution. Test command return-values and file status. Conditionally mailing the superuser. The she-bang (#!) sign. Manage location, ownership, execution and suid rights of scripts.

**Keywords:** **while** (p.83), **for** (p.84), **test** (p.82), **chmod** (p.81 and file permissions in LPI101)

#### 6. Administrative Tasks

##### Manage users and group accounts and related system files

Add, remove, suspend and change user accounts. Manage groups. Change user/group info in `passwd`/group databases. Create special purpose and limited accounts.

**Keywords:** **chage** (p.33), **gpasswd**(p.), **groupadd** (p.29), **groupdel** (p.29), **groupmod**(p.32), **grpconv** (p.31), **grpunconv** (p.31), **passwd** (p.27), **pwconv**(p.30), **pwunconv**(p.30), **useradd**(p.27,p.31), **userdel** (p.33), **usermod** (p.32)

`/etc/passwd` (p.30), `/etc/shadow` (p.30), `/etc/group` (p.31), `/etc/gshadow` (p.31)

##### Tune the user environment and system environment variables

Modify global and user profiles. Set up environment variables. Maintain the `skel` directory. Set command search path.

**Keywords:** **env** (p.79), **export** (p.79), **set** (p.79), **unset** (p.79), `/etc/profile` (p.86), `/etc/skel` (p.31)

## LinuxIT Technical Education Centre

### Appendix

---

#### Configure and use system log files to meet administrative and security needs

Configure system logs. Manage type and level of information logged. Manually scan log files for notable activity. Monitoring log files: automatic rotation and archiving. Track down problems noted in logs.

**Keywords:** **logrotate (p.105), tail -f, /etc/syslog.conf (p.103, p.104), /var/log/\* (p.103)**

#### Automate system administration tasks by scheduling jobs to run in the future

Use **cron** or **anacron** to run jobs at regular intervals. Use **at** to run jobs once. Manage **cron** and **at** jobs. Configure user access to **cron** and **at** services.

**Keywords:** **at (p.107), atq (p.107), atrm(p.107), crontab (p.106)**

**/etc/anacrontab, /etc/at.deny (108), /etc/at.allow (p.107), /etc/crontab (p.106), /etc/cron.allow (p.107), /etc/cron.deny (p.107), /var/spool/cron/\* (p.106)**

#### Maintain an effective data backup strategy

Plan a backup strategy. Automatically backup filesystems to various media. Dump a raw device to a file and vice versa. Perform partial and manual backups. Verify the integrity of backup files. Partially or fully restore backups.

**Keywords:** **cpio (p.109), dd (p.110), dump (p.109), restore (p.109), tar (p.108)**

#### Maintain system time

Maintain the system time and synchronize the clock over NTP. Set the system date and time. Set the BIOS clock to the correct time in UTC, configuring the correct time zone for the system and configuring the system to correct clock drift to match NTP clock.

**Keywords:** **date (p.97), hwclock (p.98), ntpd (p.98), ntpdate (p.99)**

**/usr/share/zoneinfo (p.98), /etc/timezone (p.), /etc/localtime(p.98), /etc/ntp.conf (p.98), /etc/ntp.drift (p.99)**

## 7. Networking Fundamentals

#### Fundamentals of TCP/IP

Understand IP-addresses, network masks and broadcast address. Determine the network address, broadcast address and netmask when given an IP-address and the number of bits. 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) (p.53) and the more common TCP and UDP ports (20, 21, 23, 25, 53, 80, 110, 119, 139, 143, 161).

**Keywords:** **/etc/services (p.54), ftp (p.61), telnet (p.60), host, ping, dig, traceroute, whois**

#### TCP/IP configuration and troubleshooting

View, change and verify configuration settings for various network interfaces. Manual and onboot configuration for interfaces and routing tables. Configure and correct routing tables. Configure Linux as a DHCP client.

**Keywords:** **/etc/HOSTNAME (p.38) or /etc/hostname, /etc/hosts (p.38), /etc/networks (p.41), /etc/host.conf (see DNS section p.66), /etc/resolv.conf (p.38), /etc/nsswitch.conf (see DNS section p.67)**

**ifconfig (p.39), route (p.40), dhcpcd (p.40), dhcpcclient (p.40), pump (p.40), host, hostname (domainname, dnsdomainname), netstat (p.43), ping (p.42), traceroute (p.44), tcpdump (p.42)**

#### Configure Linux as a PPP client

## LinuxIT Technical Education Centre

### Appendix

Understand the basics of the PPP protocol. Configure PPP for outbound connections. Define the chat sequence when connecting. Initialisation and termination of a PPP connection with a modem, ISDN or ADSL. Set up PPP to automatically reconnect if disconnected.

**Keywords:** `/etc/ppp/options.*` (p.120), `/etc/ppp/peers/*` (p.121), `/etc/wvdial.conf` (p.119)

**/etc/ppp/ip-up** (p.121), **/etc/ppp/ip-down** (p.121), **wvdial** (p.119), **pppd** (p.120)

#### 8. Networking Services

##### Configure and manage inetd, xinetd, and related services

Configure services available through inetd. Use tcpwrappers. 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.

**Keywords:** `/etc/inetd.conf` (p.58), `/etc/hosts.allow` (p.62), `/etc/hosts.deny` (p.62), `/etc/services` (p.59), `/etc/xinetd.conf` (p.59), `/etc/xinetd.log`

##### Operate and perform basic configuration of sendmail

Modify simple parameters in sendmail configuration files. 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 mailserver. It does not include advanced custom configuration of Sendmail.

**Keywords:** `/etc/sendmail.cf` (p.71), `/etc/aliases` (p.71), `/etc/mail/*` (p.71), `~/forward` (p.72)

**mailq** (p.72), **sendmail** (p.71), **newaliases** (p.72)

##### Operate and perform basic configuration of Apache

Modify simple parameters in Apache configuration files. Start, stop, and restart httpd. Does not include advanced custom configuration of Apache.

**Keywords:** **apachectl** (p.73), **httpd**, `httpd.conf` (p.72)

##### Properly manage the NFS, smb, and nmb daemons

Mount remote filesystems using NFS. Configure NFS for exporting local filesystems. Start, stop, and restart the NFS services. Install and configure Samba using GUI tools or direct edit of the `/etc/smb.conf` file.

Sharing of home directories and printers, as well as correctly setting the nmbd as a WINS client.

**Keywords:** `/etc/exports` (p.63), `/etc/fstab` (p.63), `/etc/smb.conf` (p.65), **mount** (p.63), **umount**

##### Setup and configure basic DNS services

Configure hostname lookups and troubleshoot problems with local caching-only name server. Understand the domain registration and DNS translation process. Differences between bind 4 and bind 8 configuration files.

**Keywords:** `/etc/hosts` (p.67), `/etc/resolv.conf` (p.67), `/etc/nsswitch.conf` (p.67), `/etc/named.boot` (v.4) (p.68) or `/etc/named.conf` (v.8) (p.68), **named**

##### Set up secure shell (OpenSSH)

## LinuxIT Technical Education Centre

### Appendix

Obtain and configure OpenSSH. Basic OpenSSH installation and troubleshooting. Configure **sshd** to start at system boot.

**Keywords:** `/etc/hosts.allow` (p.97), `/etc/hosts.deny` (p.97), `/etc/nologin`, `/etc/ssh/sshd_config` (p.97), `/etc/ssh_known_hosts` (p.96), `/etc/sshrc` **sshd** (p.96), **ssh-keygen** (p.96)

## 9. Security

### Perform security administration tasks

Ensure local security policies. Configure TCP wrappers. Find files with SUID/SGID bit set. Verify packages. Set or change user passwords and password ageing information. Update binaries as recommended by CERT, BUGTRAQ or distribution's security alerts. Basic knowledge of **ipchains** and **iptables**.

**Keywords:** `/proc/net/ip_fwchains`, `/proc/net/ip_fwnames`, `/proc/net/ip_masquerade`, **find** (p.90), **ipchains** (p.92), **passwd**, **socket**, **iptables** (p.92)

### Setup host security

Set up a basic level of host security. Configure syslog, shadowed passwords. Set up a mail alias for root. Turn off unused network services.

**Keywords:** `/etc/inetd.conf` or `/etc/inet.d/*`, `/etc/nologin` (p.90), `/etc/passwd`, `/etc/shadow`, `/etc/syslog.conf`

### Setup user level security (p.90)

Configure user level security. Limits on user logins, processes, and memory usage.

**Keywords:** **quota**, **usermod** (see lpi 101)

Copyright (c) 2003 LinuxIT.  
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being History, Acknowledgements, with the Front-Cover Texts being "released under the GFDL by LinuxIT".

## GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the

# Licence Agreement



notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties; any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must



# Licence Agreement



take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- **O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already

# Licence Agreement



includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate

# Licence Agreement



your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## Index

<b>A</b>		
apachectl	73	
apropos	111	
arp	43	
at	107	
atd	107	
atq	107	
atrm	107	
<b>B</b>		
backup strategies	108	
bootloaders		
1024 cylinders	20	
broadcast address	50	
<b>C</b>		
chage	33	
chattr	89	
classless subnets	52	
cron	103	
crond	106	
crontab	106	
<b>D</b>		
date	97	
dd	110	
depmod	3	
dhcpcd	40	
dhcpcclient	40	
dig	70	
<b>DNS</b>		
/etc/named.boot	68	
/etc/named.conf	69	
dig	70	
DNS Configuration Files	68	
DNS zone files	69	
Hierarchical structure	68	
host	70	
Types of DNS servers	68	
Dotted Quad	49	
driftfile	98	
dump	109	
<b>E</b>		
env	79	
export	79	
exportfs	63	
<b>F</b>		
<b>Files</b>		
/etc/hosts.allow	62	
/etc/aliases	72	
/etc/at.allow	108	
/etc/at.deny	108	
/etc/bashrc	80	
/etc/cron.allow	107	
/etc/cron.deny	107	
/etc/crontab	106	
/etc/dumpdates	109	
/etc/exports	63	
/etc/host.conf	66	
/etc/hosts	67	
/etc/hosts.deny	62	
/etc/inetd.conf	58	
/etc/inputrc	80	
/etc/issue	113	
/etc/issue.net	113	
/etc/localtime	98	
/etc/logrotate.conf	105	
/etc/mail/*	71	
/etc/motd	113	
/etc/named.boot	68	
/etc/named.conf	69	
/etc/networks	41	
/etc/nsswitch.conf	66	
/etc/ntp.conf	98	
/etc/ntp.drift	99	
/etc/ntp.drift	99	
/etc/ppp/ip-down	121	
/etc/ppp/ip-up	121	
/etc/printcap	125, 127	
/etc/profile	79	
/etc/resolv	67	
/etc/resolv.conf	38, 67	
/etc/security/access.conf	90	
/etc/security/limits.conf	90	
/etc/sendmail.cf	71	
/etc/services	55	
/etc/shadow	30	
/etc/smb.conf	65	
/etc/ssh	96	
/etc/sysctl.conf	39	
/etc/syslog.conf	103p.	
/etc/wvdial.conf	121	
/etc/xinetd.conf	59	
/lib/modules/	2p.	
/proc/sys/net/ipv4/ip_forward	39	
/usr/share/man	111	
/usr/share/zoneinfo/	98	
/var/log/httpd	73	
/var/log/messages	90	
/var/log/secure	90	
/var/log/wtmp	90	
/var/run/utmp	90	
/var/spool/at/	107	
/var/spool/cron/<username>	106	
/var/spool/mail/	72	
/var/spool/mqueue	72	
~/.bash_profile	79	
~/.bashrc	80	
~/.forward	72	
~/.inputrc	80	
\$HOME/.ssh	96	
access.conf	73	
authorized_keys2	96	
httpd.conf	73	
id_dsa	96	
id_dsa.pub	96	
known_hosts	96	
modutils (package)	3	
srm.conf	73	

ssh_config	97		mpage	127	
sshd_config	97		N		
find	90		netmask	49	
ftp	61		netstat	43	
G			network address		50
gpsswd	29		News Groups		112
groupadd	32		NFS	62	
groupadd	29		noexec	89	
groups	28		nosuid	89	
H			NTP - network time protocol		98
host	70		ntpd	98	
HOWTOs	112		ntpdate	99	
hwclock	98		P		
I			passwd	27	
ICMP	54		ping	42	
id	28		portmap	63	
ifconfig	39		PPP	54	
inetd	58		pppd		
info	112		/etc/wvdial.conf	121	
init	16		chap-secrets	121	
init (boot parameters)		21	chat	120	
init.d	17		minicom	119	
insmod	3		pap-secrets	121	
IP	54		peers	121	
ipchains	92		wvdial	119	
iptables	92		pump	40	
K			R		
kernel build			restore	109	
LILO	10		rmmod	3	
make clean	7		route	41	
make config	6		S		
make dep	7		Samba		
make menuconfig	6		smbclient	64	
make modules	8		smbmount	64	
make modules_install		8	scripting		
make oldconfig	6		\$(( ))	85	
make xconfig	6		expr	85	
L			for loop	84	
last	90		if then	83	
libwrap	91		until loop	84	
LILO	20p.		while loop	83	
logger	105		sendmail	72	
logrotate	105		sendmail.cf		
lpc	127		Cw	71	
lpd	126		Ds71		
lpq	126		Fw	71	
lpr	126		shutdown	18	
lprm	126		socket	59	
lsmod	3		ssh	96	
M			ssh-keygen		96
mailq	72		sshd	96	
man -k	111		subneting	52	
Manpages	110		subnets	52	
MANPATH		111	sysctl	39	
modinfo	3		syslog.conf		103
modprobe	3		syslogd	103	
modprobe		3	T		
modules.conf		3	tar	108	
modules.dep		3	TCP	54	
			TCP wrappers		61

## LinuxIT Technical Education Centre

### Index



---

tcp_wrapper	91		UDP	54
TCP/IP model (4 layer)		53	unset	79
TCP/IP Suite	53		useradd	27, 32
tcpdump	42		usermod	32
telnet	60		W	
test	82		whatis	111
The Linux Documentation Project		112	who	90
traceroute	44		X	
U			xinetd	59