L120: Linux System Administration II

course materials

originally released under the GFDL by LinuxIT

modified and released under the GFDL by University of Zagreb University Computing Centre SRCE
("the publisher")
Introduction

Acknowledgements
The original material was made available by LinuxIT's technical training centre 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 user friendly 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 original manual is available online at http://savannah.nongnu.org/projects/lpi-manuals/. The modified version of this manual is available online at http://www.srce.unizg.hr/linux-akademija/.

History
First release (version 0.0) October 2003. Reviewed by Adrian Thomasset.
Second release (revision1) January 2003. Reviewed by Andrew Meredith
Reviewed in January-June 2005 by Adrian Thomasset

January 2014. Title: L120: Linux System Administration II (version 1.0). Revised and modified at University of Zagreb University Computing Centre SRCE ("the publisher") by Darko Culej.

November 2016. Title: L120: Linux System Administration II (version 1.1). Revised and modified at University of Zagreb University Computing Centre SRCE ("the publisher") by Ivan Rako, Darko Culej and Branimir Radic.

No Guarantee
The manual comes with no guarantee at all.

Resources
www.lpi.org
www.linux-praxis.de
www.lpiforums.com
www.tldp.org
www.fsf.org
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
```
University Computing Centre SRCE

As the major national infrastructural ICT institution in the area of research and higher education in Croatia, the University Computing Centre SRCE is providing a modern, sustainable and reliable e-infrastructure for research and education community.

This includes computing and cloud services, high performance computing, advanced networking, communication systems and services, middleware, data and information systems and infrastructure. At the same time SRCE acts as the computing and information centre of the largest Croatian university – the University of Zagreb, and is responsible for the coordination of the development and usage of e-infrastructure at the University.

Furthermore, by applying cutting edge technologies SRCE continuously enriches academic and research e-infrastructure and its own service portfolio. This enables the active participation of Croatia and Croatian scientists in European and global research and higher education area and projects.

Since its founding in 1971 as a part of the University of Zagreb, at that time the only Croatian university, SRCE has provided an extended advisory and educational support to institutions and individuals from the academic and research community in the use of ICT for education and research purposes.

From its beginnings, and still today, SRCE has been recognized as an important factor of the development of modern e-infrastructure at the national level, participating in different projects and providing services like Croatian Internet eXchange (CIX).

SRCE has a 45 year old tradition of organizing professional courses from the field of ICT.

University Computing Centre SRCE
Josipa Marohnića 5
10000 Zagreb
Croatia

http://www.srce.unizg.hr
e-mail: ured@srce.hr
phone: +385 1 6165 555
# Table of Contents

**Introduction** ........................................................................................................................... iii

Acknowledgements ....................................................................................................................... iii

History ............................................................................................................................................ iii

No Guarantee ................................................................................................................................. iii

Resources ......................................................................................................................................... iii

Notations ........................................................................................................................................... iii

University Computing Centre SRCE ............................................................................................... iv

The Linux Kernel ............................................................................................................................ 1

1. Kernel Concepts ....................................................................................................................... 2
2. The Modular Kernel ................................................................................................................... 3
3. Routine Kernel Recompilation ................................................................................................. 5
4. Exercises and Summary ........................................................................................................... 10

Booting Linux ............................................................................................................................... 11

1. Understanding Runlevels ......................................................................................................... 12
2. Services and Runtime Control Scripts ................................................................................... 13
3. The joys of initab ...................................................................................................................... 15
4. GRUB .......................................................................................................................................... 16
5. From boot to bash ..................................................................................................................... 19
6. Exercises and Summary ........................................................................................................... 21

Managing Groups and Users ....................................................................................................... 22

1. Creating new users ................................................................................................................... 23
2. Working with groups ................................................................................................................ 24
3. Configuration files ................................................................................................................... 26
4. Command options .................................................................................................................... 28
5. Modifying accounts and default settings .............................................................................. 28
6. Exercises and Summary ........................................................................................................... 30

Bash Scripting ............................................................................................................................... 31

1. The bash environment .............................................................................................................. 32
2. Scripting Essentials .................................................................................................................. 34
3. Logical evaluations ................................................................................................................... 35
4. Flow Control and Loops .......................................................................................................... 36
5. Expecting user input ............................................................................................................... 38
6. Working with Numbers ........................................................................................................... 38
7. Exercises and Summary ........................................................................................................... 39

TCP/IP Networks .......................................................................................................................... 40

1. Binary Numbers and the Dotted Quad .................................................................................... 41
2. Broadcast Address, Network Address and Netmask ............................................................... 41
3. Network Classes ....................................................................................................................... 43
4. Classless Subnets ....................................................................................................................... 44
5. The TCP/IP Suite ..................................................................................................................... 45
6. TCP/IP Services and Ports ......................................................................................................... 47

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Acknowledgements</td>
</tr>
<tr>
<td>History</td>
</tr>
<tr>
<td>No Guarantee</td>
</tr>
<tr>
<td>Resources</td>
</tr>
<tr>
<td>Notations</td>
</tr>
<tr>
<td>University Computing Centre SRCE</td>
</tr>
<tr>
<td>The Linux Kernel</td>
</tr>
<tr>
<td>1. Kernel Concepts</td>
</tr>
<tr>
<td>2. The Modular Kernel</td>
</tr>
<tr>
<td>3. Routine Kernel Recompilation</td>
</tr>
<tr>
<td>4. Exercises and Summary</td>
</tr>
<tr>
<td>Booting Linux</td>
</tr>
<tr>
<td>1. Understanding Runlevels</td>
</tr>
<tr>
<td>2. Services and Runtime Control Scripts</td>
</tr>
<tr>
<td>3. The joys of initab</td>
</tr>
<tr>
<td>4. GRUB</td>
</tr>
<tr>
<td>5. From boot to bash</td>
</tr>
<tr>
<td>6. Exercises and Summary</td>
</tr>
<tr>
<td>Managing Groups and Users</td>
</tr>
<tr>
<td>1. Creating new users</td>
</tr>
<tr>
<td>2. Working with groups</td>
</tr>
<tr>
<td>3. Configuration files</td>
</tr>
<tr>
<td>4. Command options</td>
</tr>
<tr>
<td>5. Modifying accounts and default settings</td>
</tr>
<tr>
<td>6. Exercises and Summary</td>
</tr>
<tr>
<td>Bash Scripting</td>
</tr>
<tr>
<td>1. The bash environment</td>
</tr>
<tr>
<td>2. Scripting Essentials</td>
</tr>
<tr>
<td>3. Logical evaluations</td>
</tr>
<tr>
<td>4. Flow Control and Loops</td>
</tr>
<tr>
<td>5. Expecting user input</td>
</tr>
<tr>
<td>6. Working with Numbers</td>
</tr>
<tr>
<td>7. Exercises and Summary</td>
</tr>
<tr>
<td>TCP/IP Networks</td>
</tr>
<tr>
<td>1. Binary Numbers and the Dotted Quad</td>
</tr>
<tr>
<td>2. Broadcast Address, Network Address and Netmask</td>
</tr>
<tr>
<td>3. Network Classes</td>
</tr>
<tr>
<td>4. Classless Subnets</td>
</tr>
<tr>
<td>5. The TCP/IP Suite</td>
</tr>
<tr>
<td>6. TCP/IP Services and Ports</td>
</tr>
</tbody>
</table>
# Table of Contents

Network Configuration ............................................................................................................ 48
  1. The Network Interface ................................................................................................. 49
  2. Host Information ......................................................................................................... 50
  3. Stop and Start Networking ......................................................................................... 51
  4. Routing ......................................................................................................................... 52
  5. Common Network Tools ............................................................................................. 55
  6. Exercises and Summary .............................................................................................. 57

Network Services .................................................................................................................. 58
  1. The inetd daemon (old) .............................................................................................. 59
  2. The xinetd Daemon ...................................................................................................... 60
  3. Telnet and FTP ............................................................................................................ 61
  4. TCP wrappers ............................................................................................................. 63
  5. Setting up NFS ........................................................................................................... 64
  6. SMB and NMB ............................................................................................................ 65
  7. DNS services ............................................................................................................ 67
  8. Sendmail main Configuration ..................................................................................... 72
  9. The Apache server ...................................................................................................... 74
  10. Exercises and Summary ............................................................................................ 75

Basic Security ....................................................................................................................... 77
  1. Local Security ............................................................................................................. 78
  2. Network Security ........................................................................................................ 80
  3. The Secure Shell ......................................................................................................... 84
  4. Time Configuration ..................................................................................................... 86
  5. Exercises and Summary ............................................................................................. 88

Linux System Administration ............................................................................................... 90
  1. Logfiles and configuration files .................................................................................. 91
  2. Log Utilities ............................................................................................................... 94
  3. Automatic Tasks ......................................................................................................... 95
  4. Backups and Compressions ....................................................................................... 97
  5. Documentation ......................................................................................................... 100
  6. Exercises and Summary ............................................................................................ 103

Setting up PPP ...................................................................................................................... 105
  1. Detecting Modems ..................................................................................................... 106
  2. Dialup Configuration ................................................................................................. 107
  3. pppd and chat ........................................................................................................... 108
  4. PPPD peers ............................................................................................................... 109
  5. Wvdial ......................................................................................................................... 110
  6. Exercises and Summary ............................................................................................ 111

Printing .................................................................................................................................. 112
  1. Filters and gs ............................................................................................................. 113
  2. Printers and print queues ......................................................................................... 113
  3. Printing Tools .......................................................................................................... 114
  4. CUPS ......................................................................................................................... 115

Index .................................................................................................................................... 118

Vježbe ................................................................................................................................... 122

GNU Free Documentation License ..................................................................................... 137
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
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
  - `rmmod` 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 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, /etc/modprobe.conf or /etc/modprobe.d 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**

```bash
class 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.
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 \textit{zImage} and \textit{bzImage} respectively. These are compressed kernels using gzip and bzip2. See the next section \textit{Installing the New Kernel} to find out how to proceed with these files.

\textbf{make modules}

The modules are compiled with \textit{make modules}.

\textbf{make modules\_install}

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

The sequence of commands are depicted in Fig 3.

\textit{Kernel compilation commands}:

\begin{verbatim}
make dep
make clean
make bzImage
make modules
make modules\_install
\end{verbatim}

\subsection*{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. Install the new kernel on the system:

\begin{verbatim}
make install
\end{verbatim}

The make install command will create the following files in the /boot directory:

- \texttt{vmlinuz-3.9.3} – The actual kernel
- \texttt{System.map-3.9.3} – The symbols exported by the kernel
- \texttt{initrd.img-3.9.3} – \texttt{initrd image} is temporary root file system used during boot process
- \texttt{config-3.9.3} – The kernel configuration file

The command “make install” will also update the grub.cfg by default. So we don’t need to manually edit the grub.cfg file.
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 `\k` 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.
4. Exercises and Summary

<table>
<thead>
<tr>
<th>Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/modules.conf</td>
<td>used by modprobe before inserting a module</td>
</tr>
<tr>
<td>/lib/modules/&lt;kernel-version&gt;/</td>
<td>directory where the modules for given kernel version are stored</td>
</tr>
<tr>
<td>/lib/modules/&lt;kernel-version&gt;/modules.dep</td>
<td>list of module dependencies created by depmod</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>depmod</td>
<td>depmod(8) – kernel modules can provide services (called &quot;symbols&quot;) 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. Depmod 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 modules.dep in the same directory.</td>
</tr>
<tr>
<td>insmod</td>
<td>insmod(8) – 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 modprobe(8) instead, which is cleverer</td>
</tr>
<tr>
<td>make clean</td>
<td>delete all object files in the source tree</td>
</tr>
<tr>
<td>make config</td>
<td>configure the Linux kernel</td>
</tr>
<tr>
<td>make dep</td>
<td>creates a list of extra headers in files called .depend needed to satisfy module dependencies</td>
</tr>
<tr>
<td>make menuconfig</td>
<td>configure the Linux kernel using a menu</td>
</tr>
<tr>
<td>make modules</td>
<td>compile all the external/dynamic modules for this kernel</td>
</tr>
<tr>
<td>make modules_install</td>
<td>install the compiled modules in /lib/module/kernel-version</td>
</tr>
<tr>
<td>make oldconfig</td>
<td>create a default .config 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</td>
</tr>
<tr>
<td>make xconfig</td>
<td>configure a Linux kernel using a menu</td>
</tr>
<tr>
<td>lsmod</td>
<td>list all dynamically loaded modules</td>
</tr>
<tr>
<td>modinfo</td>
<td>print information about a kernel module such as the author (-a), the description (-d), the license (-l) or parameters (-p)</td>
</tr>
<tr>
<td>modprobe</td>
<td>modprobe(8) - will automatically load all base modules needed in a module stack, as described by the dependency file modules.dep. If the loading of one of these modules fails, the whole current stack of modules loaded in the current session will be unloaded automatically</td>
</tr>
<tr>
<td>rmmod</td>
<td>rmmod(8) – 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</td>
</tr>
</tbody>
</table>
**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
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.

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

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
```

<table>
<thead>
<tr>
<th>Expected arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>restart</td>
<td>do stop the start</td>
</tr>
<tr>
<td>stop</td>
<td>stop the daemon associated with the service</td>
</tr>
<tr>
<td>start</td>
<td>start the service</td>
</tr>
<tr>
<td>status</td>
<td>return the status of the services (running or stopped)</td>
</tr>
</tbody>
</table>

Typical services in `/etc/rc.d/init.d`:

```
ls /etc/rc.d/init.d/
```

<table>
<thead>
<tr>
<th>Service</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>anacron</td>
<td>cups</td>
</tr>
<tr>
<td>apmd</td>
<td>identd</td>
</tr>
<tr>
<td>arpwatch</td>
<td>keytable</td>
</tr>
<tr>
<td>atd</td>
<td>killall</td>
</tr>
<tr>
<td>autosfs</td>
<td>killall</td>
</tr>
<tr>
<td>cron</td>
<td>kprop</td>
</tr>
<tr>
<td>dhcpd</td>
<td>ldap</td>
</tr>
<tr>
<td>functions</td>
<td>ldaf</td>
</tr>
<tr>
<td>ipchains</td>
<td>linuxconf</td>
</tr>
<tr>
<td>isdn</td>
<td>network</td>
</tr>
<tr>
<td>krb524</td>
<td>portmap</td>
</tr>
<tr>
<td>krb5kdc</td>
<td>rlogin</td>
</tr>
<tr>
<td>kprop</td>
<td>rsh</td>
</tr>
<tr>
<td>kudzu</td>
<td>rsync</td>
</tr>
<tr>
<td>ldap</td>
<td>rsyslog</td>
</tr>
<tr>
<td>.listener</td>
<td>rwho</td>
</tr>
<tr>
<td>kadmin</td>
<td>samba</td>
</tr>
<tr>
<td>mcserv</td>
<td>sendmail</td>
</tr>
<tr>
<td>named</td>
<td>single</td>
</tr>
<tr>
<td>ntcs</td>
<td>smb</td>
</tr>
<tr>
<td>nscd</td>
<td>squid</td>
</tr>
<tr>
<td>openldap</td>
<td>syslog</td>
</tr>
<tr>
<td>random</td>
<td>squid</td>
</tr>
<tr>
<td>rawdevices</td>
<td>syslog</td>
</tr>
<tr>
<td>rpcbind</td>
<td>syslog</td>
</tr>
<tr>
<td>rawdevices</td>
<td>syslog</td>
</tr>
<tr>
<td>xinetd</td>
<td>syslog</td>
</tr>
<tr>
<td>xfs</td>
<td>syslog</td>
</tr>
<tr>
<td>xinetd</td>
<td>syslog</td>
</tr>
</tbody>
</table>

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

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:

### The /etc/inittab file

<table>
<thead>
<tr>
<th>id</th>
<th>runlevel</th>
<th>action</th>
<th>command</th>
</tr>
</thead>
<tbody>
<tr>
<td>id:3: initdefault:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># System initialization.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>si: sysinit: /etc/rc.d/rc.sysinit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:0: wait:/etc/rc.d/rc 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:1: wait:/etc/rc.d/rc 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:2: wait:/etc/rc.d/rc 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:3: wait:/etc/rc.d/rc 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:4: wait:/etc/rc.d/rc 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:5: wait:/etc/rc.d/rc 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:6: wait:/etc/rc.d/rc 6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-----------------------

snip----------------------------------

# Trap CTRL-ALT-DELETE
ca: ccontraldde: /sbin/shutdown -t3 -r now

-----------------------

snip----------------------------------

# Run getty's 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 ...

**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. 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
```

<table>
<thead>
<tr>
<th>Common dynamic modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ext3</code></td>
<td>Third extended filesystem type</td>
</tr>
<tr>
<td><code>lvm</code></td>
<td>Logical volume support</td>
</tr>
<tr>
<td><code>raidx</code></td>
<td>Software raid level x support</td>
</tr>
<tr>
<td><code>scsi</code></td>
<td>SCSI support</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>image that will boot by default (the first entry is 0)</td>
</tr>
<tr>
<td>timeout</td>
<td>prompt timeout in seconds</td>
</tr>
</tbody>
</table>

2. Image

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>name of the image</td>
</tr>
<tr>
<td>root</td>
<td>where the 2nd stage bootloader and kernel are e.g (hd0,0) is /dev/hda</td>
</tr>
<tr>
<td>kernel</td>
<td>path for the kernel starting from the previous root e.g /vmlinuz</td>
</tr>
<tr>
<td>ro</td>
<td>read-only</td>
</tr>
<tr>
<td>root</td>
<td>the filesystem root</td>
</tr>
<tr>
<td>initrd</td>
<td>path to the initial root disk</td>
</tr>
</tbody>
</table>
Bootloader Options

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

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 passe 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.

<table>
<thead>
<tr>
<th>Common kernel parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>acpi= enable/disable ACPI</td>
</tr>
<tr>
<td>init= tell the kernel which program to start from the root device</td>
</tr>
<tr>
<td>mem= specify amount of RAM to use</td>
</tr>
<tr>
<td>root= specify the root device</td>
</tr>
</tbody>
</table>
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 it's 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`.

![Diagram of boot process](image)
6. Exercises and Summary

**Files**

<table>
<thead>
<tr>
<th>Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/init.d</td>
<td>directory containing all the scripts used to stop and start services at boot time</td>
</tr>
<tr>
<td>/etc/inittab</td>
<td><code>inittab(5)</code> - 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</td>
</tr>
</tbody>
</table>

**Commands**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td><code>init(8)</code> – is the parent of all processes. Its primary role is to create processes from a script stored in the file <code>/etc/inittab</code></td>
</tr>
<tr>
<td>shutdown</td>
<td><code>shutdown(8)</code> – 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</td>
</tr>
</tbody>
</table>

**References**

Take a look at the `boot(7)` manpage, it covers most of what we did in this module.
Managing Groups and Users

Prerequisites

- None

Goals

- Manage user accounts
- Manage group accounts
- Modify accounts settings
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`.

```
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$b7ryoKGTd2h:505:505:Dr G Michael:/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:

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

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

```
/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/password` (600 or 400).
The `/etc/group` and `/gshadow` files:

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Group name</td>
</tr>
<tr>
<td>2</td>
<td>The group password (or x if gshadow file exists)</td>
</tr>
<tr>
<td>3</td>
<td>The GID</td>
</tr>
<tr>
<td>4</td>
<td>A comma separated list of members</td>
</tr>
</tbody>
</table>

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)

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

groupadd (options)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g</td>
<td>assign a GID</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d</td>
<td>the users directory</td>
</tr>
<tr>
<td>-g</td>
<td>the users initial GID</td>
</tr>
<tr>
<td>-l</td>
<td>the user’s login name</td>
</tr>
<tr>
<td>-u</td>
<td>the user’s UID</td>
</tr>
<tr>
<td>-s</td>
<td>the default shell.</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g</td>
<td>the GID</td>
</tr>
<tr>
<td>-n</td>
<td>the group name.</td>
</tr>
</tbody>
</table>
## 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:

<table>
<thead>
<tr>
<th>Lock</th>
<th>Unlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd -l</td>
<td>passwd -u</td>
</tr>
<tr>
<td>usermod -L</td>
<td>usermod -U</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/group</td>
<td>contains the names of all the groups on the system</td>
</tr>
<tr>
<td>/etc/gshadow</td>
<td>contains (optionally) passwords associated to a group</td>
</tr>
<tr>
<td>/etc/login.defs</td>
<td>contains predefined values needed when adding a new user such as the minimum and maximum UID and GID, the minimum password length, etc</td>
</tr>
<tr>
<td>/etc/passwd</td>
<td>passwd(5) – 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</td>
</tr>
<tr>
<td>/etc/shadow</td>
<td>shadow(5) – contains the encrypted password information for user’s accounts and optional the password aging information</td>
</tr>
<tr>
<td>/etc/skel/</td>
<td>directory containing files and directories to be copied into the home directory of every newly created user</td>
</tr>
</tbody>
</table>

Commands

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chage</td>
<td>chage(1) – changes a user’s password expiry information</td>
</tr>
<tr>
<td>gpasswd</td>
<td>gpasswd(1) – administer the /etc/group file</td>
</tr>
<tr>
<td>groupadd</td>
<td>add a new group to the system</td>
</tr>
<tr>
<td>groupmod</td>
<td>modify an existing group</td>
</tr>
<tr>
<td>groups</td>
<td>print out all the groups a user belongs to</td>
</tr>
<tr>
<td>id</td>
<td>print out the UID as well as the GIDs of all the groups a user belongs to</td>
</tr>
<tr>
<td>passwd</td>
<td>change the password for an account</td>
</tr>
<tr>
<td>useradd</td>
<td>add a new user to the system</td>
</tr>
<tr>
<td>usermod</td>
<td>modify an existing user account</td>
</tr>
</tbody>
</table>
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
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 on. 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
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 inputrc:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set editing-mode vi</td>
<td>change the initial editor style (default is emacs)</td>
</tr>
<tr>
<td>Control-o: &quot;&gt; output&quot;</td>
<td>bind the sequence Ctrl+o will cause the string &quot;&gt; output &quot; to be printed</td>
</tr>
<tr>
<td>TAB: complete</td>
<td>automatically complete commands and file names (is set by default)</td>
</tr>
<tr>
<td>set bell-style none</td>
<td>input errors are not audible (other option is audible)</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bash script-name</code></td>
<td>Start a new interactive <code>bash</code> which will run the script then exit</td>
</tr>
<tr>
<td><code>source script-name</code></td>
<td>Force your current shell to run the script</td>
</tr>
<tr>
<td><code>. script-name</code></td>
<td>Same as <code>source</code></td>
</tr>
<tr>
<td><code>exec ./script-name</code></td>
<td>Same as <code>./script-name</code> except that the current shell will exit once the script has run</td>
</tr>
</tbody>
</table>

**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:**

```bash
#!/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
```
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:**

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

or

```bash
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:

- `$*` List of all variables entered at the command line
- `$#` Number of arguments entered at the command line
- `$0` The name of the script
- `$!` PID of the most recent background command
- `$$` PID of the current shell
- `$?` 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 \( \text{\$}(n+1) \rightarrow \text{\$}n \)

### 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:

<table>
<thead>
<tr>
<th>Using <code>test</code></th>
<th>Using <code>[]</code></th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>test -f /bin/bash</code></td>
<td><code>-f /bin/bash</code></td>
<td>test if <code>/bin/bash</code> is a file</td>
</tr>
<tr>
<td><code>test -x /etc/passwd</code></td>
<td><code>-x /etc/passwd</code></td>
<td>test if <code>/etc/passwd</code> is executable</td>
</tr>
</tbody>
</table>
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
```

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
```

4. Flow Control and Loops

**if then**

Syntax: if CONDITION ; then
  command1
  command2
fi

```bash
#!/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: Aligne 10 hashes (#) then exit

```bash
#!/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**

<table>
<thead>
<tr>
<th>Test operators</th>
<th>Numbers</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lt</td>
<td>&lt;</td>
<td>(sort strings lexicographically)</td>
</tr>
<tr>
<td>-gt</td>
<td>&gt;</td>
<td>(sort strings lexicographically)</td>
</tr>
<tr>
<td>-le</td>
<td>N/a</td>
<td></td>
</tr>
<tr>
<td>-ge</td>
<td>N/a</td>
<td></td>
</tr>
<tr>
<td>-eq</td>
<td>==</td>
<td></td>
</tr>
<tr>
<td>-ne</td>
<td>!=</td>
<td></td>
</tr>
</tbody>
</table>
# 7. Exercises and Summary

## Files

<table>
<thead>
<tr>
<th>Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/etc/bashrc</code></td>
<td>a system wide startup file for interactive bash sessions (used for setting up the PS1 prompt)</td>
</tr>
<tr>
<td><code>/etc/inputrc</code></td>
<td>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)</td>
</tr>
<tr>
<td><code>/etc/profile</code></td>
<td>system wide configuration file for bash. It contains exported variables such as the PATH and is always read at login</td>
</tr>
<tr>
<td><code>~/.bash_profile</code></td>
<td>the user's customised configuration file for bash. It contains exported variables an is always read at login</td>
</tr>
<tr>
<td><code>~/.bashrc</code></td>
<td>the user's customised startup file for bash. It is read every time a new interactive shell is started unless the <code>–norc</code> option is given</td>
</tr>
<tr>
<td><code>~/.inputrc</code></td>
<td>the user's customised startup file for the readline library</td>
</tr>
</tbody>
</table>

## Scripting items

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$(())</code></td>
<td>operator used to substitute the result of a numerical evaluation in a command line</td>
</tr>
<tr>
<td><code>expr</code></td>
<td>perform a numerical evaluation</td>
</tr>
<tr>
<td><code>for</code></td>
<td>see p.72</td>
</tr>
<tr>
<td><code>if</code></td>
<td>see p.71</td>
</tr>
<tr>
<td><code>until</code></td>
<td>see p.72</td>
</tr>
<tr>
<td><code>while</code></td>
<td>see p.71</td>
</tr>
</tbody>
</table>

## Commands

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

- Network configuration (p. 31)

Goals

- Understand formal TCP/IP network concepts
- Manage subnets
- Understand the four layer TCP/IP model
- Introduce service port numbers
1. Binary Numbers and the Dotted Quad

**Binary numbers**

\[
\begin{align*}
10 & = 2^1 \\
100 & = 2^2 \\
101 & = 2^2 + 1 \\
11 & = 100 + 010 + 001
\end{align*}
\]

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

\[
\begin{align*}
10000000 & = 2^7 = 128 \\
01000000 & = 2^6 = 64 \\
00100000 & = 2^5 = 32 \\
00010000 & = 2^4 = 16 \\
00001000 & = 2^3 = 8 \\
00000100 & = 2^2 = 4 \\
00000010 & = 2^1 = 2 \\
00000001 & = 2^0 = 1
\end{align*}
\]

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.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.1</td>
<td>11000000.10101000.00000001.00000001</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Netmask</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>255.255.0.0</td>
<td>16-bit</td>
</tr>
<tr>
<td>255.255.128.0</td>
<td>17-bit</td>
</tr>
</tbody>
</table>

The netmask is usually given in decimal.

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00100000</td>
<td>10000000</td>
<td>00000001</td>
<td>00000001</td>
</tr>
<tr>
<td>00100000</td>
<td>10000000</td>
<td>00000000</td>
<td>00000011</td>
</tr>
</tbody>
</table>

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:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00100000</td>
<td>.</td>
<td>10000000</td>
<td>.</td>
<td>00000001</td>
</tr>
<tr>
<td>00100000</td>
<td>.</td>
<td>10000000</td>
<td>.</td>
<td>00000000</td>
</tr>
</tbody>
</table>

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.

\[
\text{Network Address} = \text{IP} \land \text{Netmask}
\]

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

\[
\text{Broadcast Address} = \text{Network} \lor \neg[\text{Netmask}]
\]

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:

<table>
<thead>
<tr>
<th>Network address</th>
<th>=</th>
<th>IP</th>
<th>AND</th>
<th>MASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>11000000.10101000.00000011.00000010</td>
<td>.</td>
<td>10101000.00000011.00000011</td>
<td>AND</td>
<td>11000000.10101000.00000011.00000000</td>
</tr>
<tr>
<td>11111111.11111111.11111111.00000000</td>
<td>(192.168.3.5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11111111.11111111.11111111.00000000</td>
<td>(255.255.255.000)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11000000.10101000.00000011.00000000</td>
<td>(192.168.3.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Broadcast Address = IP OR NOT-MASK

<table>
<thead>
<tr>
<th>OR</th>
<th>11000000. 10101000.00000011.00000101</th>
<th>(192.168.3.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11000000.10101000.00000011.11111111</td>
<td>(000.000.000.255)</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Table1: Reserved addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>255</td>
</tr>
</tbody>
</table>

- **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 255.255.255.255 and 0.0.0.0 are invalid host numbers there are $2^{24} - 2 = 16777214$ 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 $2^{16} - 2 = 65534$ 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”.

43
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: 11111111.11111111.11111111.11111111.10000000 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 192.168.1.0xxxxxx range result in a 192.168.1.0 network. We say the network number is 0
2. Host addresses in the 192.168.1.1xxxxxx range result in a 192.168.1.128 network. We say the network number is 128

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

<table>
<thead>
<tr>
<th>Network address</th>
<th>Substitute with 0’s</th>
<th>Substitute with 1’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Network: 0</td>
<td>Broadcast: 127</td>
</tr>
<tr>
<td>128</td>
<td>Network: 128</td>
<td>Broadcast: 255</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>application level (FTP, SMTP, SNMP)</td>
</tr>
<tr>
<td>Transport</td>
<td>handles hosts (TCP, UDP)</td>
</tr>
<tr>
<td>Internet</td>
<td>routing (IP, ICMP, IGMP, ARP)</td>
</tr>
<tr>
<td>Network Access</td>
<td>network cards, e.g Ethernet, token ring …</td>
</tr>
</tbody>
</table>
Protocol Overview

**IP**
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.

**TCP**
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…

**UDP**
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.

**ICMP**
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.

**PPP**
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 65536 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:

<table>
<thead>
<tr>
<th>Port</th>
<th>State</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>21/tcp</td>
<td>open</td>
<td>ftp</td>
</tr>
<tr>
<td>22/tcp</td>
<td>open</td>
<td>ssh</td>
</tr>
<tr>
<td>23/tcp</td>
<td>open</td>
<td>telnet</td>
</tr>
<tr>
<td>25/tcp</td>
<td>open</td>
<td>smtp</td>
</tr>
<tr>
<td>70/tcp</td>
<td>open</td>
<td>gopher</td>
</tr>
<tr>
<td>79/tcp</td>
<td>open</td>
<td>finger</td>
</tr>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
</tbody>
</table>

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

The /etc/services main ports:

- **ftp-data**: 20/tcp
- **ftp**: 21/tcp
- **ssh**: 22/udp
- **ssh**: 22/tcp
- **telnet**: 23/tcp
- **smtp**: 25/tcp
- **domain**: 53/tcp
- **domain**: 53/udp
- **http**: 80/tcp
- **pop-3**: 110/tcp
- **sunrpc**: 111/tcp
- **sftp**: 115/tcp
- **uucp-path**: 117/tcp
- **nntp**: 119/tcp
- **ntp**: 123/tcp
- **netbios-ns**: 137/tcp
- **netbios-ns**: 137/udp
- **netbios-dgm**: 138/tcp
- **netbios-dgm**: 138/udp
- **netbios-ssn**: 139/tcp
- **imap**: 143/tcp
- **NeWS**: 144/tcp
- **snmp**: 161/udp
- **snmp-trap**: 162/udp

# Additional notes:
- The **http** port is used by many services.
- The **smtp** port is used for secure email.
- The **domain** port is used for DNS services.
- The **ftp** port is used for file transfer.
- The **pop-3** port is used for email retrieval.
- The **telnet** port is used for terminal access.
- The **smtp** port is used for mail transfer.
- The **domain** port is used for DNS resolution.
- The **http** port is used for web services.
- The **smtp** port is used for mail services.
- The **domain** port is used for DNS resolution.
Network Configuration

Prerequisites

- Hardware configuration (see LPI 101)

Goals

- Configure a Linux system for networking
- Understand routing
- Use network troubleshooting tools
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/lsmod` 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/lsmod
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/sysconfig/network` 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 loosing 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`
- `dhcpcd`

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

Static route configuration is stored in a /etc/sysconfig/network-scripts/route-interface file. For example, static routes for the eth0 interface would be stored in the /etc/sysconfig/network-scripts/route-eth0 file. The route-interface file has two formats: IP command arguments and network/netmask directives.

Example /etc/sysconfig/network-scripts/route-eth0 IP command file:

```
default 192.168.0.1 dev eth0
10.10.10.0/24 via 192.168.0.1 dev eth0
172.16.1.0/24 via 192.168.0.1 dev eth0
```

Example /etc/sysconfig/network-scripts/route-eth0 network/netmask IP command file:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
ADDRESS1=172.16.1.0
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.1
```

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. 40).

<table>
<thead>
<tr>
<th>/etc/networks format</th>
</tr>
</thead>
<tbody>
<tr>
<td>network-name network-number aliases</td>
</tr>
</tbody>
</table>

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:

```
10.0.0.0/255.0.0.0
10.0.0.1
10.0.0.111
10.8.10.34

192.168.1.0/255.255.255.0
192.168.1.2
192.168.1.108

network  gw
10.0.0.0  0.0.0.0
0.0.0.0  10.0.0.1

route add 192.168.1.0/255.255.255.0 gw 10.0.0.111

network  gw
192.168.1.0  0.0.0.0
0.0.0.0  192.168.1.108

route add 10.0.0.0/255.0.0.0 gw 10.8.10.34
```
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.

<table>
<thead>
<tr>
<th>Options for ping</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-b</td>
<td>ping a broadcast address</td>
</tr>
<tr>
<td>-c N</td>
<td>send N packets</td>
</tr>
<tr>
<td>-q</td>
<td>quiet mode: display only start and end messages</td>
</tr>
</tbody>
</table>

**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
  - `tcpdump`

- Specify a network interface to capture packets from
  - `tcpdump -i wlan0`

- Give an expression to match
  - `tcpdump host 192.168.10.1 and port 80`

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.

<table>
<thead>
<tr>
<th>Options for netstat</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r</td>
<td>same as /sbin/route</td>
</tr>
<tr>
<td>-l</td>
<td>display list of interfaces</td>
</tr>
<tr>
<td>-n</td>
<td>don’t resolve IP addresses</td>
</tr>
<tr>
<td>-p</td>
<td>returns the PID and names of programs (only for root)</td>
</tr>
<tr>
<td>-v</td>
<td>verbose</td>
</tr>
<tr>
<td>-c</td>
<td>continuous update</td>
</tr>
</tbody>
</table>
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 traceroute:**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-f ttl</td>
<td>change the initial time to live value to ttl instead of 1</td>
</tr>
<tr>
<td>-n</td>
<td>do not resolve IP numbers</td>
</tr>
<tr>
<td>-v</td>
<td>verbose</td>
</tr>
<tr>
<td>-w sec</td>
<td>set the timeout on returned packets to sec</td>
</tr>
</tbody>
</table>
6. Exercises and Summary

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/resolv.conf</td>
<td>file containing a list of DNS servers used to resolve computer host names</td>
</tr>
<tr>
<td>/etc/sysctl.conf</td>
<td>configuration file for the sysctl tool used to customise kernel settings in /proc/sys/</td>
</tr>
<tr>
<td>/proc/sys/net/ipv4/ip_forward</td>
<td>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</td>
</tr>
</tbody>
</table>

Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arp</td>
<td>print the kernel ARP cache</td>
</tr>
<tr>
<td>dhcpd</td>
<td>a DHCP client daemon</td>
</tr>
<tr>
<td>dhcpclient</td>
<td>a DHCP client daemon</td>
</tr>
<tr>
<td>ifconfig</td>
<td>ifconfig(8) – is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary</td>
</tr>
<tr>
<td>netstat</td>
<td>netstat(8) – print information about network connections, routing tables, interface statistics, etc</td>
</tr>
<tr>
<td>ping</td>
<td>ping(8) – 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</td>
</tr>
<tr>
<td>pump</td>
<td>pump(8) – 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</td>
</tr>
<tr>
<td>route</td>
<td>route(8) – 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</td>
</tr>
<tr>
<td>sysctl</td>
<td>sysctl(8) – is used to modify kernel parameters at runtime. The parameters available are those listed under /proc/sys/</td>
</tr>
<tr>
<td>traceroute</td>
<td>traceroute(8) - 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</td>
</tr>
</tbody>
</table>


Network Services

Prerequisite

- Booting Linux (p.11)
- Network Configuration (p. 31)

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
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:

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

**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:

<table>
<thead>
<tr>
<th>service-name</th>
<th>port/protocol</th>
<th>[aliases]</th>
</tr>
</thead>
</table>
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.

<table>
<thead>
<tr>
<th>Structure of service file in xinetd.d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service-name {</td>
</tr>
<tr>
<td>disable = yes/no</td>
</tr>
<tr>
<td>socket_type = stream for TCP and dgram for UDP</td>
</tr>
<tr>
<td>protocol = valid protocol from /etc/protocols</td>
</tr>
<tr>
<td>wait = &lt;yes or no&gt;</td>
</tr>
<tr>
<td>user= the user the application runs as</td>
</tr>
<tr>
<td>group= the group the application runs as</td>
</tr>
<tr>
<td>server= the name of the program to be run for this service</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
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:

```
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)
```
4. 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:

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

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

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

```
mount -t nfs nfs-server:/shared/dir /mnt/nfs
```

- **Server settings**

A NFS server needs `portmap` (rpcbind) 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/exportfs` 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
```
6. 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 starts the NMB services. This is the NetBIOS Message Block which enables name resolution in the Windows realm.

Figure 1: 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

**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
7. 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.

<table>
<thead>
<tr>
<th>Common Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd</td>
<td>user names</td>
</tr>
<tr>
<td>group</td>
<td>group names</td>
</tr>
<tr>
<td>hosts</td>
<td>host names</td>
</tr>
<tr>
<td>networks</td>
<td>network names</td>
</tr>
</tbody>
</table>

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 database listed below.

<table>
<thead>
<tr>
<th>Common databases</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>files</td>
<td>flat files, generally in /etc</td>
</tr>
<tr>
<td>nis</td>
<td>a map from a NIS server</td>
</tr>
<tr>
<td>dns</td>
<td>a DNS server</td>
</tr>
</tbody>
</table>

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

- **com**: Commercial organisations
- **edu**: US educational institutions
- **gov**: US government institutions
- **mil**: US military institutions
- **net**: Gateways and network providers
- **org**: Non commercial sites
- **uk**: 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 file:

<table>
<thead>
<tr>
<th>directory</th>
<th>/var/named</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>named.ca</td>
</tr>
<tr>
<td>primary myco.org</td>
<td>named.myco</td>
</tr>
<tr>
<td>primary 0.0.127.in-addr.arpa</td>
<td>named.local</td>
</tr>
<tr>
<td>primary 1.168.192.in-addr.arpa</td>
<td>named.rev</td>
</tr>
</tbody>
</table>

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:

- **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:

```plaintext
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:**

```plaintext
@       IN      SOA     localhost. root.localhost.  (
    2001022700 ; Serial
    28800 ; Refresh
    14400 ; Retry
    3600000 ; Expire
    86400 ) ; Minimum

IN      NS      localhost.
1       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).
Table 2: Common Record Types

<table>
<thead>
<tr>
<th>NS</th>
<th>Specify the zones primary name server</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTR</td>
<td>Reverse mapping of IP numbers to hostnames</td>
</tr>
<tr>
<td>MX</td>
<td>Mail exchange record</td>
</tr>
<tr>
<td>A</td>
<td>Associate an IP address with a hostname</td>
</tr>
<tr>
<td>CNAME</td>
<td>Associate an alias with the host’s main name</td>
</tr>
</tbody>
</table>

Table 2: Zone parameters

<table>
<thead>
<tr>
<th>@</th>
<th>IN</th>
<th>SOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td></td>
<td>Start Of Authority. Identifies the zone followed by options enclosed in brackets.</td>
</tr>
<tr>
<td>Refresh</td>
<td></td>
<td>Time in seconds before the secondary server should query the SOA record of the primary domain. This should be at least a day.</td>
</tr>
<tr>
<td>Retry</td>
<td></td>
<td>Time interval in seconds before attempting a new zone transfer if the previous download failed</td>
</tr>
<tr>
<td>Expire</td>
<td></td>
<td>Time after which the secondary server discards all zone data if it contact the primary server. Should be a week at least</td>
</tr>
<tr>
<td>Minimum</td>
<td></td>
<td>This is the ttl for the cached data. The default is one day (86400 seconds) but should be longer on stable LANs</td>
</tr>
</tbody>
</table>

Testing

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

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

```
$ dig @127.0.0.1 srce.hr MX
```
```
$ host -t mx srce.hr 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.
8. 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.

<table>
<thead>
<tr>
<th>WARNING</th>
</tr>
</thead>
</table>
| 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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Cw</code></td>
<td>the mailer hostname. Can also contain a list of hostnames or domain names the mailer will assume but it is better to use <code>Fw</code> for this</td>
</tr>
<tr>
<td><code>Fw</code></td>
<td>path to the file containing domain names sendmail will receive mail for</td>
</tr>
<tr>
<td><code>Ds</code></td>
<td>address for ‘smart host’, this is a mailer that will relay our outgoing mail</td>
</tr>
</tbody>
</table>

**Files in `/etc/mail`**

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>access</code></td>
<td>list of hosts authorised to use the server to relay mail</td>
</tr>
<tr>
<td><code>local-host-names</code></td>
<td>list of domain names</td>
</tr>
</tbody>
</table>

**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
9. 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"

```xml
<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.
## 10. Exercises and Summary

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>apachectl</code></td>
<td><code>apachectl(8)</code> – apache HTTP server control interface. On the command line the script will simply pass all the given arguments to the <code>httpd</code> server</td>
</tr>
<tr>
<td><code>dig</code></td>
<td><code>dig(1)</code> – (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</td>
</tr>
<tr>
<td><code>host</code></td>
<td><code>host(1)</code> – a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa</td>
</tr>
<tr>
<td><code>exportfs</code></td>
<td><code>exportfs(8)</code> – 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 <code>mountd</code> 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</td>
</tr>
<tr>
<td><code>inetd</code></td>
<td>see <code>xinetd</code></td>
</tr>
<tr>
<td><code>mailq</code></td>
<td><code>mailq(1)</code> – prints a summary of the mail messages queued for future delivery</td>
</tr>
<tr>
<td><code>portmap</code></td>
<td><code>portmap(8)</code> – 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</td>
</tr>
<tr>
<td><code>smbclient</code></td>
<td><code>smbclient(1)</code> – is a client that can ‘talk’ to an SMB/CIFS server. It offers an interface similar to that of the <code>ftp</code> program (see <code>ftp(1)</code>). 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</td>
</tr>
<tr>
<td><code>smbmount</code></td>
<td><code>smbmount(8)</code> – mounts a Linux SMB filesystem. It is usually invoked as <code>mount.smbfs</code> by the <code>mount(8)</code> command when using the “-t smbfs” option. This command only works in Linux, and the kernel must support the smbfs filesystem</td>
</tr>
<tr>
<td><code>sendmail</code></td>
<td><code>sendmail(8)</code> – 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</td>
</tr>
<tr>
<td><code>xinetd</code></td>
<td><code>xinetd(8)</code> – 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 being 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</td>
</tr>
</tbody>
</table>
Basic Security

Prerequisites

- None

Goals

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

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:

```
chattr +i /bin/login
chattr +i /bin/ps
chattr +a /var/log/messages
```

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

```
<table>
<thead>
<tr>
<th>Directory</th>
<th>Mount Type</th>
<th>Superblocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tmp</td>
<td>ext2</td>
<td>1 2</td>
</tr>
<tr>
<td>/home</td>
<td>ext2</td>
<td>1 2</td>
</tr>
</tbody>
</table>
```

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/wtmp 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 : dissallow logins for groups and users from specific locations.

/etc/security/limits.conf

The format of this file is

```
<domain> <type> <item> <value>
```

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

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.
Iptables have 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

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 and may also contain user-defined chains.

![Diagram of Netfilter kernel framework for iptables](image)

**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

```bash
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:

\[ \texttt{iptables -P INPUT DENY} \]

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

\[ \texttt{iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT} \]

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:
  ```
  $\texttt{IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE}
  ```

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

```bash
#!/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
```

83
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.
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:

```
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 MMDDhhmCCYY[.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` on 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**

<table>
<thead>
<tr>
<th>server ntp2.somewhere.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>driftfile /var/lib/ntp/drift</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/fstab</td>
<td>noexec – mount option which prevents any executables to execute from the device</td>
</tr>
<tr>
<td></td>
<td>nosuid – mount option which prevents the SUID and SGID bits to take effect (see LP1101)</td>
</tr>
<tr>
<td>/etc/localtime</td>
<td>contains the time zone policy used to determine the system time (with <code>date</code>)</td>
</tr>
<tr>
<td>/etc/ntp.conf</td>
<td>configuration file for the NTP daemon <code>ntpd</code></td>
</tr>
<tr>
<td>/etc/ntp.drift or /etc/ntp/drift</td>
<td>file used by <code>ntpd</code> to keep track of the hardware clock drift</td>
</tr>
<tr>
<td>/etc/security/access.conf</td>
<td>file used to grant or deny access based on the user's name and the origin (locality or remote host). One can also specify a NIS group using <code>@group</code> notation</td>
</tr>
<tr>
<td>/etc/security/limits.conf</td>
<td>file used to impose resource limits on login (see the file itself for details)</td>
</tr>
<tr>
<td>/etc/ssh</td>
<td>directory containing configuration files for both the <code>ssh</code> client and the <code>sshd</code> server</td>
</tr>
<tr>
<td>/usr/share/zoneinfo/</td>
<td>collection of time zone files. Depending on the user's location one of these files is copied to <code>/etc/localtime</code></td>
</tr>
<tr>
<td>/var/log/messages</td>
<td>the main system log file</td>
</tr>
<tr>
<td>/var/log/secure</td>
<td>log file containing information about failed logins or user accounts</td>
</tr>
<tr>
<td>/var/log/wtmp</td>
<td>the <code>wtmp</code> file records all logins and logouts.</td>
</tr>
<tr>
<td>/var/run/utmp</td>
<td><code>utmp(5)</code> – the utmp 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 utmp logging</td>
</tr>
<tr>
<td>$HOME/.ssh</td>
<td>directory containing <code>knownhosts</code>, <code>authorized_keys2</code>, <code>id_dsa</code> and <code>id_dsa.pub</code></td>
</tr>
<tr>
<td>authorized_keys2</td>
<td>contains a list a public id keys from remote users that are authorised to use this account (via ssh)</td>
</tr>
<tr>
<td>id_dsa</td>
<td>the user's private key used during the user authentication process of an ssh session</td>
</tr>
<tr>
<td>id_dsa.pub</td>
<td>the user's public key used during the user authentication process of an ssh session – this key must be present in the <code>authorized_keys2</code> file of the account one is attempting to ssh to</td>
</tr>
<tr>
<td>known_hosts</td>
<td>list of server public keys used for host authentication</td>
</tr>
<tr>
<td>ssh_config</td>
<td>configuration file for <code>ssh</code></td>
</tr>
<tr>
<td>sshd_config</td>
<td>configuration file for <code>sshd</code></td>
</tr>
</tbody>
</table>
## Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chattr</td>
<td>change file attributes on an ext2/3 filesystem (see chattr(1) for details)</td>
</tr>
<tr>
<td>date</td>
<td>print or set the system time</td>
</tr>
<tr>
<td>hwclock</td>
<td>query or set the hardware clock</td>
</tr>
<tr>
<td>ipchains</td>
<td></td>
</tr>
<tr>
<td>iptables</td>
<td>iptables(8) – administration tool for IPv4 packet filtering and NAT</td>
</tr>
<tr>
<td>last</td>
<td>last(1) – 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</td>
</tr>
<tr>
<td>ntpd</td>
<td>the NTP daemon</td>
</tr>
<tr>
<td>ntpdate</td>
<td>ntpdate(1) – 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</td>
</tr>
<tr>
<td>ssh</td>
<td>ssh(1) – 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</td>
</tr>
<tr>
<td>ssh-keygen</td>
<td>ssh-keygen(1) – 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</td>
</tr>
<tr>
<td>sshd</td>
<td>sshd(8) – 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</td>
</tr>
<tr>
<td>who</td>
<td>who(1) – show who is logged on</td>
</tr>
</tbody>
</table>
Linux System Administration

Prerequisites

- None

Goals

- Customise system logging system
- Configure cron and at
- Understand backup tools and strategies
- Finding documentation
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 `cron` executes
- `mail`: messages relating to `mail`
- `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/rsyslog.conf file

When `rsyslogd` is started it reads the `/etc/rsyslog.conf` configuration file by default. One can also start `rsyslogd` 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:

- `auth` and `authpriv`: user general and private authentication
- `cron`: cron daemon messages
- `kern`: kernel messages
- `mail`
- `news`
- `user`
- `uucp`: user processes
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/rsyslog.conf

# rsyslog configuration file

# For more information see /usr/share/doc/rsyslog-*/*rsyslog_conf.html
# If you experience problems, see http://www.rsyslog.com/doc/troubleshoot.html

#### MODULES ####

# The imjournal module bellow is now used as a message source instead of imuxsock.
$ModLoad imuxsock # provides support for local system logging (e.g. via logger command)
$ModLoad imjournal # provides access to the systemd journal
#$ModLoad imklog # reads kernel messages (the same are read from journald)
#$ModLoad immark # provides MARK message capability

# Provides UDP syslog reception
#$ModLoad imudp
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp
#$InputTCPServerRun 514

#### GLOBAL DIRECTIVES ####

# Where to place auxiliary files
$WorkDirectory /var/lib/rsyslog

# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# File syncing capability is disabled by default. This feature is usually not required,
# not useful and an extreme performance hit
#$ActionFileEnableSync on

# Include all config files in /etc/rsyslog.d/
$IncludeConfig /etc/rsyslog.d/*.conf
# Turn off message reception via local log socket;
# local messages are retrieved through imjournal now.
$OmitLocalLogging on

# File to store the position in the journal
$IMJournalStateFile imjournal.state

##### RULES #####

# 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!
*.*;mail.none;authpriv.none;cron.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
mail.* @10.11.12.13

# Log cron stuff
cron.* /var/log/cron

# Everybody gets emergency messages
*.* :omusrmsg::*

# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log
local7.* /var/log/boot.log
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
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:

<table>
<thead>
<tr>
<th>Minutes(0-59)</th>
<th>Hours(0-23)</th>
<th>Day of Month(1-31)</th>
<th>Month(1-12)</th>
<th>Day of Week(0-6)</th>
<th>command</th>
</tr>
</thead>
</table>
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 proceeded 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 straightforward. Replace the `-c` flag with an `-x`. This will cause the archive file to create directories if necessary and copy the archived files into 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:

<table>
<thead>
<tr>
<th>tar option</th>
<th>compression type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>compress</td>
</tr>
<tr>
<td>z</td>
<td>gzip</td>
</tr>
<tr>
<td>j</td>
<td>bzip2</td>
</tr>
</tbody>
</table>
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:

```plaintext
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:

<table>
<thead>
<tr>
<th></th>
<th>shareable</th>
<th>unshareable</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>/usr, /opt</td>
<td>/etc, /boot</td>
</tr>
<tr>
<td>variable</td>
<td>/var/mail</td>
<td>/var/run, /var/spool/mail</td>
</tr>
</tbody>
</table>
5. Documentation

Manpages and the whatis database

The manpages are organised in sections

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>the name of the item followed by a short one line description.</td>
</tr>
<tr>
<td>SYNOPSYS</td>
<td>the syntax for the command</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>a longer description</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>a review of all possible options and their function</td>
</tr>
<tr>
<td>FILES</td>
<td>files that are related to the current item (configuration files etc)</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>other manpages related to the current topic</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Name(key)</th>
<th>One line description</th>
</tr>
</thead>
</table>

The syntax for whatis is:

`whatis <string>`

The output is the full NAME section of the manpages where `string` matched `name(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 mkdir
mkdir (8) - make directories
```

```
man -k mkdir
mkdir (8) - make directories
```
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.

<table>
<thead>
<tr>
<th>Manpage Sections</th>
<th>Information on executables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1</td>
<td>Information on executables</td>
</tr>
<tr>
<td>Section 2</td>
<td>System calls, e.g. mkdir(2)</td>
</tr>
<tr>
<td>Section 3</td>
<td>Library calls, e.g. stdio(3)</td>
</tr>
<tr>
<td>Section 4</td>
<td>Devices (files in /dev)</td>
</tr>
<tr>
<td>Section 5</td>
<td>Configuration files and formats</td>
</tr>
<tr>
<td>Section 6</td>
<td>Games</td>
</tr>
<tr>
<td>Section 7</td>
<td>Macro packages</td>
</tr>
<tr>
<td>Section 8</td>
<td>Administration commands</td>
</tr>
</tbody>
</table>

To access a specific section, 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, CHANGEOLOG 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. 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

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/at.allow, at.deny</td>
<td>\texttt{at.allow(5)} – determine which user can submit commands for later execution via \texttt{at(1)} or \texttt{batch(1)}. The format of the files is a list of usernames, one on each line. Whitespace is not permitted. The superuser may always use \texttt{at}. If the file \texttt{/etc/at.allow} exists, only usernames mentioned in it are allowed to use \texttt{at}. If \texttt{/etc/at.allow} does not exist, \texttt{/etc/at.deny} is checked</td>
</tr>
<tr>
<td>/etc/cron.allow, cron.deny</td>
<td>\texttt{crontab(1)} – If the \texttt{cron.allow} file exists, then you must be listed therein in order to be allowed to use this command. If the \texttt{cron.allow} file does not exist but the \texttt{cron.deny} file does exist, then you must \textbf{not} be listed in the \texttt{cron.deny} file in order to use this command. If neither of these files exists, only the super user will be allowed to use this command</td>
</tr>
<tr>
<td>/etc/crontab</td>
<td>System \texttt{crontab} file read by the \texttt{crond} daemon whenever its modified time is changed</td>
</tr>
<tr>
<td>/etc/dumpdates</td>
<td>Stores information about the last full or incremental dumps</td>
</tr>
<tr>
<td>/etc/issue</td>
<td>Message printed by the \texttt{mingetty} program at the login prompt on a \texttt{tty}</td>
</tr>
<tr>
<td>/etc/issue.net</td>
<td>Message printed by the telnet daemon at the login prompt</td>
</tr>
<tr>
<td>/etc/logrotate.conf</td>
<td>Configuration file for \texttt{logrotate}</td>
</tr>
<tr>
<td>/etc/motd</td>
<td>Message displayed by \texttt{login} after a successful login</td>
</tr>
<tr>
<td>/etc/syslog.conf</td>
<td>Configuration file for \texttt{syslogd}</td>
</tr>
<tr>
<td>/usr/share/info</td>
<td>Directory where \texttt{info} pages are stored</td>
</tr>
<tr>
<td>/usr/share/man</td>
<td>Directory where the various sections of the \texttt{man} pages are stored</td>
</tr>
<tr>
<td>/var/spool/at</td>
<td>Directory containing spooled \texttt{at} and \texttt{batch} jobs</td>
</tr>
<tr>
<td>/var/spool/cron</td>
<td>Directory containing user defined \texttt{crontab} files. The \texttt{crontab} file has the name of the user that created it and can only be edited with the \texttt{crontab -e} command</td>
</tr>
</tbody>
</table>

Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apropos</td>
<td>\texttt{apropos(1)} – searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output</td>
</tr>
<tr>
<td>at</td>
<td>\texttt{at(1)} – read commands from standard input or a specified file which are to be executed at a later time</td>
</tr>
<tr>
<td>atd</td>
<td>\texttt{atd(8)} – run jobs queued by \texttt{at} for later execution</td>
</tr>
<tr>
<td>atq</td>
<td>\texttt{atq(1)} - 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</td>
</tr>
<tr>
<td>atrm</td>
<td>deletes jobs, identified by their job number</td>
</tr>
<tr>
<td>cron or crond</td>
<td>\texttt{crond(8)} – \texttt{Cron} searches \texttt{/var/spool/cron} for \texttt{crontab} files which are named after accounts in \texttt{/etc/passwd}; \texttt{crontabs} found are loaded into memory. \texttt{Cron} also searches for \texttt{/etc/crontab} and the files in the \texttt{/etc/cron.d} directory, which are in a different format</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>crontab</td>
<td>file loaded by <code>crond</code>. It is also the name of the program used to edit crontabs created by users in <code>/var/spool/cron</code></td>
</tr>
<tr>
<td>dd</td>
<td>copy files and devices with optional modifications such as block size (see <code>info coreutils dd</code>)</td>
</tr>
<tr>
<td>dump</td>
<td><code>dump(8)</code> – examines files on an ext2/3 filesystem and determines which files need to be backed up</td>
</tr>
<tr>
<td>info</td>
<td>read info documentation stored in <code>/usr/share/info</code></td>
</tr>
<tr>
<td>logger</td>
<td>allows shell scripts to log messages with <code>syslogd</code></td>
</tr>
<tr>
<td>logrotate</td>
<td><code>logrotate(8)</code> – 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</td>
</tr>
<tr>
<td>man -k</td>
<td>same as <code>apropos</code></td>
</tr>
<tr>
<td>restore</td>
<td>restore files or file systems from backups made with <code>dump</code></td>
</tr>
<tr>
<td>syslogd</td>
<td>The system logger. Programs can send messages to <code>syslogd</code> which include information such as the date and the host name. The configuration file <code>/etc/syslog.conf</code> is used to customise where messages are logged (e.g. file, device or remote logger)</td>
</tr>
<tr>
<td>tar</td>
<td><code>tar(1)</code> – 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</td>
</tr>
<tr>
<td>whatis</td>
<td><code>whatis(1)</code> – search the whatis database for complete words</td>
</tr>
</tbody>
</table>
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
1. Detecting Modems

Linux assumes in general that serial modems are connected to a serial port (one of the /dev/ttyS\textsubscript{N} 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\textsubscript{X} 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 \texttt{ATZ} or \texttt{ATDT}. One tool that will act as a terminal interface is \texttt{minicom}.

\textit{minicom} 

\begin{figure}
\centering
\includegraphics[width=\textwidth]{minicom_screenshot.png}
\caption{minicom screenshot}
\end{figure}

Another common tool is \texttt{wvdialconf}. This tool will automatically scan for modems on the ttyS's and create a configuration file called \texttt{/etc/wvdial.conf}. The next command will create or update the configuration file

\begin{verbatim}
wvdialconf /etc/wvdial.conf
\end{verbatim}

This file is used to handle password authentication and initialise the \texttt{pppd} daemon once the connection is established. If a dialer called \texttt{MYISP} is defined in \texttt{wvdial.conf} then the connection is started using

\begin{verbatim}
wvdial MYISP
\end{verbatim}
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' ''
'login:' '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 \
   crtscts \
   asyncmap 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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crtscts</td>
<td>use hardware flow control using the RTS and CTS signals</td>
</tr>
<tr>
<td>noauth</td>
<td>do not require the peer to authenticate itself</td>
</tr>
<tr>
<td>persist</td>
<td>do not exit after a connection is terminated but try to reconnect</td>
</tr>
<tr>
<td>require-chap</td>
<td>use <code>/etc/ppp/chap-secrets</code> for authentication</td>
</tr>
</tbody>
</table>

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:

```bash
# 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 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

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/ppp/options</td>
<td>options used by the pppd daemon (additional options can be passed on the command line)</td>
</tr>
<tr>
<td>/etc/ppp/chap-secrets</td>
<td>contains login information available when using the challenge handshake authentication protocol (CHAP)</td>
</tr>
<tr>
<td>/etc/ppp/pap-secrets</td>
<td>contains login information available when using the password authentication protocol (PAP)</td>
</tr>
<tr>
<td>/etc/ppp/peers/</td>
<td>contains files with connection information (user name, chat script) as well as pppd options</td>
</tr>
<tr>
<td>/etc/wvdial.conf</td>
<td>configuration file used by wvdial</td>
</tr>
</tbody>
</table>

Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chat</td>
<td>chat(8) – The chat 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 (pppd) and the remote pppd process</td>
</tr>
<tr>
<td>minicom</td>
<td>program used to communicate over a serial connection. Can be given a phone number, user name and password. Once the connection is established minicom acts as a terminal</td>
</tr>
<tr>
<td>pppd</td>
<td>pppd(8) – 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 pppd 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. Pppd can also authenticate the peer and/or supply authentication information to the peer.</td>
</tr>
<tr>
<td>wvdial</td>
<td>wvdial(1) – wvdial 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 chat(8) 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</td>
</tr>
</tbody>
</table>
Printing

Prerequisite

- None

Goals

- Understand the GNU printing tools used to submit and administrate print jobs
- Configure a LPRng print spooler
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:\n    :sd=/var/spool/lpd/lp:\n    :mx#0:\n    :sh:\n    :rm=192.168.1.20:\n    :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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-num</td>
<td>Print num copies</td>
</tr>
<tr>
<td>-Ppq</td>
<td>Specify the print queue pq</td>
</tr>
<tr>
<td>-s</td>
<td>Make a symbolic link in the spool directory rather than copy the file in</td>
</tr>
</tbody>
</table>

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/ldpd.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. CUPS**

CUPS (an acronym for Common Unix Printing System) is a modular printing system for Unix-like computer operating systems which allows a computer to act as a print server. A computer running CUPS is a host that can accept print jobs from client computers, process them, and send them to the appropriate printer.

CUPS consists of a print spooler and scheduler, a filter system that converts the print data to a format that the printer will understand, and a backend system that sends this data to the print device. CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues. It also provides the traditional command line interfaces for the System V and Berkeley print systems, and limited support for the Berkeley print system's Line Printer Daemon protocol. System administrators can configure the device drivers which CUPS supplies by editing text files in Adobe's PostScript Printer Description (PPD) format. There are a number of user interfaces for different platforms that can configure CUPS, and it has a built-in web-based interface. CUPS is free software, provided under the GNU General Public License and GNU Lesser General Public License, Version 2.

**Overview**

CUPS provides a mechanism that allows print jobs to be sent to printers in a standard fashion. The print-data goes to a scheduler which sends jobs to a filter system that converts the print job into a format the printer will understand. The filter system then passes the data on to a backend—a special filter that sends print data to a device or network connection. The system makes extensive use of PostScript and rasterization of data to convert the data into a format suitable for the destination printer.

CUPS offers a standard and modularised printing system that can process numerous data formats on the print server. Before CUPS, it was difficult to find a standard printer management system that would accom-
moderate the very wide variety of printers on the market using their own printer languages and formats. For instance, the System V and Berkeley printing systems were largely incompatible with each other, and they required complicated scripts and workarounds to convert the program's data format to a printable format. They often could not detect the file format that was being sent to the printer and thus could not automatically and correctly convert the data stream. Additionally, data conversion was performed on individual workstations rather than a central server.

CUPS allows printer manufacturers and printer-driver developers to more easily create drivers that work natively on the print server. Processing occurs on the server, allowing for easier network-based printing than with other Unix printing systems. With Samba installed, users can address printers on remote Windows computers, and generic PostScript drivers can be used for printing across the network.

**CUPS web-based administration interface**

On all platforms, CUPS has a web-based administration interface that runs on port 631. It particularly helps organisations that need to monitor print jobs and add print queues and printers remotely.

CUPS 1.0 provided a simple class, job, and printer-monitoring interface for web browsers.

CUPS 1.1 replaced this interface with an enhanced administration interface that allows users to add, modify, delete, configure, and control classes, jobs, and printers.

CUPS 1.2 and later provide a revamped web interface which features improved readability and design, support for automatically discovered printers, and a better access to system logs and advanced settings.

**CUPS web interface screenshot:**
5. Exercises and Summary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>Scripts used to prepare a document before printing</td>
</tr>
<tr>
<td>Device</td>
<td>Type of connection used to access the printer (e.g. parallel, USB or network)</td>
</tr>
<tr>
<td>Driver</td>
<td>Translates raw or postscript type formats into printer specific instructions such as PCL</td>
</tr>
</tbody>
</table>

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/printcap</td>
<td>Read by the <strong>lpd</strong> daemon at start up and contains a list of configured printers</td>
</tr>
</tbody>
</table>

**Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpc</td>
<td>line printer control program</td>
</tr>
<tr>
<td>lpd</td>
<td>line printer daemon</td>
</tr>
<tr>
<td>lpq</td>
<td>print printer queue status</td>
</tr>
<tr>
<td>lpr</td>
<td>submit files for printing</td>
</tr>
<tr>
<td>lprm</td>
<td>remove a queued print job</td>
</tr>
<tr>
<td>mpage</td>
<td>print multiple pages of a document on one page</td>
</tr>
</tbody>
</table>
Index

A
apachectl, 74
apropos, 100
arp, 56
at, 96
atd, 96
atq, 96
atrm, 96

B
backup strategies, 97
broadcast address, 42

C
chage, 29
chattr, 78
classless subnets, 44
cron, 91
crond, 95
crontab, 95

D
date, 86
dd, 99
depmod, 3
dhcpcd, 51
dhcppclient, 51
dig, 71
DNS
/etc/named.boot, 69
/etc/named.conf, 69
XE "dig"dig, 71
DNS Configuration Files, 69
DNS zone files, 70
Hierarchical structure, 69
host, 71
Types of DNS servers, 69
Dotted Quad, 41
driftfile, 87
dump, 98

E
eenv, 32
export, 32
exportfs, 64

F
Files
$HOME/.ssh, 84
/etc/aliases, 72
/etc/at.allow, 96
/etc/at.deny, 96
/etc/bashrc, 33
/etc/cron.allow, 96
/etc/cron.deny, 96
/etc/crontab, 95
/etc/dumpdates, 98
/etc/exports, 64
/etc/host.conf, 67
/etc/hosts, 68
/etc/hosts.allow, 63
/etc/hosts.deny, 63
/etc/inetd.conf, 69
/etc/inittab, 102
/etc/inittab.net, 102
/etc/localtime, 86
/etc/logrotate.conf, 94
/etc/mail/*, 72
/etc/motd, 102
XE "DNS:/etc/named.boot"/etc/named.boot, 69
XE "DNS:/etc/named.conf"/etc/named.conf, 69
/etc/networks, 53
/etc/nsswitch.conf, 67
/etc/ntp.conf, 87
/etc/ntp.drift, 87
/etc/ntp/drift, 87
/etc/ppp/ip-down, 109
/etc/ppp/ip-up, 109
/etc/printcap, 113
/etc/profile, 32
/etc/resolv, 68
/etc/resolv.conf, 50, 68
/etc/security/access.conf, 79
/etc/security/limits.conf, 79
/etc/sendmail.cf, 72
/etc/services, 47
/etc/shadow, 26
/etc/smb.conf, 66
/etc/ssh, 84
/etc/sysctl.conf, 51
/etc/syslog.conf, 91, 92
/etc/wdial.conf, 110
/etc/xinetd.conf, 60
/lib/modules/, 2, 3
/lib/prob/s/net/ipv4/ip_forward, 51
/usr/share/man, 101
/usr/share/zoneinfo/, 86
/var/log/httpd, 74
Index

/var/log/messages, 78
/var/log/secure, 78
/var/log/wtmp, 78
/var/run/utmp, 78
/var/spool/at/, 96
/var/spool/cron/<username>, 95
/var/spool/mail/, 73
/var/spool/mqueue, 73
~/.bash_profile, 32
~/.bashrc, 33
~/.forward, 73
~/.inputrc, 33
access.conf, 74
authorized_keys2, 84
httpd.conf, 74
id_dsa, 84
id_dsa.pub, 84
known_hosts, 84
modutils (package), 3
srm.conf, 74
ssh_config, 85
sshd_config, 85
find, 78
ftp, 62

G
gpasswd, 25
groupadd, 28
groupadd, 25
groups, 24

H
host, 71
HOWTOs, 102
hwclock, 86

I
ICMP, 46
id, 24
ifconfig, 51
inetd, 59
info, 101
init, 13
init (boot parameters), 18
init.d, 13
insmod, 3
IP, 46

K
kernel build
make clean, 7
make config, 6
make dep, 7
make menuconfig, 6
make modules, 8
make modules_install, 8
make oldconfig, 6
make xconfig, 6

L
last, 78
libwrap, 80
logger, 94
logrotate, 94
lpc, 115
lpd, 114
lpq, 114
lpr, 114
lprm, 114
lsmod, 3

M
mailq, 73
man --k, 100
Manpages, 100
MANPATH, 101
modinfo, 3
modprobe, 3
modprobe, 3
modules.conf, 4
modules.dep, 3
mpage, 115

N
netmask, 41
netstat, 55
network address, 42
News Groups, 102
NFS, 64
noexec, 78
nosuid, 78
NTP - network time protocol, 86
ntpd, 87
ntpd Date, 87

P
passwd, 23
ping, 55
portmap, 64
PPP, 46
pppd
XE "Files: /etc/wvdial.conf"/etc/wvdial.conf, 110
chap-secrets, 109
chat, 108
minicom, 107
Index

pap-secrets, 109
peers, 109
svdial, 107

pump, 51

R

restore, 98
rmmod, 3
route, 52

S

Samba
smbclient, 65
smbmount, 65

scripting
$s(( ))$, 38
expr, 38
for loop, 37
if then, 36
until loop, 37
while loop, 36

sendmail, 73
sendmail.cf
  Cw, 72
  Ds, 72
  Fw, 72
shutdown, 15
socket, 59
ssh, 84
sshd, 84
ssh-keygen, 84
subnetting, 44
subnets, 44
sysctl, 51
syslog.conf, 91
syslogd, 91

T

tar, 97
TCP, 46
TCP wrappers, 63
TCP/IP model (4 layer), 45
TCP/IP Suite, 45
tcp_wrapper, 80
tcpdump, 55
telnet, 61
test, 35
The Linux Documentation Project, 102
traceroute, 56

U

UDP, 46
unset, 32

useradd, 23, 28
usermod, 28

W

whatis, 100
who, 78

X

xinetd, 60
Vježbe
Vježba 1. Instalacija nove jezgre

1. Instalirajte potrebne pakete za prevođenje nove jezgre:
   ```
   yum -y groupinstall 'Development tools'
   yum -y install ncurses-devel rpm-build qt-devel
   ```

2. Preuzmite izvorni kôd Linuxove jezgre:
   ```
   cd /usr/src/kernels
   ```

3. Otpakirajte izvorni kôd:
   ```
   tar xjf linux-3.12.6.tar.bz2
   ```

   ```
   cd /usr/src/kernels/linux-3.12.6
   cp /boot/config-`uname -r` .config
   ```

5. Pokrenite alat za konfiguraciju jezgre:
   ```
   make menuconfig
   make xconfig
   ```


7. Sada možete prevesti jezgru i module koristeći naredbu:
   ```
   make
   ```

8. Sada možete instalirati novoprevedenu jezgru i pripadajuće module koristeći sljedeće naredbe:
   ```
   make modules_install
   make install
   ```
   Moduli se sada nalaze u direktoriju `/lib/modules/version`.
   Nova jezgra se nalazi u datoteci `bzImage` u direktoriju `/usr/src/kernels/linux-version/arch/x86/boot/`.
9. Sada možete urediti konfiguracijsku datoteku bootloadera GRUB:
   vi /boot/grub/menu.lst
   vi /boot/grub/grub.conf
   Zamijenite redak `default=1` u `default=0`.
   Dodajte znak `#` na početak retka `hiddenmenu`.

Vježba 2. Pokretanje Linuxa

1. Koristite naredbu `init` da promijenite trenutni runlevel (npr. prebacite se iz runlevela 5 u 3, te se vratite natrag u runlevel 5). Kako znate u kojem se runlevelu nalazite?

2. Promijenite lozinku korisniku `root` pokretanjem Linuxa u `single user mode`.

   Istražite razlike između naredbi `shutdown`, `halt` i `reboot`. Koja će opcija naredbe `shutdown` uključiti provjeru datotečnog sustava (fsck) kod sljedećeg pokretanja sustava?

4. Koristeći alate `chkconfig` i `ntsysv` ugasite `sshd` u runlevelima 2, 3, 4 i 5.
   Provjerite stanje simboličkih poveznica u direktorijima `rc2.d`, `rc3.d`, `rc4.d` i `rc5.d`.

Vježba 3. Upravljanje korisnicima i grupama

1. Koristeći naredbu adduser kreirajte korisnika tux s UID 600 i GID 550.

2. Koristeći naredbu usermod promijenite korisniku tux njegov osobni direktorij (home directory).
   
   Je li kreiran novi direktorij? ________________________________
   
   Ako nije, pokušajte s opcijom –m.
   
   Je li sadržaj direktorija /etc/skel iskopiran u korisnički direktorij? ______________________


5. Dodajte korisnika tux u grupu sales koristeći gpasswd.

6. Prijavite se kao korisnik tux i priključite se u grupu sales koristeći naredbu newgrp.

7. Dodajte korisnika uređivanjem datoteke /etc/passwd i /etc/group.

8. Napravite grupu share i dodajte korisnika tux u tu grupu uređivanjem datoteke /etc/group.


10. Zaklučajte korisnika tux i pokušajte se prijaviti na sustav kao korisnik tux.
Vježba 4. Skripte u BASH-u

1. U naredbenoj liniji izvezite (export) varijablu TEST:
   ```
   export TEST=old
   ```

2. Napišite skriptu:
   ```
   #!/bin/bash
   echo old variable: $TEST
   export TEST=new
   echo exported variable: $TEST
   ```

3. Koja je vrijednost varijable TEST nakon pokretanja skripte?

4. Sljedeća skriptu nazovite `test_shell`. Skripta ispisuje PID ljuske koja pokreće tu skriptu.
   ```
   #!/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. Podesite dozvole da se skripta može pokretati.

6. Ispitajte ponašanja skripte različitim pokretanjima.
   ```
   test_shell
   ./test_shell
   bash test_shell
   . test_shell
   source test_shell
   exec ./test_shell
   ```

   a. Pripada li adresa 161.53.17.129 toj mreži?

   b. Pripada li adresa 161.53.17.12 toj mreži?

   c. Koja je adresa razašiljanja (broadcast) u toj mreži?

   d. Je li mreža besklasna i, ako ne, koju klasu pripada?

   e. Koliko uređaja s distinktnim IP adresama stane u ovu mrežu?

   f. Koja je adresa mreže?

2. Dobili ste adresni prostor 83.10.11.0/27.

   a. Koliko postoji mreža koje imaju ista tri bajta?

   b. Koliko računala može biti u ovoj mreži?

   c. Koja je adresa razašiljanja (broadcast) u toj mreži?
Vježba 6. Konfiguracija mreže

1. Koristeći naredbu `ifconfig` pokušajte promijeniti IP adresu mrežne kartice u 192.168.0.x.

2. Koristeći naredbu `lsmod` izlistajte module jezgre i provjerite je li modul e1000 učitan.

3. Spustite mrežno sučelje koristeći naredbu:
   ```
   ifconfig eth0  down
   ```

4. Ponovno podignite mrežno sučelje koristeći naredbu:
   ```
   ifconfig eth0  up
   ```

5. Koristeći naredbu `rmmod` ubijte modul e1000 i provjerite postoji li mrežno sučelje eth0.

6. Koristeći naredbu `modprobe` učitajte modul e1000 i podesite IP adresu mrežne kartice kao u točki 1.

7. Podijelite se u dvije grupe. Jedna grupa će biti u adresnom prostoru A (192.168.1.0/24), a druga u adresnom prostoru B (10.0.0.0/8).

8. Pokušajte pingati računala iz druge mreže. Zašto to ne radi?

   ____________________________________________________________

   ____________________________________________________________
9. Jedno računalo iz mreže A će biti gateway između mreža A i B. Na tom računalu podešite dodatno sučelje eth0:0 s adresom iz mreže B (10.0.0.x, gdje je x slobodna adresa). Možete kopirati datoteku /etc/sysconfig/network-scripts/ifcfg-eth0 u ifcfg-eth0:0 i podešiti IP adresu i sučelje u novoj datoteci.

10. Dopustite IP forwarding koristeći sljedeću naredbu:
    
    echo 1 > /proc/sys/net/ipv4/ip_forward

11. Pokušajte pingati računala iz druge mreže. Radi li to sada?_____________________
Vježba 7. Mrežni servisi: xinetd

1. Ako nisu instalirani paketi **xinetd** i **telnet** potrebno ih je instalirati naredbom:
   
   ```bash
   yum -y install xinetd telnet
   ```


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

3. Podesite dozvole da se skripta može izvršavati.

4. U direktoriju `/etc/xinetd.d` kreirajte datoteku pod nazivom **fudge**. Sadržaj datoteke je sljedeći:

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

5. Dodajte servis **fudge** u datoteku `/etc/services` tako da koristi port 60000.

6. Ponovno pokrenite **xinetd** i pokušajte se naredbom **telnet** spojiti na port 60000.
Vježba 8. Mrežni servisi: DNS poslužitelj

1. Instalirajte DNS poslužitelj bind9:
   
   ```
yum -y install bind
   ```

2. Napravite sljedeće izmjene označene crvenom bojom u datoteci `/etc/named.conf`:
   
   ```
   listen-on port 53 { 127.0.0.1; 10.11.10.1xx; }; ### Master DNS IP
   allow-query { localhost; 10.11.8.0/22; }; ### IP Range
   ```

3. Na kraj datoteke `/etc/named.conf` dodajte:
   
   ```
   zone "srce.local" IN {
       type master;
       file "forward.srce";
       allow-update { none; };
   };

   zone "10.11.10.in-addr.arpa" IN {
       type master;
       file "reverse.srce";
       allow-update { none; };
   };
   ```

4. U direktoriju `/var/named` stvorite datoteku `forward.srce` sljedećeg sadržaja (xx zamijenite svojom IP adresom):
   
   ```
   $TTL 3H
   @ IN SOA @ linux1-xx.srce.local. ( 2014120800 ; serial
   1D ; refresh
   1H ; retry
   1W ; expire
   3H ) ; minimum

   IN NS linux1-xx.srce.local.
   linux1-xx IN A 10.11.10.1xx
   linux1-30 IN A 10.11.10.130
   ```
5. U direktoriju /var/named stvorite datoteku reverse.srce sljedećeg sadržaja:

```text
$TTL 3H
@ IN SOA @ linux1-xx.srce.local. (2014120800; serial
2D; refresh
1H; retry
1W; expire
3H); minimum

IN NS linux1-xx.srce.local.
linux1-xx IN A 10.11.10.1xx
1xx IN PTR linux1-xx.srce.local.
130 IN PTR linux1-30.srce.local.
```

6. Pokrenite sljedeće naredbe:

```bash
chgrp named /var/named/forward.srce
chgrp named /var/named/reverse.srce
service named start

named-checkconf /etc/named.conf
named-checkzone srce.local /var/named/forward.srce
named-checkzone 10.11.10.in-addr.arpa /var/named/reverse.srce
```

7. Dodajte redak „nameserver 127.0.0.1“ u datoteku /etc/resolv.conf.

Vježba 9. Mrežni servisi: web poslužitelj

1. Instalirajte web-poslužitelj Apache:
   ```bash
   yum -y install httpd
   ```

2. U datoteci `/etc/httpd/conf/httpd.conf` promijenite port na kojem sluša Apache (umjesto porta 80 postavite port 8080).

3. Provjerite možete li se naredbom `telnet` spojiti na port 8080.
   ```
   Trying 127.0.0.1...
   Connected to localhost.linuxit.org.
   Escape character is '^]'.
   Upišite "GET /" i pogledajte jeste li dobili HTML sadržaj.
   ```

4. Vratite port na kojem sluša Apache na 80.


6. Podesite dodatno sučelje `eth0:0` s adresom koju dobijete od predavača iz mreže 10.0.0.0/8.
   ```bash
   Možete koristiti naredbu:
   ifconfig eth0:0 10.11.10.x
   ```

7. Dodajte sljedeći odlomak u datoteku `/etc/httpd/httpd.conf`:
   ```
   <VirtualHost new-IP>
       DocumentRoot /var/www/html/virtual
       ServerName www1
   </VirtualHost>
   ```
   Za virtualni host kreirajte direktorij `/var/www/html/virtual`.

8. Ponovno pokrenite servise `httpd` i provjerite možete li se spojiti na novi virtualni host.
Vježba 10. Osnove sigurnosti

1. Pronađite sve datoteke u direktoriju `/usr` koje imaju postavljen SUID bit.

2. U datoteci `/etc/hosts.deny` podesite da se na servis `sshd` ne može spojiti.

3. U datoteci `/etc/hosts.allow` podesite da se na servis `sshd` može spojiti samo iz lokalne mreže.

4. Generirajte par ključeva (javni i privatni) i složite autentikaciju SSHD putem para ključeva.
   Koristeći naredbu `iptables` zabranite sav dolazni i odlazni promet.

5. Podesite `ntpd` tako da koristi poslužitelje `zg1.ntp.carnet.hr` i `zg2.ntp.carnet.hr`.
   Naredbom `touch` kreirajte datoteku `/tmp/neunistiva`. Koristeći naredbu `chattr`
podesite atribute tako da se datoteka ne može obrisati.
Vježba 11. Osnove administracije poslužitelja


2. Dodajte u datoteku /etc/rsyslog.conf da se logovi local5.priority šalju na virtualnu konzolu /dev/tty10.

3. Dodajte u cron da se svake dvije minute pokreće gnome-terminal. Prisjetite se da morate podesiti varijable PATH i DISPLAY.

4. Koristeći naredbu at pokrenite gnome-terminal u sljedećih 5 minuta.

5. Koristeći naredbu find pronađite sve datoteke izmijenjene u posljednja 24 sata.

6. Koristeći naredbu tar napravite arhivu svih datoteka kojima je pristupano ili koje su mijenjane u zadnjih 5 minuta.

   Napomena: Koristite naredbu find kako biste izradili arhivu datoteka. Arhivu spremite u datoteku /temp/last5. Prilikom uporabe naredbe tar koristite opciju kojom se arhiviraju datoteke zapisane u određenoj datoteci.

7. Testirajte arhivu prije raspakiravanja.

8. Raspakirajte arhivu koju ste napravili.
Vježba 12. Ispis

1. Instalirajte programski paket cups.

2. Pomoću web-preglednika pristupite localhost portu 631.


4. Istražite mogućnosti alata CUPS.
GNU FDL License Agreement

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. Secondarily, 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 worldwide, 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 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 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:

5. **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.

6. **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.

7. **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.

8. **D.** Preserve all the copyright notices of the Document.

9. **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

10. **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.

11. **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

12. **H.** Include an unaltered copy of this License.

13. **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.
14. 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.

15. 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.

16. 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.

17. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

18. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

19. 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 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 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/.

20. 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.