Network and systems administration

(A young lady's illustrated primer)

Edition 1.2, May 1998

Mark Burgess
Centre of Science and Technology
Faculty of Engineering, Oslo College

Foreword

This work is a crucial document for the schooling of young ladies, coping with the realities of modern life in cyberspace i.e. the metaverse. It assumes only a knowledge of the UNIX operating system as described in the accompanying compendium *Unix* which is available on-line at:

`http://www.iu.hioslo.no/~mark/unix/unix.html.`

The most up-to-date version of this manual can be found at:

`http://www.iu.hioslo.no/~mark/sysadmin/SystemAdmin.html.`

The images can be found at

`http://www.iu.hioslo.no/~mark/sysadmin/pictures.html.`

This document is first and foremost a text for a course in system administration at Oslo College. It is made publicly available in the hope that it might prove useful elsewhere,

but no guarantees are made about correctness or fitness for a particular purpose etc. If you are unfamiliar with this Internet tradition, refer to the GNU public license available from

```
`ftp://prep.ai.mit.edu/pub/gnu/GPL'.
```

Part of the section on security has been influenced by Tina Darmohray's thoughtful words on internet firewalls at the USENIX/LISA conference in 1997. Comments, observing suitable protocols and etiquette, to Mark.Burgess@iu.hioslo.no Oslo, June 1998

Getting started

In this manual the word "host" is used to refer to a single computer system -- i.e. a single machine which has a name termed its "hostname" See section Glossary, for other definitions.

What is system administration? The Zen thing.

@hrule @vskip 0.3cm @vskip 0.3cm

The job of a system administrator is a complex thing. It a difficult job, which requires patience, understanding and a lot of knowledge and experience. It is like working in the casualty ward of a hospital. You need to be a doctor, a psychologist, and--when your instruments fail--a mechanic. You need to work with the limited resources you have. You need to be inventive in a crisis, you need to know a lot of facts and figures about the way computers work. You need to recognize that the answers are not always written down for you to copy, that machines do not always behave the way you think they should. You need to do all this--and, on top of everything, you need to *remain calm*.

Being a system administrator is as much a state of mind as it is about being knowledgeable. It is the sound of one hand tapping (on the keyboard) while the other is holding the phone, talking to a user. You need to be ready for the unexpected, resigned to the uncertain future, and you need to be able to plan for the future. It requires organization and the ability to be systematic. There is no right answer. It's about doing something robust which works.

It's also a confidence thing. In the beginning you will be too nervous to do any real work, later, when you are more experienced, you learn to relax and just do things. It's kind of like dating or skiing, or skate-boarding, there are three stages:

1. **Under-confidence**: you place your skate-board carefully onto a flat surface, well away from spectators, step shakily onto it and rock about hopelessly.
2. **Optimal confidence**: you glance about you, then throw your plank on the ground, jump onto it and skate off with admirable self-assurance.

3. **Over-confidence**: you don't look around you and throw your daemon-board onto the ground, running, into the path of an angry cyclist who then breaks your nose for the inconvenience.

To get going, you need to know your stuff and have confidence in your abilities--but you also need to know your limitations.

If you have installed Windows, DOS or Linux on a PC, you might think that you know a lot about system administration, but in fact you know only the beginning. Networking is about cooperation and sharing in an environment with many users. Don't think that you know all the answers simply because you know how your own machine works.

This introduction is about one possible way of solving some common problems. You should not think of it as a universal recipe for happiness, only a suggestion. As a system administrator, your first responsibility is to the users of the system. Your job is to make their lives bearable(1), not to play with the system.

A philosophy

@hrule @vskip 0.3cm

This guide is a handbook for the beginner. You need to know where to begin and how to behave. We're not proposing mind-control, in the usual sense, but you need to kill some bad habits and learn some good ones. For a young lady to rise to the distinction of system operator, one needs to know the appropriate rules of etiquette. To begin with you would be ill-advised to

- Insist that there is a right or a wrong answer to every question.
- Run to someone else whenever there is a problem.
- Get fraught and upset when things do not work the way you expect.
- Listen to people who tells you how you are `supposed' to do things.
- Expect that every problem has a beginning, a middle and an end.

The well-versed and substantially educated young lady of the metaverse is a more subtle creature who (not so much attacks as) mud-wrestles problems by

- Looking for answers in manuals and newsgroups.
- Using trial and error (carefully!) to locate problems.
- Practice being cheerful and patient with the `stupid' (PC: neuronically challenged) users of your system and the even stupider machines.
- Listening to everyone who tells you that there is a problem. It might be true, even if you can't see it yourself.
- Writing down your experiences so that you will know how to solve the same problem again in the future.
- Expecting to learn lots of unrelated facts that you will need to keep carefully and assimilate at some point. You need to be organized!

- Taking responsibility for your own actions. Be prepared for accidents. They are going to happen and they will be your fault. You will have to fix them.
- Remembering the boring jobs like vacuum cleaning your hardware once a year and cleaning the hardware.

You could also begin getting used to using the English language (or American distortions thereof). English (American, alas) is the language of the net. You will need it to be able to read documentation, to be able to communicate with others and to ask questions on the internet.

This guide to successful living begins with how to get your bearings. If you don't know where you are, you can't see where you went wrong or where to go next. Start by being systematic NOW! Every time you read something, think: *how does this apply to me?* Do yourself a big favour and buy an A-Z that you can use to write down your experiences. Start now. And remember what you have learned.

Know your network

@hrule @vskip 0.3cm

The place to start is with your network. In most cases you will start with a network already in place, you will not have to build one from scratch (though when you have finished this section you might feel as though you would like to rebuild everything from scratch). You need to know what you have and where everything is, how it is organized (or not) and so on.

Here is a checklist of things you will need to find out about. Think hierarchically: you have a network of hosts running possibly several operating systems.

- How do they fit together?
- Which function does each host have on the network?
- Where are the key network services?

When you are done thinking `network', you can begin to think about individual hosts. First there is a hardware list.

- What kind of machines do you have on your network? What are their names and where are they? Do they have disks. How big? How much memory do they have? If they are PC's, which screen cards do they have?
- What operating systems are running on your network? MS-DOS, Novell, NT or UNIX (if so which unix? Linux, solaris, hpux?)
- What kind of network are you using? Is it thin/thick ethernet? Is it a star net (hubs/twisted pair), or fibre optic FDDI net?
- Where are hubs/repeaters/the router or other network control boxes located? Who is responsible for maintaining them? You? Or someone else?
- What is the topology of the network? Where do the cables run?

- Who works above you, along side you or under you? Who else can you ask for help?

Then there is a software list:

- How many different subnets does your network have?
- What are their network addresses?
- Find the router addresses (the default routes) on each segment.
- What is the netmask?
- What is your local timezone?
- What broadcast address do you use?
- Find the key servers on these networks. @itemize @bullet @item Where are the NFS network disks located? Which machine are they attached to? @item The name services (DNS/NIS/NISPLUS) @item The WWW/http service. @end itemize

Find and record this information now. Make sure you record it in a form which is easy to understand and easy to update. Remember that networks are always changing.

Common network models

Open systems is a concept promoted by Sun Microsystems for UNIX, which is about all systems being compatible and open for sharing. Some vendors, particularly Microsoft and MacIntosh are the opposite of this: they make their systems incompatible so that you are forced to buy their products. Not all UNIX OS's are open systems, so if you are interested in sharing and compatibility, choose your systems carefully.

Unix
: Any Unix system can perform any function, as server client or whatever you like. A Unix network is fully distributed, there is no requirement about centralization of resources. In principle, you can set up Unix however you see fit: as a server, a client or whatever. Unix is the most configurable of operating systems, but also the most complicated to administrate. Each host has a fixed IP address, or kan be assigned one automatically at boot time by a service such as *BOOTP* or *ARP/RARP*.

NT
: There are two types of system with separate software licenses: workstations and servers. NT revolves around a model in which programs may be run on a local workstation or on a server. The server runs network services in a centralized model. System administration is by graphical user interface, pretty-- but tiresome in the long run. NT does allow shell access! If you find the primitive shell command prompt interface a pain, you can install GNU unix commands and the `emacs` editor which offers you filename completion and lots of other goodies. All systems must run NT. Unix `rsh` protocols were recently added to NT-server, so that remote shell commands may be executed on UNIX or NT hosts. IP addresses

are fixed or may be assigned automatically by a network service such as *BOOTP* or NT's own *WINS*.

Novell
: The Novell system is a file and printer server only. The standard network protocol is IPX which is a local network protocol. It is faster than IP, but it is not a world wide protocol. Each PC can also obtain an IP address dynamically from a single server which replies to a `bootp` broadcast request. (This is like most X terminals under unix.) Several Novell file servers can coexist, and each PC couples to the first file-server which answers a request, unless the server is specified. In Novell 3 there is only one server for each PC; this handles all services and network file storage. Novell 4 can handle several servers. All services run on this one server machine. The server is independent of the basic operating system you run on your PC, it will work with DOS or Windows or Windows 95, or NT workstation. Novell is not a distributed system like Unix. It requires a special dedicated machine to perform a function as server.

MacIntosh
: Each MacIntosh is an independent system. Only simple services like `ftp` can be run in a limited way from a normal machine. Macintosh uses its own network protocol called Apple-talk which is incompatible with IP. Apple-talk servers allow networking and disk sharing. IP protocol disk sharing available but does not mix well with the MacIntosh file system. System administration (actually everything) is by GUI only.

NT is rapidly working to catch up with UNIX as a multiuser, multitasking system. Version 4.0 of NT is approaching UNIX functionality, for a price, but is still based around the idea that only one person will use a given PC at a time. Using the standard Windows user interface, you cannot run graphical applications remotely as you can using X-windows on unix, except through the single server. A basic X-windows release 6 is available for NT however, so you have the option of throwing away Windows and choosing a more distributed user interface.

Useful commands

@hrule @vskip 0.3cm

Here is a list of commands which you will find useful during your stint as a system administrator. Always check the manual page on your local system before trying these commands. Versions, optional and even names differ, especially on older systems.

Who am I?

`whoami`
: Prints your user name.

`who am i`
: Prints your real and effective user id, and terminal.

`id`

GNU program which prints all your user ids and groups.

Remote logins

The `telnet` command is the most reliable way of logging onto a remote unix host. The `rlogin` or `rsh` commands can be used to this effect, but they will sometimes hang without reason, where `telnet` works without problem.

The `rlogin` command can be used to login without a password using the `` `.rhosts' `` authority file for trusted hosts and users.

Monitoring disk usage

`df`
: Display the usage of all mounted disk partitions if no argument is given. If a directory is named, the state of the disk partition on which the given directory resides is displayed. On SVR4 systems the output of this command is hard to understand unless the `` `-k' `` option is used.

`du`
: Show disk usage on a per-file basis. The file sizes are either in kilobytes or in 512 byte blocks. The `` `-k' `` option forces output to be in kilobytes. The `` `-s' `` option prevents `du` from outputting information about every file and yields a summary of the named directory instead.

`swap -s`
: System 5 program to show swap space.

`pstat`
: BSD program to show swap space.

Disk backups

`dump`
: Raw dump of a disk partition to a file or to tape.

`rdump`
: Same as dump, but this can be done over the network, remotely without need for physical contact with the host.

`ufsdump`
: Solaris/SVR4 replaces `dump` with this command.

`restore`
: Restores a disk partition from a filesystem dump.

`cp -r`
: Copy a directory and all files recursively to a new location. This does not preserve symbolic links but makes multiple copies of the file instead. See `tar` below.

`tar`
: A simple way to copy an entire filesystem, preserving symbolic links is to do the following:

```
cd source-dir; tar cf - . | (cd destination-dir; tar xf - )
```

This pipes the output directly to the new directory using the streams interface for standard IO.

Mounting filesystems

`mount`
: Mount a local or remote disk.

`umount`
: Unmount a local or remote disk. Note the peculiar spelling.

`showmount`
: Show all hosts who are mounting filesystems from this server.

Packing and unpacking archives

`tar cf` *`tarfile`*`.tar` *`source-dir`*
: Packs all the files and sub-directories in the directory *source-dir* into a single `tape-archive' file. If the `-f' argument is missing, `tar` expects to be able to write data to a default tape-streamer device and will complain with an error message.

`tar zcf` *`tarfile`*`.tar.gz` *`source-dir`*
: Same as above, but piped through `gzip` to compress the data. This only works with GNU tar.

`tar xf` *`tarfile`*`.tar`
: Unpacks the contents of a tar-file into the current directory.

`tar zxf` *`tarfile`*`.tar.gz`
: Same as above, but pipes through `gzip` to uncompress data. This only works with GNU tar.

Shared libraries

`ldd`
: Display the shared libraries used by a compiled executable file (SunOS only?).

`ldconfig`
: Some systems require this command to be run after installing or upgrading shared libraries. It updates symbolic links to the latest version of the library and in some cases generates a cache file of library names. Esp. Linux and SunOS prior to Solaris.

Handling binaries

`strings`
: This command lists all of the strings in a binary file. It is useful for finding out information which is compiled into software.

`file`
: Prints the type of data a file contains.

`strip`
: Remove debugging information from a compiled program. This can reduce the size of the program substantially.

Files and databases

`locate`
: GNU fast-find command, part of the GNU `find` package. Locates the names of files matching the argument string in part, by reading from a database. See `` `updatedb' `` below.

`find`
: Locate by searching through every directory. Slow but powerful search facilities.

`which`
: Locate an executable file by searching through directories in the `PATH` or `path` variable lists.

`whatis`
: Gives a one-line summary of a command from the manual page (see `catman`).

`catman -M`
: This program builds the `apropos` or `man -k` `whatis' databases.

`updatedb`
: This shell script updates the `locate` fast-find database.

Process management

`ps aux`
: Show all processes on the system (BSD).

`ps -ef`
: Show all processes on the system (SysV).

`kill`
: Send a signal to the named process (pid), not necessarily to kill it. The process ID is the one listed by the `ps` command. Typical options are `` `-HUP' `` to send the hangup signal. This is used by many system daemons like `inetd` and `cron` as a signal which tells them to reread their configuration files. Another option is `` `-9' `` which is a non-ignorable kill instruction.

`nice`
: Run a program with a non-default scheduling priority. This exists both as a shell command and as a C-shell builtin. The two versions use different syntax. Normal users can only reduce the priority of their processes (make them `nicer'). Only the superuser can increase the priority of a process. The priority values differ between BSD and SysV systems. Under BSD, the nice values run from -20 (highest priority) to 19 (lowest priority) with 0 being the default. Under SysV, priorities run from 0 to 39, with 20 being the default. The C-shell builtin priorities are always from -20 to 20 for consistency.

`renice` *`new-priority`* `-p` *`pid`*
: Resets the scheduling priority of a process to a new value. The priority values used by the system (not C shell) apply here.

`crontab`
: Modern releases of Unix use the crontab command to schedule commands or scripts which are to be run at a specified time, or at regular intervals. The `crontab -l` command lists currently registered jobs. The `crontab -e` command is used to edit the crontab file. Each user has his or her own crontab file on every host. On older BSD systems, only root could alter the crontab file, which was

typically a single file `/etc/crontab' or `/usr/lib/crontab' containing usernames and jobs to be performed.

Mail management

Sometimes mail gets stuck and cannot be delivered for some reason. This might be because the receiving mailhost is down, or because there is insufficient disk space to transfer the message, or many other reasons. In that case, incoming and outgoing mail gets placed in a queue which usually lies under the unix directory `/var/spool/mail', `/var/mail' or one of these with `/var' replaced by `/usr'.

`mailq`
: Display any messages waiting in the mail queue. Same as `sendmail -bp`.

`sendmail -q -v`
: Manually process the mail queue in verbose mode.

Disk management

`format`
: Sun's interactive disk formatting and repair tool.

`fsck`
: The filesystem check program. A disk doctor. This checks the consistency of the filesystem (superblock consistency etc) and repairs simple problems.

`newfs`
: Creates a new filesystem on a disk partition, erasing any previous data. This is analogous to formatting a diskette.

`swapon`
: This command causes the system to begin using a disk partition or swap file for system swapping/paging. `swapon -a` starts swapping on all devices registered in the filesystem table `/etc/fstab' or equivalent.

`mkfile`
: Creates a special file for swapping inside a filesystem. The file has a fixed size, it cannot grow or shrink, or be edited directly. Normally swapping should be to a raw partition. Swapping to this kind of file is inefficient, but is used by (for instance) diskless clients.

Name service lookups

`nslookup`
: An interactive query program for reading domain data from the Domain Name Service (DNS/BIND)

`dnsquery`
: A non-interactive query program for reading domain data from the Domain Name Service (DNS/BIND)

`whois`
: Displays information about who is responsible for a limited number of domains in the US. For example, the highly irritating domain moneyworld.com can be found with `whois moneyworld.com`.

System statistics

`iostat`
: Displays I/O summary from the disks at an interval of *time-in-seconds*.

`vmstat`
: Displays virtual-memory summary info at an interval of *time-in-seconds*.

`netstat`
: Show all current network socket connections.

`netstat -i`
: Show statistics from all network interfaces.

`netstat -r`
: Show the static routing table.

`nfsstat`
: Show NFS statistics. The `-c` option shows client-side data, while the `-s` option shows server-side data, where appropriate.

Networks

`ping`
: Send a sonar `ping' to see if a host is alive. The `-s` option sends multiple pings on some types of UNIX.

`traceroute`
: Show the route, passing through all gateways to the named host. This command normally has to be made setuid-root in order to open the network kernel structures. Here is an example:

```
   traceroute to wombat.gnu.ai.mit.edu (128.52.46.26), 30 hops
max,
   40 byte packets
    1  ca30-gw (128.39.89.1)  3 ms  1 ms  2 ms
    2  hioslo-gw.uninett.no (158.36.84.17)  5 ms  4 ms  5 ms
    3  oslo-gw2.uninett.no (158.36.84.1)  15 ms  15 ms  19 ms
    4  no-gw2.nordu.net (128.39.0.177)  43 ms  34 ms  32 ms
    5  nord-gw.nordu.net (192.36.148.57)  40 ms  31 ms  38 ms
    6  icm-gw.nordu.net (192.36.148.193)  37 ms  21 ms  29 ms
    7  icm-uk-1-H1/0-E3.icp.net (198.67.131.41)  58 ms  57 ms  59
ms
    8  icm-pen-1-H2/0-T3.icp.net (198.67.131.25)  162 ms  136 ms
14
    9  icm-pen-10-P4/0-OC3C.icp.net (198.67.142.69)  198 ms  134
ms
   10  bbnplanet1.sprintnap.net (192.157.69.51)  146 ms  297 ms
128
   11  * nyc2-br2.bbnplanet.net (4.0.1.25)  144 ms  120 ms
   12  nyc1-br1.bbnplanet.net (4.0.1.153)  116 ms  116 ms  123 ms
   13  cambridge1-br1.bbnplanet.net (4.0.1.122)  131 ms  136 ms
203
   14  cambridge1-br1.bbnplanet.net (4.0.1.122)  133 ms  124 ms
140
   15  cambridge1-cr1.bbnplanet.net (206.34.78.23)  138 ms  129
ms
   16  cambridge2-cr2.bbnplanet.net (192.233.149.202)  128 ms
133 m
```

```
17  ihtfp.mit.edu (192.233.33.3)  129 ms  170 ms  143 ms
18  B24-RTR-FDDI.MIT.EDU (18.168.0.6)  129 ms  147 ms  148 ms
19  radole.lcs.mit.edu (18.10.0.1)  149 ms *  130 ms
20  net-chex.ai.mit.edu (18.10.0.2)  134 ms  129 ms  134 ms
21  * * *
22  * * *   <--- routing problem here
```

`etherfind`
: Dump ethernet packet activity to console, showing traffic etc, SunOS.

`snoop`
: Newer version of etherfind in Solaris.

`ifconfig`
: Configure or summarize the setup of the a network interface. e.g. `ifconfig -a` shows all interfaces. Used to set the broadcast address, netmask and internet address of the host.

`route`
: Make an entry in the static routing table. Hosts which do not act as routers need only a default route. e.g. `route add default` *`xxx.xxx.xxx`*`.1 1` or in Linux `route add default gw` *`xxx.xxx.xxx`*`.1`.

Internet searches

These days the simple way to search for software is to use the World Wide Web and a suitable search engine. For locating anonymous ftp data try the URL http://ftpsearch.ntnu.no/ftpsearch/ . Before this, the program `archie` was used to search the archives.

Health

People who use computers frequently are notoriously bad at looking after their own health and are not usually aware of how they can be damaging themselves. Unlike cigarettes, computers do not have a government health warning.

Real geeks think that it is cool to sit for long periods in front of a computer screen without a break, pouring coffee into their systems (preferably not over their systems). It's one of those macho things. Ladies of breeding know better!

Fortunately it is not difficult to avoid the worst problems. As a system administrator you need to look after your own health and concern yourself with the health of others. It is cool to be healthy. Although system folks are not responsible for users' health, they are the best source of information ignorant users have. Post information about computer health and help out your users.

Eyes. You should protect your eyes, you only have one pair and they must last you your whole life. Ironically, users who wear glasses (not contact lenses) suffer less from computer usage, because their eyes are partially protected from the radiation from the screen.

A computer screen works by shooting charged electrons at a phosphorescent surface. If you touch the screen you will notice that it is charged with static electricity. The effect of this is to charge dust particles and throw them out into your face. This can cause irritation to the eyes over long periods. Solution: wear glasses or obtain an anti-static screen with an Earth-wire which counteracts this problem.

Another major cause of eye strain is through reflections. If there is a light source behind you, it will reflect in your screen and your eyes will be distracted by the reflection. The picture you are trying to read on your screen lies on the screen surface, any reflected images lie behind the screen (as far behind the screen as the source is in front of the screen). Whether you notice it or not, this confuses your eyes into focusing back and forth between the reflection and the image you are trying to read. The result is eye-strain. The solution is to (i) eliminate all sharp light sources which can cause reflections, (ii) obtain an anti-reflective screen cover. This can be combined with an anti-static screen and it is probably the best investment you can make.

Prolonged eye strain can lead to problems reading and focusing. It can lead to headaches and neck ache from squinting.

Back. Your back (spine) is one of the most complex and important parts of the body. It supports your upper body and head, and is attached to your brain (where applicable). Your upper body is held up by muscles in your stomach and lower back. If you relax these muscles by slouching for long periods you place unnecessary strain on muscles and bones which were not meant to bear the weight of your body.

To avoid back problems, users should (i) sit in a good chair, (ii) sit upright, using those all important flat-tummy muscles and lower back muscles to support your upper body. If in doubt, and are not sure where your stomach muscles are, join an aerobics class. Remember: posture! Chin up, back straight!

Don't sit in a draft. Cold air blowing across your back and neck causes stiffness and tension.

Mouse strain Mouse strain is a strain in the tendons of the finger and forearm, which spreads to the shoulder and back and can be quite painful. It comes from using the mouse too much. The symptoms can be offset by making sure that you are not sitting too far away from the desk where your mouse lies and by having a support for your mouse forearm. The solution is simple: don't use the mouse. Use of the keyboard is far less hazardous. Learn the keyboard shortcuts instead. They are also quicker.

Pregnancy and cancer. Some studies recommend that pregnant women wear protective aprons when sitting in front of computer screens. After all, even after the development of low-radiation screens, you should realize that your are sitting in front of a source of radiation. Prolonged exposure to this kind of thing has been known to cause cancers. There is no cause for panic (you are probably more at risk sitting in the sun), but on the

other hand it is also unlikely that you will be run down by a truck, but if you sit in the middle of the road everyday for many hours, it becomes more likely.

Generally. Do not sit for long periods without taking a break. Look away from the screen (to a far away object) at regular intervals to relax your eyes. Walk around to exercise your back and relax your shoulders. Do not assume that you are immune to the problems above. Get an anti-static, anti-reflective screen. Anti-reflective coating on your screen are not good enough.

Weather

Whether you know it or not, the weather can affect your systems.

- Lightning strikes can destroy fragile equipment. No fuse will protect your hardware. Transistors and CMOS chips burn out much faster than any fuse. Electronic spike protectors can help you here.
- Power failure can cause disk damage and loss of data. A UPS (Uninterruptible power supply) can help here.
- Heat. The blazing summer heat can cause your systems to overheat and suddenly black out. You should not let the temperature rise much above about 20 degrees Centrigrade.
- Cold. Sudden changes from hot to cold are just as bad. They can cause unpredictale changes in electrical properties of chips and cause systems to crash. In the long term, these changes could lead to cracks in the circuit boards and irreparable chip damage.

Dealing with users: etiquette

@hrule @vskip 0.3cm

Although even the most stoical girl's convictions might occasionally be called into question, every young lady of breeding should properly regard *the user* as a highly pleasant and intelligent creature, whom it is a divine pleasure to work for.

The user, on the other hand, frequently believes that the system administrator has nothing better to do than to answer every question and even execute every fancy, many of which fall markedly short of the intelligence threshold. Since nail-fights and breaking the porcelain does not achieve practical results, we learn therefore to comport ourselves in a calm and pleasant manner in all encounters. Remember:

- Your posture (tummy muscles in) and the all-essential, superior smile.
- Inner contempt, outer magnanimity.
- Share your time fairly between all users.

Responsibility

As a system administrator, you wield great power. You have the ability to read everyone's mail, change anyone's files, to start and kill anyone's processes. This power can easily be abused. Any young lady with suitable upbringing abhors such behaviour, but we are all aware that the temptation could be great.

Rule zero: *Respect users and their privacy.*

Bugs

One of the first things you wil learn as a system administrator is that operating systems and programs are full of bugs. In your job as system administrator you have to learn how to stop complaining about these bugs and live with them. If you are lucky enough to be using free software from the net, these bugs will usually be solved quickly. If you are using commerical software you will probably have to wait a lot longer for a patch. Either way, you will have to be creative and work around these bugs.

Bugs can be caused by many things. Don't for one moment believe that you know everything about bugs. They may come from

- Shoddy software,
- Little known bugs in the operating system,
- Unfortunate collisions between incompatible software, i.e. one software package destroys the operation of another.
- Totally unexplainable phenomena, cosmic rays and invasions by digital lifeforms.

If you don't like the idea of living with bugs, then you should consider a future as an artist rather than a totally-digital computer whizz. Bugs are a reality, live with them.

Useful phrases and excuses

@hrule @vskip 0.3cm

- I'm so sorry, but the system is temporarily down for service. It will be up as soon as possible.
- We are cleaning up after a power failure.
- The network is a little slow through Washington -- it's all the red tape.
- The router was struck by lightening.
- `cat` ate `` `/dev/mouse' ``.
- I'd be happy to help you just as soon as I've finished checking every sector on this 32GB hard disk by hand.
- Don't be silly, of course you're not bothering me: it's my job! (Do not grind teeth.)
- So you deleted all your files? How about a nice hot cup of tea...
- To you, your adult Cindy bimbo WWW gif-gallery might be causing your hard-drive to expand, but to me it's just slipping my disk!
- No you can't use any more disk space! Die! Die! Die!

Getting your bearings

Checking out the low-life and obtaining crucial contacts

This chapter assumes that you have a reasonably functional network already. It does not tell you how to build one.

The first thing to do is to find out who you are and who everyone else is, and who is dating whom. To do that you need to know about your internet domain.

Rule one: *Remain calm and gather information.*

Know your system

Start by familiarizing yourself with the operating systems on your network. You need to know where to look for things and you need to know what kind of system you are using so that you know *who-yuh-gonna-call* when you need help or service. Start by logging onto each system and using the `uname` command to find out what OS you are using:

```
nexus% uname -a
SunOS nexus 5.5 Generic sun4m

borg% uname -a
Linux borg 1.3.62 #2 Mon Feb 12 11:06:19 MET 1996 i586
```

This tells us that host `nexus` is a SunOS kernel version 5.5 (colloquially known as Solaris 2.5) system with a sun4m series processor, and that host `borg` is a Linux system kernel version 1.3.62.

If the `uname` command doesn't exist, you know that your operating system is an old dinosaur from BSD 4.3 days and you will have to find out what it is by different means. Try the following commands: `arch` and `mach`.

- How much memory does your system have? (Most systems print this when they boot.) What disks and other devices are in use?
- Use `locate` and `find` and `which` and `whereis` to find important directories and software.
- What binary directories exist? `` `/usr' ``, `` `/usr/bin' ``?
- Where are the important files? For example, is your filesystem table called `` `/etc/fstab' ``, `` `/etc/vfstab' ``, `` `/etc/checklist' `` etc.. or something else?
- Do you have a C compiler and a C++ compiler installed?
- Is your system a system 5 or BSD host?
- Some systems have `` `/usr/spool' `` some have `` `/var/spool' ``, some have both

Using `nslookup`

`nslookup` is a program for querying the Domain Name Service (DNS). The name service provides a mapping or relationship between internet numbers and internet names, and contains useful information about domains: both your own and others. The first thing you need to know is your domain name. This is the suffix part of the internet names for your network. For instance, our domain at Oslo College, Faculty of Engineering has the domainname `` `iu.hioslo.no' ``. Hosts in this domain have names like `` `hostname.iu.hioslo.no' ``.

If you don't know your DNS domain name, you can probably find out by looking at the file `` `/etc/resolv.conf' ``. For instance:

```
borg% more /etc/resolv.conf
domain iu.hioslo.no
nameserver 128.39.89.10
nameserver 158.36.85.10
nameserver 129.241.1.99
```

This assumes that you already have a domain name and that your network is approximately set up already. If you don't have anything at all, then you need to obtain a domain name and network from an *Internet provider*. Lookup at reference [2] for more information.

Most UNIX systems have a command called `domainname`. This prints the name of the local Network Information Service (NIS) domain which is not the same thing as the DNS domainname (though, in practice, most people would use the same name for both). Do not confuse the output of this command with the DNS domain name.

Once you know your domainname, you can find out the hosts which are registered in your domain by running the program `nslookup`, or the name service lookup program. Type the command and you should see something like this:

```
borg% nslookup
Default Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

>
```

`nslookup` always prints the name and the address of the server from which it obtains its information. Then you get a new prompt `` `>' `` at which you can type commands. Typing `help` provides you with a list of commands which `nslookup` supports.

hostname/IP lookup

Type the name of a host or internet (IP) address and `nslookup` returns the equivalent translation. Example:

```
dax% nslookup
Default Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

> www.gnu.ai.mit.edu
Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

Name:    www.gnu.ai.mit.edu
Address:  206.126.32.23


>  128.39.89.238
Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

Name:    dax.iu.hioslo.no
Address:  128.39.89.238
```

In this example we look up the internet address of the host called `www.gnu.ai.mit.edu` and the name of the host which has internet address `128.39.89.238`. In both cases the default server is the name server `nexus.iu.hioslo.no` which has internet address `128.39.89.10`.

Note that the default server is the first server listed in the file `` `/etc/resolv.conf' `` which answers for queries when you start `nslookup`.

Special information

The Domain name service identified certain special hosts which perform services like the name service itself and mail-handlers (called mail exchangers). These servers are identified by special records so that people outside of a given domain can find out about them. After all, the mail service in one domain needs to know how to send mail to a neighbouring domain. It also needs to know how to find out the names and addresses of hosts for which it does not keep information personally.

We can use `nslookup` to extract this information by setting the `query type' of a request. For instance, to find out about the mail exchangers in a domain we write

```
> set q=mx
> domain name
```

For example

```
> set q=mx
> hioslo.no
Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

Non-authoritative answer:
hioslo.no       preference = 0, mail exchanger = samson.hioslo.no

Authoritative answers can be found from:
hioslo.no       nameserver = samson.hioslo.no
hioslo.no       nameserver = aun.uninett.no
samson.hioslo.no        internet address = 158.36.85.10
aun.uninett.no  internet address = 129.241.1.99
```

Here we see that the only mail server for `hioslo.no` is `samson.hioslo.no`.

Another example, is to obtain information about the nameservers in a domain. This will allow us to find out information about hosts which is not contained in our local database See section Listing hosts belonging to a domain. To get this, we set the query-type to `` `ns' ``.

```
> set q=ns
> hioslo.no
Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

Non-authoritative answer:
hioslo.no       nameserver = aun.uninett.no
hioslo.no       nameserver = samson.hioslo.no

Authoritative answers can be found from:
aun.uninett.no  internet address = 129.241.1.99
samson.hioslo.no        internet address = 158.36.85.10
>
```

Here we see that there are two authoritative nameservers for this domain called `aun.uninett.no` and `samson.hioslo.no`.

Finally, if we set the query type to `any', we get a summary of all this information.

Listing hosts belonging to a domain

To list every registered internet address and hostname for a given domain you use the `ls` command inside `nslookup`. For instance

```
> ls iu.hioslo.no
[nexus.iu.hioslo.no]
 iu.hioslo.no.                  server = samson.oslo.uninett.no
 iu.hioslo.no.                  server = nexus.iu.hioslo.no
 iu.hioslo.no.                  server = samson.hioslo.no
 pc61-74                        128.39.74.61
 pc59-74                        128.39.74.59
 pc59-75                        128.39.75.59
 pc196-73                       128.39.73.196
etc...
```

First the nameservers are listed and then the host names and corresponding IP addresses are listed.

If you try to look up hosts in a domain for which your default name server has no information, you will get an error message. For example, suppose we try to list the names of the hosts in the domain over ours:

```
 > ls hioslo.no
[nexus.iu.hioslo.no]
*** Can't list domain hioslo.no: Query refused
>
```

This does not mean that you cannot find out information about other domains, only that you cannot find out information about other domains from your default server See section Changing to a different server.

Changing to a different server

If you know the name of a server which contains authoritative information for a domain, you can tell `nslookup` to use that server instead. That way you can list the hosts in a remote domain and find out detailed information about that domain, See section Listing hosts belonging to a domain. *You always get more information from an authoritative server than you do from a remote server.* To change the server you simply type

```
> server new-server
```

or

```
> lserver new-server
```

There is a subtle difference between these two commands. If you use the first command to change the server to another host which is not running a `named` daemon (the DNS daemon), you will find yourself in a situation where you can no longer lookup hostnames or IP addresses because the host you have specified is not a server. In this case, you can use the second form which always uses the default (the first) server to look up the names

you use. This is only relevant if you use internet names on the command line and not IP addresses (which are always recognized).

We can use this now to list all of the data for a remote domain. First we change server; then once this is done we use `ls` to list the names.

```
> server samson.hioslo.no
Default Server:  samson.hioslo.no
Address:  158.36.85.10

> ls hioslo.no

(listing ..)
```

Another advantage with using the server which is directly responsible for the DNS data, is that we obtain extra information about the domain, namely a contact address for the person responsible for administrating the domain. For example:

```
> server samson.hioslo.no
Default Server:  samson.hioslo.no
Address:  158.36.85.10

> hioslo.no
Server:  samson.hioslo.no
Address:  158.36.85.10

hioslo.no       nameserver = aun.uninett.no
hioslo.no       preference = 0, mail exchanger = samson.hioslo.no
hioslo.no       nameserver = samson.hioslo.no
hioslo.no
     origin = samson.hioslo.no
     mail addr = postmaster.samson.hioslo.no
     serial = 1996120503
     refresh = 3600 (1 hour)
     retry   = 900 (15 mins)
     expire  = 604800 (7 days)
     minimum ttl = 86400 (1 day)
hioslo.no       nameserver = aun.uninett.no
hioslo.no       nameserver = samson.hioslo.no
aun.uninett.no  internet address = 129.241.1.99
samson.hioslo.no        internet address = 158.36.85.10
```

This is probably more information than you are interested in, but it does tell you that you can address queries and problems concerning this domain to `postmaster@samson.hioslo.no`. (Note that DNS does not use the @ symbol for `at' in these data.)

Contacting other domains

Sometimes you will need to contact other domains, perhaps because you believe there is a problem with their system, or perhaps because an unpleasant user from another domain is being a nuisance and you want to ask the administrators there to put that person to a long and painful death. You now know how to obtain one contact address using `nslookup`. Another good bet is to mail the one address which every domain must have: postmaster@*domain*. Any domain which does not define this mail address deserves to have its wires cut(2).

Various unofficial standards also encourage sites to have the following mail addresses which you might try:

```
webmaster
www
ftp
abuse
info
security
hostmaster
```

Apart from these sources, there is little one can do to determine who is responsible for a domain. A limited number of domains in the USA register with another network database service called the *whois* service. In some cases it is possible to obtain information this way. For example:

```
   whois moneyworld.com
   Financial Connections, Inc (MONEYWORLD-DOM)
      2508 5th Ave, #104
      Seattle, WA 98121

      Domain Name: MONEYWORLD.COM

      Administrative Contact, Technical Contact, Zone Contact, Billing
Contact:
         Williams, Bob  (BW747)  willie@MONEYWORLD.COM
         206 269 0846

      Record last updated on 13-Oct-96.
      Record created on 26-Oct-95.

      Domain servers in listed order:

      NSH.WORLDHELP.NET            206.81.217.6
      NSS.MONEYWORLD.COM           205.227.174.9


   The InterNIC Registration Services Host contains ONLY Internet
Information
   (Networks, ASN's, Domains, and POC's).
   Please use the whois server at nic.ddn.mil for MILNET Information.
```

NVRAM settings

Some hosts come with a basic `monitor' panel which is a ROM based program which can be used to set the configuration of Non Volatile RAM variables even before the system has booted up a true operating system. You should probably familiarize yourself with the kinds of variables which can be set in NVRAM for your system, if any. These could control basic choices about how your machine works, like which network interface is to be used: thick ethernet or twisted pair, etc. On a PC system running linux, this corresponds to the BIOS settings.

On solaris hosts there is a program called `eeprom` which can be used to set values in NVRAM.

Hardware awareness

To be a system administrator you do not usually need to know very much about hardware, but it is very useful to have a basic appreciation of how to install hardware and how to treat it. If you do not feel comfortable handling hardware, then don't do it!

Rule two: *If you can't cook, don't mess with the kitchen.*

Read instructions
: When dealing with hardware, always look for and *read* instructions in a manual. Do not assume that you know what you are doing. Instructions are there for a reason.

Interfaces and connectors
: Hardware is often connected to an interface or connector. Make sure that you have the correct kind of cable. Modem cables in particular can damage your computer or modem if they are incorrectly wired. Some computers supply power to certain pins which can damage equipment which does not expect to find a power supply coming across the cable. Network interfaces are often built in. Some interfaces are for thin ethernet, some for thick ethernet and others are for twisted pair connectors. If your computer has the wrong type of interface, you will need to buy a transceiver which converts the signal and connection to the right type.

Handling components
: Modern day CMOS chips work at low voltages (typically 5 volts). Standing on the floor with insulating shoes, you can pick up a static electric charge of several thousand volts. Such a charge can instantly destroy computer chips. Before touching any computer components, earth yourself by touching the metal casing of the computer. If you are installing equipment inside a computer, wear a conductive wrist strap.

Disks
: The most common disk types are IDE and SCSI (small computer software interface). IDE disks are usually cheaper than SCSI disks, but SCSI disks are

more efficient at handling multiple accesses, and are therefore better in multitasking systems. SCSI comes in several varieties, SCSI 1, SCSI 2, wide SCSI, fast-wide etc etc. The difference has to do with the width of the data-bus and the number of disks which can be attached to each controller. Each disk has its own address (or number) which must be set by changing a setting on the disk-cabinet or by changing jumper settings inside the cabinet. Disk chains must be terminated with a proper terminating connector.

Memory

Memory chips are sold on small circuit boards called SIMMs. These SIMMs are sold in different sizes and with different speeds. Your computer has a number of slots where SIMMs can be installed. When buying and installing RAM, remember @itemize @bullet @item The physical size of SIMMs is important. Most have 72 pins and some older SIMMs have 30 pins. @item SIMMs are sold in 1MB, 4MB, 16MB, 64MB sizes etc. Find out what size you can use in your system. In most cases you are not allowed to mix different sizes. @item Do not buy slower RAM than that which is recommended for your computer, or it will not work. @item On some computers you need fill up RAM slots in a particular order, otherwise the system will not be able to find them. @end itemize

Your physical network

There are two common styles of network:

Thin ethernet

In the thin ethernet scheme, hosts are chained together by stretches of cable. Each host taps into the main cable at suitable locations. The ends of the cable must be terminated by a 50-ohm resistance. Any break in the cable disables the cable, potentially for all machines. Finding cable breaks requires special tools. Most signal losses are caused through ignorance by users who break the cable integrity. Thin ethernet cable lengths are limited to 30m per segment (between machines) but signal amplifiers (called repeaters) or signal splitters (called multi-port repeaters) can be used to increase this length. A maximum of three repeaters can be used. A total maximum length of 300m cannot be exceeded. To make a longer network, one needs to join to separate networks together by a router.

Star net/ twisted pair

This scheme is far less susceptible to cable problems. Each twisted pair (ISDN) telephone style socket is connected by an independent cable to a central HUB. A twisted pair network uses a cable directly from computer to HUB, so the cable is automatically terminated. A connection board is used to wire each computer into a HUB which acts as an exchange, routing packets internally between the legs of the star.

Other network devices include *bridges* which are small-scale routers which separate two segments of a network. Bridges do no forward packets which do not need to cross the bridge, so they may be used to isolate busy segments of a network, thereby reducing traffic in another segment.

See the pictures at

`http://www.iu.hioslo.no/~mark/sysadm/pictures.html`

Understanding Networks

The art of big-time gossip and address books. Social contacts and neighbourhood watch.

If your UNIX system is connected to a network (and let's face it, every young lady's personal computer simply ought to be), then a general understanding of networks is a must. This chapter is a minimal introduction to some of the concepts you should have a feel for. You can refer to the bibliography if you would like to read more, See section Bibliography.

General concepts

Here are some things that you should know. Browse these facts and be informed.

- A network is a line of communications between hosts. It is usually some kind of common cable which is attached to several hosts simultaneously by means of a *network interface*.
- A host can be connected to several networks. It must have a network interface for each network it is attached to. Each network interface has a separate address. Thus a host which is connected to several networks will have a different address on each network.
- A host which is coupled to several networks and which forwards data from one network to another is called a router.
- Only one host can be using a given network cable at a given instant. It is like a conference telephone call: what goes out onto a network reaches all hosts on that network (more or less) simultaneously, so everyone has to share by waiting for a suitable moment to say something. (3)
- Every network message starts with a `header' which says who the message is intended for.
- Unix hosts have a `virtual internal network' called the *loopback* network, which simply loops back to the host itself. This is used for sending socket-like protocol data to itself. The loopback address is always the same for every host: 127.0.0.1
- Some internet addresses are reserved for a special purpose. These include *network addresses* (usually xxx.yyy.zzz.0), *broadcast addresses* (usually xxx.yyy.zzz.255) and *multicast addresses* (usually 224.xxx.yyy.zzz).
- Routers prevent the spread of network messages. A router knows which destination addresses lie on which of its networks and does not let message traffic spread onto irrelevant cables.
- A *bridge* is a convenient hardware device which acts like a filter on busy networks. A bridge works like a `mini-router' and separates two segments of the

same cable. A bridge knows which parts of the cable do *not* contain a destination address and prevents traffic from spreading to this part of a cable. A bridge is use to isolate traffic on busy sections of a network.

- A repeater is an amplifier which strengthens the network signal over long stretches of cable. A multi-port repeater does the same thing and also splits the cable into N sub-cables for convenience.

If you have done your homework, then you will know where all of these boxes are on your network.

IP addresses and netmasks

Every network interface on the internet needs to have a unique number which is called its address. At present the internet protocol is ipv4 and this number consists of four bytes, or 32 bits. In the future this will be extended, in a new version of the internet protocol ipv6, to allow more IP addresses since we are rapidly using up the available addresses. The addresses will also be structured differently. The form of an IP address in ipv4 is

```
aaa.bbb.ccc.mmm
```

Some IP addresses represent networks, whereas others represent hosts, or actually (if we are being 100 percent correct) they represent network interfaces. The usual situation is that an IP address represents a host attached to a network. Such an address has a host part and a network part, but the format of the address is not necessarily the same for all hosts and networks. There is some freedom to define local conventions.

In every address there are 32 bits. We can choose to use these bits in different ways: we could use all 32 bits for host addresses and keep every host on the same enormous cable, without any routers(4), or we could use all 32 bits for network addresses and have only one host per network (i.e. a router for every host). Both these extremes are silly. After all, we are trying to save resources by sharing a cable between convenient groups of hosts, but shielding other hosts from irrelevant traffic. What we want instead is to group hosts into clusters so as to restrict traffic to localized areas.

Networks fall historically into three classes called class A, class B and class C networks. Because of the possibility of dividing into *subnets*, the distinction between these network types has to do with the historical development of the internet. The difference between class A, B and C networks concerns which bits in the IP addresses on that network refer to the network itself and which bits refer to actual hosts.

IP addresses from `0.0.0.0` to `127.0.0.0` are class A networks, where the first byte is a network part and the last three bytes are the host address. This allows @math{256^3} minus reserved addresses hosts on the network. Since this is a ludicrously large number, class A networks are no longer issued and all the free addresses are wasted. Class A

networks were intended for very large organizations and are only practical with the use of a netmask which divides up the large network into manageable subnets.

IP addresses from `128.0.0.0` to `191.255.0.0` are class B networks. Here the first two bytes are the network part and the last two bytes are the host part. This gives a maximum of @math{256^2} minus reserved host addresses. Class B networks are typically given to large institutions such as universities and internet providers.

IP addresses from `192.0.0.0` to `223.255.255.0` are class C networks. Here the first three bytes are network addresses and the last byte is the host part. This gives a maximum of 254 hosts per network.

A given network can be divided into convenient *subnets* by using a *netmask*. The host part of an IP address can be divided up into two parts by moving the boundary between network and host part. The netmask is a variable which contains zeroes and ones. Every one represents a network bit and every zero represents a host bit. By changing the value of the netmask, we can trade many hosts per network for many subnets with fewer hosts.

Some IP addresses are reserved for a special purpose:

```
0.0.0.0       Default route
127.0.0.1     Loopback address
*.*.*.0       Network address
*.*.*.255     Broadcast address
*.*.*.1       Router or gateway (conventionally)
```

The default route is a default destination for outgoing packets on a subnet and is usually made equivalent to the router address. The *loopback address* is an address which every host defines. It points straight back to the host itself. It is a kind of internal fake network which allows programs to use IP protocols to address local services. The zeroth address of any network is reserved to mean the network itself, and the 255'th (or on older networks sometimes the zeroth) is used for the broadcast address.

The IP address of a host is set in the network interface. The UNIX command `ifconfig` (interface-configuration) is used to set this. Normally this is set at boot time by a shell script executed as part of the `` `rc' `` start-up files. These files are often constructed automatically during the system installation procedure. The `ifconfig` command is also used to set the broadcast address and netmask for your subnet.

Each system interface has a name. You should find out what the name of the interface is on your system. A look at the manual page can help here. Here are the network interface names commonly used by different UNIX types.

```
   Sun            le0
```

```
DEC ultrix    ln0
DEC OSF/1     ln0
HPUX          lan0
AIX           en0
Linux         eth0
IRIX          ec0
FreeBSD       ep0
Solarisx86    dnet0
```

Look at the manual entry of your system for this command. Here is a typical `ifconfig command` which sets the internet address, netmask and broadcast address on a SUN system which a Lance-ethernet interface.

```
ifconfig le0 128.39.89.10 up netmask 255.255.255.0 broadcast
128.39.89.255
```

Normally you will not need to use this command directly, since it should be in the startup-files for the system, from the time you installed the system. But you might be working in single-user mode or are trying to solve some special problem. You might discover that a system has been incorrectly configured. Note that `cfengine` can also be used to set the netmask and broadcast address.

Routing

Unless your system is going to operate as a router in some capacity, you will only need the most minimal of routing configurations. Each host must define a *default route* which is a destination to which outgoing packets will be sent for processing. This is normally the address of the router or gateway on your network segment. It is set by a command like this:

```
route add default my-gateway-address 1
```

The syntax varies slightly between systems. On Linux systems you must write:

```
/sbin/route add default gw my-gateway-address metric 1
```

You can check that your default route is correctly set using the `netstat -r` command. The result should just be a few lines like this:

```
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use
Iface
localnet        *               255.255.255.0   U     0      0      932
eth0
loopback        *               255.0.0.0       U     0      0       38
lo
default         my-gw           0.0.0.0         UG    1      0     1534
eth0
```

where `my-gw` is the address of your local gateway (usually subnet address 1).

If this default route is not set, your system will not know where to send packets and will therefore attempt to build a table of routes, using a different entry for every outgoing address. This consumes memory rapidly and makes an enourmous table. In the worst case you might not have contact with anywhere outside your subnet at all.

PART I: UNIX

Unix

Installation procedure

Shopping and putting together your crucial outfit.

Just forget the Zen state of awareness thing for a moment: no one seriously expects a young lady's wisdom to come by purely divine enlightenment. Information is the beginning of the system administrators wisdom, but you need to know how to find information yourself by looking at manuals and by asking others. You should get used to the fact that you will almost never find exactly what you are looking for, because everybody's needs are different. Let us state another rule for the record:

Rule three: *The system administrator should always be well informed.*

We begin therefore by describing some useful information for beginners and certain key procedures. These are things you will be doing quite often so wake up and pay attention!

Installing a new unix host

@hrule @vskip 0.3cm

Installing a new machine is a simple affair these days. The operating system comes on some removable medium (like a CD). You put it in the player and run a program, usually called install. Alternatively, you boot a machine directly from the CD and the program is

run automatically. Then you simply answer a few questions and the installation is done for you.

Operating systems are getting big so they are split up into packages. You are expected to choose whether you want to install everything or whether you just want to install certain packages. Most operating systems provide a fancy package installation program which helps you with this. In most cases these programs are quite stupid, they don't tell you that something will break if you don't install certain packages. For that reason it is strongly recommended that you always install the complete operating system: every single package. Whether you know it or not, you almost certainly need the whole thing--and the stuff you don't need probably doesn't take up much space anyway. Disk is cheap, but time spent trying to find out what went wrong with an installation is expensive.

In order to answer the questions about installing a new host, you need to collect some information and make some choices:

- You must decide a name for your machine.
- You will need an unused internet address.
- You need to decide how much swap space to allocate. A good rule of thumb is at least twice the amount of RAM you have installed.
- You need to know the local netmask.
- You need to know the local timezone.
- You need to know the name of your local domain.
- You need to know whether you are using the Network Information Service (NIS) and, if so, what the name of the NIS server is.

When you have this information, you are ready to begin. As an example, you can look at the installation instructions for Debian Linux at

```
http://www.debian.org
```

Once you have installed your operating system, remember:

Rule four: *Don't mix your own files with the operating system files. If you need to edit OS files, make sure you keep a master version separately in your site dependent files. Copy or link the master, so that you always have your data safe if you reinstall the OS later.*

When installing a large group of client machines, the easiest way might be to install one client by hand, and make the other clients a copy of this "client master". This saves time in selection, installation and configuration of software packages. Here is an example from our college, which uses the Debian linux operating system.

When all software packages have been installed, and all neccesary configuration has been done, make a backup of the machine with something like this command:

```
tar --exclude /iu --exclude /proc --exclude /lib/libc.so.5.4.23 \
    --exclude /etc/hostname --exclude /etc/hosts -c -v              \
    -f /iu/nexus/ud/ds/dump/vorlon.tar /
```

Note that several files must be excluded from the backup. In this case, all our NFS disks reside in /iu, and we don't want them in our backup. The file /lib/libc.so.5.4.23 is the C library, if we try to write this file back from backup, the destination computer will crash immediately. /etc/hostname and /etc/hosts contains definitions of the hostname of the destination computer, and must be left unchanged.

On the destination clients, make a minimal Linux installation. Make sure that you install NFS and network support, as the backup will be installed from an NFS disk. When your new client is finished, mount the NFS disk where the backup resides, and restore the tar archive, for example by this command:

```
(cd / ; tar xvfp /mnt/ds/dump/vorlon.tar; lilo)
```

It is essential that the lilo command is run immediately after the tar archive has been restored. If you forget it, your system will not be able to reboot!

Then, remember to check weither the file `` `/etc/init.d/network' `` is correct. You also have to create your NFS directories by hand, for example:

```
mkdir /iu/borg/local
mkdir /iu/nexus/local
mkdir /iu/nexus/ud
mkdir /iu/nexus/ua
mkdir /iu/nexus/u1
mkdir /iu/nexus/u2
mkdir /iu/nexus/u3
mkdir /iu/nexus/u4
mkdir /iu/nexus/u5
mkdir /iu/nexus/u6
```

Check that the correct swap partition is defined in `` `/etc/fstab' ``. You might also have to change `` `/etc/X11/Xserver' `` and `` `/etc/X11/XF86Config' ``.

Finally, reboot your system.

Marrying unix and NT

@hrule @vskip 0.3cm

If you are installing a PC, you will probably be obsessed with the idea of being able to choose between a whole bunch of operating systems. you will no doubt want all of these:

DOS
: To play games.

Unix
: A real operating system.

NT
: Because everyone is talking about it.

OS/2
: Because nuns use it in the jungle. And because it has a super-duper boot manager which is really nice.

You will want each of these operating systems to live on a different partition of your hard disk(s). The question is: how do we marry these operating systems in such a way that they do not try to kill each other? It would be nice to get a menu at boot-time where a girl could choose her favourite operating system and like *loose* the others prompto! This section is about the harmonious existence of these operating systems: don't you just love weddings?

According to the grapevine, the installation programs for NT and linux do not always respect each other's independence. Experience says: install NT first, then the OS/2 boot manager, then unix. The OS/2 boot manager is a wonderful program which allows you to choose operating system at boot-time. All you need is about 2 megabytes of free space on your bootable hard disk to install it. You get hold of the diskettes for OS/2/Warp and run the installation program: after a couple of messages you will end up in the `fdisk` partition manager. If you select an area of free space, you can choose install-boot-manager from the menu and the boot manager install itself. Once you have come through the `fdisk` partition program, you do not have to proceed with the rest of the OS/2 installation unless you feel that you want to lend moral support to this historical and much-underrated operating system.

You can then install linux or your favourite free-unix, being careful *not* to choose to install a `master boot record' as you move through the installation menus.

If you do not have access to the OS2 boot manager, you can use NT's own bootmanager if you trick it by copying the boot blocks from the unix filesystem to a DOS filesystem. The command

```
dd if=/dev/hda linux-partition of=/dev/hda DOS/bootsect.linux bs=512
count=1
```

copies the bootsector to `` `c:\bootsect.linux' ``. It can now be used by the bootmanager, by editing `` `C:\boot.ini' ``

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(NT partition number)\WINNT
[operating systems]
c:\bootsect.linux="Linux"
multi(0)disk(0)rdisk(0)partition(NT partition)\WINNT="Windows NT"
```

An alternative method is to copy the program `loadlin` and kernel to the DOS drive so that UNIX can be started from a windows command (which you might like to put in a batch file)

```
loadlin vmlinuz root="/dev/hdaLinux-partition"
```

Note -- it might not be a good idea to use this method from a multitasking Windows NT machine since it does not give the system a chance to shut down properly and damage to the NT partition might result. This method was meant for DOS PCs.

Problems have been known to occur with partitions using the linux disk formatting tool. Sometimes this tool fails to acknowledge a partition or free space with a message `Bad primary partition'. If this occurs, try to use a different formatter to create a partition which uses up the free space and try again. This seems to make the problem go away in the cases I have seen.

Another problem is that the lilo disk boot information might not get written to the linux partition. To fix this problem, boot with the linux boot diskette and edit the file `` `/etc/lilo.conf' ``. Make sure that the correct partition is bootable (use `fdisk`) if you need to see a list of partitions. When you are done you must type the command `lilo` and press ENTER. If the lilo boot information is not correctly stored, the boot manager might not recognize the partitions and claim that the linux partition is not formatted.

Installing a diskless client

@hrule @vskip 0.3cm

If you plan to run diskless unix workstations, i.e. workstations which have no physical disk attached but use a network server for all disk operations, then you need to remember a few things.

Generally speaking, diskless workstations are not a good idea. Disks are cheap these days and diskless workstations are expensive in terms of network bandwidth, time wasted because they run slowly and--not least--time spent fixing problems with them.

Most vendors will supply a script for creating a diskless workstation. You should use this script or you will just invite trouble. Here is a checklist of things to remember:

- A diskless workstation needs its own root filesystem and its own swap space, but it can share system files under `/usr'. You need to create disk areas for these. Call them something like `/export/swap/*hostname*' and `/export/root/*hostname*'. These areas need to be exported to the clients with root privileges granted.
- `/etc/ethers' on the server must contain the ethernet addresses of the clients.
- When a diskless system is switched on for the first time, it has no files and knows nothing about itself except the ethernet address on its network card. It proceeds by sending a RARP (Reverse address resolution protocol) out onto the subnet in the hope that a RARP server (`in.rarpd`) will respond by telling it its internet address.
- A subsequent call to the `bootparamd` daemon transfers data about where the diskless station can find its server, and what its swap-area and root directory are called in the file tree of this server.
- Diskless workstations swap to files. The command `mkfile` is used to create a fixed-size file for swapping.

Booting a machine

@hrule @vskip 0.3cm

How do you start a networked computer? You might think that this is just a case of switching it on, but you would not always be right about that! Sometimes you need to tell a system where it should boot from (a disk or the network) or even how it should boot.

WARNING! *You should read the whole of this section before attempting to boot or reboot a system. You can cause damage to files and disks by your careless actions.*

Booting unix

Normally it is sufficient to switch on the power to boot a UNIX system. Sometimes you might have to type `boot' or `b' to get it going. Unix systems can boot in several different modes or *run levels*. The most common modes are called multi-user mode and single-user mode. On different kinds of unix, these might translate into run-levels with different numbers, but there is no consensus. In single-user mode no external logins are permitted. The purpose of single-user mode is to allow the system administrator access to the system without fear of interference from other users. It is used for installing disks or when repairing filesystems, where the presence of other users on the system would cause problems.

In the olden days, the unix boot procedure was controlled entirely by a file called `/etc/rc' meaning `run commands' and subsidiary files like `rc.local'. These files were no more than shell scripts. Newer versions of unix have made the boot process insufferably complex by introducing a program called `init`. `init` reads a configuration file called `/etc/inittab' and a directory called `/etc/rc.d'. `/etc/inittab' defines a number of run-levels, and starts scripts depending on what run-level you

choose. The idea behind `inittab' is to make unix installable in packages, where each package can be started or configured by a separate script. Which packages get started depends on the run-level you choose.

The default form for booting is to boot in multi-user mode. You should find out how to boot in single-user mode on your system, in case you need to repair a disk at some point. Here are some examples.

Under the SunOS and Solaris versions of unix, one interrupts the normal booting process by typing `stop a`, where `stop` represents the `stop key' on the left hand side of the keyboard. If you do this, you should alway give the `sync' command to synchronize disk caches and minimize filesystem damage.

```
Stop a

ok? sync
ok? boot -s
```

If the system does not boot right away, you might see the line

```
type b) boot, c) continue or n) new command
```

In this case, you should type

```
b -s
```

in order to boot in single-user mode.

Under the Linux operating system, using the LILO boot system, you need to interrupt the normal boot sequence by pressing the `SHIFT` key. You should then see the prompt

```
Boot:
```

To boot, you must normally specify the name of a kernel file which is `linux'. To boot in single-user mode, you should type

```
Boot: linux single
```

Or at the LILO prompt, you can type `?` in order to see a list of kernels. There appears to be a bug in some versions of linux so that this does not have the desired effect. In some cases you will be prompted for a run-level. The correct run-level should be determined from the file `/etc/inittab'. It is normally called `S` or `1` or even `1S`.

Once you are in single-user mode, you can return to multi-user mode just by exiting the single-user login.

Shutting down unix

Anyone can start a unix system, but you have to be the superuser to shut one down correctly. Of course, you could just pull the plug, but if you are a well-informed system administrator, then you will know that this can ruin the disk filesystem. Even when no users are touching a keyboard anywhere, a unix system can be writing something to the disk--if you pull the plug, you may interrupt a crucial write-operation which destroys the disk contents.

The correct way to shutdown a unix system is to run one of the following programs.

`halt`
: Stops the system immediately and without warning. All processes are killed with the TERMinate signal 15 and disks are synchronized.

`reboot`
: As `halt`, but the system reboots in the default manner immediately.

`shutdown`
: This program is the recommended way of shutting down the system. It is just a friendly user-interface to the other programs, but it warns the users of the system about the shutdown and allows them to finish what they are doing before the system goes down.

Here are some examples of the `shutdown` command. The first is from BSD unix:

```
shutdown -h +3 "System halting in three minutes, please log out"

shutdown -r +4 "System rebooting in four minutes"
```

The option `-h` implies that the system will halt and not reboot automatically. The option `-r` implies that the system will reboot automatically. The times are specified in minutes.

System V unix R4 (e.g. solaris) has a different syntax which is based on its confusing system of run-levels. The shutdown command allows one to switch run-levels in a very general way. One of the run-levels is the `not running' or `halt' run-level. To halt the system, we have to call this.

```
shutdown -i 5 -g 120 "Powering down os...."
```

The `` `-i 5' `` option tells SVR4 to go to run-level 5, which is the power-off state. Run level 0 would also suffice here. The `` `-g 120' `` option tells `shutdown` to wait for a grace-period of 120 seconds before shutting down.

WARNING! Never assume that the run levels on one system are the same as those on another. This would be a big mistake.

Mounting NFS disks

@hrule @vskip 0.3cm

The sharing of disks over the network is the province of NFS (Network File System). Unix disks on one host may be accessed across the network by other UNIX hosts, or by PC's running PC-NFS. A disk attached physically to a host called a *server* is said to be *mounted* on a client host. In order to maintain a certain level of security, the server must give other hosts permission to mount disks. This is called *exporting* or *sharing* disks.

Server side exporting

In order to mount a disk on a server you must export the disk to the client (this is done on the server) and you must tell the client to mount the disk. Permission to mount disks is given on the server in a file which is called `` `/etc/exports' `` or on recent SVR4 hosts `` `/etc/dfs/dfstab' ``. The format for information in these files differs from system to system so you should always begin by looking at the manual page for these files. Here are two examples. The first is from linux

```
# See exports(5) for a description.
# This file contains a list of all directories exported to other
computers.
# It is used by rpc.nfsd and rpc.mountd.
/iu/borg/local daystrom(rw) worf(rw) nanite(rw)  *.iu.hioslo.no(ro)
```

In this example, a file system called `` `/iu/borg/local' `` is exported read-write explicitly to the client hosts `daystrom`, `worf`, and `nanite`. It is also exported read-only to any host in the domain `iu.hioslo.no`. This last feature is not available on most types of UNIX.

On some brands of unix (such as SunOS 4.1.*) one must run a command af>

Transfer interrupted!

ister the changes. The command is `exportfs -a` to export all filesystems. The command `exportfs` alone shows which filesystems are currently exported and to whom.

Our second example is from Solaris (SVR4). The file is called `` `/etc/dfs/dfstab' ``. Under Solaris, one can use the `share` command to export filesystems manually from the shell, using a command line of the form

```
share -F nfs -o rw=hostname filesystem
```

The `/etc/dfs/dfstab' file is in fact a shell script which simply executes such a command for each filesystem of interest. This has several advantages over traditional export files, since one may define variables, as in the example below.

```
#       place share(1M) commands here for automatic execution
#       on entering init state 3.
#
#       share [-F fstype] [ -o options] [-d "<text>"] <pathname>
[resource]
#       .e.g,
#       share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2

hostlist=dax:axis:ferengi:regula:lore:borg:worf:daystrom:worf.iu.hioslo
.no\
:daystrom.iu.hioslo.no:nostromo:voyager:aud4:aud4.iu.hioslo.no\
:aud1:aud1.iu.hioslo.no:aud2:bajor:nostromo:galron:ds9:thistledown\
:rama:takpah:takpeh:pc-steinarj:pc-hildeh:pc-torejo:pc-
torejo.iu.hioslo.no

share -F nfs -o rw=$hostlist  /iu/nexus/local
share -F nfs -o rw=$hostlist,root=dax /iu/nexus/u1
share -F nfs -o rw=$hostlist,root=dax /iu/nexus/u2
share -F nfs -o rw=$hostlist,root=dax /iu/nexus/u3
share -F nfs -o rw=$hostlist,root=dax /iu/nexus/u4
share -F nfs -o rw=$hostlist /var/mail
```

This script exports the six named filesystems, read-write to the entire list of hosts named in the variable `hostlist`. The command `shareall` runs this script, or it can be run manually by typing `sh /etc/dfs/dfstab`. The command `share` without arguments shows the currently exported filesystems. Notice that the hostname `pc-torejo` is repeated, once unqualified and again with a fully qualified hostname. This is sometimes necessary in order to make the entry recognized. The mount daemon is not particularly intelligent when it verifies hostnames. Some systems send the fully qualified name to verify and others send the unqualified name. If in doubt, list both like this.

Client side mounting

Clients may mount any subdirectory of the exported directory onto any local directory by becoming `root` and either executing a shell command of the form

```
mount server:remote-directory local-directory
```

or by adding a line to the filesystem table file, usually called `/etc/fstab'. On some brands of unix, this file has been renamed as `/etc/checklist' or

`/etc/filesystems'. On Solaris systems it is called `/etc/vfstab'. The advantage of writing the disks in the filesystem table is that the mount commands will not be lost when you reboot your system. The filesystems in the filesystem table file are mounted automatically when the system is booted. You can also mount all file systems in this file with the simple command `mount -a`.

You should begin by looking at the manual page on the appropriate file for your system, or better still looking at examples which are already in the file. The form of a typical filesystem table is as below(5) .

```
/dev/sda2                swap            swap    rw,bg   1   1
/dev/sda1                /               ext2    rw,bg   1   1
/dev/sda3                /iu/borg/local  ext2    rw,bg   1   1
nexus:/iu/nexus/u1       /iu/nexus/u1    nfs     rw,bg
nexus:/iu/nexus/u2       /iu/nexus/u2    nfs     rw,bg
nexus:/iu/nexus/u3       /iu/nexus/u3    nfs     rw,bg
nexus:/iu/nexus/local    /iu/nexus/local nfs     rw,bg
```

This example is from linux. Notice the left hand column. These are disks which are to be mounted. The first disks which begin with `/dev' are local disks, physically attached to the host concerned. Those which begin with a hostname followed by a colon (in this case host `nexus`) are NFS filesystems which lie physically on the named host. The second column in this table is the name of a directory on which the disk or remote filesystem is to be mounted on ---i.e. where the files are to appear in the local host's file-tree. The remaining columns are options and filesystem types: `rw` means mount for read and write access, `bg` means `background' which tells `mount` to continue trying to mount a filesystem in the background if it fails on a first attempt.

Editing the `/etc/fstab' (or equivalent) file is a process which can be automated very nicely with the help of the system administration tool cfengine. We shall discuss this in the next chapter.

Trouble-shooting

If you get a message telling you `Permission denied' when you try to mount a remote filesystem, you may like to check the following:

- Did you remember to add the name of the client to the `export' or `dfstab' file on the server?
- Some systems require a fully qualified hostname (i.e. hostname with domainname appended) in the export file. Try using this.
- Did you mis-spell the name of the client or the server?
- Are the correct network daemons running which support nfs? On the server side, you must be running `mountd` or `rpc.mountd`. This is an authentication daemon. The actual transfer of data is performed by `nfsd` or `rpc.nfsd`. On older systems there should be at least four of these daemons running to handle multiple requests.

Modern systems use a multi-threaded version of the program, so that only one daemon is required. On the client side, some systems use the binary input/output daemon to make transfers more efficient. This is not strictly necessary to get NFS working. This daemon is called `biod` on older systems and `nfsiod` on newer systems like FreeBSD. Solaris no longer makes use of this daemon, its activities are now integrated into a kernel thread.

Installing a new disk

@hrule @vskip 0.3cm

Adding a new disk or device to a unix machine generally involves a little planning. The first thing you should know about your system is what kind of hard-disks you use already, and what kind you intend to use in the future. If you do not know much about harddisks, it might be a good idea to talk to someone who can advise you.

The first concern is what type harddisk. There are several types of disk interface used for communicating with harddisks. Some are cheap and cheerful (IDE), while others are more expensive and reliable (SCSI).

IDE
: You can use a maximum of 2 disks and the size of the disks may be limited to less than a megabyte, but these disks are cheap and cheerful.

EIDE
: An extended version of the IDE interface. Allows 4 disks.

SCSI (Small computer systems interface)
: Most small Unix systems use SCSI disks. This interface can be used for devices other than disks too. It is better than IDE at multitasking. The original SCSI interface was limited to 7 devices in total per interface. Wide SCSI can deal with 14 disks. SCSI interfaces are developing rapidly and can be found in SCSI I and SCSI II flavours, with fast, wide, differential. Talk to a dealer about the possibilities.

On some PC's IDE disks boot before SCSI disks because of the way the BIOS is configured. You can get problems with a PC with a SCSI disk and IDE if you want to boot from the SCSI device.

Here is a typical checklist for adding a SCSI disk to a Unix system.

- Connect disk and terminate SCSI chain with proper terminator.
- Set the SCSI id of the disk so that it does not coincide with any other disks.
- On SUN machines you could now use the ROM command `probe-scsi` to check that the system is able to find all your disks.
- Partition and label the disk. Update the defect list.
- Edit the `` `/etc/fstab' `` filesystem table, or equivalent to mount the disk. See also next section.

`mount' and `umount'

To make a disk partition appear as part of the file tree it has to be "mounted". We say that a particular filesystem is *mounted on* a directory. The command `mount' mounts filesystems and disks defined in the filesystem table file. This is a file which holds data for mount to read.

The filesystem table has different names on different implementations of UNIX.

Solaris 1 (SunOS)
: /etc/fstab

Solaris 2
: /etc/vfstab

HPUX
: /etc/checklist

AIX
: /etc/filesystems

IRIX
: /etc/fstab

ULTRIX
: /etc/fstab

OSF1
: /etc/fstab

LINUX
: /etc/fstab

These files also have different syntax on different machines. The eventual standard which most systems comply with (SunOS, HPUX, OSF1) is

```
#
# SunOS 4* / Solaris 1
#
/dev/sd0a                /                 4.2 rw                1 1
/dev/sd0g                /usr              4.2 rw                1 2
# NFS
gluino:/mn/gluino/pc     /mn/gluino/pc     nfs rw,bg,hard,intr   0 0
fidibus:/var/spool/mail  /var/spool/mail   nfs rw,bg,hard,intr   0 0
fidibus:/mn/fidibus/u1   /mn/fidibus/u1    nfs rw,bg,hard,intr   0 0
fidibus:/mn/fidibus/u2   /mn/fidibus/u2    nfs rw,bg,hard,intr   0 0
```

In HPUX:

```
#
# HPUX
#
/dev/dsk/c201d6s0        /               hfs defaults  0 1
```

```
/dev/dsk/c201d5s0        /mn/hope/disk   hfs defaults  0 2
fidibus:/mn/fidibus/fys  /mn/fidibus/fys nfs rw,nosuid 0 0
fidibus:/usr/spool/mail  /usr/mail       nfs rw,suid 0 0
fidibus:/mn/fidibus/u1   /mn/fidibus/u1  nfs rw,suid 0 0
fidibus:/mn/fidibus/u2   /mn/fidibus/u2  nfs rw,suid 0 0
hassel:/mn/hassel/u1     /mn/hassel/u1   nfs rw,suid 0 0
```

The syntax of the command is

```
mount filesystem directory type (options)
```

There are two main types of filesystem -- a disk filesystem (ufs, hfs) (which means a physical disk) and the *NFS* network filesystem. If we mount a 4.2 filesystem it means that it is, by definition, a local disk on our system and is described by some logical device name like `` `/dev/something' ``. If we mount an NFS filesystem, we must specify the name of the filesystem and the name of the host to which the physical disk is attached.

Here are some examples, using the SunOS filesystem list above:

```
mount -a                # mount all in fstab
mount -at nfs           # mount all in fstab which are type nfs
mount -at 4.2           # mount all in fstab which are type 4.2
mount /var/spool/mail   # mount only this fs with options given in fstab
```

(The `` `-t' `` option does not work on all UNIX implementations.) Of course, we can type the commands manually too, if there is no entry in the filesystem table. For example, to mount an nfs filesystem on machine `gandalf' called `` `/site/gandalf/local' `` so that it appears in our filesystem at `` `/mounted/gandalf' ``, we would write

```
mount gandalf:/site/gandalf/local /mounted/gandalf
```

The directory `` `/mounted/gandalf' `` must exist for this to work. If it contains files, then these files will no longer be visible when the filesystem is mounted on top of it, but they are not destroyed. Indeed, if we then unmount using

```
umount /mounted/gandalf
```

(the spelling is correct) then the files will reappear again. Some implementations of NFS allow filesystems to be merged at the same mount point, so that the user sees a mixture of all the filesystems mounted at the same point.

Disk device names

The convention for naming disk devices in BSD and system 5 unix differs. Let us take SCSI disks as an example. Under BSD, the SCSI disks have names according to the following scheme

/dev/sd0a
: First partition of disk 0 of the standard disk controller. This is normally th root file system `/'.

/dev/sd0b
: Second partition of disk 0 on the standard disk controller. This is normally used for the swap area.

/dev/sd1c
: Third partition of disk 1 on the standard disk controller. Note that this partition is usually reserved to span the entire disk, as a a reminder of how large the disk is.

System 5 unix employs a more complex, but also more general naming scheme. Here is an example from solaris 2:

/dev/dsk/c0t3d0s0
: Disk controller 0, target (disk) 3, device 0, segment (partition) 0

/dev/dsk/c1t1d0s4
: Disk controller 1, target (disk) 1, device 0, segment (partition) 4

Not all systems distinguish between target and device. On many systems you will find only t or d but not both.

Registering a printer

@hrule @vskip 0.3cm

The way in which you register a printer depends (i) on what kind of UNIX you are using, and (ii) on whether you are running any special network printer software. The main difference is between BSD-like systems and System V.

There are also two types of printer which you might want to register: (i) a printer which is connected physically to a UNIX host by a cable attached to its printer port, and (ii) a stand-alone printer which is simply coupled to the network with its own IP address.

In order to use a printer, you need to do the following:

- Think of a name for your printer.
- Decide whether it is going to be connected directly to a host or stand alone on the network.
- Make a `spool' directory for its queue files. This should probably lie under `var/spool' or `/usr/spool'.
-
- `mkdir /var/spool/`*`printer-name`*

- Register the printer with the printing system so that the daemons which provide the print service know how to talk to it. You will need to decide whether you will be sending all data to a common central server, or whether you will be attaching printers locally to several machines, each running their own print-server.

Most unix systems assume the existence of a *default* printer which is referred to by the name `lp'. If you do not specify a particular printer when printing, your data are sent to the default printer. It is up to you to name or alias one of your printers `lp'. Each printer may have several names or aliases.

BSD printer with `lpd`

The file `/etc/printcap' is used to register a printer with a BSD system. If you are lucky, there might be a script or a user-interface for helping you to register a printer, if not you will have to follow the recipe below.

The format of the `/etc/printcap' file can be quite simple in most cases. The manual page for `printcap` contains a description of the file format. This file consist of a list of entries (each on a single line, or split over several lines using the continuation character `\'). A simple template entry looks like this:

```
printer-name-1|printer-alias-1|printer-alias-2:\
:lp=:
:sd=spool-directory:\
:rm=remote machine or IP address of printer:\
:rp=name of remote printer on remote machine:
```

This file should be installed on all hosts which need to access the printer, regardless of whether the printer is physically attached to them or not. Here is an example which registers two printers. The first is called `virtual-light' and is connected physically to the remote host `nexus`. The second is a stand-alone printer which we have named `diff-engine' and has IP address `128.39.89.99`(6).

```
#
# /etc/printcap
#
virtual-light|lp|default|SPARCprinter, a Sun SPARCprinter printer:\
        :lp=:\
        :lf=/var/adm/lpd-errs:\
        :sd=/var/spool/VirtualLight:\
        :rm=nexus:\
        :rp=virtual-light:

diff-engine|HP laser stand-alone:\
        :lp=:\
        :lf=/var/adm/lpd-errs:\
        :sd=/var/spool/otherprint:\
```

```
:rm=128.39.89.99:\
:rp=(none):
```

Note that the `rp` field exists in case a given printer has a different name on the remote host to the one you have given it locally on your machine. On a stand-alone printer this name is irrelevant.

Sys V

The `lpadmin` command is used to install printers under system 5. This commmand is complex and has many command line options to add and remove printers. If you have system 5 systems, consult the manual page on your system.

Sun Microsystems provide a script front end to help simplify this procedure called `AdminTool`.

Environment variable `PRINTER`

The BSD print command and some application programs read the environment variable `PRINTER` to determine which printer destination to send data to. The System V print command `lp` does not.

Setting up an X-terminal

@hrule @vskip 0.3cm

An X-terminal is a computer without a CPU. It cannot execute programs, but it has just enough power to display an X-windows graphical desktop. X-terminals have no disk, but they need to run software called an X-server. They need to download this program from the network. Each X-terminal manufacturer provides software for its X-terminals. As a system administrator you must decide on a *boot-server* for X-terminals: i.e. a system which will transfer X-terminals' software to them when they boot and tell them what their internet addresses are.

Most X-terminals boot using a protocol called `bootp` in which the X-terminal sends a broadcast signal to the subnet asking for a boot server. A unix machine which runs the `bootpd` daemon `bootpd` daemon looks at the ethernet address of the X-terminal sending the request and uses a configurable table to determine an internet address for it. It returns this information to the X-terminal, t>

Transfer interrupted!

ware it needs to download. Most X-terminals then download their software using the `tftp` (trivial file transfer program) protocol, which is a way of downloading files without a password from a restricted area of the server's disk.

The server keep all the X-terminals files in a directory which is usually called `` `/tftpboot' `` (7). In order to make the chosen server work for the X-terminal, it must be configured to run the `tftp` and `bootp` services. This is done as follows:

- Edit the file `` `/etc/services' `` on the server and add the protocol lines for these services if they do not already exist, See section Installing a new service.
-
-
```
bootp           67/udp  bootps  # boot program server
tftp            69/udp
```
-
- Check to see whether you have an up to date `bootpd`. The version supplied by some vendors is old. You can collect a modern version by anonymous ftp from the internet, See section Internet searches. Compile and install this program.
- The `bootp` configuration table is called `` `/etc/bootptab' ``. Remember that keeping configuration files in a safe place is always a good idea. Store this file in your site-dependent files and make a symbolic link to `` `/etc/bootptab' `` so that you don't lose the file if you re-install the OS. The format of this file is like this:
-
```
# Legend:       (see bootptab.5)
#       first field -- hostname (not indented)
#       bf -- bootfile
#       bs -- bootfile size in 512-octet blocks
#       cs -- cookie servers
#       df -- dump file name
#       dn -- domain name
#       ds -- domain name servers
#       ef -- extension file
#       gw -- gateways
#       ha -- hardware address
#       hd -- home directory for boot-files
#       hn -- host name set for client
#       ht -- hardware type
#       im -- impress servers
#       ip -- host IP address
#       lg -- log servers
#       lp -- LPR servers
#       ns -- IEN-116 name servers
#       ra -- reply address
#       rl -- resource location protocol servers
#       rp -- root path
#       sa -- boot server address
#       sm -- subnet mask
#       sw -- swap server
#       tc -- template host (points to similar host entry)
```

```
#       td -- TFTP directory
#       to -- time offset (seconds)
#       ts -- time servers
#       vm -- vendor magic number
#       Tn -- generic option tag n

  # Define two variables/macros

  .default:\
          :hn:dn=iu.hioslo.no:\
          :td=/iu/nexus/local/tftpboot:\
          :hd=/:\
          :ds=128.39.89.10:\
          :sm=255.255.255.0:\
          :gw=128.39.89.1:\
          to=auto:

  .iu             :sm=255.255.255.0:gw=128.39.89.1 :tc=.default:

  #name :   ha=ethernet-address:ip=internet-
address:bf=bootfile:tc=data

  xterm1:   ha=080006030ED4:ip=128.39.89.254
:bf=/td/servers/Xtd.05.1  :tc=.iu
  xterm2:   ha=080006030EC0:ip=128.39.89.248
:bf=/td/servers/XtdF.05.1 :tc=.iu
  xterm3:   ha=080006030FB7:ip=128.39.89.253
:bf=/td/servers/XtdF.05.1 :tc=.iu
```

- Finally you must edit the file `/etc/inetd.conf' which is used to start the daemons for the bootp and tftp services. Make sure that you have lines of the following forms (BSD style)

```
    tftp dgram udp wait root /usr/sbin/in.tftpd in.tftpd -s
/path/tftpboot
    bootp dgram udp wait root /local/bin/bootpd bootpd -i -d
```

Note that the path names above should be altered to fit in with your local configuration. You should use the true path-names here, not use symbolic links. Security issues prevent `tftpd` from reading through symbolic links. When you are finished editing the file you must restart the `inetd` daemon, or send it the hang-up signal using `kill -HUP` *`pid`*.

If you have trouble setting up X-terminals, here is a troubleshooting list.

- X-terminal does not find a bootp server: check that the configuration in `` `/etc/inetd.conf' `` is correct for the bootp-daemon. If your bootp setup is working, you should be able to type `telnet` *`server`* `bootp` and obtain a response.
- Error message permission denied on X-terminal while downloading software or fonts: Could be that the pathname to `` `tftpboot' `` directory was incorrect or was only a symbolic link to the real directory. You must use the real directory name here. Check that the software files are readable to everyone.

PPP internet over serial lines

@hrule @vskip 0.3cm

PPP stands for the Point to Point Protocol. It is a way of using a computer whose only link to the outside world is a serial line (a modem coupled to the telephone, for instance). Other protocols exist for this purpose: SLIP, for instance or Serial Line Internet Protocol, but PPP is generally regarded as better.

To run PPP, say from your portable or home computer using a modem, you need a program which talks PPP. On unix systems this program is called `pppd`. This daemon provides the backbone of the PPP service. It opens a logical network interface, anaologous to the ethernet interface `eth0`, called `ppp0`. When PPP is configured, you can see this interface just like a standard network interface by typing `ifconfig -a`. The `pppd` program is an ungraceful workhorse which requires an enormously long command line. It requires the assistance of another program called `chat` to communicate with the modem. Both `pppd` and `chat` need to be available and in your command path.

PPP setup depends on several things: the kind of unix you are starting from, the kind of system you are logging on to and the kind of modem that you have. Obviously, you do not want PPP to be using your telephone continuously, so you need to write a script which will switch the service on when needed and off when not needed. Let's take a look at such a script.

```
#!/bin/sh

# set path to pppd and chat if necessary

pppd connect 'chat -v ABORT BUSY ABORT "NO CARRIER" \
"" ATDTteleph-num CONNECT "" ">" username ">" passwd' \
-detach debug defaultroute netmask 255.255.254.0 noipdefault \
:129.240.254.11 ipcp-accept-local ipcp-accept-remote -vj \
proxyarp /dev/ttyS0 38400
```

The above script is supposed to be typed in on a single line. The backslash characters at the ends of the lines above are shell continuation characters which mean that the following lines are to be treated as one.

- This script is just an example, it is unlikely that you will be able to use it directly. Instead, you should obtain instructions from the local service you are dialing.
- In the `chat` part, double quoted strings such as `">"` are text which the modem can expect to receive and should be waited for. These are generally prompts. On other systems they might be strings like `` `Username:' `` and `` `Password:' ``.
- The username, password and telephone number and netmask values relate to the remote system.
- The device `` `/dev/ttyS0' `` is the serial device. On some systems this is called `` `/dev/modem' ``, on others it will be called something else. You will need to find out.
- The final number is the modem speed at which the connection is to be tried. There is no harm in setting an optimistic transfer rate here, since modems negotiate their own transfer rates according to what is possible at the time of connection. This is the value they start at, and if it does not succeed the modems will try successively slower rates until a working connection is established.

Maintenance procedures

Manicures, make-up and hairdos...

One of your main jobs as a system administrator is to maintain the system. This includes adding new information to databases, upgrading and installing software, adding users and tidying up when disks get full...to name just a few things. In the next chapter we'll look at a tool which will help you to automate some of these procedures: here we describe what you need to know before you can do that. Remember that, when ever you wield the power of the system administrator, you can cause great damage. Work carefully and test your changes: do not just assume that they will work. Even small changes are often followed by hundreds of mail messages from angry users because of something you broke while doing your job.

Rule five: *Never make changes just before you go on holiday for four weeks.*

Configuring DNS/BIND

The name service must be configured in order for a system to be able to look up hostnames and internet addresses. The most important file in this connection is `` `/etc/resolv.conf' ``. Ancient IRIX systems seem to have placed this file in `` `/usr/etc/resolv.conf' ``. This old location is extremely obsolete. Without the resolver configuration file, a host will probably stop dead whilst trying hopelessly to look up internet addresses. The most important features of this file are the definition of the domain-name and a list of nameservers which can perform the address translation

service. These nameservers must be listed as IP numerical addresses (since DNS can't look up any names until it knows the name of a server to look them up on, and that's what we're trying to do, nez pah?). The format of the file is as shown in the box below.

```
domain iu.hioslo.no
nameserver 128.39.89.10
nameserver 158.36.85.10
nameserver 129.241.1.99
```

As you should know, DNS has several competing services. A mapping of hostnames to IP addresses is also performed by the `/etc/hosts' database. Moreover, network version of this file can also be shared using NIS or NISPLUS. As a result, most Unix variants allow you to choose the order in which these competing services are given priority when looking up hostname data. Unfortunately there is no standard way of configuring this. Linux and public domain resolver packages for old SunOS (resolv+), use a file called `/etc/hosts.conf'. The format of this file is

```
order hosts,bind,nis
multi on
```

This example tells the lookup routines to look in the `/etc/hosts' file first, then to query DNS/BIND and then finally look at NIS. The resolver routines quit after the first match they find, they do not query all three databases every time.

Solaris uses a file called `/etc/nsswitch.conf' which is a general configuration for all database services, not just the hostname service.

```
#
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
transports.

passwd:     files
group:      files
hosts:      files dns
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
```

```
bootparams: files
```

The print queue

@hrule @vskip 0.3cm

BSD print queue

`lpr -p *printer* `file''
: Send file to named print queue.

`lpq'
: Show the printer queue for the default printer, or the printer specified in the environment variable `PRINTER` if this is set. This lists the queue-ids.

`lprm *queue-id*'
: Remove a job from the print queue. Get the queue id using `lpq`.

`lpd'
: Start the print service. (Must be killed to stop again)

`lpc'
: An incredibly stupid user interface for print administration. This program tells lies.

SysV print queue

`lp -d *printer file*'
: Send a file to the named print queue.

`lpstat -o all'
: Show the printer queue for the default printer. This lists the queue-ids.

`lpstat -a'
: Tells lies about when the print service was started.

`lpsched'
: Start the print service

`lpshut'
: Stop the print service.

`cancel *queue-id*'
: Remove a job from the print queue. Get the queue id using `lpstat`.

The solaris operating system has an optional printing system called Newsprint in addition to the SVR4 printing commands.

Building a kernel

@hrule @vskip 0.3cm

In the past it was necessary to recompile the operating system kernel for every new device you wanted to add to a unix system The kernel was an enourmous statically linked program.

In modern unix systems, the kernel is divided up into modules which can be loaded and reloaded as needed. Under system 5 (SVR4) unix, you can reconfigure some aspects of the kernel online without having to stop the system.

Solaris (SVR4)
: Edit the file `` `/etc/system' `` to change kernel parameters. Reboot if necessary with the command

```
reboot -- -r
```

: which reconfigures the system automatically.

Linux
: The linux kernel is subject to more frequent revision than other systems. It must be recompiled when new changes are to be included, or when an optimzed kernel is required. Change directory to `` `/usr/src/linux' ``. The command `make config` can be used to set kernel parameters. A more user friendly windows based program `make xconfig` is also available.

These days it is often unnecessary to build custom kernels. The default kernels supplied with many OSes are good enough. Still, performance enhancements are obtainable, particularly on busy servers.

Adding a new user

@hrule @vskip 0.3cm

Having set up a network host you will want to create user accounts on it. When the system is new, only one user account is set up: the superuser root. The first thing you should do is to set a password for root using the `passwd` command.

You should never use the root account for normal usage; use it only for system maintenance. The root account has so much power (it has the potential to destroy the system) that you cannot trust yourself not to do something silly while wielding this power. If you think that you can trust yourself, then you will have to learn the hard way.

Many systems provide shell scripts or user interfaces for installing new users, but most of these scripts will be useless to you, because they will have different ideas about how the system should work than you. Also, you will probably want to use some system for centralizing passwords, so that each user has the same password on every host on your network.

To add a new user to your local system you have to do the following.

- Find a unique username and user-id (uid number) for the new user.
- Add a line to the file `/etc/passwd' on your machine (or if you are centralizing passwords, on a server) for the new user.
- Create a login directory (home directory) for the user.
- Choose a shell for the user.
- Copy some configuration files like `.cshrc' or `.profile' into the new user's directory.

You will almost certainly want to write your own script for adding new users.

Configuring a user environment

@hrule @vskip 0.3cm

When a new user logs in for the first time, he or she expects the new account to work straight away. Printing should work, programs should work and there should be no strange error messages about files not being found or programs not existing. Most users want to start up a window environment.

Unix is about the ability to customize. Everything in UNIX is configurable and advanced users like to play around and many create their own setups, but there will also be a lot of users who do not want to do this. As system administrator, it is your job to make sure that everything works properly with acceptable defaults, right from the start. Here is a checklist for configuring a user environment

`.cshrc'
: If the default shell for users is a C shell or derivative, then you need to supply a default `read commands' file for this shell. This should set a path and a terminal type and any environment variables which your local system requires.

`.profile'
: If the default shell is a Bourne-again shell like `bash` or `ksh`, then you will need to supply this file to set a `PATH` variable and terminal type and any environment variables which your system requires.

`.xsession'
: This file specifies what windows and window manager will be used when the X-windows system is created. It is a shell script which should begin by setting up applications in the background (with a `&' symbol after them) and end up `exec`-ing a window manager in the foreground. If the window manager is called as a background process, the script will be able to exit immediately and users will be logged out immediately. Some older systems use an outdated file called `.xinitrc' but this file is officially obsolete. The official way to start the X11 window system is through the `xdm` program, which provides a login prompt window. For some reason, many distributions of Linux seem to encourage the use of the obsolete command `startx` which starts the X windows system from a tty-

shell. The older `startx` system used the `.xinitrc' file, whereas `xdm` uses `.xsession'. Most linuxes hack this so that one only needs a `.xsession' file.

`.mwmrc'
: This file configures the default menus etc for the `mwm` window manager.

`.fvmwrc'
: This file customizes the behaviour of the `fvwm` window manager.

`.fvwm2rc'
: This file customizes the behaviour of the `fvwm2` window manager.

`.fvwm95rc'
: This file customizes the behaviour of the `fvwm95` window manager. This is a mock windows-95 interface.

`.tkgrc'
: If you are running the TkGoodStuff toolbar as a part of `fvwm2` then you might like to create a default toolbar setup for users.

Executing jobs at regular times

@hrule @vskip 0.3cm

Unix has a time daemon called `cron`: it's chronometer. Cron reads a configuration file called a `crontab' file which contains a list of shell-commands to execute at regular time intervals. On modern UNIX systems, every user may create and edit a crontab file using the command

```
crontab -e
```

This command starts a text editor allowing the file to be edited. The contents of a user's crontab file may be listed at any time with the command `crontab -l`. The format of a crontab file is a number of lines of the form

```
minutes 0-59 hours 0-23 day 1-31 month 1-12 weekday Mon-Sun Bourne-shell-command
```

An asterisk or star `*' may be used as a wildcard, indicating `any'. For example:

```
# Run script every weekday morning Mon-Fri at 3:15 am:

15 3 * * Mon-Fri /usr/local/bin/script
```

A typical root crontab file looks like this:

```
#
# The root crontab should be used to perform accounting data
collection.
#
```

```
0 2 * * 0,4 /etc/cron.d/logchecker
5 4 * * 6   /usr/lib/newsyslog
0 0 * * *   /usr/local/bin/cfwrap /usr/local/bin/cfdaily
30 * * * *  /usr/local/bin/cfwrap /usr/local/bin/cfhourly
```

The first line is executed at 2a.m. on Sundays and Wednesdays, the second at 4:05 on Saturdays; the third is executed every night at 00:00 hours and the final line is executed every half hour.

In old BSD 4.3 unix, it was only possible for the system administrator to edit the crontab file. In fact there was only a single crontab file for all users, called `/usr/lib/crontab' or `/etc/crontab'. This contained an extra field, namely the username of under which the command was to be executed. This type of crontab file is largely obsolete now, but may still be found on some older BSD 4.3 derivatives such as DEC's ULTRIX.

```
0,15,30,45 * * * * root    /usr/lib/atrun
00 4 * * *         root    /bin/sh /usr/adm/daily 2>&1 | mail daily-
reports
30 3 * * 6         root    /bin/sh /usr/adm/weekly 2>&1 | mail daily-
reports
30 5 1 * *         root    /bin/sh /usr/adm/monthly 2>&1 | mail daily-
reports
```

Installing software

@hrule @vskip 0.3cm

Unlike many other systems, UNIX has grown up around people who write their own software rather than relying on off-the-shelf products. The internet contains gigabytes of software for UNIX systems which is completely free. Large companies like the oil industry and newspapers can afford off-the-shelf software for UNIX but most people can't.

There are therefore two kinds of software installation: the installation of commercial software and the installation of freeware. Commercial software is usually installed from a CD by running a simple script and by following the intructions carefully; the only decision you need to make is where you want to install the software. Freeware usually comes in an uncompiled form and must therefore be compiled. UNIX programmers have gone to great lengths to make this process as simple as possible for system administrators.

Rule six: *Be tidy: don't drop your garbage on the OS filespace, place it in the receptacle provided...`/usr/local'*

Structuring software

The first step in installing software is to decide where you want to keep it. You could put the software anywhere you like, but you should bear in mind the following:

- You should keep installed software separate from the operating system installed files, so that the OS can be reinstalled or upgraded without ruining your software installation.
- Compiled software should be grouped together, with a `bin' directory and a `lib' directory so that binaries and libraries conform to the usual UNIX conventions. This makes the system more consistent and easier to understand and it also makes it easier to administer the PATH for user-commands is programs are collected in only a few places.
- You should try to keep files and programs which are special to your site separate from files which could be used anywhere.

The directory traditionally chosen for installed software is called `/usr/local'. One then makes sub-directories `/usr/local/bin' and `/usr/local/lib' and so on.

UNIX has a de-facto naming standard for directories which you should try to stick to, so that others will understand how your system is built up.

`bin'
: Binaries or executables for normal user programs.

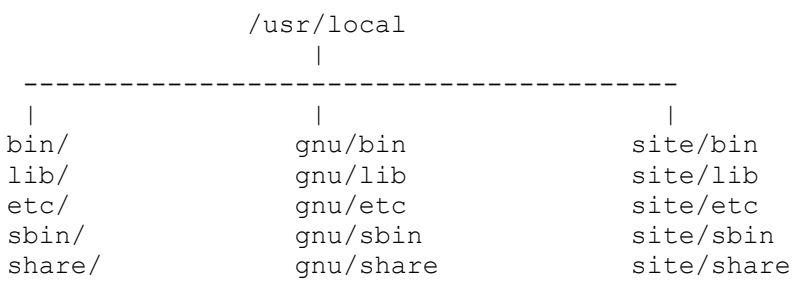
`sbin'
: Binaries or executables for programs which only system administrators require. Those files in `/sbin' are often statically linked to avoid problems with libraries which lie on umounted disks during system booting.

`lib'
: Libraries and support files for special software.

`etc'
: Configuration files.

`share'
: Files which might be shared by several programs or hosts. For instance, databases or help-information; other common resources.

Here is one suggestion for structuring installed software.



We divide these into three categories: regular installed software, GNU software and site-software. The reason for this is as follows:

- `/usr/local' is the traditional place for software which does not belong to the OS. You could keep everything here, but you will end up installing a lot of software after a while, so you might like to create two other sub-categories.
- GNU software, which is written by the Free Software Foundation, forms a self-contained set of tools which replace many of the older UNIX equivalents, like `ls` and `cp`. GNU software has its own system of installation and set of standards. GNU will also eventually become an operating system in its own right, and should therefore be kept separate.
- Site specific software includes programs and data which you build locally to replace the software or data which follows with your operating system. It also includes special data like the database of `aliases' for e-mail and the DNS tables for your site. Since it is special to your site, you should keep it separate so that you can back it up separately and you always know where to find site-specific stuff.

When you are installing software, you are expected to give the name of a *prefix* for installing the package. The prefix in the above cases is `/usr/local' for ordinary software, `/usr/local/gnu' for GNU software and `/usr/local/site' for site specific software. Most software installation scripts place files under `bin' and `lib' automatically.

To begin compiling software, you should always start by looking for a file called `README' or `INSTALL'. This will tell you what you have to do to compile and install the software. In most cases, you will only have to type a couple of commands, like in the following example.

GNU software example

Let us now illustrate the GNU method of installing software which most others have since copied. The steps are as follows

> Collect the software package by `ftp` from a site like `ftp.uu.net` or `ftp.funet.fi`. Use a program like `ncftp` for painless anonymous login.

- Unpack the file using `tar zxf` *`software`*`.tar.gz`.
- Enter the directory which is unpacked, `cd` *`software`*.
- Type: `configure --prefix=/usr/local/gnu`
- Type: `make`
- If all goes well, type `make install`. This should be enough to install the software.
- Some installation scripts leave files with the wrong permissions so that ordinary users cannot access the files. You might have to check that the files have a mode like 755 so that normal users can access them.

This procedure should be more or less the same for just about any software pick up. Older software packages sometimes provide only Makefiles which you must customize yourself. Some X11 based windowing software requires you to use the `xmkmf` X-make-makefiles command instead of configure. You should always look at the README file.

Installing shared libraries

Systems which use shared libraries or shared objects sometimes need to be reconfigured when new libraries are added to the system. This is because the names of the libraries are cached to fast access. The system will not look for a library if it is not in the cache file.

SunOS (not solaris)
: After adding a new library, you must run the command `ldconfig` *`lib-directory`*. The file `` `/etc/ld.so.cache' `` is updated.

Linux
: You can add new library directories to the file `` `/etc/ld.so.conf' ``. Then run the command `ldconfig`. The file `` `/etc/ld.so.cache' `` is updated.

How often should I upgrade?

Some software (especially free software) gets updated very often. You could easily spend your entire life just chasing the latest versions of your favourite software packages. Don't!

- It is a waste of your time.
- Sometimes new versions contain more bugs than the old one, and an even-newer-version is just around the corner.
- Users will not thank you for changing things all the time. Stability is a virtue. Everyone likes time to get used to the system before you change it!

Rule Seven: *Relax and keep your head. Don't feel like you have to have the latest fashion.*

Installing a new service

@hrule @vskip 0.3cm

Sooner or later you will find yourself installing software which requires it's own network service to be set up. You need to configure the system to accept a new service by editing the file `` `/etc/services' ``. This file contains the names of services and their protocol types and port numbers.

The format of entries is like this:

```
service      portnumber/protocol    aliases

pop3            110/tcp         postoffice
bootp           67/udp
cfinger         2003/tcp
```

There are two ways in which a service can run under unix: one is that a daemon runs all the time in the background, handling connections. This method is used for services which are used often. Another way is to start the daemon only when an outside connection wishes to use it; this method is used for less frequently used services. In the second case, a master `internet' daemon is used, which listens for connections for several services at once and starts the correct daemon only long enough to handle one connection. The aim is to save the overhead of running many daemons.

If you want to run a daemon all the time, then you just need to make sure that you start the daemon in the appropriate `rc' start-up files for the system. To add the service to the internet daemon, on the other hand, you need to add a line of the following form the the configuration file `/etc/inetd.conf'.

```
# service type protocol threading user-id server-program server-command

pop3     stream tcp nowait root /local/etc/pop3d        pop3d
cfengine stream tcp nowait root /local/iu/bin/cfpush    cfpush -x -c
```

The software installation instructions will tell you what you should add to this file.

Once you have configured a new service, you must start it by running the appropriate daemon, See section Summoning daemons.

Mail queue

@hrule @vskip 0.3cm

When mail cannot be delivered immediately it is placed in the mail queue. To see what mail is waiting in the mailqueue, you must log into the mailserver (i.e. the host which handles outgoing mail) on your network. You can see what is in the queue by typing

```
  nexus% mailq
```

or

```
  nexus% sendmail -bp
```

These two forms are equivalent. You can force sendmail to process the mail queue by typing:

```
nexus% sendmail -q -v
```

Mail aliases

@hrule @vskip 0.3cm

One of the first things you should locate on your system is the `sendmail` alias file. This is a file which contains e-mail aliases for users and system services. Common locations for this file are `` `/etc/aliases' `` and `` `/etc/mail/aliases' ``. On some systems, the mail aliases are in the NIS network database.

If this file actually lies in the `` `/etc' `` directory, or some other place amongst the system files, then you should move it to your special area for site-dependent files and make a symbolic link to `` `/etc/aliases' `` instead. Your mail aliases are valuable and you want to make sure that nothing happens to them if you reinstall the OS.

The format of the mail aliases file is as follows:

```
# Alias for mailer daemon; returned messages from our MAILER-DAEMON
# should be routed to our local Postmaster.

postmaster: mark, toreo

MAILER-DAEMON: postmaster

nobody: /dev/null

#
# alias: list of addresses
#

drift:mark@iu.hioslo.no,toreo@iu.hioslo.no
root:mark@iu.hioslo.no,toreo@iu.hioslo.no

#
# Alias for distribution list, members specified elsewhere:
# alias: :include:file of names
#

lsk: :include:/iu/nexus/u2/wiikl/www/lsk/maillist/maillist

#
# Dump mail to a file
#
```

ace: /iu/nexus/local/iu/archive/ACEmailinglist

Controlling diskspace

@hrule @vskip 0.3cm

Disks fill up at an alarming rate. Users almost never throw away files unless they have to. If you are lucky enough to have only experienced and friendly users on your system, then you just have to ask them nicely to tidy up their files once in a while and all will be well. Most administrators do not have this luxury however. Most users never think about the trouble they might be causing others by keeping lots of junk around. And then there is the user from hell who fetches hundreds of megabytes of nonsense from the internet every week and never tidies up even when asked.

To keep your system alive, you might want to limit the amount of disksapce users can have access to, or you might want to automatically delete certain files at regular intervals. For instance, when a program crashes, the unix kernel dumps its image to disk in a file called `core'. These files crop up all over the place and have no useful purpose except for majorly serious goggle eyed geeks. To others, they are just fluff on the upholstry and should be removed pronto! A lot of free disk space can be claimed by deleting these files. Many users will not delete them themselves however, because they do not even understand why they are there.

Disk quotas mean that users have a hard limit to the number of bytes they are allowed to use on the disk. They are an example of a more general concept known as system accounting whereby you can control the resources used by any user, whether they be number of printed pages sent to the printer or the number of bytes written to the disk. Disk quotas have advantages and disadvantages.

- The advantage is that users really cannot exceed their limits. There is no way around this.
- Disk quotas are very restrictive and when you hit your limit you do not often understand what has happened. Usually you will not get a message unless you are logging in. Quotas also prevent users from creating large temporary files which can be a problem when compiling programs. They also carry with them a system overhead, which makes everything run a little slower (though perhaps not very noticably these days).

In general, if you can avoid using disk quotas, do so. They are a nuisance.

Deciding whether to delete files automatically is a tough decision. It might seem a bit fascist to just delete a bunch of files without asking. On the other hand, this is often the only way to ever clear anything up. You have to assume that most users will be happy if they do not have to think about it themselves. You just have to be careful to never delete a file which cannot be regenerated or re-acquired if necessary.

A useful procedure is to only delete files (even nonsense files) if they have not been used for more than a week.

Some files can be safely deleted: netscape cache files, compilation by-products like `.o' files. These files are large but most people will not miss them.

The system robot `cfengine` can be used to tidy files in a cautious way, See section Using cfengine.

Setting clocks

@hrule @vskip 0.3cm

You need to keep your system clock synchronised. A time-server is used for this purpose. The network time protocol daemon `ntpd` is used to synchronize clocks from a reliable time server.

Another option for BSD-like systems is the `rdate` command, which sets the local clock according to the clock of another UNIX host. Some systems (like Linux) do not have this command, but it can be emulated with a script like this one, provided remote-shell access is permitted by the server.

```
#!/bin/sh
#
# Fake rdate script for linux - requires rsh access on server
#

echo Trying time server

DATE=`/bin/su -c '/usr/bin/rsh time-server date' remote-user`

echo Setting date string...

/bin/date --set="$DATE"
```

Another more reliable way of keeping clocks synchronized in the long run is the use the NTP protocol, or network time protocol. The daemon `xntpd` can be used to synchronize hosts against a master host. Two configuration files are needed to set up this service: `/etc/ntp.conf' and `/etc/ntp.drift'. `/etc/ntp.conf' looks something like this, where the IP address is that of the master time server, whose clock you trust.

```
driftfile /etc/ntp.drift
authdelay 0.000047
server 128.39.89.10
```

The `/etc/ntp.drift' file must exist, but its contents are undetermined. you should touch this file.

Posix ACLs

@hrule @vskip 0.3cm

ACLs, or access control lists are as modern replacement for file modes and permissions. With access control lists you can specify precisely the access rights to files for each user individually. Earlier in Unix systems, it was necessary to make a group if you wanted to open a file for several users, but not for everyone.

ACLs were invented in DOMAIN OS by Apollo, and were then copied by Novell, HP and other vendors. A POSIX standard for ACLs has been drafted, but as of today there is no standard for ACLs and each vendor has a different set of incompatible commands and datastructures. Sun Microsystem's solaris (NFS3) imlementation is based on the POSIX draft. We shall follow Solaris ACLs in this section. Not all Unix systems have ACLs. Linux does not have them. If you grant access to a file which is shared on the network to a machine which doesn't support them, they will be ignored.

ACLs are literally lists of access rights. Each file has a list of data structures with pairs of names and permissions:

```
                `Filename'

        Who                     Permission

  (user|group|everyone)   (read|write|execute)
```

You specify an ACL by saying what permissions you would like to grant and which user or group of users the permissions apply to. An ACL can grant access or deny access to a specific user. Because of the amount of time required to check all the permissions in an ACL, ACLs slow down file search operations.

Under solaris, the commands to read and write ACLs have the horrible names

`getfacl` *file*
: Examine the ACLs for a file

`setfacl` *file* `-s` *permission*
: Set ACL entries for a file, replacing the entire list.

`setfacl` *file* `-m` *permission*
: Set ACL entries for a file, adding to an existing list.

For example. If we create a new file, it ends up with a default ACL which is based upon the unix `umask` value. Suppose `umask` is 077, giving minimal rights to others.

```
dax% touch testfile

dax% getfacl testfile

# file: testfile
# owner: mark
# group: iu
user::rw-
group::---              #effective:---
mask:---
other:---
```

This tells us that a new file is created with read/write permission for the user (owner) of the file, and no other rights are granted. To open the file for a specific user, one writes

```
dax% setfacl -m user:ds:rw- testfile

dax% getfacl testfile

# file: testfile
# owner: mark
# group: iu
user::rw-
user:ds:rw-             #effective:---
group::---              #effective:---
mask:---
other:---
```

To open a file for reading by a group `iu`, except for one user called `robot`, one would write:

```
dax% setfacl -m group:iu:r--,user:robot:--- testfile

dax% getfacl testfile

# file: testfile
# owner: mark
# group: iu
user::rw-
user:robot:---          #effective:---
user:ds:rw-             #effective:---
group::---              #effective:---
group:iu:r--            #effective:---
mask:---
other:---
```

Notice that this is accomplished by saying that the group has read permission whilst the specific user should have no permissions.

Are you awake?

@hrule @vskip 0.3cm

HEY WAKE UP! *How much of the following section did you absorb? Did you read it through and memorize all the information or did it go right through and never touch the grey stuff? Either way, you did it wrong! You should read through once so that you know what is there and then remember where you can find the information in the future! Don't waste your time trying to remember everything. You'll go crazy.*

Using cfengine

Constructing your personal organizer, programming the vacuum cleaner and teaching the dog to drive.

Many of the routine maintenance jobs you are required to do to keep your system running smoothly can be automated by a configuration robot called cfengine. Cfengine, or the configuration-engine, can set up your system from scratch and can watch over the system as time goes on, reporting about and even fixing errors.

Rule eight: **Adopt a philosophy of prevention! Many sticky situations could be avoided with a little family planning.**

What is cfengine?

System maintenance involves a lot of jobs which are repetitive and menial. In the beginning, you might feel that it is cool to be humble and menial, thereby identifying with the common man or woman in an act of selfless love for humanity. After all, a little hard work never hurt anyone. But after a while you will realize that bashing away at those humble tasks on the keyboard ruins your nail polish and that sitting too long in front of the cathode tube makes your hair stand on end, not to mention all the split ends from the stress.

Fortunately UNIX is forgiving in this respect, and ever adaptable. There are half a dozen languages and tools for writing programs which will automatically check the state of your system and perform a limited amount of routine maintenance automatically, while you're under the drier.

Cfengine is one such tool which has acquired wide acceptance on the net. It is a very high level *language* (much higher level than shell or Perl) and a *robot* for interpreting your programs and implementing them. Cfengine is a general tool for structuring, organizing

and maintaining information system on a network. Because it is general, it does not try to solve every little problem you might come across, instead it provides you with a framework for solving all problems in a consistent and organized way. Cfengine's strength is that it encourages organization and consistency of practice--also that it may easily be combined with other languages.

Cfengine is about (i) defining the way you want all hosts on your network to be set up (configured), (ii) writing this in a single `program' which is read by every host on the network, (iii) running this program on every host in order to check and possibly fix the setup of the host. Cfengine programs make it easy to specify general rules for large groups of host and special rules for exceptional hosts. Here is a summary of cfengine's capabilities.

- Check and configure the network interface on network hosts.
- Edit textfiles for the system or for all users.
- Make and maintain symbolic links, including multiple links from a single command.
- Check and set the permissions and ownership of files.
- Tidy (delete) junk files which clutter the system.
- Systematic, automated (static) mounting of NFS filesystems.
- Checking for the presence or absence of important files and filesystems.
- Controlled execution of user scripts and shell commands.
- Process management.

By automating these procedures, you will save a lot of time and irritation, and make yourself available to do more interesting work.

A cfengine program is probably not like other programming languages you are used to. It is more like a Makefile. Instead of using low-level logic, it uses high-level classes to make decisions. Actions to be carried out are not written in the order in which they are to be carried out, but listed in bulk. The order in which commands are executed is specified in a special list called the *action-sequence*. A cfengine program is a free-format text file, usually called `cfengine.conf' and consisting of declarations of the form.

```
action-type:

    classes::

         list of actions
```

The action type tells cfengine what the commands which follow do. The action type can be from the following list.

```
binservers
```

```
broadcast
control
copy
defaultroute
directories
disable
editfiles
files
groups
homeservers
ignore
import
links
mailserver
miscmounts
mountables
processes
required
resolve
shellcommands
tidy
unmount
```

You may run cfengine scripts/programs as often as you like. Each time you run a script, the engine determines whether anything needs to be done -- if nothing needs to be done, nothing is done! If you use it to monitor and configure your entire network from a central file-base, then the natural thing is to run cfengine daily with the help of `cron`.

Cfengine configurations can save you an enourmous amount of time by freeing you from repetitive tasks. Finally you run your system chauffeur driven with your own programmable dog. Totally excellent.

The simplest way to use cfengine

The simplest cfengine configuration you can have consists of a control section and a shellcommands section, in which you collect together scripts and programs which should run on different hosts or hosttypes. Cfengine allows you to collect them all together in one file and label them in such a way that the right programs will be run on the right machines.

```
control:

   domain = ( mydomain )

   actionsequence = ( shellcommands )

shellcommands:

   # All linux machines
```

```
    linux::
            "/usr/bin/updatedb"

    # Just one host

    myhost::

             "/bin/echo Hi there"
```

While this script does not make use of cfengine's special features, it shows you how you can control many machines from a single file. Cfengine reads the same file on every host and picks out only the commands which apply.

A simple file for one host

Although cfengine is designed to organize all hosts on a network, you can also use it just on a single stand-alone host. In this case you don't need to know about classifying commands.

Let's write a simple file for checking the setup of your system. Here are some key points:

- Every cfengine must have a `control:` section with an `actionsequence` list, which tells it what to do, and in which order.
- You need to declare basic information about the way your system is set up. Try to keep this simple.

```
#!/usr/local/gnu/bin/cfengine -f
#
# Simple cfengine configuration file
#

control:

   actionsequence = ( checktimezone netconfig resolve files
shellcommands )

   domain         = ( iu.hioslo.no )
   netmask        = ( 255.255.255.0 )
   timezone       = ( MET )

#########################################################################

broadcast:

  ones

defaultroute:

   cadeler30-gw
```

```
#########################################################################

resolve:

   #
   # Add these name servers to the /etc/resolv.conf file
   #

   128.39.89.10  # nexus
   158.36.85.10  # samson.hioslo.no
   129.241.1.99

#########################################################################

files:

   /etc/passwd mode=644 owner=root action=fixall

#########################################################################

shellcommands:

   Wednesday||Sunday::

      "/usr/local/bin/DoBackupScript"
```

A file for multiple hosts

If you want to have just a single file which describes all the hosts on your network, then you need to tell cfengine which commands are intended for which hosts. Having to mention every host explicitly would be a major hassle in a young lady's life, like stressful okay? Usually though, we are trying to make hosts on a network basically the same as one another, as far as possible, but there will still be a few obvious differences which need to be accounted for.

For example, the Solaris operating system is quite different from the Linux operating system, so some rules will apply to all hosts which run Solaris, whereas others will only apply to Linux. Cfengine uses classes like `solaris::` and `linux::` to label commands which apply only to these systems.

We might also want to make other differences, based not on operating system differences but on groups of hosts belonging to certain people, or with a special significance. We can therefore create classes using groups of hosts.

```
  groups:

     Rivals = ( girl1 girl2 )
```

```
     Seriously_Uncool_Losers = ( nerd1 nerd2 nerd3 )
```

This defines a class called `Rivals` which refers collectively to the hosts `girl1` and `girl2`, as well as a class called `Seriously_Uncool_Losers` which refers to any or all of the hosts `nerd1`, `nerd2`, `nerd3`.

Other ways to label commands (other classes) include *times*: days of the week, month or year, hours and minutes of the day, the names of hosts, specific operating system information and special classes which become switched on in response to certain situations which happen on the network. Here are some example classes; look at the cfengine documentation at http://www.iu.hioslo.no/cfengine for more information.

```
 Friday::    # Day of the week
 Hr15::      # If the time is 15:xx
 Min5_10::   # If between five and ten past the hour
 Day19::     # 19th day of the month
 September:: # Month
 Yr1997::    # Year
 32_bit::    # System architecture
 solaris::   # operating system
```

We may now label actions by these classes to restrict their scope:

```
   editfiles:

      solaris::

         { /etc/motd

         PrependIfNoSuchLine "Plan 9 was a better movie and a better
OS!"
         }

      Rivals::

         { /etc/motd

         AppendIfNoSuchLine "Your rpc.spray is so last month"
         }

      Seriously_Uncool_Losers::

         { /etc/motd

         AppendIfNoSuchLine "Please keep your finger commands to
yourself!"
         }
```

Actions or commands which work under a class operator like `solaris::` are only executed on hosts which belong to the given class. This is the way one makes decisions in cfengine: by class assignment rather than by `if..then..else` clauses.

General guidelines for using cfengine

Setting up a network configuration takes a long time. Even with a tool like cfengine, you will be changing things all the time. The main strength of cfengine is that it allows you to make changes without losing sight of what you have done. Here are some tips to bear in mind while putting together your configuration.

- Get hold of the cfengine manual and read it.
- Start writing your configuration from the earliest stages of setting up your network. A configuration file takes a long time to develop and it is best to do a little at a time.
- Add to your file a little at a time so that you always understand and remember what is in your configuration.
- Begin with network services and configuration, then add file checking and symbolic links. Use `tidy` and `disable` only when you know that cfengine is working the way it should.
- Avoid mounting filesystems from host A onto host B and filesystems from host B onto host A. This can lead to NFS deadlocks.
- If you are thinking of linking files like the `` `/etc/passwd' `` file to another disk where you keep all of your site data -- don't! When a host boots, it normally has access only to the root file system (no other disks), so if you link a critical file to disk which isn't available booting will fail and you might have to reinstall the system! Do not expect all disks to be mounted during the whole boot process.

We do not have the space in this introduction to go into cfengine in great detail. You should look at the manual yourself if you find yourself in charge of a network.

Key systems and services

Starting your own community services.

Summoning daemons

Far be it for this Young Lady's Illustrated Primer to encourage ladies engage themselves in the mysteries of dark forces and black magic (except perhaps the chocolate variety). Daemonic rituals have generally been the domain of unix-wizards. But wake up! This is one of those feminist issues. Sometimes a girl just has to get her hands dirty.

Network services are run by daemons. Once you have done the deed of configuring a network service, See section Installing a new service, by editing some textfiles and

chanting some dark rites and maybe sacrificing a few doughnuts, you reach the point where you have to actually start the daemon in order to see the fruits of those labours. There are two ways to start network daemons:

- When the system boots, by adding an appropriate shellcommand to one of the system's startup scripts. When you use this method, the daemon hangs around in the background all the time waiting for connections.
- On demand: that is, only when a network request arrives. You use the `inetd` daemon to monitor requests for your new service and it starts the daemon to handle requests on a one-off basis. Not all services should be started in this way. You should normally follow the guidelines in the documentation for the service that you are running

The behaviour of unix systems at boot-time is very far from being standard. Older unix systems use a series of scripts called `` `/etc/rc*' `` (short for `read commands'). On such system one normally finds a file called `` `/etc/rc.local' `` where it is possible to add your own commands. Newer unix varieties use a program called `init` and a series of run-levels to control what happens when the machine boots, See section Booting unix.

System 5 init

The SVR4 version of the `init` program is attaining som popularity and is used by several Linux distributions, such as Debian and Redhat. The idea with this program is to start the system in one of a number of run-levels. Runs levels decide how many services will be started when the system boots. The minimum level of operation is single user mode, or run level `s'. Full operation is usually run level 2 or 3, depending on the system you are using. (NB: be sure to check this!) When entering a run level, init looks in a directory called `` `/etc/rc?.d' `` and executes scripts in this directory. For instance, if we are entering run-level 2, init would look in the directory `` `/etc/rc2.d' `` and execute scripts lying there in order to start necessary services for this run-level. All you have to do to add a new service is to make a new file here which conforms to init's simple rules. The files in these directories are usually labelled according to the following pattern:

```
Snumber-function

Knumber-function
```

Files beginning with S are for starting services and files beginning with K are for killing them again when the system is halted. The number is used to determine the order in which the scripts are read. It does not matter if two scripts have the same number. Finally the function tells us what the script does.

Each script is supposed to accept a single argument, the word `start' or the word `stop'. Let's consider an example of how we might start the `httpd` daemon using init. Here is a checklist:

1. Determine the correct run-level for the service. Let us suppose that it is run level 2.
2. Choose an unused filename, say `S99http'.
3. Create a script accepting a single argument:

```
#!/bin/sh

case $1 in

  start) /usr/local/bin/httpd -d /usr/local/lib/httpd ;;

  stop)  kill `cat /usr/local/lib/httpd/logs/httpd.pid` ;;

  *) echo Syntax error starting http

esac
```

The advantage of this system is that software packages can be added and removed transparently just by adding or removing a file. No special editing is required as is the case for BSD unix.

BSD init

The BSD style `init` program is rather simple. It starts executing a shell script called `/etc/rc' which then generally calls other child-scripts. These scripts start important daemons and configure the system.

To add your own local modifications, you have to edit the file `/etc/rc.local'. This is a Bourne shell script.

The BSD approach has a simpler structure than the system 5 inittab directories, but it is harder to manipulate package-wise.

`inetd` configuration

The internet daemon is a *service demultiplexer*. In english, that means that it is a daemon which listens on the network for messages to several services simultaneously. When it receives a message intended for a specific port, it starts the relevant daemon to handle the request just long enough to handle one request. `inetd` saves the system some resources by starting daemons only when they are required, rather than having the clutter up the process table all the time.

The format this file can differ slightly on older systems. The best way to gleaning its format is to look at the entries which are already there. Here is a common example for the format.

```
#
```

```
# Service|type|protocol|wait|user|daemon-file|command line
#
ftp     stream tcp nowait root   /usr/sbin/in.ftpd      in.ftpd
telnet  stream tcp nowait root   /usr/sbin/in.telnetd   in.telnetd
finger  stream tcp nowait nobody /local/etc/in.fingerd  in.fingerd
cfinger stream tcp nowait nobody /local/etc/in.cfingerd in.cfingerd
```

The first column is the name of the service from `/etc/services'. The next column is the type of connection (stream or dgram or tli), then comes the protocol type (tcp/udp etc). The wait column indicates whether the service is to be single or multithreaded, i.e. whether new requests should wait for an existing request to complete or whether a new daemon should be started in parallel. The last two columns contain the location of the program which should handle the request and the actual command line (including options) which should be executed.

To add a new service, you have to edit the file `/etc/inetd.conf' and then send the `inetd` process the HUP signal. To do this, you find the process id:

```
ps aux | grep inetd
```

Then you type:

```
kill -HUP process-id
```

Planning for an efficient network

The efficiency of your network can be improved greatly by planning carefully how you run networks services particularly NFS, NIS and DNS. You should think about:

- Which hosts should have disks and which host(s) should be the NFS disk server.
- Separate users' home directories over several disks and keep problem disk-users on a partition for themselves, away from honest users.
- Which services will you provide. Which hosts should be servers and for which services?
- You will need a binary server (for software) for each operating system type you maintain, since a Sun machine cannot run software compiled on a Linux host etc.
- Do you want to centralize your services on one host, or distribute them across many? The first possibility is easier to administrate, but the second might be more efficient and less prone to host crashes.
- Try not to create disk-deadlocks whereby host A needs to mount host B and host B needs to mount host A.
- Will one server be enough? Do you need a backup server?
- Normally you want to use your most powerful system as the server, but perhaps you want to reserve this for running some especially heavy software, and use a less powerful host as server.

Setting up a nameserver

@hrule @vskip 0.3cm

File structure

The Domain Name Service (DNS) is that internet service which converts fully qualified hostnames (FQHN) like `nexus.iu.hioslo.no` into IP addresses like `128.39.89.10` and vice versa. A FQHN always includes the full name of the host and the domain in which is it registered. The service consists of a database for your local domain together with a daemon `named` or `in.named` which handles lookups in the database.

Recently, the BIND software has been rewritten to solve a number of pressing problems. The resulting version is called BIND 8. Most vendor releases do not incorporate this new BIND as of 1998, but you would do well to get the bind software and install it yourself, if you intend running a nameserver.

Since the mapping of (even fully qualified) hostnames to IP addresses is not one-to-one (a host can have several aliases and network interfaces), the DNS database needs information about conversion both from FQHN to IP address and the other way around. To set up a primary name server, you will need to complete the following checklist.

- Make a directory in your site-dependent files (don't mix it up with your OS files) called `` `dns' `` or `` `named' `` and change to this directory.
- Make subdirectories `` `pz' `` and `` `sz' `` for primary and secondary data. We shall only be using the primary data at this stage. Secondary data are cached files from a different primary nameserver, which act as mirror for backup purposes.
- Find out your domain name (let's supposed it's called *domain.country*) and create a file `` `pz/`*`domain.country`*`' ``. We shall worry about its contents shortly. This file will contain data for converting names into addresses.
- Find out your network numbers and create files for them. The domain `iu.hioslo.no`, for instance, has four networks: `128.39.89.0`, `128.39.73.0`, `128.39.74.0` and `128.38.75.0`. DNS does not use the trailing zero for the network addresses, so create files `` `pz/128.39.89' ``, `` `pz/128.39.73' `` etc., one for each network. These files will contain data for converting addresses into `canonical' names, or official hostnames (as opposed to aliases). We shall call the network files `` `pz/`*`network`*`' ``
- Create a file for the loopback network (everyone needs one of these) `` `pz/127.0.0' ``
- Create a cache file `` `named.cache' `` which will contain the names of the Internet's primary (root) name servers.
- (Old BIND v 4.x) Create a boot configuration file for the nameservice daemon `` `named.boot' ``. We shall later link this file to `` `/etc/named.boot' `` where the daemon expects to find it. We place it here, however, so that it doesn't get lost or destroyed if we should choose to upgrade the operating system.

- (New BIND v 8.x) Create a configuration file `named.conf'. We shall later link this file to `/etc/named.conf' where the daemon expects to find it. We place it here, however, so that it doesn't get lost or destroyed if we should choose to upgrade the operating system.
- Link the boot/conf file to the `/etc' directory, so that it appears to be at the location `/etc/named.boot'. Start the nameservice daemon by typing in.named.

At this stage you should have the following directory structure in your site dependent files.

Names	Example
named/named.boot	named/named.boot (old BIND)
named/named.conf	named/named.conf (new BIND)
named/named.cache	named/named.cache
named/pz/domain.country	named/pz/iu.hioslo.no
named/pz/network	named/pz/128.39.73
	named/pz/128.39.74
	named/pz/128.39.75
	named/pz/128.39.89

The DNS tables also tell E-mail services how to deliver mail. You will need to have a so-called `mail-exchanger' record in the DNS tables in order to tell E-mail which host handles E-mail for the domain. An entry of the form

domain-name MX *priority mailhost*

tells E-mail services that mail sent to *name*@*domain-name* should be routed to the host *mailhost*. For instance,

```
iu.hioslo.no. MX 10 nexus
```

tells our server that mail addresses of the form *name*@iu.hioslo.no should be handled by host nexus. The priority number 10 is chosen at random. Several records can be added with backup servers if the first server does not respond.

Mail records are also possible on a per-host basis. If you want mail sent to host XXX handled by host YYY, you would add a record,

```
XXX MX 10 YYY
```

This would mean that mail sent to *name*@XXX.*domain-name* would be handled by `YYY`. For instance, *name*`@XXX.iu.hioslo.no` would be handled by `@YYY.iu.hioslo.no`.

`Sample named.boot' for BIND v 4.x

The boot file tells the nameservice daemon which files provide information for which networks. The syntax of this file is somewhat bizarre and has its roots in history. It begins with the name of the directory in which you have chosen to store your data. Next it contains a line telling the daemon the name of the cache file. Finally we list all primary and secondary domains and networks. In our case we have only primary data, no mirrored data from other servers. Suffice it to say that the network addresses have to be written backwards for reverse lookup (i.e. IP address to FQHN resolution) and the string `in-addr.arpa` is appended(8). Here is an example file from the primary server at `iu.hioslo.no`.

```
;
; Boot file for primary name server
; Note that there should be one primary entry for each SOA record.
;
; type          domain                          source file or host
;
directory       /usr/local/iu/dns               ; running directory for
named
;
; cache (mandatory). Needed to "prime" name server with startup info so
it
; can reach the root name servers.
;
cache           .                       named.cache
;
; Primary and secondary name/address zone files follow.
;
primary         0.0.127.in-addr.arpa                    pz/127.0.0
primary         73.39.128.in-addr.arpa                  pz/128.39.73
primary         74.39.128.in-addr.arpa                  pz/128.39.74
primary         75.39.128.in-addr.arpa                  pz/128.39.75
primary         89.39.128.in-addr.arpa                  pz/128.39.89
primary         121.39.128.in-addr.arpa
        pz/128.39.121
primary          iu.hioslo.no                             pz/iu.hioslo.no
```

Note that comments are written after a semi-colon in DNS files.

Sample `named.conf' for BIND v 8.x

If you are going to use the new BIND software (recommended if you run a nameserver) then you need to replace the `named.boot' file with a new format file called `named.conf'. The information contained in this file is the same as that for

`named.boot', but many more options can be set in the new file, particularly in connection with logging. Here is a translation of the above `named.boot' file into the new format:

```
options
   {
   directory "/local/iu/dns";
   check-names master ignore;
   check-names response ignore;
   check-names slave warn;
   named-xfer "/local/iu/bind/bin"; /* Location of daemon */
   fake-iquery no;                  /* security */
   notify yes;
   };

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

//
// Primary and secondary name/address zone files follow.
//

zone "0.0.127.in-addr.arpa"
   {
   type master;
   file "pz/127.0.0";
   };

zone "73.39.128.in-addr.arpa"
   {
   type master;
   file "pz/128.39.73";
   };

zone "74.39.128.in-addr.arpa"
   {
   type master;
   file "pz/128.39.74";
   };

zone "75.39.128.in-addr.arpa"
   {
   type master;
   file "pz/128.39.75";
   };

zone "89.39.128.in-addr.arpa"
   {
   type master;
   file "pz/128.39.89";
   };
```

```
zone "iu.hioslo.no"
   {
   type master;
   file "pz/iu.hioslo.no";
   };

// dns.iu.hioslo.no server options

server 192.16.202.11
   {
   transfer-format many-answers;
   };

logging
   {
   channel admin_stuff
       {
       file "/local/iu/logs/admin" versions 7;
       severity debug;
       print-time yes;
       print-category yes;
       print-severity yes;
       };

    channel xfers
        {
        file "/local/iu/logs/xfer" versions 7;
        severity debug;
        print-time yes;
        print-category yes;
        print-severity yes;
        };

    channel updates
        {
        file "/local/iu/logs/updates" versions 10;
        severity debug;
        print-time yes;
        print-category yes;
        print-severity yes;
        };

    channel security
        {
        file "/local/iu/logs/security" versions 7;
        severity debug;
        print-time yes;
        print-category yes;
        print-severity yes;
        };

    category config
        {
        admin_stuff;
        };
```

```
 category parser
     {
     admin_stuff;
     };

 category update
     {
     updates;
     };

 category load
     {
     updates;
     };

 category security
     {
     security;
     };

category xfer-in
     {
     xfers;
     };

category xfer-out
     {
     xfers;
     };

category db
     {
     updates;
     };

category lame-servers
     {
     null;
     };

category cname
     {
     null;
     };
};
```

Sample `named.cache'

The data in the cache file can be retrieved by anonymous ftp from the most illustrious repository of internet information `nic.ddn.mil`. The list of internet root servers (which bind together all internet domains) are listed in a file called `` `netinfo/root-servers.txt' `` The retrieved data have to be written in the following form

```
;
; Compiled from nic.ddn.mil's netinfo/root-servers.txt, as of December
96 (mark)
;

.                       999999  IN      NS      A.ROOT-SERVERS.NET.
.                       999999  IN      NS      B.ROOT-SERVERS.NET.
.                       999999  IN      NS      C.ROOT-SERVERS.NET.
.                       999999  IN      NS      D.ROOT-SERVERS.NET.
.                       999999  IN      NS      E.ROOT-SERVERS.NET.
.                       999999  IN      NS      F.ROOT-SERVERS.NET.
.                       999999  IN      NS      G.ROOT-SERVERS.NET.
.                       999999  IN      NS      H.ROOT-SERVERS.NET.
.                       999999  IN      NS      I.ROOT-SERVERS.NET.

A.ROOT-SERVERS.NET.           999999  IN      A               198.41.0.4
B.ROOT-SERVERS.NET.           999999  IN      A               128.9.0.107
C.ROOT-SERVERS.NET.           999999  IN      A               192.33.4.12
D.ROOT-SERVERS.NET.           999999  IN      A               128.8.10.90
E.ROOT-SERVERS.NET.           999999  IN      A
192.203.230.10
F.ROOT-SERVERS.NET.           999999  IN      A               192.5.5.241
G.ROOT-SERVERS.NET.           999999  IN      A
192.112.36.4
H.ROOT-SERVERS.NET.           999999  IN      A               128.63.2.53
I.ROOT-SERVERS.NET.           999999  IN      A
192.36.148.17
```

Sample `pz/domain.country'

The main domain file contains data identifying the IP addresses of hosts in your domain, it defines possible aliases for those names and it also identifies special servers such as mail-exchangers which mail-relay programs use to learn where to send electronic mail to your domain.

Each host has a *canonical name* or CNAME which is its official name. One may then define any number of aliases to this canonical name. For instance, it is common to create aliases to hosts which provide well known services, like `www.domain.country` and `ftp.domain.country` so that no one needs to remember a special host name in order to access these services in your domain. Here is an abbreviated example file. There are several kind of records here

`SOA`
: Indicates authority for this domain (referred to as `@').

`NS`
: Lists a nameserver for this domain.

`MX`
: Lists a mail exchanger for this domain (with priority).

`A`
: Create an A record, i.e. define the canonical name of a host with a given IP address.

CNAME
: Associate an alias with a canonical name.

HINFO
: Host information.

```
$ORIGIN iu.hioslo.no.    ; @ is an alias for this

@               IN      SOA     nexus.iu.hioslo.no.
drift.nexus.iu.hioslo.no.
                                (
                                1996111300 ; Serialnumber
                                3600    ; Refresh, 1 hr
                                600     ; Retry, 10 min
                                604800  ; Expire 1 week
                                86400   ; Minimum TTL, 1 day
                                )
; Name servers:

                IN      NS      nexus.iu.hioslo.no.
                IN      NS      samson.hioslo.no.
                IN      NS      samson.oslo.uninett.no.

; Mail exchangers:

@               MX      10      nexus

; Domain data:

                HINFO   Sun4    Solaris 2
www             CNAME   nexus   ; aliases
ftp             CNAME   nexus
mailhost        CNAME   nexus

; Router

ca30-gw         A       128.39.89.1
                A       128.39.73.129
                A       158.36.84.18
                HINFO   Cisco-4700      IOS
iu-gw           CNAME   ca30-gw

localhost       A       127.0.0.1

; 89 net

nexus           A       128.39.89.10
thistledown     A       128.39.89.233
voyager         A       128.39.89.234
nostromo        A       128.39.89.235
daystrom        A       128.39.89.236
borg            A       128.39.89.237
dax             A       128.39.89.238
axis            A       128.39.89.239

; 73 net
```

```
pc78-73         A       128.39.73.78
pc79-73         A       128.39.73.79
pc80-73         A       128.39.73.80
```

Sample `pz/network'

The network files are responsible for producing a fully qualified domain name given an IP address. This is accomplished with so-called PTR records. The reverse lookup-file looks like this:

```
$ORIGIN 89.39.128.in-addr.arpa.
               IN      SOA     nexus.iu.hioslo.no.
drift.nexus.iu.hioslo.no.
                                (
                                1996111300 ; Serialnumber
                                3600    ; Refresh, 1 hr
                                600     ; Retry, 10 min
                                604800  ; Expire 1 week
                                86400 ) ; Minimum TTL, 1 day
; Name servers:


                IN      NS      nexus.iu.hioslo.no.
                IN      NS      samson.hioslo.no.
                IN      NS      samson.oslo.uninett.no.

;
; Glue records:

;
; Mail exchangers:

;
; Domain data:
1       PTR     ca30-gw.iu.hioslo.no.
4       PTR     charon.iu.hioslo.no.
5       PTR     lore.iu.hioslo.no.
6       PTR     bajor.iu.hioslo.no.
7       PTR     galron.iu.hioslo.no.
8       PTR     ds9.iu.hioslo.no.
9       PTR     nova.iu.hioslo.no.
10      PTR     nexus.iu.hioslo.no.
11      PTR     pc.iu.hioslo.no.
12      PTR     pc-georgm.iu.hioslo.no.
15      PTR     pc-hildeh.iu.hioslo.no.
38      PTR     pc-torejo.iu.hioslo.no.
41      PTR     pc-steinarj.iu.hioslo.no.
```

Note carefully how the names end with a full-stop. If you forget this, the name server appends the domain name to the end, resulting in something like
`lore.iu.hioslo.no.iu.hioslo.no.`

Sample `pz/127.0.0'

```
; Zone file for "localhost" entry.

$ORIGIN 0.0.127.IN-ADDR.ARPA.
@       IN      SOA     nexus.iu.hioslo.no. drift.nexus.hioslo.no. (
                                1995070300 ; Serialnumber
                                3600    ; Refresh
                                300     ; Retry
                                3600000 ; Expire
                                14400 ) ; Minimum
        IN      NS      nexus.iu.hioslo.no.
;
; Glue
;
; (none needed, no name servers in this domain)

;
; Domain data
;
1               IN      PTR     localhost.

0.0.127.in-addr.arpa. IN NS nexus.iu.hioslo.no.
0.0.127.in-addr.arpa  IN NS samson.hioslo.no.

1.0.0.127.in-addr.arpa. IN PTR localhost.
```

Getting and compiling BIND v 8.x

The new BIND software can be collected from the Internet Software Consortium `www.isc.org'. This software contains everything you need to build replacement name-daemon software and `nslookup`. It also allows you to build resolver libraries for system lookup. Unless you really know what you are doing, I don't recommend replacing the resolver libraries on systems with shared-libraries (Solaris, for instance). If you are running Solaris 2.6 or better, you can make do without these. The most important element is the name server daemon `named`.

Here is a brief checklist for building and installing the nameserver:

- Make a new directory somewhere, called `bind'. Collect the gzipped tar file from the WWW site above. The tar file unpacks into a sub-directory called `src', since it is part of a larger tree of code, so make sure that you unpack in a fresh directory.
- Decide on a directory where the compiled code will reside. For instance `/site/host/bind'.
- You will need to be `root` in order to follow the build procedure. (Silly!)
- Change directory to the sources and follow the instructions there:
-

```
host# cd src
host# make DST=/site/host/bind SRC=`pwd` links
host# cd /site/host/bind
host# make clean
host# make depend
host# make
```

- The installation requires that you have byacc installed on your system. If you do not have this, you need to edit one of the Makefiles and replace it with yacc or bison. Note that, if you use bison, you should use the command `bison -y -d` for compatibility. Once the sources are build, you can move them to a permanent location. For instance

```
host# mv bin/named/named /local/sbin/named
```

and so on for the other programs. Remember to set appropriate permissions on the files. None of them need to be setuid root! Remember to set the correct path to the compiled xfer-daemon in the file `` `named.conf' ``. Remove any startup code you have on the system for older `named` versions. Add code to start up the new daemon.

Setting up a WWW server

@hrule @vskip 0.3cm

The World Wide Web (or W3) service is mediated by the http-daemon. There are several publicly available daemons which mediate the WWW service. This text is based on the freely available Apache daemon which is widely regarded as the best and most up-to-date. you can get it from

```
http://www.apache.org
```

At the practical level, the web service is just another daemon which you have to start on some server. To start a WWW service you need some html-files containing web data and a server which is configured with the filenames and locations of these data. You then need to set some configuration files which tell the daemon where to find the web pages it will be publishing, and to tell it what you do *not* want it to tell the outside world. The

security of your system depends on which files and directories outsiders should have access to. It is up to you to decide this. Here is your checklist:

- Create a directory in your site-dependent files called `www', where you will keep your web pages. In particular you will need your master file `www/index.html' which is the root of your web site.
- Create a directory in your site dependent files called `httpd' where the web server program will be installed. This is collected as a package, so you should keep the files together in this directory. Unpack the files into this directory.
- Edit the file `httpd/conf/access.conf' with your favourite text editor. Set up the directories to point to your data, See section Sample `access.conf' file.
- Edit the file `httpd/conf/srm.conf' and set the paths to point to your data, See section `srm.conf' file.

Sample `access.conf' file

```
# access.conf: Global access configuration
# Online docs at http://www.apache.org/

<Directory /local/iu/httpd/cgi-bin>
Options Indexes FollowSymLinks
</Directory>

<Directory /local/iu/www>
Options Indexes FollowSymLinks Includes
</Directory>

<Directory /local/iu/www/cgi-bin-public>
Options Indexes FollowSymLinks Includes
AllowOverride All
<Limit GET>
order allow,deny
allow from all
</Limit>
</Directory>
```

`srm.conf' file

```
#
# With this document, you define the name space that users see of your
http
# server.

# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory,
but
# symbolic links and aliases may be used to point to other locations.
```

```
DocumentRoot /local/iu/www/

# UserDir: The name of the directory which is appended onto a user's
home
# directory if a ~user request is recieved.

UserDir www

# DirectoryIndex: Name of the file to use as a pre-written HTML
# directory index

DirectoryIndex index.html

# FancyIndexing is whether you want fancy directory indexing or
standard

FancyIndexing on

# AddIcon tells the server which icon to show for different files or
filename
# extensions

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#AddIconByType (TXT,/icons/text.xbm) text/*
#AddIconByType (IMG,/icons/image.xbm) image/*
#AddIconByType (SND,/icons/sound.xbm) audio/*
#AddIconByType (VID,/icons/movie.gif) video/*
#AddIcon /icons/movie.xbm .mpg .qt
#AddIcon /icons/binary.xbm .bin
#
#AddIcon /icons/back.xbm ..
#AddIcon /icons/menu.xbm ^^DIRECTORY^^
#AddIcon /icons/blank.xbm ^^BLANKICON^^

# DefaultIcon is which icon to show for files which do not have an icon
```

```
# explicitly set.

DefaultIcon /icons/unknown.gif

# AddDescription allows you to place a short description after a file
in
# server-generated indexes.
# Format: AddDescription "description" filename

# ReadmeName is the name of the README file the server will look for by
# default. Format: ReadmeName name
#
# The server will first look for name.html, include it if found, and it
will
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.

ReadmeName README
HeaderName HEADER

# IndexIgnore is a set of filenames which directory indexing should
ignore
# Format: IndexIgnore name1 name2...

IndexIgnore */.??* *~ *# */HEADER* */README*

# AccessFileName: The name of the file to look for in each directory
# for access control information.

AccessFileName .htaccess

# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.

DefaultType text/plain

# AddType allows you to tweak mime.types without actually editing it,
or to
# make certain files to be certain types.
# Format: AddType type/subtype ext1
AddType text/plain doc
AddType text/html htm

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+)
uncompress
# information on the fly. Note: Not all browsers support this.

AddEncoding x-compress Z
AddEncoding x-gzip gz

# AddLanguage allows you to specify the language of a document. You can
# then use content negotiation to give a browser a file in a language
# it can understand.  Note that the suffix does not have to be the same
# as the language keyword -- those with documents in Polish (whose
```

```
# net-standard language code is pl) may wish to use "AddLanguage pl
.po"
# to avoid the ambiguity with the common suffix for perl scripts.

AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
AddLanguage da .da
AddLanguage el .el
AddLanguage it .it

# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
# Just list the languages in decreasing order of preference.

LanguagePriority en fr de

# Redirect allows you to tell clients about documents which used to
exist in
# your server's namespace, but do not anymore. This allows you to tell
the
# clients where to look for the relocated document.
# Format: Redirect fakename url

# Aliases: Add here as many aliases as you need, up to 20. The format
is
# Alias fakename realname

Alias /icons/ /local/iu/httpd/icons/

# ScriptAlias: This controls which directories contain server scripts.
# Format: ScriptAlias fakename realname

ScriptAlias /cgi-bin/ /local/iu/httpd/cgi-bin/

# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.

AddType text/x-server-parsed-html .shtml
AddType application/x-httpd-cgi .cgi

# If you want to have files/scripts sent instead of the built-in
version
# in case of errors, uncomment the following lines and set them as you
# will.  Note: scripts must be able to be run as if the were called

# directly (in ScriptAlias directory, for instance)

# 302 - REDIRECT
# 400 - BAD_REQUEST
# 401 - AUTH_REQUIRED
# 403 - FORBIDDEN
# 404 - NOT_FOUND
# 500 - SERVER_ERROR
# 501 - NOT_IMPLEMENTED

#ErrorDocument 302 /cgi-bin/redirect.cgi
```

```
#ErrorDocument 500 /errors/server.html
#ErrorDocument 403 /errors/forbidden.html

# MetaDir: specifies the name of the directory in which Apache can find
# meta information files. These files contain additional HTTP headers
# to include when sending the document

#MetaDir .web

# MetaSuffix: specifies the file name suffix for the file containing
the
# meta information.

#MetaSuffix .meta

# Customizable error response (Apache style)
#  these come in three flavors
#
#    1) plain text
#ErrorDocument 500 "The server made a boo boo.
#  n.b.  the (") marks it as text, it does not get output
#
#    2) local redirects
#ErrorDocument 404 /missing.html
#  to redirect to local url /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#  n.b. can redirect to a script or a document using server-side-
includes.
#
#    3) external redirects
#ErrorDocument 402 http://some.other_server.com/subscription_info.html
#

# mod_mime_magic allows the server to use various hints from the file
itself
# to determine its type.
#MimeMagicFile conf/magic

# The following directive disables keepalives and HTTP header flushes
for
# Netscape 2.x and browsers which spoof it. There are known problems
with
# these

BrowserMatch Mozilla/2 nokeepalive

# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.

IndexOptions FancyIndexing IconsAreLinks

ScriptAlias /cgi-bin-public /local/iu/www/cgi-bin-public

ScriptAlias /cgi-bin-nobody /local/www/cgi-bin
ScriptAlias /cgi-bin-noaccess /local/www/cgi-bin
```

```
ScriptAlias /cgi-bin-mark /site/server/home/mark/www/cgi-bin-mark
```

Perl script for generating script aliases

A convenient way of generating script aliases for all users is to write a short Perl script which rewrites the `srm.conf' file by looking through the password file and adding an entry for every user. In addition, a general cgi-bin directory is often desirable, where it is possible to place scripts which anyone can use. In the example below we call this alias `cgi-bin-public`. Each user has a script alias called `cgi-bin-`*`username`*.

```
#!/local/bin/perl
#
# Build script aliases from password file
#

# Path to the srm.conf file

$srmconf = "/local/httpd/conf/srm.conf";

open (OUT,">$srmconf") || die;

print OUT <<ENDMARKER;

#
#
#    NOTE !!!! Do not edit this file, it is generated automatically
#
#    by /local/bin/MakeScriptAlias
#
#
#
#
#
# With this document, you define the name space that users see of your
http
# server.

# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory,
but
# symbolic links and aliases may be used to point to other locations.

DocumentRoot /local/iu/www/

# UserDir: The name of the directory which is appended onto a user's
home
# directory if a ~user request is recieved.

UserDir www

# DirectoryIndex: Name of the file to use as a pre-written HTML
```

```
# directory index

DirectoryIndex index.html

# FancyIndexing is whether you want fancy directory indexing or
standard

FancyIndexing on

# AddIcon tells the server which icon to show for different files or
filename
# extensions

AddIconByType (TXT,/icons/text.xbm) text/*
AddIconByType (IMG,/icons/image.xbm) image/*
AddIconByType (SND,/icons/sound.xbm) audio/*
AddIcon /icons/movie.xbm .mpg .qt
AddIcon /icons/binary.xbm .bin

AddIcon /icons/back.xbm ..
AddIcon /icons/menu.xbm ^^DIRECTORY^^
AddIcon /icons/blank.xbm ^^BLANKICON^^

# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.

DefaultIcon /icons/unknown.xbm

# AddDescription allows you to place a short description after a file
in
# server-generated indexes.
# Format: AddDescription "description" filename

# ReadmeName is the name of the README file the server will look for by
# default. Format: ReadmeName name
#
# The server will first look for name.html, include it if found, and it
will
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.

ReadmeName README
HeaderName HEADER

# IndexIgnore is a set of filenames which directory indexing should
ignore
# Format: IndexIgnore name1 name2...

IndexIgnore */.??* *~ *# */HEADER* */README*

# AccessFileName: The name of the file to look for in each directory
# for access control information.

AccessFileName .htaccess
```

```
# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.

DefaultType text/plain

# AddType allows you to tweak mime.types without actually editing it,
or to
# make certain files to be certain types.
# Format: AddType type/subtype ext1
AddType text/plain doc
AddType text/html htm

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+)
uncompress
# information on the fly. Note: Not all browsers support this.

#AddEncoding x-compress Z
#AddEncoding x-gzip gz

# Redirect allows you to tell clients about documents which used to
exist in
# your server's namespace, but do not anymore. This allows you to tell
the
# clients where to look for the relocated document.
# Format: Redirect fakename url

# Aliases: Add here as many aliases as you need, up to 20. The format
is
# Alias fakename realname

Alias /icons/ /local/iu/httpd/icons/

# ScriptAlias: This controls which directories contain server scripts.
# Format: ScriptAlias fakename realname

ScriptAlias /cgi-bin/ /local/iu/httpd/cgi-bin/

# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.

AddType text/x-server-parsed-html .shtml
AddType application/x-httpd-cgi .cgi

# If you want to have files/scripts sent instead of the built-in
version
# in case of errors, uncomment the following lines and set them as you
# will.  Note: scripts must be able to be run as if the were called

# directly (in ScriptAlias directory, for instance)

# 302 - REDIRECT
# 400 - BAD_REQUEST
# 401 - AUTH_REQUIRED
# 403 - FORBIDDEN
# 404 - NOT_FOUND
# 500 - SERVER_ERROR
# 501 - NOT_IMPLEMENTED
```

```
#ErrorDocument 302 /cgi-bin/redirect.cgi
#ErrorDocument 500 /errors/server.html
#ErrorDocument 403 /errors/forbidden.html

IndexOptions FancyIndexing IconsAreLinks

ScriptAlias /cgi-bin-public /local/iu/www/cgi-bin-public

ENDMARKER

setpwent();

while (($name,$pw,$uid,$gid,$qu,$com,$full,$dir) = getpwent)
   {
   # SKip system accounts

   next if ($uid < 100);

   print OUT "ScriptAlias /cgi-bin-$name $dir/www/cgi-bin\n";

   last if ($name eq "");
   }

close OUT;
```

`mime.types' file

This file tells the server how to respond to file requests containing special data. It consists of a list of protocol names followed by a list of file extensions. Unrecognized files are displayed in your browser as text/ascii files. If you see graphics files (like vrml files) displayed as text, then you need to add a line here to inform the server about the existence of such files. Here is a brief excerpt:

```
   video/mpeg                   mpeg mpg mpe
   video/quicktime              qt mov
   video/x-msvideo              avi
   video/x-sgi-movie            movie
   x-world/x-vrml               wrl
```

E-mail configuration

@hrule @vskip 0.3cm

Nightmares on ELM street!

Electronic mail configuration is one of those things which you do not want to change very often. Once you have a working system, it is well worth leaving it alone.

Configuration of E-mail is one of the traditionally complex issues for the system administrator.

Why should it be so complex? Part of the trouble is that, in the past, there were many different kinds of networks, many different ways of connecting up to different hosts. This made it quite a complex issue to relay messages all over the world. Today things are much simpler: most sites use the internet protocols and mail configuration can be simplified quite a lot.

If you are lucky, and you are using an operating system like Linux, then a program will automatically help you set up E-mail. But you will still need some information about the way mail is handled at your site. There are two main models for handling electronic mail at a domain. One is that every host receives mail independently. This method is a bit old-fashioned since, usually, all users have the same password and account on all of the hosts on a network. Hence the second method.

The second approach is to have a mail `hub', or central mail processor. In this model, all incoming mail is diverted to the hub and all outgoing mail is sent via the hub. With this approach, you put all of your hard work into fixing mail on the hub, and all other machines have a simple configuration to pass all mail onto the hub.

In this section we shall look only at the mail agent called *Berkeley sendmail*. This is the most up to date version of sendmail. If you are using non-free software from a vendor, you should strongly consider collecting Berkeley sendmail from the network and compiling it in place of your vendors version. Sendmail is very susceptible to attack from the network, and only the Berkeley version is well enough equipped with deal with this threat.

Collecting sendmail

You can find out about sendmail and also collect the latest version from the web site.

```
http://www.sendmail.org
```

Compiling and installing sendmail

Once you have unpacked the distribution, you need to run make to compile it. Before doing this, you should make sure that you have all of the libraries you need to compile. Sendmail uses BIND and TCP-wrappers libraries. You should consider searching for the latest versions of these libraries on the internet before compiling. BIND is the resolver library. The official place to get BIND is `` `ftp://ftp.vix.com/pub/bind/release' ``. This also contains a library `lib44bsd.a` which you should make available. You can pick up the latest version of TCP wrappers from `` `http://ciac.llnl.gov/ciac/ToolsUnixNetSec.html' ``.

Many of the database lookup features require the Berkeley `db' package. You should get this from `ftp://ftp.cs.berkeley.edu/ucb/4bsd/db.tar.Z' and compile it.

Here is an example for sendmail-8.8.6:

```
host# cd sendmail-8.8.6
host# ls

FAQ       READ_ME       cf.tar  mail.local praliases src
KNOWNBUGS RELEASE_NOTES contrib mailstats  rmail     test
Makefile  cf            doc     makemap    smrsh

host# cd src
host# sh makesendmail
Configuration: os=SunOS, rel=5.5.1, rbase=5, rroot=5.5,
 arch=sun4, sfx=
Making in obj.SunOS.5.5.1.sun4
...
```

The script `makesendmail' selects your operating system type and compiles the program. In the process it creates a directory for the compilation. In the example above, it creates `obj.SunOS.5.5.1.sun4'.

You might still have to edit the Makefile in this new directory, so do a CTRL-C to stop the compilation and edit the file which corresponds to `obj.SunOS.5.5.1.sun4/Makefile' in the example above. You will have to choose the exact Makefile based on your own operating system.

In the Makefile, you can switch on several features. The first thing you should switch off is NIS alias lookups. The best way to do alias lookups is to use *only* the Berkeley db database. That means editing out the line beginning DBMDEF= as in the example below. Use of NIS, NIS+ or other databases is not recommended.

You must also decide where you want your distribution to be placed. A good place is `/usr/local/mail' or `/usr/local/lib/mail'. Create this directory now and make a subdirectory `bin' where you will keep your executable file.

Here is an example Makefile:

```
#
#  This Makefile is designed to work on the old "make" program.  It
does
#  not use the obj subdirectory.  It also does not install
documentation
```

```
#  automatically -- think of it as a quick start for sites that have
the
#  old make program (I recommend that you get and port the new make if
you
#  are going to be doing any signficant work on sendmail).
#
#  This has been tested on Solaris 2.5.
#
#       @(#)Makefile.SunOS.5.5  8.10 (Berkeley) 4/13/97
#

# use O=-O (usual) or O=-g (debugging)
# warning: do not use -O with versions of gcc prior to 2.6
O=      -O

CC=     gcc
DESTDIR = /local/mail

# define the database mechanism used for alias lookups:
#       -DNDBM -- use new DBM
#       -DNEWDB -- use new Berkeley DB
#       -DNIS -- include NIS support
# The really old (V7) DBM library is no longer supported.
# See READ_ME for a description of how these flags interact.
#
#DBMDEF=        -DNDBM -DNIS -DNISPLUS
DBMDEF= -DNEWDB

# environment definitions (e.g., -D_AIX3)
ENVDEF= -DSOLARIS=20500

# see also conf.h for additional compilation flags

# include directories
INCDIRS=-I/usr/sww/include

# library directories
LIBDIRS=-L/usr/sww/lib -L/local/lib

# libraries required on your system
#  delete -l44bsd if you are not running BIND 4.9.x
#  add -ldb if you add -DNEWDB above (in DBMDEF)
#LIBS=  -lresolv -l44bsd -lsocket -lnsl -lkstat
LIBS=   -lwrap -lresolv -l44bsd -lsocket -lnsl -lkstat -ldb

# location of sendmail binary (usually /usr/sbin or /usr/lib)
BINDIR= ${DESTDIR}/lib

# location of sendmail.st file (usually /var/log or /usr/lib)
STDIR=  ${DESTDIR}/log

# location of sendmail.hf file (usually /usr/share/misc or /usr/lib)
HFDIR=  ${DESTDIR}/etc

...
```

When you have compiled the program successfully, the finished executable files must be installed. Store these executables in your local files, by copying them like this:

```
cp obj.SunOS.5.5.1.sun4/sendmail /usr/local/mail/bin/sendmail
cp obj.SunOS.5.5.1.sun4/makemap  /usr/local/mail/bin/makemap
```

Your operating system most likely expects to find the sendmail executable file in either the `/usr/lib/' directory, or the `/usr/sbin' directory on newer systems. You should replace the old executable in these directories by making a link to the new executable. For example:

```
mv /usr/lib/sendmail /usr/lib/sendmail.org
ln -s /usr/local/mail/bin/sendmail /usr/lib/sendmail
```

Configuring sendmail

To finish of the installation, you need to create configuration files for your mail domain. Begin by going back to the sendmail distribution and copying the `cf' directory to your own directory, like this:

```
cp -r sendmail-8.8.6/cf /usr/local/mail
```

Next make a `lib' directory.

```
mkdir /usr/local/mail/lib
```

To create a `sendmail.cf' file, you need to create a so-called macro file `/usr/local/mail/lib/domain.mc'. Here is an example file for domain `iu.hioslo.no'. You should only need to change the domain name and the OS name of your mailhost in the first trhee lines. Using this file you will be able to build the sendmail configuration more or less automatically.

```
divert(-1)
include(`/usr/local/mail/cf/m4/cf.m4')

VERSIONID(`$Id: nexus.mc,v 1.1 1997/04/08 08:52:28 mroot Exp mroot $')
```

```
OSTYPE(solaris2)dnl
DOMAIN(iu.hioslo.no)dnl
MASQUERADE_AS(iu.hioslo.no)

FEATURE(use_cw_file)
FEATURE(use_ct_file)
FEATURE(redirect)
FEATURE(always_add_domain)
FEATURE(allmasquerade)
FEATURE(masquerade_envelope)
FEATURE(domaintable, `hash -o /usr/local/mail/lib/domaintable')
FEATURE(mailertable, `hash -o /usr/local/mail/lib/mailertable')
FEATURE(genericstable, `hash -o /usr/local/mail/lib/genericstable')
FEATURE(virtusertable, `hash -o /usr/local/mail/lib/virtusertable')
GENERICS_DOMAIN_FILE(/usr/local/mail/lib/sendmail.cG)
EXPOSED_USER(root)

define(`ALIAS_FILE', /usr/local/mail/lib/aliases)dnl
define(`HELP_FILE', /usr/local/mail/lib/sendmail.hf)dnl
define(`STATUS_FILE', /usr/local/mail/etc/sendmail.st)dnl
define(`QUEUE_DIR', /var/spool/mqueue)
define(`LOCAL_MAILER_CHARSET', iso-8859-1)dnl
define(`SMTP_MAIL_CHARSET', iso-8859-1)dnl
define(`confMAX_MESSAGE_SIZE', `10000000')
define(`confHOST_STATUS_DIRECTORY', `.hoststat')
define(`confPRIVACY_FLAGS', `authwarnings,noexpn,novrfy')
define(`confME_TOO', `True')
define(`confMIME_FORMAT_ERRORS', `False')
define(`confTIME_ZONE', `MET-1METDST')
define(`confDEF_CHAR_SET', `iso-8859-1')
define(`confEIGHT_BIT_HANDLING', `m')
define(`confCW_FILE', `/usr/local/mail/lib/sendmail.cw')
define(`confCT_FILE', `/usr/local/mail/lib/sendmail.ct')
define(`confUSERDB_SPEC', `/usr/local/mail/lib/userdb.db')
define(`confSMTP_MAILER', `esmtp')
MAILER(local)dnl
MAILER(smtp)dnl
LOCAL_CONFIG
F{Stupid} /usr/local/mail/lib/stupid
F{Spammers} /usr/local/mail/lib/spammers
F{SpamDomains} /usr/local/mail/lib/spam_domains
LOCAL_RULESETS
Scheck_mail
#
# .. continued on next page



#
# Reject known spammers
#
R<$={Spammers}>              $#error $@ 5.7.1 $: We don't accept junk
mail
R<$={Stupid}>         $#error $@ 5.7.1 $: "571 Bad sender's system
address
#
#  Name canonicalization
```

```
 #
 R$*  [ $+ ] $*        $: $1  $[ [ $2 ] $] $3
 R$*                   $: $>3 $1
 #
 # Reject Spam Domains and their subdomains
 #
 R$*<@$={SpamDomains}.>$*      $#error $@ 5.7.1 $: We don't accept junk
mail
 R$*<@$*.$={SpamDomains}.>$*   $#error $@ 5.7.1 $: We don't accept junk
mail
 #
 #
 # Reject unresolvable domain (no dot at end of hostname after
canonicalization)
 #
 R$* < @ $* $~P > $*   $#error $ 4.1.8 $: "418 Bad sender's system
address"

 LOCAL_RULE_3
 #
 # Repair defective TV2 addresses :-)
 #
 R$* < @ tv2mail . tv2 . no > $*      $: $1 <  tv2 . no > $2
 #
 #
```

Next create `/usr/local/mail/Makefile' which will build the configuration for you:

```
MAKEMAP=        bin/makemap
SENDMAIL=       bin/sendmail
PIDFILE=        /etc/mail/sendmail.pid
MCFILE=         lib/domain.mc
ALIASES=        lib/aliases
USERDB=         lib/userdb
CF_DIR=         cf/

all:    sendmail.cf $(ALIASES).db $(USERDB).db .restart

$(ALIASES).db: $(ALIASES)
        $(SENDMAIL) -bi

$(USERDB).db: $(USERDB)
        $(MAKEMAP) btree $(USERDB) < $(USERDB)

#sendmail.cf: $(MCFILE) cf/hack/rewrite_from.m4
sendmail.cf: $(MCFILE)
        m4 -D_CF_DIR_=$(CF_DIR) cf/m4/cf.m4 $(MCFILE) > sendmail.cf

.restart: sendmail.cf lib/sendmail.cw
        kill -1 `head -1 $(PIDFILE)
        touch .restart
```

Typing `make` in the `` `/usr/local/mail' `` directory should now result in a configuration file `/usr/local/mail/sendmail.cf`. You should wait until you have read the next section before doing this.

You will need to create a file `` `lib/sendmail.cw' `` which contains a list of possible machines or domains for which the sendmail program will accept mail. It is, amongst other things, this file which allows you to send mail of the form `mark@iu.hioslo.no`, i.e. to an entire domain, without specifying a particular machine. This file should contain a list of all the valid addresses, like this:

```
iu.hioslo.no
mailhost.iu.hioslo.no
www.iu.hioslo.no
nexus.iu.hioslo.no
dax.iu.hioslo.no
borg.iu.hioslo.no
worf.iu.hioslo.no
daystrom.iu.hioslo.no
regula.iu.hioslo.no
ferengi.iu.hioslo.no
lore.iu.hioslo.no
```

Finally, you will need to make the key files readable for normal users. There is no harm in giving everyone read access to all the files and directories.

Rewriting outgoing addresses

The `` `lib/userdb' `` file and `` `lib/aliases' `` file are used by sendmail for resolving aliases. It is common for sites to create mail aliases for all their users, by taking the full name of each user and joining the pieces with dots. For example, user `` `mark' `` with fullname `Mark Burgess' would map to an alias `` `Mark.Burgess' ``.

Aliases of this kind look nice and are user friendly to outsiders, but they do not work unless you set up the system yourself. To do this you need to create files `` `lib/aliases' `` and `` `lib/userdb' ``

For each user, the `` `userdb' `` file should contain a line of the form.

```
Mark.Burgess@iu.hioslo.no:maildrop mark@mailhost.iu.hioslo.no
```

which tells sendmail where to deliver incoming mail which is sent to the user alias `Mark.Burgess' at the mail hub.

Outgoing messages are processed if you have a line of the form:

```
mark:mailname Mark.Burgess@iu.hioslo.no
```

This means that mail messages which originate from `mark' will be rewritten so that they look as though they orginate from `Mark.Burgess@iu.hioslo.no'. This form usually only applies to mail which originates from the local mail host, since that is the only case where your outgoing name is just your short user-name `mark'. Mail which is sent from other hosts to the mail-hub for outgoing processing generally produces from-information in the form `mark@iu.hioslo.no'. In order for this to be rewritten, you need a line of the form as well.

```
mark@iu.hioslo.no:mailname Mark.Burgess@iu.hioslo.no
```

The `aliases' file is set up as in the earlier section, See section Mail aliases.

Security and reliability

Security is about keeping the system safe. This includes protecting it against

- Malicious attacks.
- Accidental erasure of data.
- Disk crashes.
- User ignorance.

There are two kinds of approach to maintaining system security: prevention and recovery.

Security is an increasingly important problem. Just in the last few years the number of attacks and break-ins to computer systems has risen to millions of cases a year. Crackers(9) have foud their way inside the computers of the pentagon, the world's security services, warships, figher plane command computers, banks and major services such as electrical power grids. With this kind of access the potential for causing damage is great. Computer warfare is the next major battlefield we have to conquer. It is happening now, as you read these words. It is here, like it or not. Moreover, it is estimated that the banks lose millions of dollars a year to computer crime.

If that is not enough to scare you into awareness, you will just have to learn the hard way.

Security is a huge subject, because modern computer systems are very complex and the connectivity of the internet means that millions of people can try to break into networked systems. This chapter is not a comprehensive guide to security. There is not time in this introductory course to cover security in detail. There are several reasons for this. One is the sheer size of the subject, the other is that security awareness requires a level of experience which we have not had time to develop. A third reason is that one could easily spend every waking moment worrying about security, but in the end it comes down to a choice: how much security do you need?

It is a good idea to have a security policy so that you know exactly what you mean by security. That way, when security is breached, you will know what to do. Some sites which contain sensitive data require strict security and spend a lot of time enforcing it, others do not particularly care about their data and would rather not waste their time on pointless measures to protect them.

Systems which do not implement security tend to attract only low-level crackers--and those who manage to break in, tend to use the systems only as a springboard to go other places. The more security you implement, the more of a challenge it is for a cracker. So spending a lot of time on security might only have the effect of asking for trouble.

When it comes down to it there is this: if you have extremely sensitive data, do not put them onto a network capable computer. If you do, live with the consequences.

User security

@hrule @vskip 0.3cm

Password security

Password security is the first line of defense against intruders. Experience shows that many users have little or no idea about the importance of using a good password.

Let's face it, users can be pretty lame. I mean. if you're going to have a password, then you don't print it on your T-shirt, right? Let's take a look at some examples from a survey of passwords at a university. About 40 physicists had the password `Einstein', around 10 had `Newton' and several had `Kepler'. Hundreds of users used their login-name as their password, some of them really went to town and added `123' to the end. Let's face it, users can be pretty lame.

If you want privacy, you don't leave the door unlocked!

Passwords are not visible to ordinary users, but their encrypted form is usually visible. There are many publicly available programs which can guess passwords and compare them with the encrypted forms. No one with an easy password is safe. Passwords should never be any word in a dictionary or a simple variation of such a word or name. It takes just a few seconds to guess these.

Some new operating systems like FreeBSD, NetBSD and Solaris and linux have `shadow password files' which are not readable by normal users. The regular password file contains an `x' instead of a password, and the encrypted password is kept in an unreadable file. This makes it much harder to scan the password file for weak passwords.

Once a malicious user has gained access to an account, it is very much easier to exploit other weaknesses in security. Good passwords are the key to a safe system.

On the network, it is possible to fetch programs such as `crack` which are designed to break passwords. It is unusual not to find a few accounts with trivial passwords in the space of a few seconds.

A useful way to choose a password is to use the PIN code number as a part of your password. This means that you don't have to remember too much--and it means that you have secret numbersin your passsword. The worst passwords are names and words which are in any language dictionary.

xhost access list

Although many users are not aware of it, it is possible to actually download the screen picture of another user using the X-windows system. It is equally possible to `bug' the keyboard and listen to all the keypresses.

The problem is an out-dated security mechanism which has long since been replaced, but which is still used by some old-fashioned users. The problem is the `xhost` program. This is used to grant other hosts permission to draw on your X server--in other words, if you are remotely logged on to a host other than the one you are using as a display, you must grant the remote host access to write on your screen.

In the old X windows system, prior to release 5, one had to grant access to a particular host. One this was done, *anyone* on that host had access to your server, not just you. This was later replaced by the `xauth` magic-cookie mechanism which works on a user basis. Some users still insist on using `xhost` however, with a command like this:

```
xhost +
```

Any user writing this, opens their display to everyone in the world. The antidote, of course is the command `xhost -`.

Secure shell

This is a secure replacement for the `rsh` commands. It protects against IP spoofing where a remote host pretends to be trusted host by faking IP datagrams; DNS spoofing where an attacker forges name entries in the nameservice; the interception of passwords in network packets and several other kinds of attack.

Accidental deletion of files

As you should know, once you delete a file in UNIX, you cannot get it back. There is no way to undelete a file. Some system administrators like to protect ignorant users by making an alias (in C shell)

```
alias rm    rm -i
```

which causes the `rm` command to prompt whether it should delete files before actually doing so. This is a simple idea and it is not foolproof. The only real security against deletion is to keep extensive backups of user disks.

System security

@hrule @vskip 0.3cm

Since the explosion of interest in the internet, the possibility of hosts being attacked from outside sources, has become a significant problem. With literally millions of users on the net, the tiny percentage of malicious users becomes a large number.

Security Policy

The place to start in implementing security(10) is to indentify what it is you are trying to accomplish.

First of all: what resources are you trying to protect?

Secrets
: Some sites have secrets they wish to protect. They might be government or trade secrets or the solutions to a college exam.

Personell data
: In your country there are probably rules about what you must do to safeguard sensitive personal information. This goes for any information about employees, patients, customers or anyone else you deal with. Information about people is private.

CPU usage/System downtime
: You might not have any data that you are afraid will fall into the wrong hands. It might simply be that your system is so important to you that you cannot afford the loss of time incurred by having someone screw it up. If the system is down, everything stops. Perhaps you can't afford that.

Abuse of system
: It might simply be that you do not want anyone using your system to do something for which they are not authorized, like breaking into other systems.

Next: what will happen if the system is compromised?

- Loss of money
- Threat of legal action against you
- Missed deadlines

- Loss of reputation

How much work will you need to put into protecting the system. Who are the people trying to break in?

- Sophisticated spies
- Tourists, just poking around
- Braggers, trying to impress

Finally: what risk is acceptable to you? If you have a secret which is worth 4 Lira, would you be interested in spending 5 Lira to secure it? Where do you draw the line? How much is your security worth to you?

You should never forget the social term in the security' equation. You might spend a hundred thousand dollars on the most super fancy firewall system to protect you data, but someone could walk into the building and look over your shoulder, or use a receiver to collect the stray radiation from your monitors. Are you leaving printouts lying around? Are you willing to place your entire building in a Faraday cage to avoid remote detection of the radiation expelled by monitors? In the final(!) instance someone could just point a gun at your head and ask you nicely for your secrets.

There's a few examples of software security which you might want to look at.

`Site security handbook, RFC 1244`
`gopher://gopher.eff.org/11/CAF/policies`
`http://musie.phlab.missouri.edu/Policy/copies/tamucollection1.html`
`http://www.usenix.org`
: A guide to developing computing security documents

Security holes

One way that outside users can attack your system is by exploiting security holes in software. Classic examples usually involve *setuid-root* programs, which give normal users temporary superuser access to the system. Typical examples are programs like `sendmail` and `finger`. These programs are constantly being fixed, but even so, new security holes are found with alarming regularity.

Faults in software leave back-doors open to intruders. The only effective way of eliminating such attacks is to build a so-called *firewall* around your network, See section Firewalls.

The computer emergency response team (CERT) was established in the wake of the Internet Worm incident to monitor potential security threats. CERT publish warnings to a mailing list about known security holes. This is also available on the newsgroup

```
comp.security.announce
```

It is a good idea to follow these. Subscription messages may be sent to:

```
cert-advisory-request@cert.org
```

There are also relevant web sites at

```
www.sans.org
www.cert.org
ciac.llnl.gov
```

PLEASE: do not subscribe to this mailing list from this college! If you are interested in receiving these notices, ask.

Firewalls

A firewall is a network configuration which isolates some machines from the rest of the network. You might want to isolate your network from the internet, or you might want to isolate just a few hosts from the rest of your local network. Whatever you decide, a firewall is a gatekeeper which limits access to and from your network. You might think that this sounds like a lousy idea: after all, what is the point of being connected to the internet if you just want to isolate yourself from it!

There is no single firewall solution. The name `firewall' is a collective description for a variety of methods which restrict access to a network. They all involve placing restrictions on the way in which network packets are routed.

A firewall might be a computer which is programmed to act like a router, or it might be a dedicated router or a combination of routers and software systems. The idea with a firewall is to keep important data behind a barrier which has passport-control and can examine and restrict network packets, allowing only `harmless' packets to pass.

- All traffic from inside to outside or vice versa must pass through the firewall.
- Only authorized traffic is allowed to pass.
- Potentially risky network services (like mail) are rendered safer using intermediatary systems.
- The firewall itself should be immune to attack.

A firewall cannot help with the following:

- Badly configured hosts or misconfigured networks.
- Data based attacks (where the attack involves sending some harmful information, like the code word which makes you take your own life, or an E-mail which bolts a Trojan horse).

There are two firewall philosophies: *block everything unless we make an explicit exception* and *Pass everything unless we make a specific exception*. The first of these is clearly the most secure or at least the most paranoid of the two.

Here's a few concepts which get bandied around in firewall-speak:

Screening router/Choke
: A router which can be programmed to filter or reject packets directed at certain IP ports.

Bastion host
: A computer, specially modified to be secure.

Dual-homed host
: A computer with two net-cards, which can be used to link an isolated network to a larger network.

Application gateway
: A filter, usually run on a bastion host, which has the ability to reject or forward packets at a high level (i.e. at the application level).

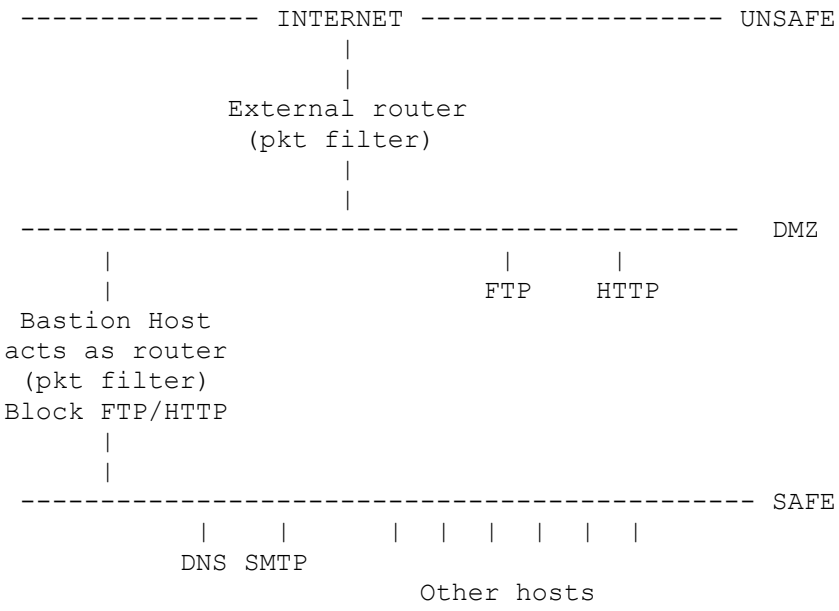
Screened subnet/DMZ
: An isolated subnet, between the internet and the private network. Also called a DMZ (de-militarized zone). A DMZ is the bit between a screening router and the firewall, bastion host. This is a good place for external WWW services.

The firewall philosophy maintains that it is easier to secure one host (the bastion host) than it is to secure hundreds or thousands on a local network. We focus on one machine and make sure that it is the only one effectively on the network. We force all network traffic to stop at the bastion host, so if someone tries to attack the system by sending some kind of IP attack there can be little damage to the rest of the network because the private network will never see the attack. This is simplifying a bit though. It is important to realise that installing a firewall does not give you abolsute protection and it does not let you off the hook of configuring and securing the hosts on the inside of the firewall.

Of course we do not want all traffic to stop, we want some things like E-mail and maybe FTP to pass through. To allow this, one uses a so-called `proxy' service or a `gateway'. A common solution is to give the bastion host two network interfaces. One is connected to the unsafe part of the network and the other is connected to the safe part. A service is said to be proxied if the bastion host forwards the packets from the unsafe network to the safe one. It only does this for a packets which meet the requirements of your security policy. For instance, you might decide that the services you require to cross the firewall are inbound/outbound telnet, inbound/outbound SMTP (mail), DNS, HTTP and FTP but no others.

Proxying requires some special software, often at the level of the kernel where the validity of connections can be established. For instance, packets with forged addresses can be blocked. Data arriving at ports where there was no registered connection can be discarded. Connections can be discarded if they do not relate to a known user-account.

A simple firewall configuration is shown below.



In this example we have effectively two routers, a DMZ and a protected network. The first packet filtering router will route packets between the internet and one of three hosts. FTP is routed directly to a special FTP server. The same applies to HTTP packets. These services are dealt with by separate hosts, so that (if something should go wrong and the machines are broken into) it is no worse than having to restore these single hosts from backup. None of the servers in the DMZ have user accounts, so there would be no help to crackers trying to crack password files there, if they managed to break in. The bastion host gets all packets which are not for the other services. The bastion host forwards okay-looking packets to the internal router which is really just a further packet filter (a backup in case of failure of the bastion host). The internal router accepts only packets passing between the safe network and the bastion host, all others are rejected. The bastion host proxies all of the appropriate protocols including FTP and HTTP between the safe network and the DMZ. This is just an example. In practice you might not have all the hardware you need to separate things as cleanly as shown here.

Although there is a public domain firewall toolkit,

http://www.tis.com

most firewall software is commerical in nature because it needs to live in the kernel and make use of code which is proprietory.

Firewall management is a complex issue. You can't just set up a firewall and then forget about it. Firewalls need constant maintenance and they are susceptible to bugs just like any other software. It is best to build up a firewall system slowly understanding each step as you go. A good place to start is with packet filtering routers to eliminate the most offensive or least secure service requests from outside your local network. These include NFS, IRC, ping, finger etc.

Access control with tcpd

If you don't want the hassle of a firewall, then you can go half way by verifying the authenticity of packets coming into your network. This is a front line against `spoofing' -- i.e. internet impersonantion. The TCP wrappers package is a tool which can be used to this end.

The `tcpd` program is a `wrapper' for internet services which provides host-based access control. Instead of starting services directly from the `/etc/inetd.conf' file, one starts the tcp-daemon with instructions to start a given service. The `tcpd` daemon checks where the request comes from and only starts it if it comes from a trusted host. A trusted host is one which is listed in `/etc/hosts.allow'. Another file `/etc/hosts.deny' lists services which are to be denied to non-trusted hosts.

```
   replace:

     finger  stream  tcp  nowait  nobody  /usr/etc/in.fingerd
in.fingerd

   with:

     finger  stream  tcp  nowait  nobody  /some/where/tcpd
in.fingerd
```

Attacks

@hrule @vskip 0.3cm

Ping attacks

Some older network interfaces can be made to crash certain operating systems by sending a `ping' request with a very large packet size. Some users deliberately try to crash people's systems by sending such packets. This problem can be combated with a packet filtering router. See http://www.sophist.demon.co.uk/ping/ .

Denial of service attacks

Another type of attack is to overload a system with so many service requests that it grinds to a halt. One example is mail spamming(11), in which an attacker sends large numbers of repetitive e-mail messages, filling up the server's disk and causing the `sendmail` daemon to spawn rapidly and slow the system to a stand-still.

Newer versions of Berkeley sendmail have built in anti-spamming mechanisms to help protect from this problem. Vendors' sendmails are less advanced.

Denial of service attacks are almost impossible to protect against. It is the reponsibility of local administrators to prevent their users from initiating such attacks whereever possible.

Protecting against attacks

- Disable unused services which might contain security leaks, like UUCP.
- Monitor connections using `netstat -a` to show all listening connections.
- Monitor processes running on your system. How many copies of important processes are running? How many should be running. Often it is possible to see that one is under attack by looking at what processes are running and who is running them. For instance an attempt at port sniffing or spamming might be seen with a bunch of proceses like this:
-
-
-
```
nobody  .... /usr/sbin/inetd
nobody  .... /usr/sbin/inetd
nobody  .... /usr/sbin/inetd
nobody  .... /usr/sbin/inetd
nobody  .... /usr/sbin/inetd
```
-

  `inetd` is a multiplexer which starts internet services on many ports. Normally it is only root who runs this. The above indicates that a user is trying to use the well-known account `nobody` to start services, or to overload the system with requests.

- Check filesystems for suspicious looking hidden files. i.e. files with names like `` `.. .' `` These are often used to hide dangerous programs or shells which users can use to gain root privileges. Cfengine performs this task automatically when it examines filesystems.

Password sniffing

Many communication protocols (telnet, ftp etc) were introduced before security was a concern amongst those on the internet. So many of these protocols are very insecure.

Passwords are often sent over the network as plain text. This means that a sophisticated cracker could find out passwords simply by listening to everything happening on the network and waiting for passwords to go by.

One way to avoid this problem is to use a system of one-time passwords. This is a system which is designed to eliminate the need for you to send your password over the network. Instead of typing your actual password, you use a program on your local machine, together with your password on a remote machine to generate a sequence of throw-away passwords which will be used in place of your actual password. The passwords are used only once, so even if someone gets to overhear your password, it will already be too late: the password will have expired. S/KEY is such a system. Here is an example of how it works:

1. You want to make a connection from host A to host B.
2. You have earlier set a password on host B.
3. You telnet to host B from host A.
4. Host B prompts you with a code string: 659 ta55095 and asks for your user name. you type your user name and host B asks you for the one-time password.
5. You now need to find the one-time password by running a local program on host A with the code string as an argument:

```
key 659 ta55095
passwd: *******
```

   The key program prompts you for your secret password on host B. When you type this it does not go across the network. The key program returns a clear text, one-time password valid for one session: "EASE FREY WRY NUN ANTE POT"

11. You type "EASE FREY WRY NUN ANTE POT" on host B (sent over the network) and the password is accepted.
12. Next time you follow the same procedure and get a different password.

The social aspect

@hrule @vskip 0.3cm

System crackers are people. That might seem like an obvious thing to say, but it is something which netfreaks tend to forget. We are usually so consumed by the world of the network, that we forget that people can just walk into a building and steal something in the *real world* (not to be confused with the MTV television series). There is more than one way to crack a system.

Cracking is a social problem. It needs a social solution, or at the very least social awareness.

Spamming is a kind of network childishness, a selfishness or perhaps a tantrum. Someone sends a whole load of unwanted mail, often the same mail many times. Either way, it is best handled in an adult way, either by ignoring the sender or by locating the sender through appropriate channels. you should *never* spam someone back to get even. That is falling straight into the trap which spammers are trying to set. In general, security requires some social awareness.

The only secure computer is a computer which is locked into a room, not connected to a network, shielded from all electromagnetic radiation. In social studies of large companies, it has been demonstrated that--in spite of expensive firewall software and sophisticated anti-cracking technology--all most crackers had to do to break into the system was to make a phone call to an unwary employee of the company and ask for their username and password. Some crackers posed as system administrators trying to fix a bug, others simply questioned them as in a marketing survey until they gave away information which allowed the crackers to guess their passwords. Some crackers will go one step further and visit the building they are trying to break into, going through the garbage/refuse to look for documents which would give clues about security. Most people do not understand the lengths that people will go to to break into systems if they really want to.

Another social phenomenon is bragging. People (spotty kids) who have broken into the system like to tell people that they have been there and done that. They sometimes try to scare administrators by telling them how much damage they have caused. Here the trick is not to panic and do something hasty, but to try to verify what the crackers claim. In most cases it is nonsense, empty words.

There is a few things you can do to counteract social threats.

- Examine system logs, check the system, run cfengine.
- Make hardcopies of messages sent with all the headers printed out. Most people don't know how to hide their true identity on the internet.
- Make proper backups of the system regularly.
- Inform the whole system of attacks so that anyone who knows something can help you.
- Do not assume that what crackers tell you is true. Make a judgement as to whether you want to act or ignore the event.
- Have a clear security policy. Death threats, serious or not, should probably be reported to the company responsible for sending the message, and perhaps even the police, but not the person sending the message.

PART II: NT OVERVIEW

NT vs Unix showdown

A little bird told me that NT stands for `new technology'. While you are cleaning the spilled coffee from your dress, ponder this: most users in the world do not want the power of UNIX--they want simple tools for simple jobs. They have therefore grown up with DOS and MacIntosh and will continue to use these platforms, not because they are best, but because they are simple. One should not (publicly) scorn the PC user simply because they use an inferior system.

Word on the street has it that it is fashionable to be wearing Windows NT. Of course a girl knows that to wear the *right thing*, she should avoid main street shopping. But there is no doubt that the masses are into the latest *wearz* from Microsoft. Call it bitching, but Windows NT is not Jean Paul Gaultier or Ralph Lauren: it's simple things for simple minds(12).

NT is based around an old concept: namely that every PC is a personal workstation for just one user. You couple each workstation to a network server which serves files, printers, and network services like mail to each PC centrally. The model is old-fashioned because, a network is for sharing resources evenly and this centralized model means that *one PC* has the entire load of of file service, while every other PC sits idle, running empty cycles for more than 90 percent of the time. This is wastage, bigtime(13). This centralized model was originally created for the PC because PCs running DOS could not multitask. Now that they can, there is no need to have such centralization. While NT clings to this model, everyone else is into distributed services and full-blown networking. Did I say last season?(14)

NT is both easier and harder to cope with for system administrators. It is easier because the old fashioned model of having one server at a time means that all configuration information can be left in one place (on the server), and each workstation can be made (at least to a limited degree) to configure itself from the server's files.

It is harder to run because system adminisration tasks are tied into the *user-interface*! That means that to do any simple thing, you have to open about thirty windows and press a hundred buttons. Most system files are not editable ASCII files, so you *have* to use the user interface. This makes system files more efficient for the machine, but harder to maintain.

By forcing users to use the window interface, the burden of administrating an NT machine falls on the person who has the machine in front of him/her. NT has no remote login feature, so this job cannot be handled by someone else, unless the PC is tied specifically to a central server for all of its maintainance.

The central model means that user registration only has to be done on the central server which is an advantage, as with unix systems like NIS, but system administration is also about maintaining each machine separately. This involves cleaning up the disk, making sure that everything is locally in place... These things cannot be done from a central `server'.

NT is quite new and Microsoft have made a decision to reinvent every system rather using tried and tested solutions, so it will take sometime to catch up with Unix. In the mean time, we have to live with these systems, so we'd better start getting comfy. One thing is certain, NT has a long way to go before it becomes mature, so we can expect some big changes in the next few years.

This part of the book is not about running NT in the way that Microsoft suggest. Many books have already been written on this very large topic. Rather, it is about integrating NT into a network, thus opening NT to a larger world so that it can communicate with general network services and open Unix systems. It is about living with NT and Unix as cooperating systems rather than as rival systems.

Tools for UNIX junkies

GNU tools for NT

If, after two minutes of starting Windows NT, you're looking for the command shell, after another two minutes you're looking for a decent shell with command completion etc... then you are a *Unix junkie*. But don't dispair, Major Tom, you are in good company. Those very nice GNU people have ported many of the standard unix commands as well as the `emacs` text editor (which includes a decent shell) to NT. You can save a lot of clicking time by fetching the GNU for NT software.

Since you don't get anything for free on NT, you will need the help of a Unix machine in order to collect the necessary software and unpack it. The software is ready compiled and comes complete with some batch files to help set up appropriate environment variables. You will need to log on as Administrator.

- Make directory `` `C:\usr\local' `` where the unix commands will be installed. (New software should really go into the `` `Program Files' `` directory, according to the NT setup, but Unix junkies seem to need their `` `/usr/local' ``!)
- Start the internet explorer. Go to a GNU ftp site, for instance `` `ftp://ftp.funet.fi' ``. Go to `` `pub/win-nt/gcc' `` folder and get everything.
-
-
- `gnu-bin.tar.Z` `gnu-lib.tar.Z` `tar.exe`
- `gnu-emacs.tar.Z` `gnu-man.tar.Z` `win32gnu.dll`
-

If you can find `gunzip` or `uncompress` get them, but otherwise you will need to download the files to a Unix machine first and uncompress them there with `gunzip` You can then use the explorer or the Windows NT `ftp` program to collect the unpacked tar file.

- Move the tar file and `` `tar.exe' `` file to the `` `C:\' `` directory and start an NT/DOS command prompt. Type
-
-
```
tar xf gnu-bin.tar
```

and so on. The files unpack themselves.

- To start a shell window which recognizes the unix commands, you will need to set up a `short-cut' to NT's `CMD.EXE` program and tell the command window to execute a setup script. (NT command windows do not seem to execute a setup script analogous to `` `.cshrc' `` automatically.) Click on the right (menu) mouse button over the Window interface background to bring up the icon menu. Select `New` from this menu and `Shortcut` from the sub-menu which pops up. As the command line, type
-
-
-
```
C:\\winNT\system32\cmd.exe /K C:\usr\local\congruent\setenv C:
```
-

and choose a name for the short cut e.g. Unix-shell. A little icon pops up and when you click on this, it should now start a shell where you can use all of the installed unix commands including the `ntemacs` editor.

- The command shell you just started is still not very intelligent, although you can set up on-line editing and command history, it is not possible to use filename or command completion. `ntemacs` can do this however. To get such a shell, type `ntemacs` and then `ESC x shell`.

Tcsh for NT

If you want a real Unix shell with completion and internationalization, there is a port of the popular `tcsh` program for windows NT. This could be a useful environment for performing system tasks. You can collect a ready compiled binary from `` `ftp:/ftp.blarg.net/users/amol/tcsh' ``. This shell supports all of the usual unix syntax and job control goodies, with the exception of certain signal handlers which NT does not have.

UWIN for NT

David Korn of AT&T (the author of `ksh`, the Korn shell) has written an beautiful package of code for NT which allows you to run many standard unix services for free! There is a development package for porting Unix programs to NT.

Whether you like unix or not, this package is invaluable since it gives you access to `telnet` services, `rsh` and many other goodies. Many of the GNU tools are bundled with this package too. In other words, you can now remote log onto an NT machine and do all of the things you are used to doing on unix except for running window based software.

The Uwin package belongs to AT&T, it is not free software in the sense of having access to source code or having the right to redistribute the package. But this is mainly a formality and anyone can download the libraries. You can collect everything from

```
http://www.research.att.com/sw/tools/uwin
```

Policy decisions and How-Tos

Disk layout

The first thing to learn about a new operating system is where all the important files live on the disk, where you should place new software and where you should install new users.

The layout of the NT filesystem has changed through the different versions, in an effort to improve the structure. This description relates to NT 4.0.

`I386'
: This directory conatins binary code and data for the NT operating system. This should normally be left alone.

`Program Files'
: This is NT's official location for new software. Program packages which you buy will install themselves in subdirectories of this directory.

`Temp'
: Temporary scratch space, like Unix's `/tmp'.

`Winnt'
: This is the root directory for the NT system. This is mainly for operating system files, so you should not place new files under this directory yourself unless you really know what you are doing. Some software packages might install themselves here.

`Winnt\config'

Configuration information for programs. These are generally binary files so the contents of NT configuration files is not very interesting.

`Winnt\system32'
: This is the so-called system root. This is where most system applications and datafiles are kept.

In addition to these files and directories, there is a set of files which is normally invisible called the system registry. The system registry is a place where configuration information, preferences and options are kept. These files contain the analogue of Unix dot-files and `/etc' setup files. These files can only be viewed by starting the registry editor program `winnt\system32\regedt32.exe'.

`HKEY_CURRENT_USER'
: Contains data about the user who is currently logged in on the console.

`HKEY_LOCAL_MACHINE'
: Contains configuration data for the local machine. This contains information about installed software, analogous to Linux's `Packages.gz' file, so that new software can be downloaded or upgraded from a server. It also contains setup information concerning running services, hardware and applications.

`HKEY_USERS'
: This subtree contains user ID's (security ID's) for users who are known to the machine. This is analogous to the `/etc/passwd' and `/etc/shadow' files on unix.

`HKEY_CURRENT_CONFIG'
: Data about the local machine configuration which might change in the future.

`HKEY_DYN_DATA'
: Sub tree containing dynamically created data. For NT's private use only.

The information in the registry is updated mainly by application programs, but can also be edited using the Registry editor mentioned above.

User login directories

NT has no specified place for user directories, so you may place them whereever you like. A naming scheme, such as the one described earlier for Unix disk partitions can still be used even though NT cannot mount disks at arbitrary places in a file system.

It is a little confusing just what intentions Microsoft have for NT as far as networking are concerned. Since each PC is essentially meant for one person, you might want to set up all your files locally on your own machine. If the PC is part of a terminal room where anyone could use the system, then you will probably want to keep all files on a file server.

Mounting disks

Disks must be mounted in NT just as in UNIX, but this occurs more transparently. Disks become visible when you join an appropriate NT domain. This is a nice feature, but there

is a price for the simplification. It is not possible to add a remote (network) disk to your file tree at an arbitrary point, as one can with Unix file systems. Each new device is assigned a drive number, a hang over from the old DOS ways. Disks may be assigned A:, B:, C:, D: etc. Moreover, you don't have any control over what designation a device gets. Two machines might assign different drive names to the same device, so be careful.

Samba

Windows NT uses a system of network file sharing based on their own SMB (Server message block) protocol. `Samba` is a unix daemon based service which makes unix disks visible to Windows NT.

Samba maps usernames, so you need an account with the same name on the NT server and the unix server. It maps username textually, without any fancy security.

Samba configuration is in true unix style, by editing the texfile `` `/etc/smb.conf' ``. Here is an example file.

Note carefully the `hosts allow' line which restricts access to disks to specific IP addresses. Without this option, anyone could fake packets and mess with your disks.

```
[global]
  printing = bsd
  printcap name = /etc/printcap
  load printers = yes
  guest account = nobody
  invalid users = root
  workgroup = UNIX
  hosts allow = 128.39.

[homes]
  comment = Home Directories
  browseable = no
  read only = no
  create mode = 0644

[printers]
  comment = All Printers
  browseable = no
  path = /tmp
  printable = yes
  public = no
  writable = no
  create mode = 0644
```

ACLs and ACEs

Access control lists, or Access control entries are set and checked with either the `Explorer' program (File/Properties/Security/Permissions menu) or the `cacls` command. This command works in more or less the same way as the POSIX `setfacl` command, but with different switches. Also the lists apply to usernames not groups. The switches are

`/G`
: Grant access to user.

`/E`
: Edit ACE instead of replacing.

`/T`
: Act on all files and subdirectories

`/R`
: Revoke (remove) access rights to a user.

`/D`
: Deny access rights to a given user.

The access rights are not read, write, execute as in Unix, but `N` (no rights, `R` (read), `C` (change/write), `F` (full).

```
hybrid> CACLS testfile
C:\home\mark\testfile BUILTIN\Administrators:F
                      Everyone:C
                      MT AUTHORITY\SYSTEM:F

hybrid> CACLS testfile /G ds:F
wait for 30 seconds..
Are you sure(Y/N)?

hybrid> CACLS testfile
C:\home\mark\testfile HYBRID\ds:F
```

In this example the original ACL consisted of three entries. We then replace it with a single entry for user `ds` on the local machine `HYBRID`, granting full rights. The result is shown in the last line. If, instead of replacing the ACE, we want to supplememt it, we write

```
hybrid> CACLS testfile /E /G mark:R
wait for 30 seconds
Are you sure(Y/N)?

hybrid> CACLS testfile
C:\home\mark\testfile HYBRID\ds:F
                      HYBRID\mark:R
```

Unix/NT correspondence

It does not seem to be possible to rename directories under NT. Instead one has to cut and paste them using the explorer program! If you have collected the Unix shell programs, you can use the usual `mv` command.

Project for `iu.hioslo.no`

This project forms the basis for an evaluation/examination for a one-semester course in system administration. When writing your report, you should be as clear as possible. Always say why you make the choices you do and explain clearly how you solve every problem. System administration requires clear skills in structuring information and you will be evaluated also on the clarity of your report. You might want to make a box or special page for each specific problem you solve, with the format

```
Problem:

Solution:

Why we chose this approach:
```

1. Your first task is to analyse the present state of the local network as suggested in section 1.3. (Use the network of your college department for this part of the project.) The aim of this part of the project is to learn about the system the way it is set up, so that when you reinstall a system on the network from scratch, it will fit in with the remaining machines smoothly. You should always do this when you plan to install a system. You should collect and present the following information as clearly as possible: @itemize @bullet @item The names and IP addresses of all hosts on your network. Use DNS (`nslookup`) to help you. @item Find out about the network hardware on your network: where is the router located? Are there any bridges or repeaters? @item Identify which services are in operation (WWW, DNS, NIS, NISPLUS, anonymous FTP etc) and which hosts are servers for those services. @item You should find out which hosts have disks and which hosts mount those disks as clients (as in the previous point). @item Find out the domain name of your network (NIS and DNS might have separate domain names). @item Find the netmask and broadcast address for the subnet. @item Find and examine the cfengine configuration files for the local domain. Explain how the work in general terms, without going into too much detail. @end itemize **You should now have a convenient overview of the network and its various server/client relationships.**
2. You will now be assigned one host with hostname and IP address. It will be your task to install the operating system on this system. You will then set up the system as a standalone system for a while. Then finally you will remove your standalone setup and use the central cfengine files for `iu.hioslo.no` to integrate your

machine into the network. The aim of this part of the project is to learn what is required to manually run a system with superuser privileges, to automate some of those tasks, and then finally to give up those privileges and integrate the host into the existing network. *If you have two unix machines at home, you can do most of this with your own machines at home.* @itemize @bullet @item Install the OS now. @item Set a root password immediately and keep it secret! @item Explain how you would add an entry for your host into the DNS tables under @smallexample /iu/nexus/local/iu/named @end smallexample and the host table under @smallexample /iu/nexus/local/iu/etc/hosts @end smallexample @item Locate the filesystem table on your system. @item Locate the export file on your system. @item All files which you create on your system should be kept clearly separate from the OS you have installed. Keep them under a directory root called `` `/iu/*your-hostname*' ``. Create this directory and any subdirectories you need. Explain what you are storing in all the directories you create (see next point) and briefly why you choose the structure you do. @item Write a perl or shell script to add new users to your system. You should grant accounts on your system to the other groups, if they ask. Your script should assign a unique user ID to each user and build the login directory. You should create a default `` `.cshrc' `` or `` `.profile' `` file for new users. Home (login) directories should be under @smallexample /iu/*hostname*/home @end smallexample or equivalent name. @item Use ftp to `` `ftp.iu.hioslo.no' ``, directory `` `pub' ``, to fetch the program package cfengine. If you are doing this part of the project at home, you might need to take it home on a diskette. Compile and install this software package. Do not print out the manual. If you want a copy of this, you can borrow a copy or look at the on-line version: @smallexample http://www.iu.hioslo.no/~mark/cfengine @end smallexample @item Write a cfengine configuration file which links `` `/home' `` to your home directory @smallexample /iu/*hostname*/home @end smallexample and which monitors the permissions of the password file. @item Join forces with the other groups and create one file which integrates the configurations of all hosts. Use a group or class name to make general rules. One group should act as keeper of the master file version. @item Make a directory containing exportable data (could contain anything which you might share with others) called `` `/iu/*hostname*/export' `` and export this to the other groups' hosts. The other groups will do the same. @item Add the other groups' exported filesystems to your cfengine configuration so that cfengine mounts these filesystems automatically. @item Imagine you now wanted to have the same password on every system on the network. Sketch out a *brief* plan for two ways in which you might implement such as system: one using NIS and one without NIS. (You do not need to implement this in practice). @end itemize

Bibliography

1. *DNS and BIND*, Paul Albitz and Cricket Liu, O'Reilly & Assoc.
2. *TCP/IP Network administration*, Craig Hunt, O'Reilly & Assoc.

3. *Practical UNIX security*, Simson Garfinkel and Gene Spafford, O'Reilly & Assoc.
4. *Building Internet Firewalls*, D.B. Chapman and E.D.Zwicky, O'Reilly & Assoc.

Glossary

booting
: Bootstrapping a machine. This comes from the expression `to lift yourself by your bootstraps', which is supposed to reflect the way computers are able to start themselves from scratch.

BIND
: Berkeley Internet Name Domain. The library part of DNS, the routines which perform nameservice lookups.

C/MOS
: Complementary Metal Oxide Semiconductor. p-n back-to-back transistor technology, low dissipation.

Cyberspace
: William Gibson's name for the virtual world of the net. From his novel Neuromancer.

DNS
: The Domain Name Service, which converts internet names into IP addresses and vice versa.

GUI
: Graphical user interface.

IP address
: Internet address. Something like `128.39.89.10`

Metaverse
: Neal Stephenson's virtual world of the internet. A cooler name for cyberspace than `cyberspace'. Also his vision was deeper. From his novel Snow Crash.

RAID
: Redundant array of inexpensive (ha-ha) disks. A disk array with automatic redundancy and error correction. Can tolerate a failure of one disk in the array without loss of data.

SCSI
: Small Computer Systems Interface. Used mainly for disks on multiuser systems and musical instruments.

SIMM
: Memory chip arrays.

Striping
: A way of spreading data over several disk controllers to increase through out. Striping can be dangerous, since files are stored over several disks, meaning that if one disk fails, all data are lost.

SVR4
: System 5 release 4 unix. AT&T's code release.

TTL
: Time to live/Transistor-Transistor Logic

X11
: The Unix windows system.

Index

a

b

c

d

e

f

g

h

i

k

l

m

n

o

p

q

r

S

t

u

v

w

x

This document was generated on 29 May 1998 using the texi2html translator version 1.50.