Principles of system administration

Edition 2.1

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

Foreword

For many years system administration has been passed on to new generations through manual pages, technical handbooks and by word of mouth. In all but the most disciplined institutions this has been a haphazard affair with a disregard for theory and no common standard of practice. Put simply, no one has seriously considered system administration to be a worthy academic pursuit. In recent years, the arrival of the internet has made this belief untenable however. The level of complexity of networked operating systems together with the increasing problem of intrusion (`cracking') has properly elevated the status of system administration to that of a discipline in its own right.

This text is an introduction to the aims and guiding principles of system administration. In this respect it is not like other books which focus mainly on procedures. It needs to be used together with any and all information which you find elsewhere. It's main aim is to illuminate a sound and logical way through the murky swamps of networked operating systems. It does not replace other books nor is it like any other. It is aimed at students in a College environment, or apprentice administrators not at system administrators working in the field.

The course assumes a basic familiarity with Windows and Unix as well as a core knowledge of basic operating system concepts from the two preceding courses *Operating systems* and *Unix programming* . It is intended to run intensively over one semester. You will be required to study an existing network, then set up a GNU/Linux machine and integrate it into the network. You will learn system administration by trial and error. No one should underestimate the value of making errors during the learning process. This is often the only way to really understand.

The practical work for the course is based on the Debian distribution of the GNU/Linux operating system. This choice has been made in order to maximize the understanding of the important issues rather than the ease or simplicity of installation or of use. i.e. the aim is to gain experience, not to make things as easy as possible.

Much of the section on security has been influenced by the thoughtful words exchanged at the USENIX/LISA conferences. Comments on this course, observing suitable protocols and etiquette, to Mark.Burgess@iu.hioslo.no

/M\, Oslo, Dec 1998

PART I: THEORY

Theory

What is system administration about?

@hrule @vskip 0.3cm *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 job of a system administrator is complex. 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 learn a hundred new things a year.

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 and there is a queue of people waiting for help. 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, but there is often a wrong answer. It's about making something robust which works. In spite of stereotypes, today's system administrator is neither haphazard nor messy. Today's computing systems require the very best of organizational skills and the most professional of attitudes.

It's also a matter of self-confidence. In the beginning a novice will be too nervous to do any real work, later, when more experienced, one learns to relax and just do things. It's like dating 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 GNU/Linux on a PC, you might think that you know a lot about system administration, but in fact you know only the very 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 something about how your own machine works.

This book is an introduction to the *concepts* of system administration. As a system administrator, your first responsibility is to the greater network community and then to the users of your system. An administrator's job is to make their lives bearable and to allow them to perform productive work..

A philosophy

How does one begin and how does one behave? What are bad habits and what are good habits? The value of an adequate theoretical understanding should not be underplayed. The days when one could fly a system by the seat of one's pants are over. There is a strong need for practical skills, but nothing can replace real understanding. Today, keeping up with the theory is as important as upgrading the software. One also needs to approach the problem correctly. The following are ill-advised:

- Insisting that there is a right or a wrong answer to every question.
- Running to someone else whenever there is a problem.
- Getting fraught and upset when things do not work the way one expects.
- Expecting that every problem has a beginning, a middle and an end.

In contrast it is recommended to begin

- Looking for answers in manuals and newsgroups.
- Using trial and error (carefully!) to locate problems.
- Practice being cheerful and patient with the users of your system and the machines.
- Listening to people who tell 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.
- Taking responsibility for one's 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 tedious jobs like vacuum cleaning the hardware once a year.

However much one might like one's own language, it is important to get used to the English language. English (actually American) is the language of the net. System administrators need it to be able to read documentation, to be able to communicate with others and to ask questions on the Internet.

It is important to start by being systematic from the beginning. Every time you read something, or learn some isolated fact, think: *how does this apply to me?* Do yourself a favour and buy an A-Z that you can use to write down your experiences. Start now. And try to find your own method for remembering what you have learned.

What are the challenges of system administration?

System administration is not about installing operating systems. It is about planning and designing an efficient network of computers so that real *users* will be able to get their jobs done. That means:

- Designing a network which is logical and efficient.
- Deploying large numbers of machines which can be easily upgraded later.
- Deciding what services are needed.
- Planning and implementing adequate security.
- Providing a comfortable environment for users.
- Developing ways of fixing errors and problems which occur.
- Keeping track of and understanding how to use the enormous amount of knowledge which increases every year.

Some system administrators are responsible for both the hardware of the network and the computers which it connects, i.e. the cables as well as the computers. Some are only responsible for the computers. Either way, an understanding of how data flow from machine to machine is essential as well as an understanding of how each machine affects every other. A central axiom for network administration is:

A computer which is plugged into the network is not just yours. It is part of a society of machines which shares resources and communicates with the whole. What your machine does affects other machines. What other machines do affects your machine.

The ethical issues associated with connection to the network are not trivial. Administrators are responsible for their organization's conduct to the entire rest of the Internet. This is a great responsibility which must be borne wisely.

Is system administration a career?

The internet has grown considerably in the last ten years and operating systems have grown more and more complex. Unfortunately the number of qualified system administrators has not grown in proportion, so there is a gap.

There is currently quite an appreciable market for consulting services in security and automation of system administrative tasks. Remember that most companies have more money than time or expertise and that they need to be able to *buy* something in order to satisfy their board members and accountants. For a business, doing something to address a problem means spending some money. That means that they are looking to pay someone to carry out a service. Since the best system administration tools are free companies actively seek to pay consultants to set up and maintain administration tools for them. Not only is system administration a very interesting line of work (it is extremely varied), it is also very satisfying.

NEVER-DOs in system administration

Some rules of thumb are useful to prevent embarrassing accidents.

- The administrator or root account has unlimited privileges. Never log into the system as the root user. Use the `su` command to gain root privileges when you need them and then quit at once. This is because one should never run ordinary programs with root privileges. If one does, it is possible to allow the system to be invaded by viruses or to cause accidental damage to the system.

- Never leave root shells on the console. It is possible to accidentally do something destructive without realizing that one has root privileges. (Put the sledgehammer down when you are not using it, Eugene!)
- Never leave root shells running so that others might gain access to them in an open room.
- Never leave services running if they are not used for anything. They provide a possible back-door into the system for intruders.
- Never give users physical access to a machine which stores important data. If users can touch the system, it's theirs.
- Windows 95, Windows 98 and the MacIntosh are inherently insecure systems. They cannot be secure by virtue of their design (they have no access control of any kind--all access is privileged). When setting up a network in a potentially hostile environment, use an operating system like Unix or NT.

Know your network

System administration requires its operatives to know a lot of facts about hardware and software. The road to real knowledge is long and hard, so how does one get started?

A top down approach is useful for understanding the network interrelationships. The best place to start is at the network level. In most daily situations, one starts with a network already in place--i.e. one doesn't have to build one from scratch. It is important to know what hardware one has to work with and where everything is to be found; how it is organized (or not) and so on. Here is a checklist:

- How does the network fit together? (What is its topology?)
- Which function does each host/machine have on the network?
- Where are the key network services?

Having thought about the network as a whole, one can begin to think about individual hosts/machines. First there is a hardware list.

- What kind of machines are on the network? What are their names and addresses 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 the network? MS-DOS, Novell, NT or Unix (if so which Unix? GNU/Linux, Solaris, HPUX?)
- What kind of network cables are used? Is it thin/thick Ethernet? Is it a star net (hubs/twisted pair), or fiber optic FDDI net?
- Where are hubs/repeaters/the router or other network control boxes located? Who is responsible for maintaining them?
- What is the hierarchy of responsibility?

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 the local timezone?
- What broadcast address convention is used? 255 or the older 0?
- 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

Finding and recording this information is an important learning process. The information will change as time goes by. Networks are not static; they grow and evolve with time.

Health

Frequent computer users are not usually aware of how they can be damaging their own health. Unlike cigarettes, computers do not have a government health warning. Whether or not this is an issue for system administrators is open for discussion, but often the system administrator is the only person who thinks about the users and the hardware they use. Certainly every administrator needs to look after his/her own health and, along the way, it is natural to think of the health of others. Fortunately it is not difficult to avoid the worst problems.

Eyes should be protected, We only have one pair and they must last our entire lives. 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 one touches the screen one notices that it is charged with static electricity. The effect of this is to charge dust particles and throw them out into users' faces. 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 reflection. If there is a light source behind a user, it will reflect in the screen and the eyes will be distracted by the reflection. The image on the 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). This confuses the eyes into focusing back and forth between the reflection and the image. 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 a user can make.

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

Back. The back (spine) is one of the most complex and important parts of the body. It supports the upper body and head, and is attached to the brain (where applicable). The upper body is held up by muscles in the stomach and lower back. If these muscles are relaxed by slouching for long periods unnecessary strain in placed on muscles and bones which were not meant to bear the weight of the 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. Don't sit in a draft. Cold air blowing across the 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 lessened by making sure that users do not sit too far away from the desk where the mouse lies and by having a support for the mouse forearm. The ultimate solution is simple: don't use the mouse. Use of the keyboard is far less hazardous. Learning keyboard shortcuts is good for prolonged work.

Pregnancy and cancer. Some studies recommend that pregnant women wear protective aprons when sitting in front of computer screens.

Generally. Users should not sit for long periods without taking a break. Look away from the screen (to a far away object) at regular intervals relaxes the eyes. Walking around exercises the back and relaxes the shoulders. Use of anti-static, anti-reflective screens is recommended.

Weather

Weather and environment affect computers.

Lightning
: strikes can destroy fragile equipment. No fuse will protect hardware from a lightning strike. Transistors and CMOS chips burn out much faster than any fuse. Electronic spike protectors can help here.

Power
: failure can cause disk damage and loss of data. A UPS (Uninterruptible power supply) can help.

Heat.
: The blazing summer heat or a poorly placed heater oven can cause systems to overheat and suddenly black out. One should not let the ambient temperature near a computer to rise much above about 25 degrees Centigrade. Heat can cause RAM to operate incorrectly and disks to misread/miswrite. Good ventilation is essential for computers and screens for avoiding electrical faults.

Cold.
: Sudden changes from hot to cold are just as bad. They can cause unpredictable 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

Although even the most stoical administrator's convictions might occasionally be called into question, system administration is a social service and it is important to remain calm and reasonable. Users frequently believe that the system administrator has nothing better to do than to answer every question and execute every whim and fancy. Dealing with users is not a small task.

Ethics and Responsibility

A system administrator wields great power. He or she has the means to read everyone's mail, change anyone's files, to start and kill anyone's processes. This power can easily be abused and that temptation could be great.

Another danger is the temptation for an administrator to think that the system exists primarily for him or her and that the users are simply a nuisance to the smooth running of things; if network service is interrupted, or if a silly mistake is made which leads to damage in the course of an administrator's work, that is okay: the users should accept these mistakes because they were made trying to improve the system. When wielding such power there is always the chance that such arrogance will build up. Some simple rules of thumb are useful.

The ethical integrity of a system administrator is clearly an important issue. Administrators for top secret government organizations and administrators for small businesses have the same responsibilities towards their users and their organizations. One has only to look at the governing institutions around the world to see that power corrupts. Few individuals, however good their intentions, are immune to the temptations of such power at one time or other. As with governments it is perhaps a case of: those who wish the power are least suited to deal with it.

Administrators `watch over' backups, email, private communications and they have access to everyone's files. While it is almost never necessary to look at a user's private files, it is possible at any time and users do not usually consider the fact that their files are available to other individuals in this way. Users need to be able to trust the system and its administrator.

- What kind of rules can you fairly impose on users?
- What responsibilities do you have to the rest of the network community, i.e. the rest of the world?
- Censoring of information or views
- Restriction of personal freedom
- Taking sides in personal disputes
- Extreme views (some institutions have policies about this)
- Unlawful behaviour

Objectivity of the administrator means avoiding taking sides in ethical, moral, religious or political debates, in the role of system administrator. Personal views should be kept separate from professional views. However, the extent to which this is possible depends strongly on the individual and organizations have to be aware of this. Some organizations dictate policy for their employees. This is also an issue to be cautious with: if a policy is too loose it can lead to laziness and unprofessional behaviour; if it is too paranoid or restrictive it can lead to bad feelings in the organization. Historically, unhappy employees have been responsible for the largest cases of computer crime. Other references:

** http://www.acm.org/constitution/code.html*
** http://www4.ncsu.edu/unity/users/j/jherkert/ethics.html*

Bugs

Operating systems and programs are full of bugs. Learning to tolerate bugs is a matter of survival for system administrators. If one is lucky enough to be using free software from the net, these bugs will usually be solved quickly and one can eliminate them by upgrading. If one uses commercial software it will probably be necessary to wait a lot longer for a patch. Either way, one must be creative and work around these bugs. Bugs can be caused by many things. 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 life-forms.

Operating systems

@hrule @vskip 0.3cm

For an operating system to be managed consistently it has to be possible to prevent its destruction by restricting the privileges of its users. Different operating systems vary in their provisions for restricting privilege. In operating systems where any user can change any file, there is little or no possibility of gaining true control over the system. Any accident or whim on the part of a user can make uncontrollable changes.

Insecure operating systems

The home computer revolution was an important development which spread cheap computing power to a large part of the world. As with all rapid commercial developments, the focus in developing home operating systems was on immediate functionality not on planning for the future. The home computer revolution preceded the network revolution by a number of years and home computer operating systems did not address security issues. Operating systems developed during this period include Windows, MacIntosh, DOS, Amiga-DOS. All of these systems are completely insecure: they place *no limits* on what a determined user can do.

Fortunately these systems will slowly be replaced by operating systems which were designed with resource sharing (including networking) in mind. Still, there is a large number of insecure computers in use and many of them are now connected to the network. This should be a major concern for a system administrator. In an age where one is forced to take security extremely seriously, leaving insecure systems where they can be accessed

- physically
- by network

is a potentially dangerous situation. Such machines should not be allowed to hold important data and they should not be allowed any privileged access to network services. We shall return to this issue in the chapter on security.

Administrator/root privilege

To distinguish them from insecure operating systems we shall refer to operating systems like Unix and NT as *secure* operating systems. This should not give the impression that Unix and NT are completely secure by any stretch of the imagination: complete security is a fairy tale, a pipe dream, it will never happen in any operating system. Nevertheless, these operating systems do have the mechanisms which make a basic level of security possible.

The most fundamental tenet of security is the ability to restrict access to certain parts of the system's resources. The main reason why DOS, Windows 9x and MacIntosh are so susceptible to virus attacks is because any user can change the operating system's files. Properly configured and bug free Unix/NT systems are theoretically immune to such attacks because ordinary users do not have the privileges required to change system files. Unfortunately the key phrases *properly configured* and *bug-free* highlight the flaw in this theory.

In order to restrict access to the system we require a notion of *ownership* and *permission* . Ordinary users should not have access to the hardware devices of a secure operating system's files, only their own files, and then they will not be able do anything to compromise the security of the system. System administrators need access to the whole system in order to watch over it, make backups and keep it running. Secure operating systems thus need a privileged account which can be used by the system administrator when he/she is required to make changes to the system. In Unix the privileged account is called *root* . In NT, the *Administrator* account is similar to Unix's root, except that the administrator does not have automatic access to everything as does root. Instead he must be first granted himself access to an object. However the administrator always has the right to grant him or herself access to a resource so in practice this feature just adds an extra level of caution.

Administrator and root accounts should not be used for normal work, since they have far too much power. To use these an normal user accounts would be to make the systems as insecure as the insecure systems we have mentioned above. In summary: *It is desirable to restrict users' privileges for the greater good of everyone on the network.*

System identities

Whenever more than one computer is coupled together, there is a need for each computer to have a unique identity. As it turns out, this need has been recognized many times in different contexts and the result is that todays computer systems can have many different names which identify them in different contexts. The result is a confusion. For internet enabled machines, the IP address of the host is usually sufficient for most purposes.

Host ID
: Circuit board identity number. Often used in software licensing.

Install name
: Configured at install time. This is often compiled into the kernel, or placed in a file like `` `/etc/hostname' ``.

Application level name
: Any name used by application software when talking to other hosts.

Local mapping
: Originally the Unix `` `/etc/hosts' `` file was used to map IP addresses to names and vice versa.

Network Information Service
: A local area network database service developed by Sun Microsystems.

Transport level address(es)
: Each network interface can be configured with an IP address. This numerical converts into a text name through some naming service.

Network level address(es)
: Each network interface (Ethernet/FDDI etc) has a hardware address burned into it at the factory. Some services (e.g. RARP) will turn this into a name or an IP address through a secondary naming service like DNS.

DNS name(s)
: The name returned by a Domain name server (BIND) based on an IP address key.

WINS name(s)
: The name returned by a WINS server (Microsoft's name server) based on IP address.

In order to choose from all of these, it is necessary to understand why they are required (if at all). The host ID and network level addresses simply exist. They are unique and nothing can be done about them, short of changing the hardware. For the most part they can be ignored. The network level address is used by the network transport system for end-point data delivery, but this is not something which need concern most system administrators. The network hardware takes care of itself.

At boot time, each host needs to obtain a unique identity. In today's networks that means a unique IP address and an associated name for convenience. The only purpose for this name is to uniquely identify the host amongst all of the others on the world-wide network. Although every host has a unique Ethernet

address or token ring address, these addresses do not fall into a hardware-independent hierarchical structure. In other words Ethernet addresses cannot be used to route messages from one side of the planet to the other in a simple way. In order to make that happen, a system like TCP/IP is required. At boot-time then each host needs to obtain an identity. It has two choices

- Ask for an address to be provided from a list of free addresses. (DHCP or BOOTP protocols)
- Always use the same address, stored on its system configuration files. (Requires correct information on the disk)

The first of these possibilities is useful for terminal rooms containing large numbers of identical machines. In that case, the specific IP address is unimportant as long as it is unique.

The second of these is the preferred choice for any host which is special in some way, particularly hosts which provide network services. Network services should always be at a well-known location.

From the IP address a name can be automatically attached to the host through an internet naming service. There are several services which will perform this conversion. DNS, NIS, WINS. In these remaining cases one needs to make a choice. As far as anyone on the network is concerned a host is its IP address. Any names which are used internally by the kernel are quite irrelevant. The only difficulty with naming conflicts is that they can cause internal problems. This is operating system dependent but it is always best not to risk it.

The only world-wide service is DNS (Domain Name Service) whose common implementation is called BIND (Berkeley Internet Name Domain); this should always be used. This service binds a local network to the world-wide internet in several important ways. WINS (Windows Internet Name Service) is a proprietary system built by Microsoft for NT. Any host can register data in this service; it is completely insecure and is therefore inadvisable in any network which is trusted in some manner (either to provide a service, or to hold important data). DNS (which already existed prior to WINS) is a better option regardless. WINS will be abandoned as of NT 5.0. NIS also provides a local service which has multiple database functions, not just hostname/IP address conversion. It can also be avoided in favour of DNS for hostname mapping.

Under NT each system has a textual name which is chosen during the installation. A domain server will provide an SID (security ID) for the name which helps prevent spoofing. When NT boots it broadcasts the name across the network to see whether it is already in use. If the name is in use, the user of the workstation is prompted for a new name(!)

Quotas

Although one should not forget that computers exist for users, it is also important to understand that users are often the biggest threat to the stability of their computers. Two generations have now grown up with computers in their homes, but these computers were private machines which were not (until recently) attached to a network. They were not part of an organization with many users--they were used by perhaps one or two family members. In short users have grown up thinking that what they do with their computers is their business. That is not a good attitude in a network community.

In a shared environment, all users share the same machine resources. If one user is selfish that affects all of the other users. Given the opportunity users will consume all of the disk space and all of the memory and CPU cycles somehow, whether through greed or simply through inexperience. Thus it is in the interests of the *user community* to limit the ability of users to spoil things for other users.

One way of protecting operating systems from users and from faulty software is to place quotas on the amount of system resources which they are allowed.

Disk quotas
: place fixed limits on the amount of disk space which can be used per user. The advantage of this is that the user cannot use more storage than this limit; the disadvantage is that many software systems need to generate/cache large temporary files (e.g. compilers, or web browsers) and a fixed limit means that these systems will fail to work as a user approaches his/her quota.

CPU time limit
: Some faulty software packages leave processes running which consume valuable CPU cycles to no use. Users of multiuser computer systems occasionally steal CPU time by running huge programs which make the system unusable for others. The C-shell `limit cputime` function can be globally configured to help prevent accidents.

Policy decisions
: Users collect garbage. To limit the amount of it, one can specify a system policy which includes items of the form: `Users may not have mp3, wav, mpeg etc.. files on the system for more than one day'. To enforce such a policy, See section Controlling user resources.

Quotas have an unpleasant effect on system morale, since they restrict personal freedom. They should probably only be used as a last resort. There are other ways of controlling the build up of garbage, See section Controlling user resources.

Internationalization

Internationalization support in computing involves three issue:

- Choice of keyboard: American, British, German, Norwegian, Thai...etc.
- Fonts: Roman, Cryllic, Greek, Arabic, far east...etc.
- Translating program text messages into native language.

Inexperienced computer users usually want to be able to use computers in their own language. Experienced computer users often prefer the American versions of keyboards and software.

Logging and Auditing

Operating system kernels share resources and offer services. They can be asked to keep lists of transactions which have taken place so that one can later go back and see exactly what happened at a given time. This is called logging or auditing.

Full system auditing involves logging every single operation which the computer performs. This consumes vast amounts of disk space and CPU time and is generally inadvisable unless you have a specific reason to audit the system. Part of auditing used to be called system accounting from the days when computer accounts really were accounts for real money. In the mainframe days, users would pay for system time in dollars and thus accounting was important since it showed who owed what.

Auditing has become an issue again in connection with security. Organizations have becomes afraid of break-ins from system crackers and want to be able

to trace the activities of the system in order to be able to look back and find out the identity of a cracker. The other side of the coin is that system accounting is so resource consuming that the loss of performance might be more important to an organization than the threat of intrusion.

Executing jobs at regular times

The ability of a host to execute jobs at predetermined times is very important in automating the task of system administration. 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  Shellcommand
```

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
#
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/local/sbin/daily   2>&1 | mail root
30 3 * * 6         root /bin/sh /usr/local/sbin/weekly  2>&1 | mail root
30 5 1 * *         root /bin/sh /usr/local/sbin/monthly 2>&1 | mail root
```

A version of cron has been ported to Windows.

File system planning

Most operating systems have hierarchical file systems with directories and subdirectories. This is a powerful tool for organizing data. Disks can also be divided up into partitions. In many operating systems the largest supported partition size is 4GB since that is the maximum size which can be represented in a 32 bit register. The point behind directories and partitions is to separate files so as not not mix together things which are logically separate. For example,

- User home directories
- Development work
- Commerical software
- Free software
- Local scripts and databases

One of the challenges of system design is in finding an appropriate directory structure for all data which are not the operating system.

Data which are separate from the operating system should be kept in a separate directory tree, preferably on a separate disk partition. If they are mixed with the operating system file-tree if makes re-installation or upgrade of the operating system unnecessarily difficult.

Apart from anything else, it makes no sense to mix logically separate file trees. Operating systems often have a special place for installed software. Regrettably they often break the above rule and mix software with the operating system's file tree. In Unix machines the place for installed software is traditionally `` `/usr/local' ``; fortunately in Unix separate disk partition can be placed anywhere in the file tree on a directory boundary, so this is not a practical problem as long as everything lies under a common directory. In NT software is often installed in the same directory as the operating system itself; also NT does not support partition mixing in the same way as Unix so the re-installation of NT means re-installation of all the software as well.

Data which are installed or created locally are not subject to any constraints however: they may be installed anywhere. One can therefore find a naming scheme which gives the system logical clarity. This benefits users and management issues. Again we may use directories for this purpose. Operating systems which descended from DOS also have the concept of drive numbers like A:, B:, C: etc. These are assigned to different disk partitions. Some Unix operating systems have virtual file systems which allow one to add disks transparently without ever reaching a practical limit. Users never see partition boundaries. This has both advantages and disadvantages since small partitions are a cheap way to contain groups of misbehaving users, without resorting to disk quotas.

Each operating system has a model for laying out its files in a standard pattern, but user files are usually left unspecified. System administrators are usually made responsible for both the safety and integrity of data on a network of computers. Choosing a sound layout for data can make the difference between an

incomprehensible chaos and a neat orderly structure. An orderly structure is useful not only for the users of the system, but also when making backups. Some relevant issues are:

- Disk partitions are associated with drives or directory trees when connected to operating systems. These need names.
- Naming schemes for files and disks are operating system dependent.
- The name of a partition should reflect its function or contents.
- In a network the name of a partition ought to be a URL i.e. contain the name of the host.
- It is good practice to consolidate file storage into a few special locations rather than spreading it out all over the network. Data kept on many machines can be difficult to manage, compared to data collected on a few dedicated file servers. Also, insecure operating systems offer files on a local disk no protection.

The URL model of file naming has several advantages. It means that one always knows the host-provider and function of a network resource. Also it fall nicely into a hierarchical directory pattern. A simple but effective scheme is to use a three level mount-point for adding disks: each user disk is mapped onto a directory with a name of the form

```
/site /host /purpose
```

This scheme is adequate even for large organizations and can be extended in obvious ways.

In DOS-derived operating systems one does not have the freedom to `mount' network filesystems into the structure of the local disk, network disks always become a special `drive', like H: or I: etc. It is difficult to make a consistent view of the disk resources with this system, however it is rumoured that NT 5.0 will have this possibility and one can already use filesystems like the DFS on NT which do support this model.

Within an organization a URL structure provides a global naming scheme, like those used in true network filesystems like AFS and DFS. These use the name of the host on which a resource is physically located to provide a point of reference. This is also an excellent way of labelling backups of partitions since it is then immediately clear where the data belong. A few rules of thumb allow this naming scheme to live painlessly along side traditional Unix naming schemes.

- When mounting a remote filesystem on a host, the client and server directories should always have exactly the same name. Anything else only causes confusion and problems later.
- The name of every filesystem mount-point should be unique and tell us something meaningful about where it is located and what its function is.
- One can always make links from special unique names to more general names like `/usr/local'.
- It doesn't matter whether software compiles in the path names of special directories into software as long as you follow the points above.

For example, the following scheme is used in Oslo at the University and at the College. The first link in the mount point is the part of the organization or university faculty which the host belongs to, the second link is the name of the host to which the disk is physically connected, and the third and final link is a name which reflects the contents of the partition. Some examples:

```
/site /hostname /purpose

/research/grumpy/local
/research/happy/home1
/research/happy/home2

/sales/slimy/home1

/physics/einstein/data
/biology/pauling/gene-db
```

The problem of drive names in NT and Windows is an awkward one if one is looking for Unix/NT interoperability. In this case Sun's PCNFS might be an answer. In practice many networks based on NT and Windows will use Microsoft's model throughout, and while it might not gleam with elegance it does the job. The problem of backups is confined to the domain servers, so the fact that Windows is not a fully distributed operating system restricts the problem to manageable proportions.

Networks

@hrule @vskip 0.3cm

During the 1970's it was realized that expensive computer hardware could be used most cost-efficiently (by the maximum number of people) if it was available *remotely* , i.e. if one could communicate with the computer from a distant location. Inter-system communication became possible through the use of modems and UUCP and later wide area networks.

The large mainframe computers which served sometimes hundreds of users were painfully slow for interactive tasks, although they were efficient at time-sharing. As hardware became cheaper many institutions moved towards a model of smaller computers coupled to file-servers and printers by a network. This solution was relatively cheap but had problems of its own. At this time the demise of the mainframe was predicted. Today, however, mainframe computers are very much alive for computationally intensive tasks, while the small networked workstation provides access to a world of resources via the internet.

Dealing with networks is one of the most important aspects of system administration today. In order to be a system administrator it is necessary to understand how and *why* networks are implemented, using a world-wide protocol: the internet protocol family.

Terminology

Although it changes fairly frequently, the popular jargon of system administration is useful to know when reading the commercial press. Here we define a few terms which can be found in the literature.

Vendor
A company which sells hardware or software. A *seller* .

PC An Intel based personal computer, used by a single user.

Workstation
A non-Intel based personal computer which might be used by several users. Workstations might be based on for example SPARC (Sun Microsystems) or Alpha (Digital/Compaq) chip sets.

Enterprise
A small business network environment. Enterprise management is a popular concept today because NT has been aimed at this market. If nothing else, Microsoft know about marketing and have highlighted this important market far more effectively than the Unix vendors have done before them. Enterprise management typically involves running a web server, a database, a disk server and a group of workstations and common resources like printers and so on. Many magazines think of enterprise management as the network model, but when people talk about Enterprise management they are really thinking of small businesses with fairly uniform systems.

LISA Large Installation System Administration. This refers to environments with many (hundreds or thousands of) computers. The environments typically consist of many different kinds of system from multiple vendors. These systems are usually owned by huge companies, organizations like NASA or universities.

Consolidated system
A centralized mainframe type of solution for concentrating computing power in one place. This kind of solution makes sense for heavy calculations, performed in engineering and for computer graphics.

Distributed system
A de-centralized solution, in which many workstations spread the computing power evenly throughout the network.

Open systems
is a concept promoted originally by Sun Microsystems for Unix. It is about software systems being compatible through the use of freely available standards. Competitors are not prevented from knowing how to implement and include a technology in their products or from selling it under license.

Proprietary systems
is the opposite of open systems. These systems are secret and the details of their operation is not disclosed to competitors. The most infamous example is Microsoft, but there are many others.

Domains
A domain is a logical group of hosts. This word is used with several different meanings in connection with different software systems. The most common meaning is connected with DNS, the Domain Name Service. Here a domain refers to an Internet suffix, like `.iu.hioslo.no', or `.nasa.gov'. Internet domains denote organizations. Domain is also used in NT to refer to a group of hosts sharing the attributes of a common file server. Try not to confuse Domain nameserver (DNS) server with NT Domain server.

Finally, it is important to distinguish between a user interface and an operating system. A window system is a *graphical user interface* (GUI), an operating system shares resources and provides functionality. This issue has been confused by the arrival of operating systems called Windows and including a graphical user interface. An operating system can be good or bad independently of whether its windowing system(s) is/are good or bad.

Common network sharing models

Different operating systems support different ideas about how networks should be used.

Unix Unix is not tied to any specific model for utilizing network resources. Any Unix system can perform any function, as server, client or both. A Unix network is fully distributed, there is no requirement about centralization of resources, but central models are commonly used. Unix is a multitasking, multiuser system. It is is the most broadly configurable of the popular operating systems and many tools are freely available for administratrating it. Unix contains tools for making many hosts work together and share resources, but each host can also be configured as a standalone system. Each host either has a fixed IP address, or can be assigned one automatically at boot time by a service such as *BOOTP* or *DHCP* with *ARP/RARP* .

NT There are two types of NT system with separate software licenses: *workstations* and *servers* . NT revolves around a model in which programs are run on a local workstation, but where network services are run on a centralized server. All systems must run NT or understand the proprietary protocols which NT uses. IP addresses are fixed or may be assigned automatically by a network service such as *BOOTP* or *DHCP* . Several NT servers can coexist. Each server serves a `domain' and clients subscribe to as many domains as they wish. NT is not a distributed system in the sense that services are localized on server machines. NT supports two kinds of organizational groups: *workgroups* in which hosts share a simple peer-to-peer network, perhaps with Windows 9x machines, and *domains* which have a central authority through a domain server. Domains provide a common framework including user-id's (SID's in NT language), passwords and user profiles. Domains have a common user-database and a common security policy. Any host which subscribes to a domain inherits the users and the security policy of the domain.

Novell The Novell Netware software is not an operating system. It is a network server for PCs which adds file storage, printing and other network services on top of Windows, DOS, MacIntosh or GNU/Linux. The network protocol for local traffic is IPX which is faster than IP and is an inter-networking protocol, but it is not a world wide protocol, thus Novell run PC's still need IP configurable interfaces. Each PC can also obtain an IP address dynamically from a single server which replies to a BOOTP or DHCP broadcast request. Several Novell file servers can coexist. All services run on these servers (which support a form of multitasking). The server is independent of the basic operating system run on the PCs it supports, 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. 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 and IPX. 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.

Recently, several companies (e.g. Auspex, Network Appliance) have begun producing solutions for integrating disk storage both for Unix and for Windows worlds. IBM has traditionally produced software which works both on Unix and Microsoft platforms.

Another issue for networked systems is where configuration (preference) data should be stored. There are two possibilities here which correspond approximately to the Unix approach and the approach used by all other systems.

Windows/Mac/Personal
Under Windows (9x/NT) and MacIntosh systems, each user is assumed to have his or her own personal workstation which will not normally be used by other users. Configuration data or preferences which the user selects are thus stored locally on the system disk in a location provided by the operating system. This location is common to all users. Only NT distinguishes between different users.

Unix/Shared
Under Unix, each user sets up personal preferences in his or her personal *dot files* which are stored in private user space. More general global preferences are stored in a directory of the administrator's choice. Traditionally this has been the directory `/etc'.

The difficulties associated with the first of these approaches (having a fixed location for the configuration information which lies in the system files) are several. In any single user operating system, one user can overwrite another users' preferences simply by changing them since the system is not capable of telling the difference between users. This is a fundamental problem though which indicates that single user operating systems are basically unsuited to networking. More pertinent to a networked world are the following point:

- When the operating system is reinstalled, configuration information can easily be lost or overwritten if they are stored in an operating system directory.
- In a distributed environment, where users might not sit at the same physical workstation day after day, the user's personal configuration data will not follow him or her from machine to machine.

NT partly solves these problems by maintaining *user profiles* which are stored on the domain server in a `\profiles' subdirectory of the system-root. These data are copied into the local workstation when a user logs on to a domain server. On a Unix system, it is easy to specify the locations of configuration files and these can then be kept separate from operating system files, e.g. on a different disk partition so that they are immune to accidental deletion by system re-installation.

The primary difference between Unix and Windows/MacIntosh systems is that Unix is a *multiuser* system, i.e. any Unix machine can be used by an arbitrary number of users from an arbitrary location on the network. Microsoft and Apple systems allow only a single user to use a workstation interactively, from the console of the machine itself.

At the time of writing we are in the middle of a war of words between Unix and NT. In 1997, NT gained a lot of ground with newly started companies, eager to get going on the Internet as quickly as possible. Microsoft efficient marketing and existing dominance of the PC world made NT an interesting option. NT has suffered from setbacks as a result of bugs which affect security and stability, in the face of a vigorous marketing campaign. A major problem is the need for compatibility with DOS, through Windows 9x to NT. Since both DOS and Windows 9x are insecurable systems, this has led to conflicts of interest. Unix vendors have tried to keep step, in spite of poor public image of Unix (often the result of private dominance wars between different Unix vendors) but the specially designed hardware platforms built by Unix vendors have had a hard time competing with inferior but cheaper technology from the PC world.

In 1998 we see sales of both Unix and NT increasing but many users who have tried NT report that they are turning to cheap Unix solutions on Intel platforms (GNU/Linux, FreeBSD etc) instead because of its proven reliability. Unix tends to work better in a distributed environment and delivers especially good performance as a database server on 64-bit multiprocessor hardware. NT can run on 32-bit multiprocessor systems, but independent benchmarks show that it does not scale as well as many Unix variants do. Also the issue of remote login is significant for some sites. Using standard Windows user interfaces on NT, one cannot run graphical applications remotely as with X-windows on Unix. A basic X-windows release 6 is available for NT however, so one has the option of replacing the Windows interface and choosing a more distributed user interface. Today some systems which are based on the Mach micro-kernel can run native Unix and NT simultaneously. For example, Digital (now Compaq) Unix, GNU/Linux and NT can run on the Alpha processor.

User sociology

Most branches of computer science deal only with software systems and algorithms. System administration is made more difficult than this by the fact that it deals with computer communities and is therefore strongly affected by what human beings do. In short, a large part of system administration is *sociology* .

A newly installed machine does not usually require attention until it is first used, but as soon as a user starts running programs and storing data, the reliability and efficiency of the system are tested. This is where the challenge of system administration lies.

The load on computers and on networks is a social phenomenon: it peaks in response to patterns of human behaviour. For example, at universities and colleges network traffic usually peaks during lunch breaks, when students rush to the terminal rooms to surf on the web or to read E-mail. In industry the reverse can be true, as workers flee the slavery of their computers for a breath of fresh air (or carbonized air). In order to understand the behaviour of the network, the load placed on servers and the availability of resources, we have to take into account the users' patterns of behaviour.

Models of system administration

There are several approaches to the management of computers in a network environment. Here are three broad categories which are used today.

Reboot
: Because the number of local networks has outgrown the number of experienced technicians, there are many administrators who are not skilled in the systems they manage. A disturbing but common belief, which originated in the 1980's microcomputer era, is that problems with a computer can be fixed by simply rebooting the operating system. Since home computer systems tend to crash with alarming regularity, this is a habit which has been acquired from painful experience. The habit should be stifled however since more mature preemptive systems like Unix and NT should almost never need to be rebooted(1), indeed it can be damaging to do so. Rebooting a multi-user system is extremely dangerous since users might be logged in from remote locations and lose data. Moreover, the number of problems which can be fixed by rebooting a system is small compared to the number of problems which eventually arise. For this reason we classify this point as a regrettable but real development in network management.

Control
: Another approach to system administration is the use of control systems such as Tivoli, HP OpenView and Sun Solstice. In this approach the system administrator follows the state of the network on a graphical user interface. The administrator defines error conditions to look for and a process on each host reports errors as they occur to the administrator. In this way the administrator has an overview of every problem on the network from his/her single location and can either fix the problems by hand as they occur (if the system supports remote login), or distribute scripts and antidotes which provide a partial automation of the process. The disadvantage with this system is that a human administrator usually has to start the repair procedures by hand and thus this creates a bottleneck, since all the alarms go to one place to be dealt with serially. This means that the amount of work required to run this system increased roughly linearly with the number of hosts on the network.

Immunology
: A relatively new approach to system management which is gaining growing popularity is the idea of equipping networked operating systems with a simple immune system. By analogy with the human body, an immune system is an automatic system which every host possesses which attempts to deal with emergencies. An immune system is the Fire, Police and Paramedic services as well as the garbage collection agencies. In an immune system, every host is responsible for automatically repairing its own problems, without sending out warnings about what is going on. This avoids a serial bottleneck created by a human administrator. The time spent on implementing this model is independent of the number of hosts on the network. Unix administrators have run background scripts for many years. Such scripts (often called *sanity checking* scripts) run daily or hourly and make sure that each system is properly configured, perform garbage cleaning and report any serious problems to an administrator. In an immunological model, the aim is to minimize the involvement of a human being as far as possible. The tool Cfengine introduced technology which promotes this approach.

These different approaches are easily implemented on Unix systems because of the large number of scripting languages and tools available. On Microsoft systems the amount of freely available languages ported from Unix has now increased to the point where there are few problems there either. For MacIntosh systems there seems to have been little development and one is locked into the commercial products available there. The main problem in implementing monitoring software on insecure operating systems is that they are basically unstable. If a network administrator were informed every time a Windows 9x machine crashed or was rebooted on a network of hundreds, there would be a considerable amount of irrelevant traffic. Since there is nothing an administrator can do to change this fact, it has to be defined as a `feature' of these operating systems and tolerated.

Network details

This section offers a minimal introduction to some of the concepts required to discuss networking. You can refer to the bibliography if you would like to read more, See section Recommended reading.

Cables

A network is a line of communications between two or more hosts. It is usually some kind of shared cable which is attached to several hosts simultaneously by means of a *network interface* . Since it is impractical to have a private cable between every pair of hosts on a network (this would require N network interfaces and cables per host and would be unmanageable) there are two main ways of connecting hosts together.

Bus/Ethernet approach
: In the bus approach, every host is connected to a common cable or bus. 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 the line by waiting for a suitable moment to say something. You should not think data transmission over the network as being a little stream of bytes rollin' down the track one behind the other, from origin to destination, like Thomas the Tank-Engine. Every bit, every 1 or 0, is a signal (a voltage or light pulse) on a cable which fills the entire cable at a good fraction of the speed of light. It's like sending Morse code with a lighthouse. Everyone sees the signal, but only the recipient bothers to read it.

Token ring/FDDI approach
: In the token ring approach, each host has two network interfaces and the hosts are connected in a ring. If the signal arriving at one of the interfaces is for the host itself then it is read. If it is not intended for the host itself, the signal is forwarded to the next host where the same applies. The most common token ring in use today is the optical FDDI (Fiber distributed data interface).

Both of these approaches are susceptible to spying devices. If a host on the network wants to overhear a conversation between two others they have only to listen. In some cases extra hardware *switches* can be used to isolate private connections (see below). In what follows an Ethernet type of network will be assumed, since this is the commonest form in user-workstation environments.

Even with the bus approach, any host can be connected to several independent network segments. It must have a network interface for each network it is attached to. Each network interface has a separate network 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* .

Network signals are carried by a variety of means. These days copper cables are being replaced by fiber-optic glass transmission for long distance communication. In local area networks it is still copper cables which carry the signals. These cables are often called Ethernet cables, but in some cases the name is used mainly for historical reasons since the Ethernet has been superseded by the twisted pair technology. Thick yellow cables (about 1cm thick) carry thick Ethernet, thin black cables (0.5cm) with BNC connectors carry thin Ethernet. Fibre optic cables (FDDI) have varying appearances. Twisted pair lines are sometimes referred to at 10baseT, 100baseT etc, or for short T1, T10, T100. The numbers indicate the capacity of the line, `base' indicates that the cable is used in a *baseband* system and the `T' stands for twisted-pair. Twisted pair cables are very thin, so they are not tapped as are thin and thick Ethernet cables. Instead each host has a single cable connecting it to a *multi-way repeater* or *hub* .

Connectivity

The cables of the network are joined together in segments by hardware which makes sure that messages are transmitted in the right direction to reach their destinations. A host which is coupled to several network segments and which forwards data from one network to another is called a *router* . Routers not only forward data but they prevent the spread of network messages which other network segments do not need to know about. The limits the number of hosts which are sharing any given cable segment, and thus limits the traffic which any given host sees. A router knows which destination addresses lie on which of the networks it is connected to and it does not let message traffic spread onto irrelevant cables.

A *bridge* is a 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 or conversely to splice networks together.

A *repeater* is an amplifier which strengthens the network signal over long stretches of cable. A *multi-port repeater* also called a *hub* does the same thing and also splits one cable into N sub-cables for convenience. Hubs are common in twisted pair networks where it is necessary to fan a cable out into a star pattern from the hub to send one cable to each host. A *switch* is a hub which can direct a message from one host cable directly to the intended host by routing the signal directly. The advantage with this is that other machines do not have to see the traffic between two hosts. Each pair of hosts has an virtual private cable. Switched networks are immune to spies, net-sniffing or network listening devices.

When learning about a new network you should obtain a plan of the physical setup. If you have done your homework, then you will know where all of these boxes are on your network.

Protocols

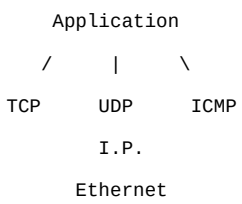
Information transactions take place by agree standards or *protocols* . Protocols exist to make sure that transmitted data are understood by the receiver in the way that the sender intended. On a network, protocols are required to make sure that data are understood, not only by the receiver, but by all the network hardware which carry them between source and destination. The data are wrapped up in envelope information which contains the address of the destination. Each transmission layer in the protocol stack (protocol hierarchy) is prefixed with a some header information which contains the destination address and other data which identify it. The Ethernet protocol also has a trailer.

```
     ------------------------------------------------
<- |ethernet |IP header| TCP    | data          |ethernet|
<- | header  |         | header| (ftp/telnet..)|trailer |
```

```
      ---------------------------------------------------

                              | Appl. layer   |
                      |   Transport   layer   |
              |   N e t w o r k         layer   |
      |    D  a  t  a  l  i  n  k         layer           |
```

Wrapping data inside envelope information is called *encapsulation* and it is important to understand the basics of this mechanisms. Ten years ago network administrators did not need to concern themselves with protocols and their like; today however network attacks make clever use of the features and flaws in these protocols and system administrators need to understand them in order to protect systems from the attacks.

The Internet Family of protocols has been the basis of Unix networking for many years, since they were implemented as part of the Berkeley Software Distribution (BSD) Unix. The hierarchy has the following form:



The transmission control protocol TCP is for reliable connection oriented transfer. The user datagram protocol UDP is a rather cheaper connectionless service and the internet control message protocol (ICMP) is used to transmit error messages and routing information for TCP/IP. These protocols have an address structure which is hierarchical and *routable*, that means that IP addresses can find their way from any host in the world to any other so long as they are connected. The ethernet protocol does not know much more about the world than the cable it is attached to.

NT supports three network protocols

NETBEUI
: NETBIOS Extended User Interface, Microsoft's own network protocol. This was designed for small networks and is not routable.

NWLink/IPX
: Novell/Xerox's IPX/SPX protocol suite. Routable.

TCP/IP
: Standard Internet protocols. The default for NT 4.

Data formats

There are many problems which arise in networking when hardware and software from different manufacturers have to exist and work together. Some of the largest computer companies have tried to use this to their advantage on many occasions in order to make customers buy only their products. An obvious example is the choice of network protocols used for communication. Both Apple and Microsoft have tried to introduce their own proprietary networking protocols. TCP/IP has won the contest because it was an *inter*-network protocol (i.e. capable of working on and joining together any hardware type) and also because it is a freely open standard. Neither Appletalk nor Microsoft's protocols has either of these features.

This illustrates how networking demands standards. Another example of this is the way in which different operating systems represent numerical data. Operating systems (actually the hardware they run on) fall into two categories known as *big endian* and *little endian*. The names refer to the *byte-order* of numerical representations. The names indicate how large integers (which require say 32 bits or more) are stored in memory. Little endian systems store the least significant byte first, while big endian systems store the most significant byte first. For example, the representation of the number 34,677,374 has either of these forms.

```
         -----------------------------------
Big    |   2    |   17   |  34    |   126  |
         -----------------------------------

         -----------------------------------
Little |   126  |   34   |  17    |   2    |
         -----------------------------------
```

Obviously if one is transferring data from one host to another, both hosts have to agree on the data representation otherwise there would be disastrous consequences. This means that there has to be a common standard of *network byte ordering*. For example, Solaris (SPARC hardware) uses network byte ordering (big endian), while GNU/Linux (Intel hardware) uses the opposite (little endian). This means that Intel systems have to convert the format every time something is transmitted over the network.

IP addresses

Every network interface on the Internet needs to have a unique number which is called its address. At present the internet protocol is at version 4 and this address 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 IPv4 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 (this would be physically impossible in practice), 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. This distinction has proven to be mistake. 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.

Some IP addresses are reserved for a special purpose:

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

The default route is a default destination for outgoing packets on a subnet and is usually made equal 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. 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, but in older networks it was xxx.yyy.zzz.0) and *multicast addresses* (usually 224.xxx.yyy.zzz).

Subnets

In practice, a given `network' might consist of very many separate cable systems, coupled together by routers and bridges. One problem with very large networks, like class B or class A networks is that *broadcast messages* (i.e. messages which are sent to every host) create traffic which can slow a busy network. In most cases broadcast messages only need to be sent to a subset of hosts which have some logical or administrative relationship, but unless something is done a broadcast message will by definition be transmitted to all hosts on the network. What is needed then is a method of assigning groups of IP addresses to specific cables and limiting broadcasts to hosts belonging to the group. The purpose of subnets is to divide up networks into regions which naturally belong together and to isolate regions which are independent. This reduces the propagation of useless traffic.

This logical partitioning can be achieved by dividing hosts up, through routers, into subnets. Each network can be divided into *subnets* by using a *netmask* . Each address consists of two parts: a *network address* and a *host address* . A system variable called the *netmask* decides how IP addresses are interpreted locally. The netmask decides the boundary between how many bits of the IP address will be kept for hosts and how many will be kept for the network location name. There is thus a trade off between the number of allowed domains and the number of hosts which can be coupled to each subnet. Subnets are usually separated by routers, so the question is how many machines do we want on one side of a router?

The netmask only has a meaning as a binary number. When you look at the netmask, you have to ask yourself -- which bits are ones and which are zeroes? The bits which are *ones* decide which bits can be used to specify the domain and the subnets within the domain. The bits which are zeroes decide which are hostnames on each subnet. The local network administrator decides how the netmask is to be used.

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.

```
          Class B address
 ----------------------------------
|   net  |   net  |  host  |  host  |
 ----------------------------------
    Subnet mask 255.255.254.0
 ----------------------------------
|11111111|11111111|11111110|00000000|
 ----------------------------------
          Interpretation
 ----------------------------------
| net id | net id | subnet|  host   |
 ----------------------------------
          Broadcast address
 ----------------------------------
|????????|????????|???????1|11111111|
 ----------------------------------
```

A subnet mask can be used to separate hosts which also lie on the same physical network, thereby forcing them to communicate through the router. This might be useful for security or administrative purposes.

The IP address of a host is set in the network interface. The UNIX command `ifconfig` (interface-configuration) or the NT command `ipconfig` 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
GNU/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.

The internet protocol (IPv4)

The internet protocol consists of two user protocols (TCP/UDP) and a private control protocol (ICMP). The simplest protocol is the unreliable, connectionless protocol UDP (User datagram protocol). UDP datagrams know their where they come from and where they are going, but have no idea about how they relate to other datagrams or how what they are carrying fits into the big picture. UDP has no mechanism for determining whether datagrams arrive at their destination or not.

TCP packets are reliable connection oriented data. They form *streams* or continuous data-flows with handshaking. This is accomplished by using a three-way handshake based om so-called SYN (synchronize) and ACK (acknowledge) bits in the TCP header. Suppose *host A* wishes to set up a connection with *host B* . *Host A* sends a TCP segment to *host B* with its SYN bit set and a sequence number X which will be used to keep track of the order of the segments. *Host B* replies to this with its SYN and ACK bits set, with Acknowledgement=X+1 and a new sequence number Y. *Host A* then replies to *host B* with the first data and the Acknowledge field=Y+1. The reason why each side acknowledges every segment with a sequence number which is one greater than the previous number sent is that the Acknowledgement field actually determines the next sequence number expected. This sequence is actually a weakness which network attackers have been able to exploit through different connections, in `sequence number guessing' attacks. Now many implementations of TCP allow random initial sequence numbers.

The purpose of this circuit connection is to ensure that both hosts know about every packet which is sent from source to destination. Because TCP guarantees delivery, it retransmits any segment for which is has not received an ACK after a certain period of time (the TCP timeout).

At the end of a transmission the sender sends a FIN (finished) bit, which is replied to with FIN/ACK. In fact closing connections is quite complex since both sides must close their end of the connection reliably. See the reference literature for further details of this.

The next generation internet protocol (IPv6)

The current implementation of the internet protocol has a number of problems. It is straightforward to calculate that, because of the structure of the IP addresses, divided into class A,B and C networks, something under two percent of the possible addresses can actually be used in practice. A recent survey from Unix Review, March 1998 shows

```
Year           Allocated            Already allocated
-------------------------------------------------------
          A       B       C          A       B      C

'82       0       0       6          0       0      0
 83       1      15      30          0       0    930
 84       2       7      43          0       0    975
 85       1      22      93          0       0    539
 86       1      64     237          0       0   1464
 87       0     145     482          0       0    812
 88       4     421    1240          0      28   1753
 89       5     765    1176          0     360   3344
 90       4    1375    3009          0     187   1344
 91       8    1436    2555          0     161   4006
 92       4    2036   11626          0      82  74524
 93       1    1107   20289          0     334 144201
 94       1     484    9314          0      75 182494
 95       9     113   14546          0     136 124154
 96       1      85    9824          0       2  31690

(Unknown Date)
```

```
         22     500    2451         63     210 115051
==================================================
         64    8575   76921         63    1575 687281
```

This means that, of the total numbers of addresses, these are already allocated:

```
           Max possible   Percent allocated
Class A         127        100%
Class B       16382         62%
Class C     2097150         36%
```

Of course, this does not mean that all of the allocated addresses are in active use. After all, what organization has 65,535 hosts? In fact the survey showed that under two percent of these addresses were actually in use. This is an enormous wastage of IP addresses. Amongst the class C networks, where smaller companies would like address space, the available addresses are being used up quickly, but amongst the class A networks, the addresses will probably never be used. A new addressing structure is therefore required to solve this problem. Other problems with IPv4 are that it is too easy to take control of a connection by guessing sequence numbers. Moreover there is no native support for encryption or mobile computing.

In order to address these issues the IETF (Internet Engineering Task Force) has put together a workgroup to design a new 128-bit protocol which will be called IPv6. Not only will the addressing structure be different but there will be a considerable number of extra addresses. Even with a certain inefficiency of allocation, it is estimated that there will be enough IPv6 addresses to support a density of 10,000 IP addresses per square meter which ought to be enough for every toaster and wristwatch on the planet and beyond.

Routing

Unless a host operates as a router in some capacity, it only requires a minimal routing configuration. Each host must define a *default route* which is a destination to which outgoing packets will be sent for processing when they do not belong to the subnet. This is the address of the router or gateway on the same network segment. It is set by a command like this:

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

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

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

The default route can be checked 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, a host 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 leads to great inefficiency. In the worst case the host might not have contact with anywhere outside its subnet at all.

ARP/RARP

ARP is the (IP) address resolution protocol. ARP takes an IP address and turns it into an ethernet (hardware) address. The ARP service is mirrored by a reverse ARP service (RARP). RARP takes a hardware address and turns it into an IP address.

Ethernet (or generally hardware) addresses are required when routing traffic from one device to another. While it is the IP addresses which contain the structure of the internet and permit routing, it is the hardware address to which one must deliver packets in the final instance; this is the address which is burned into the network interface.

The hardware addresses are cached by each host on the network so that repeated calls to the service ARP translation service are not required. Addresses are checked later however, so that if an address from a host claiming to have a certain IP address originates from an incorrect hardware address (i.e. the packet does not agree with the information in the cache) then this is detected and a warning can be issued to the effect that two devices are trying to use the same IP address. ARP sends out packets on a local network asking the question `Who has IP address xxx.yyy.zzz.mmm?' The host concerned replies with its hardware address.

For hosts which know their own IP address at boot-time these services only serve as confirmations of identity. Diskless clients (which have no place to store their IP address) do not have this information when they are first switched on and need to ask for it. All they know originally is the unique hardware (ethernet) address which is burned into their network interface. In order to bring up and configure an internet logical-interface they must first use RARP to find out their IP addresses from a RARP server. Services like BOOTP or DHCP are used for this. Also the Unix file `` `/etc/ethers' `` and `rarpd` can be used.

SNMP

The ability to read information about the performance of network hardware via the network itself is an attractive idea. Suppose one could look at a router on the second floor of a building half a mile away and immediately see the load statistics, or number of rejected packets it has seen. That would be useful diagnostic information. Similar information could be obtained about software systems on any host.

SNMP (Simple Network Management Protocol) is a protocol designed to do just this. SNMP supports two operations *get* and *put*, thus it can read and perhaps modify the data stored on a host. SNMP access is mediated by a server process on each hardware node. Modern operating systems usually run SNMP daemons or services. The services are protected by a rather weak password which is called the *community string*.

SNMP information is stored in data structures called MIBs (Management Information Bases). The MIBs are catalogued in RFC 1213 and give hardware and software profiles. They are dynamically updated. An SNMP request specifies the information it wants to read/write by giving the name of a MIB. In SNMP v2 there are standard MIBs for address translation tables, TCP/IP statistics and so on. There are 27 default parameters that can be altered by SNMP: system name, location and human contact; interface state (up/down), hardware and IP address, IP state (forwarding gateway/not) IP TTL, IP next HOP address, IP route age and mask, TCP state, neighbour state, SNMP trap enabling.

Operating systems define their own MIBs for system performance data. Some commercial network management systems like Tivoli and Hewlett Packard's OpenView work by reading and writing MIBs using SNMP client-server technology. Most Unix variants now support SNMP. NT supports SNMP version 1. Its MIBs can be used to collect information from NT systems, such as the names of users who are logged on.

SNMP seems to be increasing in popularity, but like any public information database, it can be abused by network attackers. Some sites choose to disable SNMP services altogether. All sites should filter SNMP packets to and from external networks to avoid illegal access of these services from intruders.

@node High level protocols

High level protocols

Internet networks use many high level protocols to provide the distributed services which most users take for granted.

HTTP The world wide web protocol for exchanging hypertext and multimedia data. All data are sent in clear text.
SHTTP
The secure world wide web protocol for exchanging hypertext and multimedia data. All data are encrypting using Netscape's secure socket layer (SSL).
FTP The File transfer protocol. Passwords are sent in clear text.
SSH The secure shell. A replacement for the remote shell (rsh) Unix protocol. The secure shell provides full encryption and forwarding of X11 display data through a secure pipe.
LDAP The *Lightweight Directory Access Protocol* is a generalized protocol for looking up data in simple databases. It is a lightweight version of the service written for X.500 and is currently at Version 3. It can be used to register user information, passwords, telephone numbers etc and interfaces through gateways to the NDS (Novell Directory Service), Microsoft's Exchange server and NIS (Sun's Network Information Service). The advantage of LDAP will be a uniform protocol for accessing table lookups. Currently the spread of LDAP is hindered by few up-to-date implementations of the protocol. Only Netscape's directory server supports the version 3 standard.

Security and reliability

@hrule @vskip 0.3cm

Security is about keeping the network of machines and their data safe. This includes protecting 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(2) have found their way inside the computers of the pentagon, the world's security services, warships, fighter 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

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.

Consider 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.

Passwords are not visible to ordinary users, but their encrypted form is often 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 GNU/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 from a little-used credit card as a part of your password. This means that you don't have to remember too much--and it means that you do have secret numbers in your password. 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 often possible to download the screen image of another user who is using the X-windows system. It is equally possible to `bug' the keyboard and listen to all the key-presses. The problem is an out-dated security mechanism which has long since been replaced, but which is still used by very many 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 -`. Users of the secure shell `ssh`, See section Secure shell, can now have automatic X11 forwarding with authentication cookies. Everyone should therefore execute `xhost -` once and never use the xhost mechnism again.

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 name-service; the interception of passwords in network packets and several other kinds of attack. Note that the secure shell is NOT free software. It is only freely usable in academic contexts for Unix-like systems. Anything else you have to pay for.

To install this, collect it from an ftp site and perform the usual steps:

```
(get ftp file ssh-2.0.9.tar.gz)

host% tar zxf ssh-2.0.9.tar.gz
host% cd ssh-2.0.9
host% ./configure
host% make
host% su
Passwd: ********
host# make install
```

You then need to start the daemon by adding a command of the form

```
/usr/local/sbin/sshd
```

to the startup scripts on your system.

Accidental deletion of files

Once a file is deleted in UNIX, it is not possible to 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

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 is to identify 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.

Who are you trying to protect them from?

- Competitors, who might gain an advantage by learning your secrets.
- Malicious intruders. Note that people with malicious intent might come from inside or outside your organization. Do not think that the enemy is simply everyone outside of your domain. Too many organizations think `inside/outside' instead of having a proper form of access control. If you *always ensure that systems and data are protected* on a need-to-know basis, then there is no reason to discriminate between inside or outside of an organization.
- Old employees with a grudge. against your organization

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 does one draw the line? How much is security worth?

The social term in the security' equation should never be forgotten. One can spend a hundred thousand dollars on the top of the range firewall to protect data, but someone could walk into the building and look over an unsuspecting shoulder to obtain it instead, or use a receiver to collect the stray radiation from your monitors. Are you leaving sensitive 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.

Some examples of security policies are here:

```
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 through obscurity

There is a commonly held belief that, if one makes it difficult for intruders to find out information, they will not bother to try. This is completely wrong. Often the reverse is true. If attackers can see that there is nothing worth finding they will leave systems alone. If everything is concealed they will assume that there must be something interesting worth breaking in for.

Security through obscurity is a naive form of security which at best delays break-ins. Shadow passwords are an example of this. By making the encrypted password list inaccessible to normal users one makes it harder for them to automate the search for poor passwords, but one does not prevent it! It is still possible to guess passwords in exactly the same way as before, but it takes much longer. The NT password database is not in a readable format. Some people have claimed that this makes it more secure than the Unix password file. Since then tools have been written which rewrite the NT password file in Unix format with visible encrypted passwords. In other words, making it difficult for people to break in does not make it impossible.

Clearly there is no need to give away information to potential intruders. Information should be available to *everyone* on a need-to-know basis, whether they be local users or people from outside the organization. But at the same time, obscurity is no real protection. Even the invisible man could get shot.

Time spent securing systems is better than time spent obscuring them. Obscurity might attract more attention than you want.

Security holes

One way that outside users can attack a 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* . Several other organizations, often run by staff who work as security consultants, are now involved in computer security monitoring:

www.sans.org
: System administration and network security organization.

www.cert.org
: The computer emergency response team.

ciac.llnl.gov
: The computer incident center.

www.rootshell.com
: A source of all kinds of the latest security information.

Bugtraq
: http://www.netspace.org/cgi-bin/wa?S1=bugtraq

PHRACK
: http://www.phrack.com

System homogeneity

In a site with a lot of different kinds of platforms, perhaps several Unix variants, NT and Windows 9x, the job of closing security holes is much harder. Inhomogeneity often provides the determined intruder with more possibilities for bug-finding. You might ask yourself whether you need so many different kinds of platform. If you do, then perhaps a firewall solution would provide an extra level of protection, giving you a better chance of being able to upgrade your systems before something serious happens.

Modem pools

Some companies expend considerable effort to secure their network connections, but forget that they have dial-in modems. Modem pools are a prime target for attackers because they are often easy targets. There are many problems associated with modem pools. Sometimes they are quite unexpected. For example, if one has network access to Windows systems using the same modem then those systems are automatically on a shared segment and can use one another's resources, regardless of whether they have any logical relationship. Modems can also succumb to denial of service attacks by repetitive dialling.

Modems should never allow users to gain access to a part of the network which needs to be secure. Modems should never be back-doors into firewalled networks.

Laptops

Laptop computers are increasingly popular and they are popular targets for thieves. There have been cases of laptop computers being stolen containing sensitive information, often enough to give crackers access to further systems, or simply to give competitors the information they wanted! One anecdote tells of a high ranking officer in the US military whose laptop was stolen with sensitive national security information. It was later returned with a note from the thief who did not want to be labelled a traitor.

Backups

If you make backups of important data (private data) then you must take steps to secure the backups also. If an intruder can steal your backups, then he/she doesn't need to steal the originals.

Firewalls

A firewall is a network configuration which isolates some machines from the rest of the network. A firewall is a gate-keeper which limits access to and from a network. You might think that this sounds like a strange idea: after all, what is the point of being connected to the internet if you just want to isolate yourself from it! The point, of course, is not to cut oneself off from the network but to provide controls on what is allowed in and what is allowed out.

A firewall is a concept. It is not one specific thing; 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 some kind of 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 intermediary 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).

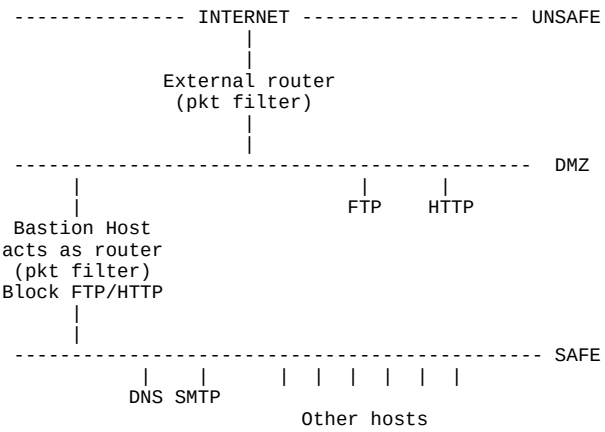
Screened subnet/DMZ
: An isolated subnet, between the internet and the private network. Also called a DMZ (de-militarized zone). A DMZ is the bit between a screening router and the firewall, bastion host. This is a good place for external WWW services.

The firewall philosophy builds on the idea that it is easier to secure one host (the bastion host) than it is to secure hundreds or thousands of hosts on a local network. One focuses on a single machine and ensures that it is the only one effectively coupled directly to the network. One forces 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 of course a simplification. It is important to realize that installing a firewall does not give absolute protection and it does not remove the importance of configuring and securing the hosts on the inside of the firewall.

Of course we do not want all traffic to stop, some things like E-mail and maybe HTTP should be able 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 commercial in nature because it needs to live in the kernel and make use of code which is proprietary.

Firewall management is a complex issue. One cannot 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. 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 (RPC), IRC, ping, finger etc.

Access control with tcpd

Even without a firewall one can go half way by verifying the authenticity of packets coming into your network. This is a front line against `spoofing' -- i.e. internet impersonation. 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
```

`pidentd` authentication server

One of the problems with socket based communication is authentication. How do we determine the identity of the user making the connection? Normally the only certain way is to require a password to be given, i.e. some kind of shared secret. The RFC documents RFC981 and RFC1413 specify a standard protocol for identifying the user*name* of a connecting user. In order for this to work, the a server which is being connected to has to contact the host which is attempting to connect and check who owns the transmitting socket. This is done with the help of the `identd` daemon, or its implementation `pidentd` (the portable identity daemon).

The identity or authentication service (port 113) is alas not installed as standard by most vendors. You need to get this daemon and install it yourself. This is quite straightforward.

```
ftp ftp.lysator.liu.se/pub/ident/servers/

tar zxf pidentd-3.0.4.tar.gz
cd pidentd-3.0.4
configure --with-des=no
make
make install
```

The latest versions of the server are multithreaded and are most easily run outside of the inetd service. The daemon is started by

```
identd
```

Although it can be started from inetd, you shuld never invoke this daemon with TCP wrappers, since TCP wrappers attempts to contact with daemon. This will result in infinite loop. You might have to register the service in `` `/etc/services' ``

```
auth   113/tcp   authentication tap ident
```

The identification service is a public service which you provide for your own and for others' benefit. It allows you and others to authenticate connections from your hosts.

Honey-pots and sacrificial lambs

A Honey Pot is a host which is made to look attractive to attackers. It is usually placed on a network with the intention of catching an intruder or distracting them from more important systems. A Sacrificial Lamb host is a host which is not considered to be particularly important to the domain. If it is compromised by an attacker then that is an acceptable loss and no real harm is done.

Some network administrators believe that the use of such machines contributes to security. For example, WWW servers are often placed on Sacrificial Lamb machines which are placed outside firewalls. If the machine is compromised then it can simply be reinstalled and the data reloaded from a secure backup.

This practice can seem rather dubious. There is certainly no evidence to support the idea that either Honey Pot machines or Sacrificial Lamb systems actually improve security.

Trust relationships

A trust relationship lies at the basis of many software systems which grant access to services or resources. Trust relationships are important to grasp because they can lead to security holes. Whenever one installs a new service which is available to more than one user it is appropriate to ask the questions:

- Do I need this service?
- Whom or what information do I have to trust in order to use this?
- What will happen if someone abuses that trust?

For example, the `rlogin` feature of Unix has a file called `` `.rhosts' `` in which a user can add a list of trusted hosts. That user can log in to the host with the `` `.rhosts' `` file from any one of those trusted hosts without giving a password. The user is clearly willing to trust this list of hosts. But that is not the only trust relationship here. Unix uses DNS (the Domain Name Service) in order to verify the identity of connecting machines, so the rlogin service *implicitly* trusts the DNS service. If someone could corrupt that service, there would be a potential security problem, See section DNS cache poisoning.

Another example is in software distribution, both for Unix and NT. In order to distribute software from a central server to many clients, the clients have to trust the information being sent to them from the server. They have to give the server permission to install unknown files which might be security hazards.

SNMP control systems accept information from a controller, based only on a fairly weak password (community string). The password has a default value of `public' which many sites forget to change (a potentially huge security risk). This information can be used to change control functions of key network components and is even used for performing remote system administration in certain products.

Cfengine places all of its trust in the correctness of its input file, it does not accept input from the network at all. In software distribution it will trust files from a software server of its own choosing, but arbitrary servers cannot send data to it uninvited.

Attacks

There are many ways to attack a networked computer in order to gain access to it, or simply disable it. Some well-known examples are listed below. The actual attack mechanisms used by attackers are often intricate and ingenious, but the common theme in all of them is to exploit naive limitations in the way network services are implemented. Time and again one sees crackers make use of software systems which were written in good faith, by forcing them into unnatural situations where the software fails through inadequate checking.

Ping attacks

The RFC 791 specifies that internet datagrams shall not exceed 64kB. Some implementations of the protocol can send packets which are larger than this, but not all implementations can receive them.

```
ping -s 65510 targethost
```

Some older network interfaces can be made to crash certain operating systems by sending them a `ping' request like this with a very large packet size. Most modern operating systems are now immune to this problem (e.g. NT 3.51 is vulnerable, but NT 4 is not). If not, it can be combatted with a packet filtering router. See http://www.sophist.demon.co.uk/ping/ .

Denial of service (DoS) 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(3), 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 responsibility of local administrators to prevent their users from initiating such attacks wherever possible.

Denial of service attacks on NT can be embarrassingly simple

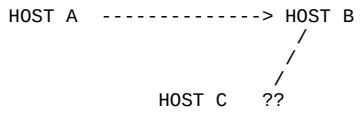
```
%myhost telnet nt-host 1028
Trying ???.???.???.???...
Connected to nt-host
Escape character is '^]'.
Hello there.

^]quit

myhost%
```

This sends NT4 inetinfo server into a loop which can stop all services. Service Pack 2 fixes this. There are many other examples. Starting full system auditing can also bring a respectable Pentium system to a virtual standstill.

TCP/IP spoofing

Most network resources are protected on the basis of the host IP addresses of those resources. Access is granted to by a server to a client if the IP address is contained in an access control list (ACL). Since the operating system kernel itself declares its own identity when packets are sent, it has not been common to verify whether packets actually do arrive from the hosts which they claim to arrive from. Ordinary users have not traditionally had access to privileges which allow them to alter network protocols. Today everyone can run a PC with privileged access to the networking hardware.

Normally an IP datagram passing from *host A* to host B has a destination address `host B' and source address `host A'. IP spoofing is the act of forging IP datagrams in such a way that they appear to come from a third party host. i.e. an attacker at *host A* creates a packet with destination address `host B' and source address `host C'. The reasons for this a varied. Sometimes an attacker wants to appear to be *host C* in order to gain access to a special resource which *host C* has privileged access to. Another reason might be to attack *host C* , as part of a more elaborate attack. Usually it is not quite this simple however, since the forgery is quickly detected.



The TCP handshake is such that *host A* sends a packet to *host B* and then replies to the source address with a sequence number which has to match the next number of an agreed sequence. If another packet is not received with an agreed sequence number the connection will be reset and abandoned. Indeed, if *host C* received the confirmation reply for a message which it never sent, it would send a reset signal back immediately, saying effectively `I know nothing about this'. To prevent this from happening it is common to take out *host C* first by attacking it with some kind of Denial of Service method, or simply choosing an address which is not used by any host. This prevents it from sending a reset message. The advantage of choosing a real *host C* is that the blame for the attack is placed on *host C* .

SYN flooding

IP spoofing can also be use>

Transfer interrupted!

hoosing an address for *host C* which is not in use so that it cannot reply with a reset, *host A* can send SYN packets (new connections) on the same and other ports repeatedly. The *RECV* queue quickly fills up and cannot be emptied since the connections cannot be completed. Because the queues are filled the services are effectively cut off.

These attacks could be prevented if routers were configured so as to disallow packets with forged source addresses.

TCP sequence guessing

Random sequences

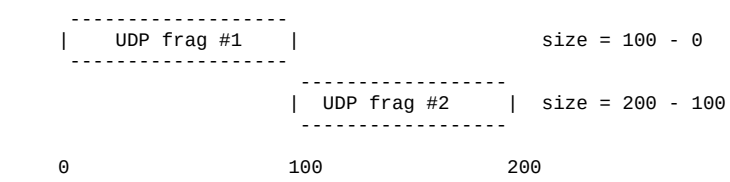
This attack allows an attacker to make a TCP connection to a host by guessing the initial TCP sequence number used by the other end of the connection. This is a form of IP spoofing. The attack was first described in the references below. It was made famous by the break in into Tsutomo Shinomrua's computers which led to the arrest of Kevin Mitnick. This attack is used to impersonate other hosts for trusted access.

1. R.T. Morris, "A weakness in the 4.2 BSD Unix TCP/IP software", Computer Science Technical Report #117, ftp://ftp.research.att.com/dist/internet_security/117.ps.Z
2. S.M. Bellovin, "Security problems in the TCP/IP protocol suite", Computer Communications Review 19:2, April 1989, pp. 32-48. http://www.research.att.com/~smb/papers/ipext.pdf
3. http://www.kohala.com/~rstevens/shimomura.95jan25.txt
4. Jonathan Littman, *The Fugitive Game* , http://www.well.com/user/jlittman/game

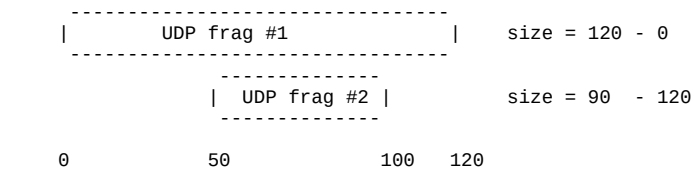
UDP fragmentation (Teardrop)

The Teardrop attack was responsible for the now famous twelve hour attack which `blue-screened' thousands of NT machines all over the world. This attack uses the idea of datagram fragmentation. Fragmentation is something which happens as a datagram passes through a router from one network to another network where the transmission rate is lower. Large packets can be split up into smaller packets for more efficient network performance. In the Teardrop attack, the attacker forges two UDP datagrams which appear to be fragments of a larger packet, but with data offsets which overlap.

When fragmentation occurs it is always the end host which reassembles the packets. In order to allocate memory for the data, the kernel calculates the difference between the end of the datagram and the offset at which the datagram fragment started. In a normal situation that would look like this:



In a Teardrop attack the packets are forged so that they overlap like this:



The assumption that the next fragment would follow on from the previous one leads to a negative number for the size of the fragment. As the kernel tries to allocate memory for this it calls `malloc(`*size*`)` where the size is now a negative number. The kernel panics and the system crashes on implementations which did not properly check the bounds.

ICMP flooding (Smurf)

ICMP flooding is another denial of service attack. The ICMP protocol is the part of TCP/IP which is used to transmit error messages and control information between hosts. Well known services like `ping` and `echo` use ICMP. Normally all hosts respond to ping and echo requests without question, since they are useful for debugging. In an ICMP flooding attack, the attacker sends a spoofed ICMP packet to the broadcast address of a large network. The source address of the packet is forged so that it appears to come from the host which the attacker wishes to attack. Every host on the large network receives the ping/echo request and replies to the same host simultaneously. The host is then flooded with requests. The requests consume all the system resources.

DNS cache poisoning

This attack is an example of the exploitation of a trusted service in order to gain access to a foreign host. Again it uses a common theme, that of forging a network service request. This time however the idea is to ask a server to cache some information which is incorrect so that future look-ups will result in incorrect information being given instead of the correct information.

DNS is a hierarchical service which attempts to answer queries about IP names and addresses locally. If a local server does not have the information requested it asks an authoritative server for that information. Having received the information from the authoritative server it caches it locally to avoid having to contact the other server again, after all, since the information was required once, it is likely that the same information will be required again soon. The information is thus placed in the cache for a period of time called the TTL (*Time To Live*). After that time has expired it has to be obtained again from the authoritative server.

In a cache poisoning attack, the aim is to insert incorrect information into the cache of a server. Once it is there it will be there for the TTL period. In order to arrange this an attacker does the following.

1. The attacker launches his/her attack from the authoritative nameserver for his/her network. This gives him/her the chance to send information to another nameserver which will be trusted.
2. The attacker sends a query for the IP address of the victim host to the victim's default DNS server in order to obtain a DNS query ID. This provides a point of reference for guessing i.e. forging the next few query IDs from that server.
3. The attacker then sends a query asking for the address of a host which the victim machine trusts, i.e. the host which the attacker would like to impersonate.
4. The attacker hopes that the victim host will soon need to look up the IP address of the host it trusts; he/she sends a fake `reply' to such a DNS lookup request, forged with the query ID to look as though it comes from a lookup of the trusted host's address. The answer for the IP address of the trusted host is altered so that it is the IP address of the attacker's host.
5. Later when the victim host actually sends such a DNS request it finds that it has already received a UDP reply to that request (this is the nature of UDP) and it ignores the real reply because it arrives later. Now the victim's DNS cache has been poisoned.
6. The attacker now attempts to connect directly to the victim host, posing as the trusted host. The victim host tries to verify the IP address of the host by looking up the address in its DNS server. This now responds from its cache with the forged address.
7. The attacker's system is accepted.

This kind of attack requires the notion of external login based on trust, e.g. with Unix `.rhosts' files. This doesn't help with NT because NT doesn't have trusted hosts in the same way. On the other hand, NT is much easier to gain access to through NULL sessions.

Protecting against attacks

- Look out for users with weak passwords. This is the easiest way for attacker to enter the system.
- Do not give trusted access to other hosts unless absolutely necessary. Make sure there are no NIS wildcards `+' in `/etc/hosts.equiv'. Avoid using `.rhosts' files altogether.
- Disable unused services in `/etc/inetd.conf' which might contain security leaks, like UUCP, TFTP.
- Make sure the router filters all unnecessary traffic. Usually there is no reason to permit RPC traffic outside of the local domain for instance.
- Make sure that the latest security patches are installed on all systems.
- Monitor connections using `netstat -a` to show all listening connections. Use tcpd logging.
- 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 processes 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.
- Make sure that `.' is not in root's path. It is possible to inadvertently execute a Trojan horse program.
- Make sure that files like `/var/adm/utmp' are not world writable allowing crackers to cover their tracks.

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. If a cracker has privileged access to at least one machine with a network interface on the same subnet he/she use `tcpdump` to capture all network traffic. Normal users do not have this privilege for precisely this reason. These days however, anyone with a laptop, an ethernet card and a GNU/Linux installation could do this. Switched networks are immune to this problem since traffic is routed directly from host to host.

Programs which dump all network traffic include `tcpdump`, `etherfind`, `snoop`. Here is a sample of the output from Solaris' `snoop` program showing the ethernet traffic on segment of cable. Snoop recognizes common high level protocols (SMTP/FTP/ARP etc) and lists them explicitly. Unknown protocol types (in this case IPX) are simply listed as ETHER. In the right hand column is the information which an intruder would try to use to sniff passwords.

```
Using device /dev/hme (promiscuous mode)
 post.eet.no -> nexus        SMTP C port=4552 oJyhnJycoZyhnKCcnGCc
torget.drammensnett.no -> nexus        SMTP C port=54621 AGoHRPVU9VT3
       nexus -> torget.drammensnett.no SMTP R port=54621
pc111-75.iu.hioslo.no -> nexus        FTP C port=1093
       nexus -> pc111-75 FTP R port=1093 226 Transfer complet
       nexus -> post.eet.no  SMTP R port=4552
 post.eet.no -> nexus        SMTP C port=4546 UHAQcBB/UB9QcBBwH1AQ
       nexus -> post.eet.no  SMTP R port=4546
 post.eet.no -> nexus        SMTP C port=4546 H2AQcBBwH1AfYBAQH1Af
   fw.nki.no -> nexus        SMTP C port=11424 O3Jw+XF7cMFCCwe\r\nEQ/
       nexus -> fw.nki.no    SMTP R port=11424
 post.eet.no -> nexus        SMTP C port=4552 \niYmJgomChomChoaChoK

       nexus -> post.eet.no  SMTP R port=4546
       nexus -> (broadcast)  ARP C Who is 128.39.89.230, takpeh ?
       nexus -> post.eet.no  SMTP R port=4552

         ? -> *            ETHER Type=0000 (LLC/802.3), size = 86 bytes
         ? -> *            ETHER Type=0000 (LLC/802.3), size = 128 bytes
         ? -> *            ETHER Type=0000 (LLC/802.3), size = 80 bytes
```

One way to avoid the problem of password sniffing is to use fullt encrypted links such as `ssh` and SSL (Secure Socket Layer) enabled services which replace the standard services. Another is to use a system of one-time passwords. One time passwords are designed to eliminate the need for users to send their passwords over the network at all. Instead of typing an actual password, one types the remote password for a host into a program on a local machine, in order to generate a sequence of throw-away passwords which can be used in place of the actual remote password. The passwords are used only once so, even if someone gets to overhear them, it will already be too late: the password will have expired. Also the system is ingeniously designed so that the actual remote password (which is used to generate the one-time passwords) never gets sent over the network at all. 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"
6. You type "EASE FREY WRY NUN ANTE POT" on host B (sent over the network) and the password is accepted.
7. Next time you follow the same procedure and get a different password.

Port scanning

In order to find back-doors into vulnerable systems many network attackers scan ports on network hosts in order to find out which services are running on them. Programs for performing such scans can be obtained freely from the network, as can many other intrusion tools, so crackers require little or no intelligence in order to carry out these simple attacks these days.

In a poorly configured system a cracker might find active services which even the system owner did not realize were running. UUCP and TFTP are typical examples. These services can often be exploited to install files in illegal places. Known faults in services can be exploited if one knows about the services which are running.

Primitive port scanning is detectable if one follows network activity closely. Strings of attempted `connect' requests to one port after the other are easily spotted. Recently however the trend has expanded to include `stealth scanning' in which scans are performed at random over long periods of time to avoid attracting attention. Port scanning is only dangerous if there are poorly configured hosts on the network.

Social engineering

Network attackers i.e. system crackers are people. We are often 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* . If one can avoid complex technical expertise in order to break in to a system, then why not do it? There is more than one way to crack a system.

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 which motivates break-ins is bragging. Crackers 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 many cases it is nonsense, empty words.

There is a few things which can be done to counteract social threats.

- Never disclose information over the telephone, especially through a voice-mail system. Phone calls can be spoofed.
- Examine system logs, check the system regularly, run cfengine to ensure consistency.
- Make hard-copies 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 judgment 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.

Current attack profiles

The popular operating systems to attack are

- Windows 9x
- Windows NT
- GNU/Linux
- Irix (Silicon graphics)

In the Wheelgroup and NetSolve Security Survey, which includes an analysis of a half-million security alarms, it was found that serious attacks to companies take place at a rate of between 0.5-5.0 per month. Electronic commerce sites (sites which deal with money) are those most at risk. The attack methods follow standard, well-known bugs

IMAP
BIND
qpopper
DoS/SMURF
FTP (Bounce)

Intrusion detection

In the last year or two the reality of network intrusion has led to several attempts to build systems which can detect break-ins, either while they are in progress or afterwards. In order to detect an intrusion in progress, programs like *Network Flight Recorder* (NFR) and *Big Brother* (Bro) attempt to examine every packet on the network in order to look for tell-tale signatures of network break-in activity. This is an extremely resource consuming task and it is best with a number of problems. One problem is that of *fragmentation* . Fragmentation is something which occurs to IP datagrams which pass between networks with different transmission rates. Larger packets can be broken up into smaller packets in order to optimize transmission. These fragments are reassembled at the final destination. This presents a problem for intrusion detection systems because the fragmented packets might not contain enough data to identify them as hostile. This would allow them to get past the detection system. An intruder might be able to generate packets which were fragmented in such as way as to confound the attempts at detection. In spite of the difficulties network intrusion detection is a hot research topic.

Network forensics is what one does after an intrusion. The idea is to examine logs and system audits in order to name the intruder and determine the damage. Network forensics is perhaps most important for the purpose of possible legal action against intruders. The cost of keeping the necessary logs and audits is very great and the work required after a break-in is far from trivial.

User administration

@hrule @vskip 0.3cm

User registration

One of the most important issues for a system administrator is to issue new accounts for users. Surprisingly this is an area where vendors provide virtually no help. The tools provided by operating systems for this task are at best primitive and almost never suitable for use without some modification.

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 one needs to

- Find a unique username, user-id (uid number) and password for the new user.
- Update the system database of user accounts, e.g. add a line to the file `` `/etc/passwd' `` in Unix (or on the centralized password server of a network) for the new user.
- Create a login directory (home directory) for the user.
- Choose a shell for the user (if appropriate).
- Copy some configuration files like `` `.cshrc' `` or `` `.profile' `` into the new user's directory, or update the system registry.

Because every site is different, user registration requires different tools and techniques in almost every case. For example: where should users' home directories be located? GNU/Linux has an `` `adduser' `` script which assumes that the user will be installed on the local machine under `` `/home/user' ``, but many users belong to a network and their disk space lies physically on a different host which is mounted by NFS. In NT all users begin in the root directory by default. It is customary to create a `` `\users' `` directory for home directories. If one creates an account on one host, what about the other hundred hosts at the site? Should users be allowed to log on to all of them? Should they have the same files on every host, or different ones? All of these questions need to be answered.

Perl scripts are excellent ways of making user installation scripts. Interactive programs are almost useless since users are seldom installed one by one. At universities hundreds of students are registered at the same time. No system administrator would type in all the names by hand. More likely they would be input from some administrative list generated by the admissions department. The format of that list is not a universal standard, so no off-the-shelf software package is going to help here.

Local and network accounts

Both Unix and NT support the creation of accounts both locally on a single host, or `globally' within a network domain. With a local account, a user has permission to use only the local host. With a network account, the user can use any host which belongs to the network `domain'.

Local accounts are configured on the local host itself. Unix reisters local users by added them to the files `` `/etc/passwd' `` and `` `/etc/shadow' ``. In NT the Security Accounts Manager (SAM) is used to add local accounts to a given workstation.

For network accounts, Unix uses NIS (Network Information Service, formerly called Yellow Pages or simply YP). NT uses its model of domain servers, which is like a NIS server only much more. A user in the SAM of a primary domain controller is registered within that domain and has an account on any host which subscribes to that domain.

An NT domain server involves not only shared databases but also shared administrative policies and shared security models. A host can subscribe to one or more domains and one domain can be associated with one another by a trust relationship. When one NT domain `trusts' another then accounts and groups defined in the *trusted* domain can be used in the *trusting* domain.

The DCE (Distributed Computing Environment) can also be used to provide a seamless world-wide distributed network domain. The DCE has been ported to both Unix and NT and requires a special login after normal login to Unix/NT.

Login environment

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 GNU/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 GNU/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.

Controlling user resources

Diskspace

Disks fill up at an alarming rate. Users almost never throw away files unless they have to. If one is lucky enough to have only very experienced and extremely friendly users on the system, then one can try asking them nicely to tidy up their files. Most administrators do not have this luxury however. Most users never think about the trouble they might cause others by keeping lots of junk around. After all, multiuser systems and network servers are designed to give every user the impression that they have their own private machine.

To keep hosts working it is necessary to remove files, not just add them. Quotas limit the amount of disk-space users can have access to, but this does not solve the real problem. The real problem is that in the course of using a computer many files are created as temporary data but are never deleted afterwards. The solution is to delete them.

Some files are temporary by definition.
: For example the byproducts of compilation, `*.o' files, files which can easily be regenerated from source like TeX `*.dvi' files, cache files in `.netscape/' loaded in by Netscape's browser program, etc.

Some files can be defined as temporary as a matter of policy.
: Files which users collect for personal pleasure like `*.mp3', video formats and pornography.

When a Unix program crashes, the kernel dumps its image to disk in a file called `core'. These files crop up all over the place and have no useful purpose. To most users they are just fluff on the upholstery and should be removed. 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 a user exceeds their limit they do not often understand what has happened. Usually users do not even get a message unless they are logging in. Quotas also prevent users from creating large temporary files which can be a problem when compiling programs. They carry with them a system overhead, which makes everything run a little slower.

Deciding whether to delete files automatically is a policy decision. It might be deemed totalitarian to delete files without asking. On the other hand, this is often the only way to ever clear anything up. Many users will be happy if they do not have to think about the problem themselves, however one has to be careful never to delete file which cannot be regenerated or re-acquired if necessary.

A useful strategy is to delete files one is not sure about only if they have not be *accessed* for a certain period of time, say a week. This allows users to use files freely as long as they need to, but prevents them from keeping the files around for ever. Cfengine can be used to perform this task, See section Using cfengine. For example:

```
control:

   actionsequence = ( tidy )
```

```
  #
  # 3 days minimum, remember weekends
  #

 tidy:

     home                 pattern=core  recurse=inf age=0
     home                 pattern=a.out recurse=inf age=3
     home                 pattern=*%    recurse=inf age=3
     home                 pattern=*~    recurse=inf age=3
     home                 pattern=*.o   recurse=inf age=1
     home                 pattern=*.aux recurse=inf age=3
     home                 pattern=*.mp3 recurse=inf age=14

     home/Desktop/Trash   pattern=*     recurse=inf age=14
     home/.netscape/cache pattern=*     recurse=inf age=0
```

Killing old processes

Cfengine can also be used to clean up old processes. Processes sometimes do not get terminated when they should. There are several reasons for this. Sometimes users forget to log out, sometimes poorly written terminal software does not properly kill its processes when a user logs out. Sometimes background programs simply crash or go into loops from which they never return. One way to clean up processes in a work environment is to look for user processes which have run for more than a day. (Note that the assumption here is that everyone is supposed to log out each day and then log in again the next day--that is not always the case.) Cfengine's processes commands are used to match processes in the process table (which can be seen by running `ps ax`). Here is an example:

```
 control:

   actionsequence = ( processes )

 processes:

  "Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec"

      signal=kill

      include=tcsh
      include=xterm
      include=netscape
      include=ftp
      include=tkrat
      include=pine
      include=irc
      include=kfm
      include=java
```

This rule works by noticing that, when processes are more than a day old, the date (i.e. the name of the month) appears in the process listing. Thus to find processes which are more than a day old we only need to search for any line containing the name of a month and then pick out a subset of those which contain the strings `tcsh', `xterm' etc. This is an extremely useful way of cleaning up old processes which poor terminal software leave behind, or which forgetful users leave behind when they forget to log out. In the latter case, user security is also protected by this clean up operation.

PART II: PRACTICE

Practice

Network orientation

@hrule @vskip 0.3cm

This chapter assumes that you have a reasonably functional network already. It is about how to understand how a network works. We begin from the perspective of someone who has just moved to a new job where the network is unfamiliar.

The first step in familiarizing oneself with a computer network is to find out about the local domain, all the hosts connected to it and who is responsible for what. Then one needs an overview of where the cables run and what operating systems are running.

Knowing the domain

The familiarization process begins by analysing the network and all of its inter-relationships. It is especially important to know who is responsible for maintaining different parts of the network. It might be you or it might be someone else. Who runs the DNS domain above yours? Who do you call if the internet connection is broken? What service contracts might exist on hardware, what upgrade possibilities are there on software? What system is in use for making backups? How does one obtain a backup should the need arise? In short, it is essential to know where to begin in solving any problem which might arises, and who to call if the responsibility for a problem lies with someone else.

After these organizational points, one needs an overview of the hosts. A host list can probably be obtained from the DNS using `nslookup`. If there are Unix systems, one can start by logging onto each machine and using the `uname` command to find out what OS is being used:

```
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 GNU/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 a host have? (Most systems print this when they boot. Sometimes the information can be coaxed out of the system in other ways.) What disks and other devices are in use?
- Use `locate` and `find` and `which` and `whereis` to find important directories and software. How is the software layed out?
- 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 Unix systems have a C compiler and a C++ compiler installed? This is often needed for installing software.

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
```

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.org
Server:  nexus.iu.hioslo.no
Address:  128.39.89.10

Name:    www.gnu.org
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.org` 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(4).

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:
      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 Info
(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 GNU/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 one does 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!

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 (integrated drive electronics) 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

Physical network issues

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 or repeater.

Star net/twisted pair (xBaseT)
: 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

Using cfengine

@hrule @vskip 0.3cm

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.

What is cfengine?

System maintenance involves a lot of jobs which are repetitive and menial. 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. 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 enormous 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 host-types. 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 GNU/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 tedious business. Usually though, we are trying to make hosts on a network basically the same as one another so we can make generic rules which cover many hosts at a time. Nonetheless 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 GNU/Linux operating system, so some rules will apply to all hosts which run Solaris, whereas others will only apply to GNU/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.

Classes

The idea of classes is central to the operation of cfengine. Saying that cfengine is `class oriented' means that it doesn't make decisions using `if...then...else` constructions the way other languages do, but only carries out an action if the host running the program is in the same class as the action itself. To understand what this means, imagine sorting through a list of all the hosts at your site. Imagine also that you are looking for the *class* of hosts which belong to the computing department, which run GNU/Linux operating system and which have yellow spots! To figure out whether a particular host satisfies all of these criteria you first delete all of the hosts which are not GNU/Linux, then you delete all of the remaining ones which don't belong to the computing department, then you delete all the remaining ones which don't have yellow spots. If you are on the remaining list, then you are in the class of all computer-science-Linux-yellow-spotted hosts and you can carry out the action.

Cfengine works in this way, narrowing things down by asking if a host is in several classes at the same time. Although some information (like the kind of operating system you are running) can be obtained directly, clearly, to make this work we need to have lists of which hosts belong to the computer department and which ones have yellow spots.

So how does this work in a cfengine program? A program or configuration script consists of a set of declarations for what we refer to as *actions* which are to be carried out only for certain classes of host. Any host can execute a particular program, but only certain action are extracted -- namely those which refer to that particular host. This happens automatically because cfengine builds up a list of the classes to which it belongs as it goes along, so it avoids having to make many decisions over and over again.

By defining classes which classify the hosts on your network in some easy to understand way, you can make a single action apply to many hosts in one go -- i.e. just the hosts you need. You can make generic rules for specific type of operating system, you can group together clusters of workstations according to who will be using them and you can paint yellow spots on them -- what ever works for you.

A *cfengine action* looks like this:

```
action-type :

   compound-class ::

       declaration
```

A single class can be one of several things:

- The name of an operating system architecture e.g. `ultrix`, `sun4` etc. This is referred to henceforth as a *hard class* .
- The (unqualified) name of a particular host. If your system returns a fully qualified domain name for your host, cfengine truncates it so as to unqualify the name.
- The name of a user-defined group of hosts.
- A day of the week (in the form `Monday Tuesday Wednesday..`).
- An hour of the day (in the form Hr00, Hr01 ... Hr23).
- Minutes in the hour (in the form Min00, Min17 ... Min45).
- A five minute interval in the hour (in the form Min00_05, Min05_10 ... Min55_00)
- A day of the month (in the form Day1 ... Day31).
- A month (in the form January, February, ... December).
- A year (in the form Yr1997, Yr2001).
- An arbitrary user-defined string.

A compound class is a sequence of simple classes connected by dots or `pipe' symbols (vertical bars). For example:

```
myclass.sun4.Monday::

sun4|ultrix|osf::
```

A compound class evaluates to `true' if all of the individual classes are separately true, thus in the above example the actions which follow `compound_class::` are only carried out if the host concerned is in `myclass`, is of type `sun4` and the day is Monday! In the second example, the host parsing the file must be either of type `sun4` *or* `ultrix` *or* `osf`. In other words, compound classes support two operators: AND and OR, written `` `.' `` and `` `|' `` respectively. Cfengine doesn't care how many of these operators you use (since it skips over blank class names), so you could write either

```
solaris|irix::
```

or

```
solaris||irix::
```

depending on your taste. On the other hand, the order in which cfengine evaluates AND and OR operations *does* matter, and the rule is that AND takes priority over OR, so that `.' binds classes together tightly and all AND operations are evaluated before ORing the final results together. This is the usual behaviour in programming languages. You can use round parentheses in cfengine classes to override these preferences.

Cfengine allows you to define switch on and off dummy classes so that you can use them to select certain subsets of action. In particular, note that by defining your own classes, using them to make compound rules of this type, and then switching them on and off, you can also switch on and off the corresponding actions in a controlled way. The command line options `-D` and `-N` can be used for this purpose.

A logical NOT operator has been added to allow you to exclude certain specific hosts in a more flexible way. The logical NOT operator is (as in C and C++) `!'. For instance, the following example would allow all hosts except for `myhost`:

```
   action :

    !myhost::

        command
```

and similarly, so allow all hosts in a user-defined group `mygroup`, *except* for `specialhost`, you would write

```
   action :

    mygroup.!myhost::

        command
```

which reads `mygroup AND NOT myhost'. The NOT operator can also be combined with OR. For instance

```
   class1 |!class2
```

would select hosts which were either in class 1, or those which were not in class 2.

Finally, there is a number of reserved classes. The following are hard classes for various operating system architectures. They do not need to be defined because each host knows what operating system it is running. Thus the appropriate one of these will always be defined on each host. Similarly the day of the week is clearly not open to definition, unless you are running cfengine from outer space. The reserved classes are:

```
ultrix, sun4, sun3, hpux, hpux10, aix, solaris, osf, irix4, irix, irix64
        freebsd, netbsd, openbsd, bsd4_3, newsos, solarisx86, aos,
            nextstep, bsdos, linux, debian, cray, unix_sv, GnU
```

If these classes are not sufficient to distinguish the hosts on your network, cfengine provides more specific classes which contain the name and release of the operating system. To find out what these look like for your systems you can run cfengine in `parse-only-verbose' mode:

```
  cfengine -p -v
```

and these will be displayed. For example, solaris 2.4 systems generate the additional classes `sunos_5_4` and `sunos_sun4m`, `sunos_sun4m_5_4`.

Cfengine uses both the unqualified and fully host names as classes. Some sites and operating systems use fully qualified names for their hosts. i.e. `uname -n` returns to full domain qualified hostname. This spoils the class matching algorithms for cfengine, so cfengine automatically truncates names which contain a dot `.' at the first `.' it encounters. If your hostnames contain dots (which do not refer to a domain name, then cfengine will be confused. The moral is: don't have dots in your host names! *NOTE: in order to ensure that the fully qualified name of the host becomes a class you must define the domain variable.* The dots in this string will be replaced by underscores.

In summary, the operator ordering in cfengine classes is as follows:

`()' Parentheses override everything.
`!' The NOT operator binds tightest.
`.' The AND operator binds more tightly than OR.
`|' OR is the weakest operator.

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.

Using cfengine as a front end for `cron`

One of cfengine's strengths is its use of classes to identify systems from a single file or set of files. Many administrators think that it would be nice if the cron daemon also worked in this way. One possible way of setting up cron from a global configuration would be to use the cfengine `editfiles` facility to edit each cron file separately. A much better way is to use cfengine's time classes to work like a user interface for cron. This allows you to have a single, central cfengine file which contains all the cron jobs on your system without losing any of the fine control which cron affords you. All of the usual advantages apply:

- It is easier to keep track of what cron jobs are running on the system when you have everything in one place.
- You can use all of your carefully crafted groups and user-defined classes to identify which host should run which programs.

The central idea behind this scheme is to set up a regular cron job on every system which executes cfengine at frequent intervals. Each time cfengine is started, it evaluates time classes and executes the shell commands defined in its configuration file. In this way we use cfengine as a wrapper for the cron scripts, so that we can use cfengine's classes to control jobs for multiple hosts. Cfengine's time classes are at least as powerful as `cron`'s time specification possibilities, so this does not restrict you in any way. The only price is the overhead of parsing the cfengine configuration file.

To be more concrete, imagine installing the following `` `crontab' `` file onto every host on your network:

```
  #
  # Global Cron file
  #
  0,15,30,45 * * * * /usr/local/cfengine/inputs/run-cfengine
```

This file contains just a single cron job, namely a script which calls cfengine. Here we are assuming that you will not want to execute any cron script more often than every fifteen minutes. If this is too restrictive, the above can be changed. We refer to the time interval between runs of the script `` `run-cfengine' `` as the `scheduling interval' and discuss its implications in more detail below.

The script `` `run-cfengine' `` would replace any `` `cfdaily' `` or `` `cfhourly' `` scripts which you might have, and can as simple as this

```
 #!/bin/sh
 #
 # Script run-cfengine

 export CFINPUTS=/usr/local/cfengine/inputs

 /usr/local/gnu/bin/cfengine

 #
 # Should we pipe mail to a special user?
 #
```

or it could be more fancy. You could also use the `` `cfwrap' `` script, if you have perl on all your systems, to pipe mail to the mail address described in the cfengine file.

```
 #
 # Global Cron file
 #
 0,15,30,45 * * * * path /cfwrap path /run-cfengine
```

You might not want to run your entire system configuration `` `cfengine.conf' `` every time cron fires up cfengine. An alternative would be to keep a separate file for cron jobs called, say, `` `cf.cron' ``. You would then replace the `` `run-cfengine' `` file by

```
#!/bin/sh
#
# Script run-cfengine

 export CFINPUTS=/usr/local/cfengine/inputs

 /usr/local/gnu/bin/cfengine -f cf.cron

 #
 # Should we pipe mail to a special user?
 #
```

There is no particular advantage to doing this unless you are running cfengine on some very slow hardware. A better way to approach the problem is to think of the `cf.cron' file as a module which can be imported into the main configuration file. This gives you the maximum amount of flexibility, since it allows you to decide exactly what you want to happen any any given time from the central file.

Building flexible time classes

Each time cfengine is run, it reads the system clock and defines the following classes based on the time and date:

Yr*xx* ::
: The current year, e.g. `Yr1997', `Yr2001'. This class is probably not useful very often, but it might help you to turn on the new-year lights, or shine up your systems for the new millennium!

Month ::
: The current month can be used for defining very long term variations in the system configuration, e.g. `January', `February'. These classes could be used to determine when students have their summer vacation, for instance, in order to perform extra tidying, or to specially maintain some administrative policy for the duration of a conference.

Day :: The day of the week may be used as a class, e.g. `Monday', `Sunday'.

Day*xx* ::
: A day in the month (date) may be used to single out by date, e.g. the first day of each month defines `Day1', the 21st `Day21' etc.

Hr*xx* ::
: An hour of the day, in 24-hour clock notation: `Hr00'...`Hr23'.

Min*xx* ::
: The precise minute a which cfengine was started: `Min0' ... `Min59'. This is probably not useful alone, but these values may be combined to define arbitrary intervals of time.

Min*xx* _*xx* ::
: The five-minute interval in the hour at which cfengine was executed, in the form `Min0_5', `Min5_10' .. `Min55_0'.

Time classes based on the precise minute at which cfengine started are unlikely to be useful, since it is improbable that you will want to ask cron to run cfengine every single minute of every day: there would be no time for anything to complete before it was started again. Moreover, many things could conspire to delay the precise time at which cfengine were started. The real purpose in being able to detect the precise start time is to define composite classes which refer to arbitrary intervals of time. To do this, we use the `group` or `classes` action to create an alias for a group of time values. Here are some creative examples:

```
classes:  # synonym groups:

  LunchAndTeaBreaks = ( Hr12 Hr10 Hr15 )

  NightShift        = ( Hr22 Hr23 Hr00 Hr01 Hr02 Hr03 Hr04 Hr05 Hr06 )

  ConferenceDays    = ( Day26 Day27 Day29 Day30 )

  QuarterHours      = ( Min00 Min15 Min30 Min45 )

  TimeSlices        = ( Min01 Min02 Min03 Min33 Min34 Min35)
```

In these examples, the left hand sides of the assignments are effectively the OR-ed result of the right hand side. This if any classes in the parentheses are defined, the left hand side class will become defined. This provides an excellent and readable way of pinpointing intervals of time within a program, without having to use `|' and `.' operators everywhere.

Choosing a scheduling interval

How often should you call your global cron script? There are several things to think about:

- How much fine control do you need? Running cron jobs once each hour is usually enough for most tasks, but you might need to exercise finer control for a few special tasks.
- Are you going to run the entire cfengine configuration file or a special light-weight file?
- System latency. How long will it take to load, parse and run the cfengine script?

Cfengine has an intelligent locking and timeout policy which should be sufficient to handle hanging shell commands from previous crons so that no overlap can take place.

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. See http://www.iu.hioslo.no/cfengine
- 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

@hrule @vskip 0.3cm

Summoning daemons

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 some popularity and is used by several GNU/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
#
# NB wu-ftpd needs -a now
#
ftp     stream tcp nowait root   /usr/sbin/in.ftpd      in.ftpd -a
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`*

TCP wrapper security

One of the problems with `inetd` is that it accepts connections from any host and passes them to to services registered in its configuration file without question. In todays network climate this is a dangerous step and it is usually desirable to limit the availability of certain services. For instance, services which are purely local (like RPC) should never be left open so that outside users could try to exploit them. In short, services should only be made available to those who need them. If they are left open to those who do not need them, you invite attacks on your system.

TCP wrappers is a solution to this problem. In short it gives you the possibility of adding Access Control Lists (ACLs) to your network services.

TCP wrapper expects to find daemons in a special directory.

```
  # hosts.allow
  in.fingerd: .iu.hioslo.no .hioslo.no LOCAL
  in.cfingerd: .iu.hioslo.no LOCAL
  sendmail: ALL
  cfd: .iu.hioslo.no LOCAL
  in.rlogin: ALL
  in.telnetd: ALL
  sshd: ALL
  sshdfwd-X11: .iu.hioslo.no
```

All other services are denied access:

```
  ALL: ALL
```

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 GNU/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

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 name-service 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 name-service 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 name-service 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(5). 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

```
;
; 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

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. This can also be used to specify any special error handling, such as customized error pages.

The sample files for in which the root files for WWW are attached to a special user called `www'. The daemon will run with the access rights of this user for all operations. The UserDir directive tells the daemon that all users' personal WWW pages will be kept in a subdirectory of their home directories called `www'. Thus using this arrangement (user www, with subdir www) the DocumentRoot for the main WWW pages becomes

```
(home)/www/index.html
```

which, using the file scheme */site/host/contents* becomes

```
/site/server/www/www/index.html
```

Having installed the WWW server, you might want to provide a connection with a database engine, such as an MSQL database. This is covered briefly in the last section.

Sample `access.conf' file

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

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

```
</Directory>

<Directory /site/server/home/www/www>
Options Indexes FollowSymLinks Includes
</Directory>

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

`srm.conf' file

```
#
###
## srm.conf -- Apache HTTP server configuration file
##

# With this document, you define the name space that users see of your http
# server.  This file also defines server settings which affect how requests are
# serviced, and how results should be formatted.

# See the tutorials at http://www.apache.org/ for
# more information.

# Originally by Rob McCool; Adapted for Apache

# 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 /iu/nexus/ua/www/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 or files to use as a pre-written HTML
# directory index.  Separate multiple entries with spaces.

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

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

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^^

# 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* RCS

# 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

# 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 (with no limit). The format is
# Alias fakename realname

# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL.  So "/icons" isn't aliased in this
# example.

Alias /icons/ /local/iu/share/apache/icons/

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

ScriptAlias /cgi-bin/ /site/server/home/www/www/cgi-bin-public/

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

# 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

# For example, the PHP3 module (not part of the Apache distribution)
# will typically use:
```

```
AddType application/x-httpd-php3 .phtml
AddType application/x-httpd-php3-source .phps
AddType application/x-httpd-php3 .php3 .php

# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
# Format: AddHandler action-name ext1

# To use CGI scripts:
#AddHandler cgi-script .cgi

# To use server-parsed HTML files
AddType text/html .shtml
AddHandler server-parsed .shtml

# Uncomment the following line to enable Apache's send-asis HTTP file
# feature

#AddHandler send-as-is asis

# If you wish to use server-parsed imagemap files, use
AddHandler imap-file map

# To enable type maps, you might want to use
#AddHandler type-map var

# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location

# 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
#

ErrorDocument 404 /400.html
ErrorDocument 401 /400.html
ErrorDocument 500 /500.html

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

#MimeMagicFile /local/iu/etc/apache/magic

# The following directives disable keepalives and HTTP header flushes.
# The first directive disables it for Netscape 2.x and browsers which
# spoof it. There are known problems with these.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4.0b2;" nokeepalive downgrade-1.0 force-response-1.0

# 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.

BrowserMatch "RealPlayer 4.0" force-response-1.0
BrowserMatch "Java/1.0" force-response-1.0
BrowserMatch "JDK/1.0" force-response-1.0
```

```
ScriptAlias /cgi-bin-public /iu/nexus/ua/www/www/cgi-bin-public

# More script aliases here...
```

Notice that this file can be used to create customized error messages, tailored to your own special environment, perhaps with personal logo etc. The `ErrorDocument` directive is used for this. It traps error numbers and maps them to special pages. For example, to map to a standard local file in the root of the server's pages one would add

```
ErrorDocument 500 /errorpage.html
ErrorDocument 401 /errorpage.html
ErrorDocument 402 /errorpage.html
```

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";
$srmbase = "/local/iu/lib/apache/srm.conf";

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

open (BASE,"$srm") || die;

  while (<BASE>)    # Copy base file to output
     {
     print OUT $_;
     }

close (BASE);

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
```

Private directories

In some cases you will want certain information to be made available to local users of your domain but not to general outside users. This can be accomplished by using a `` `.htaccess' `` file to override the default access rights set in the server configuration files. The assumes that you have set `AllowOverride All`.

Creating a directory that is only available from the local domain is a simple matter of creating the directory and creating a `` `.htaccess' `` file owned by the `www' user (i.e. the user running the daemon) with read permission for `www', containing the lines:

```
order deny,allow
deny from all
allow from .local.domain
```

SQL/PHP

For many sites the possibility to combine a database of information with its web pages is a powerful one. The language PHP is designed for this purpose. In order to use PHP it has to be coupled to a local WWW server and to a specific database. The order in which these three components is configured is important. Here is an example using MySQL, PHP and the Apache web server. This is a powerful and popular combination of free components.

For some of its graphical functions, PHP uses a library called gd. This can be obtained from http://www.boutell.com/gd/ and added with a configuration option as secribed below.

The first step is to compile the database engine. Note that the source tree has to be remain on the system, so it should be unpacked in its final location. The other software packages expect to find the msql server in the directory `/usr/local/mysql'. You should make a special user for the mysql daemon so that it does not need to run as root.

```
cd /usr/local
tar zxf mysql-xxx.tar.gz

cd /local/mysql-xxx
configure --with-pthread --with-unix-socket-path=/home/mysql.socket  --with-mysqld-user=mysql --prefix=/usr/local/mysql
make
make install

# After check that the daemon really is started as mysql user!!!
```

Next it is necessary to make a temporary build of the WWW server. After this is done, PHP can be compiled and installed. Then one must go back and make a proper build of the WWW server. These can be kept in `/usr/local' or in site dependent files `/local/site', as you see fit. Again, the source trees have to remain on the system.

```
cd /local/site
tar zxf apache_1.3.3.tar.gz
cd /local/site/apache_1.3.3

./configure --prefix=/local/site

cd /local/site
tar zxf php3.tar.gz

cd /local/site/php3
./configure --prefix=/local/site --with-apache=/local/site/apache_1.3.3 --with-mysql=/local/mysql --with-zlib --with-gd=/usr/l
make install

cd /local/site/apache_1.3.3
./configure --prefix=/local/site --activate-module=src/modules/php3/libphp3.a
make install
```

There are lots of things which can go wrong with options here. These options work for building a php interpreter into the apache server. You might also want to use php as a CGI scripting language. In that case you need to build a binary separately. To do this you configure as follows:

```
cd /local/site/php3
./configure --prefix=/local/site --with-mysql=/local/mysql --with-zlib --with-gd=/usr/local/lib --enable-discard-path
make install
```

E-mail configuration

Nightmares on ELM street!

Electronic mail configuration is something 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 most 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 GNU/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 order for mail to be diverted to a hub, one has to arrange for the mail exchanger data in DNS to point to the hub, for every system. i.e. for every host in DNS one should have an MX record accompanying the A record:

```
hostname   A   xxx.yyy.zzx.mmm
           MX  mailhub.domain
```

Without such an MX record, mail which is addressed to user@hostname.domain will be sent directly to hostname. With such a record the mail for hostname is sent to mailhub.domain instead. It can later be forwarded to hostname if desired using a mailertable. This has several advantages: first of all it means that mail configuration can be centralized, spam filtering can be performed even for dumb hosts and aliases can be expanded here without the need for a distributed alias database like NIS. The second advantage concerns security. If all mail is forced to pass through this hub then a secure set-up here will prevent SNMP attacks on weaker hosts, thus this simplifies the security administration of mail also. One may concentrate most of one's effort on securing this hub, knowing that nothing very bad will happen. A further precaution is then to configure the site router to accept SNMP traffic only for the mailhub since it is supposed to go there anyway. That way, if one forgets an MX record in DNS there will be no back-doors for would-be attackers.

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-8.9.1:

```
host# cd sendmail-8.9.1
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 for several reasons. The main reason is that it not required as long as mail passes through a central hub, which is a good approach to mail configuration for small to medium sized sites.

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 significant 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-8.9.1/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. This example is for sendmail 8.9.1 for a mail hub:

```
divert(-1)
include(`/local/iu/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)
MASQUERADE_DOMAIN(pc.iu.hioslo.no)
```

```
MASQUERADE_DOMAIN(ca30s.iu.hioslo.no)

FEATURE(use_cw_file)
FEATURE(use_ct_file)
FEATURE(redirect)
FEATURE(relay_entire_domain)
FEATURE(always_add_domain)
FEATURE(allmasquerade)
FEATURE(masquerade_envelope)
FEATURE(domaintable, `hash -o /local/iu/mail/lib/domaintable')
FEATURE(mailertable, `hash -o /local/iu/mail/lib/mailertable')
FEATURE(access_db, `hash -o /local/iu/mail/lib/access_db')
FEATURE(genericstable, `hash -o /local/iu/mail/lib/genericstable')
FEATURE(virtusertable, `hash -o /local/iu/mail/lib/virtusertable')

FEATURE(local_procmail,`/local/bin/procmail')

GENERICS_DOMAIN_FILE(/local/iu/mail/lib/sendmail.cG)

EXPOSED_USER(root)

define(`ALIAS_FILE', /local/iu/mail/lib/aliases)dnl
define(`HELP_FILE', /local/iu/mail/lib/sendmail.hf)dnl
define(`STATUS_FILE', /local/iu/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(`SMTP_MAIL_MAX',`2000000')
define(`confMAX_MESSAGE_SIZE', `20000000')
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', `/local/iu/mail/lib/sendmail.cw')
define(`confCT_FILE', `/local/iu/mail/lib/sendmail.ct')
define(`confUSERDB_SPEC', `/local/iu/mail/lib/userdb.db')
define(`confSMTP_MAILER', `esmtp')
define(`LOCAL_SHELL_PATH',`/local/iu/mail/bin/smrsh')

MAILER(local)
MAILER(smtp)

FEATURE(rbl) dnl vixie's black hole database for spammers
```

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
ACCESSDB=       lib/accessdb
CF_DIR=               cf/

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

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

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

$(ACCESSDB).db: $(ACCESSDB)
       $(MAKEMAP) hash $(ACCESSDB) < $(ACCESSDB)

#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
```

This is a shorter example for a system attached to a mail hub, whose only function is to send the mail to the hub for processing.

```
divert(0)

OSTYPE(solaris)dnl
FEATURE(nullclient, mailhost.iu.hioslo.no)dnl
```

```
MASQUERADE_AS(iu.hioslo.no)dnl
define(`MAIL_HUB',`mailhost.iu.hioslo.no')
define(`SMART_HOST',`mailhost.iu.hioslo.no')
```

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.

smrsh

The sendmail remote shell is a security measure to prevent system crackers from executing an arbitrary program on your system. The `smrsh` program is contained in the sendmail distribution and is configured by using the FEATURE

Spam and junk mail

Spam mail is E-mail which is sent repetetively as a would-be denial of service attack. The word comes from the Monty Python spam song sketch. Junk mail is unwanted mail, often advertisements about financial opportunities or pornography, sometimes hoaxes. Often these two kinds of unwanted mail are lumped together and called collectively `spam'.

Spam has become a major problem since it is very easy to send E-mail and very hard to pick out what it useful from what is useless. There are two approaches to the filtering of spam, both of which are needed together:

- Site rules for rejecting mail (ACLs).
- Private user-rules for rejecting mail.

The reason why both of these is needed is that what one user wants to reject, another user might be glad to receive. Users prospecting for finanical opportunities or collecting the latest `artwork' might live for the messages which most of us get annoyed with.

Sendmail has rules for filtering mail at the site level. These include the ability to deny access to connecting mailers from certain domains.

At the user level, users of `procmail` can use `junkfilter` to create their own rules for rejecting spam. See http://www.pobox.com/~gsutter/junkfilter/ for details.

Policy decisions

In order to protect your site from E-mail attacks, even ones made in innocence you might want to restrict mail by other criteria. For example, multimedia attachments can now allow users to sent huge files by email. This is a very inefficient way of sending large amounts of data and it causes problems for mailbox storage space. A possibility is to limit the size of mail messages handled by sendmail so that mail which is too large will be rejected with an error message. For example, the following rules limit E-mail to approximately 20MB. Even with such a large reject size a handful of messages per month are rejected on the basis of this rule.

```
define(`SMTP_MAIL_MAX',`2000000')
define(`confMAX_MESSAGE_SIZE', `20000000')
```

Again, this must be a policy decision like garbage collection of users' files. It is never desirable to restrict the personal freedom of users, but it becomes a matter of survival. If one provides an opening, it will be exploited either through ignorance or malice.

PART III: UNIX

Unix

Overview of Unix

@hrule @vskip 0.3cm

Useful Unix commands

Typed commands are infinitely more flexible than graphical (GUI) based programs. You can tell the system what you want to do, rather than having to search through the menus to find out whether or not you are allowed to do what you want. As a system administrator you will find most GUI programs useless for any real tasks.

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. GNU/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
 8  icm-pen-1-H2/0-T3.icp.net (198.67.131.25)  162 ms  136 ms
 9  icm-pen-10-P4/0-OC3C.icp.net (198.67.142.69)  198 ms  134
10  bbnplanet1.sprintnap.net (192.157.69.51)  146 ms  297 ms
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
13  cambridge1-br1.bbnplanet.net (4.0.1.122)  131 ms  136 ms
14  cambridge1-br1.bbnplanet.net (4.0.1.122)  133 ms  124 ms
15  cambridge1-cr1.bbnplanet.net (206.34.78.23)  138 ms  129
16  cambridge2-cr2.bbnplanet.net (192.233.149.202)  128 ms
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
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 GNU/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.

Strengths and weaknesses of Unix

The main strength of Unix in a network environment is its stability and experience. It uses open standards which means that bugs are more quickly found. It scales well under load.

Amongst the disadvantages: GNU/Linux is an attractive target for hackers. Unix has expensive commercial software. Courses and books on Unix tend to be ten years out of date.

Installation procedure

@hrule @vskip 0.3cm

Information is the beginning of the system administrators wisdom, but you need to know how to find new 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:

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

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 GNU/Linux at

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

Once you have installed your operating system, remember:

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 GNU/Linux operating system.

When all software packages have been installed, and all necessary 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/my-machine.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 GNU/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/my-machine.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 that 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

If you are installing a PC, you will probably be interested in the idea of being able to choose between operating systems. You might want DOS to play games, Unix for serious work, NT or Windows for its well-known applications and perhaps OS/2. 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?

A word of warning: because of the wide range of system cracking tools available to users, it can be risky to install dual-boot operating systems on any important host. NT is particularly vulnerable to password editing on dual boot systems.

The installation programs for NT and GNU/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 GNU/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/hda Linux-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 GNU/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 GNU/Linux partition. To fix this problem, boot with the GNU/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 GNU/Linux partition is not formatted.

Installing a diskless client

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.

Cloning systems

We are almost never interested in installing every machine separately. A real system administrator usually has to install ten, twenty or even a hundred machines at a time. She would also like them to be as far as possible the same, so that users will always know what to expect. This might sound like a

straightforward problem, but it is not. There are several approaches.

- Some operating systems provide a solution to this using package templates so that the installation procedure becomes standardized.
- You copy the hard-disks of one machine physically and then change the hostname and address afterwards.
- You can put all your software on one machine and share the disk via NFS.

Each of these approaches has its attractions. The NFS approach is without doubt the least amount of work, since it involves installing the software only once, but it is also the slowest in operation for users.

As an example of the first, here is how Debian GNU/Linux tackles this problem using the Debian package system:

```
Install one system

  dpkg --get-selections > file

On the remaining machines you type

  cat file | dpkg --set-selections

before you call dselect After that, just install.
```

Booting a machine

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 GNU/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 GNU/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

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 GNU/Linux

```
# See exports(5) for a description.
# This file contains a list of all dirs 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 after editing this file in order to register 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: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

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(6) .

```
/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 GNU/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

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 hard-disks, it might be a good idea to talk to someone who can advise you.

The first concern is what type hard-disk. There are several types of disk interface used for communicating with hard-disks. 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

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 `myprinter' 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`(7).

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

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`.

LPRng

A recent and welcome addition to the printer debate is the Next Generation LPR package by Patrick Powell. LPRng is a drop in replacement for both BSD and system 5 print systems. It is configured quite simply in a manner very similar to (but not identical to) the Berkeley printcap system. LPRng can be obtained from @uref{http://www.astart.com/lprng/LPRng.html} and it is a real god send if you have system 5 (e.g. Solaris) hosts to grapple with. The software uses a printcap file and two other optional files called `` `lpd.conf' `` and `` `lpd.perms' ``. The printcap file is like a regular printcap file but without the backslash continuation characters.

LPRng provides effectively both `` `lpr' ``, `` `lpd' ``, `` `lpq' `` and `` `lprm' `` commands from Berkeley and `` `lp' ``, `` `lpstat' `` and `` `cancel' `` commands from system 5. The daemon reads the three configuration files an handles spooling. The configuration is challenging but straightforward and there is extensive documentation. Here is a simple example for a network printer (with its own IP address) which allows logged on users to start and delete their own printjobs:

```
# /etc/printcap (lprng)

myprinter|lp
     :if=/local/bin/lpf          # LF/CR filter
     :af=/var/spool/lpd/acctfil
     :lf=/var/spool/lpd/printlog
     :sd=/var/spool/myprinter
     :lp=xxx.yyy.zzz.mmm%9100
     :rw
     :sh
```

The IP address of the printer is *xxx.yyy.zzz.mmm* and it must be written in numerical form. The percent symbol marks the standard port 9100. The `` `lpd.conf' `` file is slightly mysterious but has a number of useful options. Most, if not all of these can be set in the printcap file also, but options set here apply for all printers. One nice feature for instance is the ability to reject printouts of binary (non-printable) files. This can save a few rain forests if someone is kind enough to dump `` `/bin/ls' `` to the printer.

```
#
# lpd.conf
#

# Purpose: name of accounting file (see also la, ar)
af=/var/spool/lpd/acctfil

# Purpose: accounting at start (see also af, la, ar)
as=jobstart $H $n $P $k $b $t

# Purpose: check for nonprintable file
check_for_nonprintable

# Purpose: default printer
default_printer=local

# Purpose: error log file (servers, filters and prefilters)
lf=/var/adm/printlog

# Purpose: lpd lock file
lockfile=/var/spool/lpd/lpd.lock.%h

# Purpose: lpd log file
logfile=/var/spool/lpd/lpd.log.%h

# Purpose: /etc/printcap files
printcap_path=/etc/printcap

# Purpose: suppress headers and/or banner page
sh
```

The `` `lpd.perms' `` file sets limits on who can access the printers and from where, unlike the traditional services which are open to everyone.

```
#
# lpd.perms
#
# allow root on server to control jobs
ACCEPT SERVICE=C SERVER REMOTEUSER=root
# allow anybody to get status
ACCEPT SERVICE=S
# reject all others, including lpc commands permitted by user_lpc
REJECT SERVICE=CSU
#
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M
```

```
# All other operations disallowed
DEFAULT REJECT    # orACCEPT
```

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

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, together with the filename of the software 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'(8). 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.5.1:tc=.iu
xterm2:ha=080006030EC0:ip=128.39.89.248:bf=/td/servers/XtdF.5.1:tc=.iu
xterm3:ha=080006030FB7:ip=128.39.89.253:bf=/td/servers/XtdF.5.1: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

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, analogous 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

@hrule @vskip 0.3cm

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.

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. GNU/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

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

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 enormous 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.

GNU/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 optimized 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.

Installing software

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 instructions 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.

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:

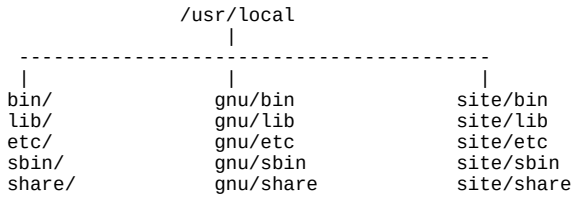
- You should keep installed software separate from the operating system installed files, so that the OS can be reinstalled or upgraded without ruining your software installation.
- Compiled software should be grouped together, with a `bin' directory and a `lib' directory so that binaries and libraries 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.
GNU/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!

Installing a new service

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

When mail cannot be delivered immediately it is placed in the mail queue. To see what mail is waiting in the mail-queue, 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

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
```

Setting clocks

You need to keep your system clock synchronized. 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 GNU/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

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 introduced in the 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) implementation is based on the POSIX draft. We shall follow Solaris ACLs in this section. Not all Unix systems have ACLs. GNU/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.

PART IV: NT

NT

NT Overview

@hrule @vskip 0.3cm

NT is a multitasking operating system from Microsoft which allows one user at a time to log in to a console. The consoles may be joined together in a network with common resources shared by a `domain'. An NT host is either a network domain server or a personal workstation. NT is quite new, at least in terms of age, and Microsoft has reinvented many well known systems rather than use tried and tested solutions. This has led to a history of bugs and security issues and some disillusionment in the popular press. Together with GNU/Linux and IRIX, NT is the most popular target for network attacks. NT has a long way to go before it becomes mature, so significant changes can be expected in the next few years. The pace of development is already rapid and NT is getting steadily closer to Unix. Here we present a provisional overview as a point of reference and comparison.

The basic NT distribution contains few tools which can be used for network administration. The NT Resource Kit is an extra package which, for a tidy sum of money, provides many essential tools. Other tools can be obtained free of charge.

NT can be both easier and harder to administrate than Unix. It can be easier because the centralized model of having one domain server running the network services means that all configuration information can be left in one place (on the server), and that each workstation can be made (at least to a limited degree) to configure itself from the server's files.

It is harder to administrate because the only tools provided for system administration tasks work by the GUI (*graphical user interface*) and this is not a suitable tool for addressing the issues of hundreds of hosts. The Resource Kit and free tools go some way to helping with this problem, but the Resource kit adds a substantial sum to the cost of running NT. It provides tools analogous to `cron` and script languages for automating tasks.

NT does not have a remote shell login feature like Unix, so the administrative job cannot be handled remotely, unless the PC is tied specifically to a central server for all of its maintenance. Fortunately the free Perl Win32 package and related tools provides tools for solving a number of problems with NT from a script viewpoint.

Logging in

In order to log on to NT the user must type CTRL-ALT-DEL. This is a `feature' which is intended to serve a security purpose. Any login screen which one might see prior to pressing this key combination is likely to be a Trojan horse program. Windows NT is not a multiuser operating system, only one user may be logged in at any time. The superuser account is called the Administrator account. In fact any account which is a member of the group Administrators has the same privileges. These accounts have the power to do anything with the system, just as in Unix. However it is possible to remove access rights to the Administrator as a precautionary measure; the Administrator then has to grant him/herself access before continuing.

System layout

The first thing to learn about a new operating system is where all the important files live on the disk, where new software should be installed and where new users should have home directories.

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. The system root is usually in `C:\WinNT' and is referred to in the system environment variable *%SystemRoot%* .

`I386' This directory contains 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 should install themselves in subdirectories of this directory. More often than not they choose their own locations however, often with a distressing lack of discipline.

`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.

The registry

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, boot info and options are kept. These files contain the analogue of Unix dot-files and `/etc' setup files, as well as the kind of information which `dmesg` provides. They replace the `.INI' files of earlier windows systems. The hive

There are advantages and disadvantages to the registry approach to host configuration. A nice feature is that the database lies in one well-defined place. A disadvantage is that it lies in the system workspace--meaning that it is not a distributed database. If users want special preferences their registry data have to be installed manually on each host. This can be contrasted with Unix, say, where configuration files exist per-host and per-user. Thus what one gains in

tidiness, is lost in convenience.

The registry has a hierarchical structure made from `keys'. Keys are analogous to directories: they may hold values or sub-keys. The values of keys are protected by ACLs. The registry files are called *hives* and lie in the directory `C:\WINNT\system32\config'. The default ACLs on the registry allow everyone to read them.

These files can only be viewed by starting the registry editor program `regedt32'. There are six base keys which cannot be changed

`HKEY_CLASSES_ROOT'
: Contains file associations and OLE (Object Linking and Embedding) data (Link to HKEY_LOCAL_MACHINE\SOFTWARE\Classes)

`HKEY_CURRENT_USER'
: Contains data about the user who is currently logged in on the console. It is linked to HKEY_USERS.

`HKEY_LOCAL_MACHINE'
: Contains configuration data for the local machine. This contains information about installed software, analogous to GNU/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. It does not contain info about the SAM (security access manager).

`HKEY_CURRENT_CONFIG'
: Data about the local machine configuration which might change in the future.

`HKEY_DYN_DATA'
: Sub tree containing dynamically created performance 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. Perl for WIN32 can also manipulate the registry data, and as of service pack 4 and NT 5.0, the registry can be managed remotely.

It is hazardous to change the registry without considerable care since ordinary users can lose control of the machine if the ACLs are changed.

User login directories

NT has no specified place for user directories, so you may place them where-ever 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.

Starting and stopping under NT

Like Unix, processes under NT can live in the foreground or in the background. A background process can be started with

```
start /B
```

In order to kill the process it is necessary to purchase the Resource kit which contains a `kill` command. A background process detaches itself from a login session and can continue to run even when the user is logged out.

The shutdown of the whole system must be performed from the Windows menu. Any logged on user can shut down the a host. This is not a major problem since only one user can use the system at a time. However, background processes die when this happens, including other users' background processes. A `shutdown` command also exists for shutting down local or remote systems.

Mounting disks

Network disks and local disks must be mounted in NT just as in UNIX, but this occurs more transparently. Network 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. Users home directories are attached to H: if possible. Although only a few years old NT has nearly as many historical quirks as Unix.

It is rumoured that NT 5.0 will have support form transparent mounting at an arbitrary point within a directory structure as is the case with Unix file systems.

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 text-file `/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 supplement 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

As NT improves it has been steadily converging on Unix. It is not only interesting, but useful to compare the Unix and NT software interfaces.

Strangely 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.

Here is a list of some Unix commands and their NT equivalents.

`alias` `doskey` Create a shell alias.
`at` The `at` and `soon` commands in the resource pack for scheduling background batch tasks
`cat` `type` to print a file or `copy a+b+c` to concatenate
`cd` `cd` Change directory
`chgrp` None: files don't have group attributes.
`chmod` `cacls` Change access rights on a file.
`un/compress` `compact`/`expand` Compress and uncompress files. Also `gzip`
`cp` `copy` Copy a file
`cp -r` `xcopy` Copy directories recursively
`cp -p` `scopy` Copy preserving attributes
`cron` `at` Periodic task scheduler
`df` Show disk usage. Must be simulated with script or free software additions.

`exportfs`
: `net share` Make disks available to other hosts via network.

`fsck` `chkdsk` Check disk filesystem integrity.

`ifconfig`
: `ipconfig /all` Ethernet interface configuration.

`kill` A kill command is in the Resource Kit.

`ln -s` None. Short cuts are similar to symbolic links.

`mkdir` `mkdir` Make a new directory.

`mount` `net use` Add a remote disk to the local host.

`netstat`
: `netstat` Show network connections and statistics.

`nslookup`
: `nslookup` Query the DNS/BIND service.

`passwd` `net user` Change password.

`ps` `pstat`, `pview` in the Resource Kit show active processes.

`rsh` A version of `rsh` is provided for login to Unix systems.

`tar` GNU `tar`, and `pax` available for packing/unpacking tape archives.

`traceroute`
: `tracert` Trace route through TCP/IP network.

`uptime` `net statistics` Show system statistics.

UNIX tools on NT

@hrule @vskip 0.3cm

GNU tools for NT

The Unix GNU tools from Cygnus software are obtainable free of charge. To obtain them, 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 this package follows Unix formalities and uses `` `/usr/local' ``.)
- Start the internet explorer or netscape if you have it. 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
```

Recommended reading

1. *Computer security* , D. Gollmann, Wiley.
2. *DNS and BIND* , Paul Albitz and Cricket Liu, O'Reilly & Assoc.
3. *TCP/IP Network administration* , Craig Hunt, O'Reilly & Assoc.
4. *Practical UNIX security* , Simson Garfinkel and Gene Spafford, O'Reilly & Assoc.
5. *Building Internet Firewalls* , D.B. Chapman and E.D.Zwicky, O'Reilly & Assoc.
6. *Windows NT: User administration* , A.J. Meggitt and T.D. Ritchey, O'Reilly & Assoc.
7. *Computer networks, a systems approach* , L.L. Peterson and B.S. Davie, Morgan Kaufman.
8. *Security reference* , http://www.rootshell.com

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 running their programs 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.

Consolidated
: A centralized mainframe type of solution for concentrating computing power in one place. This kind of solution makes sense for heavy calculations, performed in engineering of computer graphics.

Cyberspace
: William Gibson's name for the virtual world of the net. From his novel Neuromancer.

Distributed
: A de-centralized solution, in which many workstations spread the computing power evenly throughout the network.

DNS The Domain Name Service, which converts internet names into IP addresses and vice versa.

Enterprise
: A small business network environment. Enterprise management is a popular concept today because NT has been aimed at this market. If nothing else, Microsoft know about marketing and have highlighted this important market far more effectively than the Unix vendors have done before them. Enterprise management typically involves running a web server, a database, a disk server and a group of workstations and common resources like printers and so on. Many magazines think of enterprise management as the network model, but when people talk about Enterprise management they are really thinking of small businesses with fairly uniform systems.

Free software
: This is a concept promoted by the Free Software foundation. Free means freedom to share with your friend, rather than gratis. Free software might cost money (usually it does not), but it comes with a license which gives everyone the right to copy, distribute and alter the program for whatever purpose they wish. As Richard Stallman (who founded the FSF) puts it, think `free speech' not `free beer'. Several companies make money out of supporting and marketing Free Software, e.g. Cygnus and Red Hat.

GUI Graphical user interface.

Internetworking protocol
: A protocol which can send messages across quite different physical networks, binding them together into a unified communications base.

IP address
: Internet address. Something like `128.39.89.10`

LISA Large Installation System Administration. This refers to environments with many (hundreds or thousands of) computers. The environments typically consist of many different kinds of system from multiple vendors. These systems are usually owned by huge companies, organizations like NASA or universities.

Open Source
: This is like free software. It is a kind of registration stamp for software for which the source code is available to the user. Mostly users are not allowed to modify the code and resell it however. See http://www.OpenSource.com

Open systems
: is a concept promoted originally by Sun Microsystems for Unix, which is about all systems being compatible through the use of freely available standards. While competitors might have to pay to implement a feature, they are not prevented from knowing how to include that feature or from selling it later.

PC An Intel based personal computer, used by a single user.

PID Process identity number.

Proprietary systems
: is the opposite of open systems. These systems are secret and the details of their operation is not disclosed to competitors. The most infamous example is Microsoft, but there are many others.

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.

SID Security identity number (NT)

SIMM Memory chip arrays.

Spoofing
: Impersonation, faking, posing as a false identity.

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

UID User identity number (Unix)

URL Uniform resource locator. A network `filename' including the name of the host on which the resource resides and the network service (port number) which provides it.

Vendor
: A company which sells hardware or software. A *seller* .

Workstation
: A non-Intel based personal computer which might be used by several users. Workstations might be based on for example SPARC (Sun Microsystems) or Alpha (Digital/Compaq) chip sets.

X11
: The Unix windows system.

Index

!

- !

.

- `.cshrc', `.cshrc'
- `.fvwm2rc'
- `.fvwm95rc'
- `.fvwmrc'
- `.mwmrc', `.mwmrc'
- `.profile', `.profile'
- `.rhosts'
- `.tkgrc'
- `.xinitrc'
- `.xsession'

/

- `/etc/aliases'
- `/etc/checklist'
- `/etc/dfs/dfstab'
- `/etc/ethers', `/etc/ethers'
- `/etc/exports'
- `/etc/filesystems'
- `/etc/fstab', `/etc/fstab'
- `/etc/hosts.allow'
- `/etc/hosts.deny'
- `/etc/inetd.conf', `/etc/inetd.conf'
- `/etc/inittab'
- `/etc/named.boot'
- `/etc/named.conf'
- `/etc/nsswitch.conf'
- `/etc/printcap'
- `/etc/rc' files
- `/etc/rc.local'
- `/etc/resolv.conf', `/etc/resolv.conf'
- `/etc/services'
- `/etc/vfstab'
- `/tftpboot'
- `/usr/etc/resolv.conf' on IRIX
- `/usr/local'
- `/usr/local/gnu'
- `/usr/local/site'
- `/var/mail'
- `/var/spool/mail'

a

- A record
- Access control for services
- Access control lists
- ACEs in NT
- ACLs
- ACLS in NT
- ACLs, network services
- actionsequence
- Aliases in mail
- Alive, checking a host
- Anonymous ftp databases
- arch program
- ARP
- ARP/RARP

b

- Backdoors
- Background process, NT
- Backups
- Bad primary partition error

- Big endian
- BIND
- BIND version 8
- BIND, setting up
- `biod`
- Boot scripts
- Booting unix
- `bootp` protocol
- BOOTP protocol
- Bridge
- Bridges
- Broadcast address, Broadcast address
- BSD 4.3
- Byte order

c

- Cache file, DNS
- Cache poisoning
- `CACLS` command
- *cancel*
- Canonical name
- Canonical names
- `catman` command
- cfengine
- `` `cfengine.conf' ``
- `chat` program
- Checking the mode of installed software
- Checking whether host is alive
- Class A,B,C networks
- Classes
- Classes, compound
- Classes, defining and undefining
- Clock synchronization
- CNAME, CNAME
- Community string
- Community strings
- Components, handling
- Compound classes
- `configure`
- Contact with the outside world
- `cp` command
- crack
- cron
- `cron`, `cron`
- Cron jobs, controlling with cfengine
- `` `crontab' ``
- `crontab` command
- Cyberspace

d

- Daemons and services
- David Korn
- Day of the week
- Death to the users
- Default nameserver
- Default printer, Default printer
- Default route, Default route, Default route
- Denial of service attack
- Devices
- `df` command
- Disk backups
- Disk doctor
- Disk partition names
- Disk quotas
- Disk repair
- Disk statistics
- DNS, DNS, DNS
- DNS cache file
- DNS, BIND setup
- DNS, mail records
- `dnsquery`
- Domain
- Domain name
- Domain name service
- Domain name, definition
- Domain OS
- Domain, listing hosts in

- domainname
- DoS attack
- Dots in hostnames
- Down, checking a host
- du command
- dump command

e

- eeprom
- etherfind command
- exportfs command
- Exporting on GNU/Linux
- External hosts do not seem to exist

f

- File type problem in WWW
- Filesystem table
- find command
- Finding a mail server
- Finding domain information
- Finding the name server for other domains
- Firewalls
- format program, Sun
- Formatting a filesystem
- FQHN
- Fragmentation of UDP
- Free software foundation
- fsck program.
- FSF
- ftp
- FTP
- ftp databases
- `ftp.funet.fi'
- `ftp.uu.net'
- Fully qualified names
- fvwm window manager
- fvwm2 window manager
- fvwm95 window manager

g

- Gateway
- GNU software
- Grouping time values
- groups and time intervals
- groups in cfengine

h

- Handling components
- Hangup signal
- HINFO
- Home directory
- Host name gets truncated
- Host name lookup
- Hostname lookup
- HTTP
- HUB
- Hub

i

- IDE
- ifconfig
- ifconfig command, ifconfig command
- in.rarpd
- inetd
- inetd master-daemon
- `INSTALL'
- Interface configuration, Interface configuration
- Internet domain
- iostat command
- IP address
- IP address lookup
- IP address, setting
- IP addresses, IP addresses

- IPv6

j

- junkfilter

k

- `kill` command
- kill, NT process
- Korn Shell

l

- LDAP
- `ldconfig` command
- `ldd` command
- Linux, exports
- Little endian
- loadlin
- `locate` command
- Logical NOT
- Login directory
- Looking up name/domain information
- Lookup hosts in a domain
- Loopback address
- Loopback network in DNS
- `lp`, `lp`
- lp default printer
- `lpc`
- `lpd`
- `lpq`
- `lpr`, `lpr`
- `lprm`
- `lpsched`
- `lpshut`
- `lpstat -a`
- `lpstat -o all`

m

- `mach` program
- Macintosh
- Mail address of administrator, Mail address of administrator
- Mail aliases
- Mail exchangers
- Mail queue, Mail queue
- Mail records in DNS
- Mail spool directory
- Mail, finding the server
- `make`
- Management information base
- MIB
- `mkfile` command
- Months
- `mount -a`
- `mount` command
- `mountd`
- Mounting filesystems
- Mounting filesystems.
- Mounting problems
- Multi-port repeater, Multi-port repeater
- multi-user mode
- Multicast address
- `mwm` window manager
- MX
- MX records

n

- Name service lookups
- `named`
- Nameserver for other domains
- Nameserver list
- `ncftp`
- Netmask, Netmask
- `netstat -r` and routing table
- `netstat -r` command

- netstat command
- Network address
- Network byte order
- Network information service
- Network interface, Network interface
- Network interfaces
- Network numbers
- Network, transmission method
- Networks
- newfs command
- Newsprint
- NFS client/server statistics
- nfsd
- nfsiod
- nfsstat command
- nice
- NIS
- No contact with outside world
- NOT operator
- Novell, Novell
- NS
- nslookup, nslookup
- NT
- NT GNU tar file
- NT installation
- NT, ACL/ACEs
- NT, tcsh for
- ntemacs
- ntpd

o

- One time passwords, One time passwords
- Operator ordering
- OS/2 boot manager

p

- Partitions
- Password sniffing
- Permissions on installed software
- Ping attacks
- ping command, ping command
- Point to point protocol
- Port sniffing
- PPP
- Print spool area
- Print-queue listing, Print-queue listing
- Print-queue, remove job, Print-queue, remove job
- Print-queue, start, Print-queue, start
- Print-queue, stop, Print-queue, stop
- PRINTER
- Printer registration
- Printer, choosing a default
- Probe SCSI
- procmail
- ps command
- PTR records

q

- q=any, nslookup
- q=mx, nslookup
- q=ns, nslookup
- Quotas

r

- RARP, RARP, RARP, RARP
- `rc' files
- rdump command
- `README'
- Registering a printer
- Registry, NT, Registry, NT
- renice command
- Repairing a damaged disk
- Repeater, Repeater
- Resolver, setting up

- Restarting daemons
- `restore` command
- Reverse lookup, DNS
- `rlogin` command
- `rm -i` command
- Root partition
- `route` command
- Router
- Routing information
- Routing table, Routing table
- `rpc.mountd`
- `rpc.nfsd`
- `rsh` command
- Rsh for NT
- Rule 2
- Rule 3
- Rule 8
- `` `run-cfengine' `` file.
- Running jobs at specified times

s

- S/KEY
- Scheduling priority
- SCSI
- SCSI probe on SunOS
- Security holes
- `sendmail`
- Sequence guessing
- Server message block
- Service configuration
- Services and daemons
- Setuid programs
- `share`, `share`
- `shareall`
- SHTTP
- SIMM
- Simple Network Management Protocol
- Single user mode
- single-user mode
- Site specific data
- SMB protocol
- Smurf attack
- SNMP, SNMP
- `snoop` command
- SOA
- Socket connections
- SSH
- Start up files for unix
- `startx`
- Statistics, disks
- Statistics, NFS
- Statistics, virtual memory
- Subnets
- SVR4
- Swap partition.
- Swap space
- `swapon` command
- Swapping, switching on
- Switched networks
- SYN flooding
- System accounting
- System type, System type

t

- `tar` command, `tar` command
- TCP wrappers
- TCP/IP spoofing
- tcpd
- tcsh for NT
- Teardrop
- `telnet` command
- Telnet for NT
- Time classes
- Time service
- Time, executing jobs at specified
- timezone
- TkGoodStuff

- `traceroute` command
- Transceiver
- TTL
- Twisted pair

u

- `ufsdump` command
- uid
- Ultrix
- umask
- `uname`
- Undeleting files
- Unix dot-files
- Up, checking a host
- `updatedb` script
- user-id

v

- Virtual memory statistics
- `vmstat` command

w

- `whatis` command
- `which` command
- `whois` command

x

- X server
- X-terminal
- X-terminal troubleshooting
- `xdm`
- xhost access control
- `xhost` command

y

- Years

This document was generated on 25 Febuary 1999 using the texi2html translator version 1.50.