521481P
INTRODUCTION TO THE USE OF WORKSTATION

T. Hakanen, M. Heikkilä, J. Tiensyrjä
jtk-staff@ee.oulu.fi

Fall 2010

Foreword

There are a number of servers running ***Linux*** and ***Solaris 10*** operating systems at the department of electrical and information engineering. The servers are also Sun Ray Servers, which means they can be used in the workstation classrooms graphically. The servers ***st-cn0001*** (Linux) and ***stekt1*** through ***stekt4*** (Solaris) are reserved for student use. In this course a brief introduction to Linux and Solaris command line interface is given.

Before joining the course assistants in the workstation classroom, please make sure you have read this course material and understood it. You should also answer the questions either in Appendix A (in Finnish) or Appendix B (in English). Your answers will be checked in the beginning of the class.

Contents

1 Introduction to Unices

The material in this chapter is mainly borrowed from [5] and [4].

Sun Ray Servers at the department of electrical and information engineering are equipped with ***Linux*** or ***Solaris 10*** operating systems. Solaris is based upon Unix, an operating system that was originally developed in 1969 and became widely available in 1975. Even though Unix is more than 40 years old, it still enjoys considerable usage and it is continually evolving. A Unix-like operating system Linux, created by Linus Torvalds in 1991, does not directly descent from Unix; rather, it is a clone. Nowadays Linux has surpassed Solaris and other Unices in popularity.

1.1 Key System Concepts

You are probably already familiar with most of the concepts, but you might not be sure how they fit into the Unix world.

Operating System. Operating systems provide an interface between the computer hardware and software. In a sense, they are the translator that makes the hardware and software play nice together. Second, based on the first function, they enable users to run applications. So, operating systems are applications that let you run other applications.

Kernel. The kernel is the brain of the operating system. Although kernels vary among operating systems, they all have some common characteristics. Kernels are responsible mainly for managing computer input/output (I/O), allocating system resources, and managing processes.

Processes. Processes are the running parts of an application. A common misconception is that an application is a process. That's not true, because many applications will be running as multiple processes at one time. Such applications are known as multithreaded applications. Multithreading speeds up the application and allows for smoother execution. System tasks other than applications such as daemons, run as processes as well. All processes have a process identifier (PID), which is used by the kernel to identify and manipulate the process as needed.

Daemons. Depending on where you look, you can find two common definitions for daemons. The first one describes a daemon as a program that runs automatically in the background without the need for user intervention. The second definition is that a daemon provides a service. The service can be administrative, such as cleaning up temporary files, or the service can be one that provides meaningful interaction to clients, such as a print daemon. Daemons run as processes, and can either start automatically when the operating system starts, or be started manually.

Shells. The shell enables users to input information to be interpreted by the operating system. Consider the operating system to be the interface between the computer hardware and software, and the shell to be the interface between the user and the operating system. Shells also enable users to program commonly used or frequently used lists of applications to run with the execution of one command. These are called scripts, macros, or batch

files. Although there are different graphical user interfaces (GUIs) for Unix-like operating systems, the shell itself is command prompt based.

Graphical User Interfaces (GUIs). A graphical user interface (GUI) is a particular case of user interface for interacting with a computer which employs graphical images and controls in addition to text to represent the information and actions available to the user. Usually the actions are performed through direct manipulation of the graphical elements. Graphical user interfaces are sometimes called visual shells or graphical shells.

File Systems. File system is a data structure that translates the physical (sector) view of a disc into a logical (files, directories) structure, which helps both computers and users locate files. In other words, it records where files and directories are located on the disc.

Clients and Servers. On networks, computers can be divided into two broad categories: clients and servers. Some operating systems are designed to be a server only. Others are to be clients only. Linux and Solaris are versatile operating systems that can be used as either a client or a server. As a rule of thumb, end users sit at client machines and perform daily tasks. Clients will often request information (files and applications) from centralized servers, which are located in some sort of server rooms. Servers should be secured away from prying hands, because they often hold critical and sensitive information. Clients make requests of servers, and servers fulfill client requests.

1.2 Where to Get Help

There are numbers of resources that you can use when you get stuck or simply need a quick refresher. This section highlights the most common ones.

man Pages. Unix operating systems have extensive documentation known as man pages. The command used to display them is `man`. It displays reference manual pages about commands and concepts. For example, you could type `man passwd`, which would display information on the command `passwd`. You can even type `man man` to get information on the `man` command itself. Type `man intro` to see introduction to commands and application programs available with the operating system.

info Pages. In addition to `man` pages, some software prefer to keep their manuals in Info format. Type `info info` to find out more.

Online Resources. The documentation for your Linux distribution comes usually with the distribution itself. To find out more, see [4] or [3].

Literature. There is vast amount of literature available on different Unices and Linux.

1.3 Users and Groups

Users. Users are one of the most critical components of any computer system. When the system is booted, you must log in to use resources. To log in, you provide a valid username and password. Every time you attempt to access a system resource, your user

account is checked for appropriate permissions. All resources must have an owner, and without users, there are no owners. Unices are multi-user environments, meaning that multiple user can be logged in to the same system at one time. The username and password are part of your user account. Every person who uses the computer should have his or her own unique account. User accounts have the following components:

- A unique ***username*** that identifies the user on the system.
- A ***User ID (UID)*** number that identifies the user.
- A ***password*** that allows the user to access the system and proves that the user is who she claims to be.
- A ***home directory*** that can be used to contain the user's files and serve as the default directory at login.
- ***Initialization files*** that customize the user's environment upon login.

To find out your UID, you can use the command `id -a`.

Groups. Groups are designed to make system administration easier. Groups are not used to log in to a system, as that's what users are for, but ***every user account belongs to at least one group***. Assigning resource access to groups makes security administration easier than assigning permission directly to users. Each group has a name and a ***Group ID (GID)***. The GID for a group is analogous to a UID for a user. Users must belong to one group, their primary group, and can be a part of several other secondary groups. To see which groups you are a member of, issue the `groups` command at a command prompt. In our department, all students belong to group named ***students***.

1.3.1 Password

To log in to a server, users must provide a valid username and password. Since the usernames are often public knowledge, the password is the only obstacle between the potential hacker and critical network resources.

Passwords can contain any keyboard characters and should contain a combination of uppercase and lowercase letters, numbers, and special characters, such as $, %, @, or whatever the user prefers. Password must be at least eight characters, and the first six characters must contain at least two alphabetic characters and at least one numeric or special character. ***The recommended length of the password is 8-15 characters — the longer the better.***

To increase network security, users should change their passwords periodically, especially if you have accessed the system from potentially unsecure computer, e.g. from an internet café. Information that can be easily guessed should not be used for any passwords. Nonsense words make the best passwords, and users should avoid any words listed in common dictionaries. To change your password, issue the `passwd` command at a command prompt.

1.4 Files, Directories, and Security

Beyond the login (username and password), additional security measures are needed to ensure that users have access only to the information that they are supposed to have access to. This is where file and directory security comes into play.

1.4.1 File and Directory Concepts

In Unices, data is stored within a file system located on a hard disk. A file system is divided into two basic components: files and directories. Files contain data, and a directory is a specialized file that lists filenames.

Linux and Solaris regard almost everything on your system as a file. Documents and programs are files. Commands are essentially programs and are therefore files. Directories are files that contain other files. Even devices on your computer, such as your hard drive, terminal, and printer, are considered files.

To show a listing of files in a directory, use the `ls` command. The most commonly used options for `ls` are `-l` (long listing) and `-la` (long listing and including files that start with a period, which are usually hidden). For a complete listing of `ls` switches, type `man ls` at a command prompt. Here is a sample directory listing:

```
(stekt40)(markot)(132)(~/jtkdir) ls -l
total 8
-rwxrwxrwx 1 markot tklab   2 Jul 7 11:05 a
drwx------ 2 markot tklab 512 Jul 7 10:59 dir1
drwx--x--x 2 markot tklab 512 Jul 7 11:00 dir2
-rw------- 1 markot tklab  13 Jul 7 11:02 textfile.txt
(stekt40)(markot)(133)(~/jtkdir)
```

The long listing of a directory shows seven fields. On the left, you see the file's permissions. The number after the permissions indicates the number of links to the file. After the links, you see the owner of the file (`markot` in this case) and the primary group of the owner (`tklab` in this case). The file size in bytes, last modification date and time, and filename end each entry.

Using the File System. File systems are organized in a hierarchical tree. The base of this tree, or the root directory, is called ***root*** and is represented with a slash (`/`). All other files and directories can be thought of in terms of their relationship to the root. Figure 1 illustrates this hierarchical structure.

One of the basic tenets of file system is that each object must have a unique ***absolute pathname***. For example, consider the `/etc` directory. This directory named `etc` exists right off of the root (`/`). Therefore, you cannot create another directory named `etc` from within the root directory. However, if you were to switch to the `/export` directory, you could create a directory called `etc`. The absolute pathname of this new directory would be `/export/etc`. Also remember that Unix file systems are case sensitive; the directory `/export/dir1` is different from `/export/DIR1`.

A ***relative pathname*** indicates the location of a file based on your current working directory. Any pathname that does not begin at the root directory (`/`), or with a tilde (~), is a relative pathname. ***Absolute pathnames never change, but relative pathnames do, based on where you are in the directory tree structure.***

File System Navigation. To determine, where you are in the file system, use the `pwd` (print working directory) command. The ***working directory*** is also known as the current directory. ***Don't confuse the working directory with the home directory.*** Your home directory is used to store personal files, whereas the working directory is the

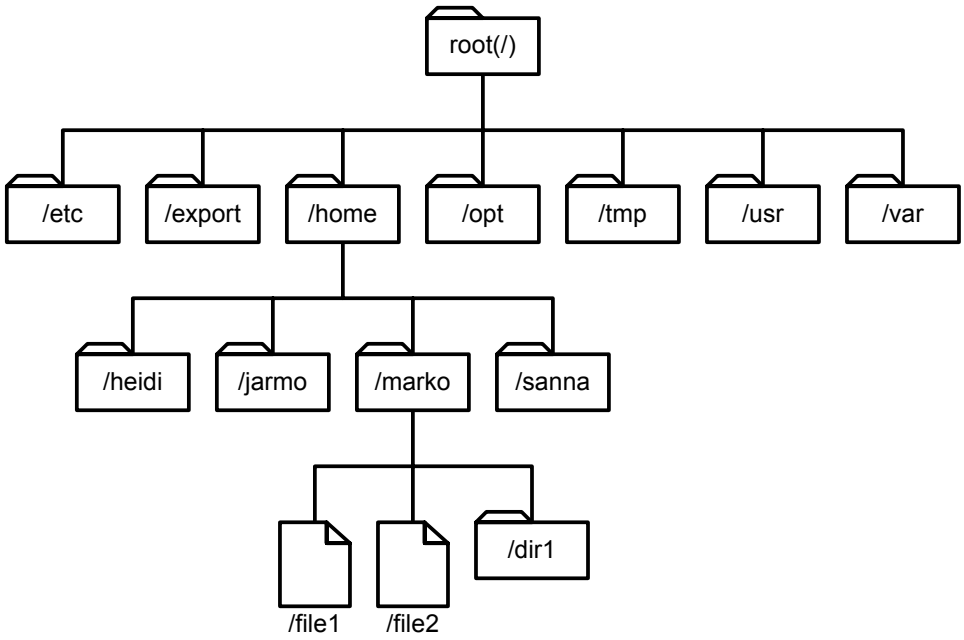

Figure 1: Hierarchical file structure.

directory you are currently working in.

The . and .. Directories. Each directory automatically has two entries: a single period (`.`) and a double period (`..`). The single period represents the directory itself, and the double period represents the directory's parent directory. When you use the `ls` command, you typically don't see these entries, because they begin with a period, and by default `ls` doesn't show entries that begin with a period.

Useful File Commands. There are literally dozens of command you can use to create, copy, move, rename, display, and modify files. The following table lists some common file-management-based commands. Use `man command` command to get more information about the commands.

Command	Description
`cat`	Concatenates and displays files. For example, `cat myfile1 myfile2 > myfile3` combines files `myfile1` and `myfile2` in a new file named `myfile3`. The command `cat myfile` simply displays the contents of a file named `myfile`.
`cd`	Changes directories. For example, `cd /public/students` changes the working directory to `/public/students`. The command `cd ..` moves up one directory in the tree, `cd` $\sim$ changes to the home directory, and `cd /` changes to the root directory.
`cp`	Copies files. The command `cp myfile1 myfile2` creates a copy of `myfile1` named `myfile2`. To copy a file named `myfile` from the current working directory (`.`) to the directory `/public/students`, type `cp myfile /public/students`. For copying directories, use `cp -r`, which copies the directory as well as any files or subdirectories to the desired destination.

Command	Description
`emacs`	Emacs is a versatile text editor.
`file`	Determines file type.
`find`	Finds files. For example, to find a file named `myfile` from the working directory, use `find . -name myfile`.
`grep`	The `grep` utility searches text files for a pattern and prints all lines that contain that pattern. For example, to display all lines that contain a string `keyword` from a file named `myfile`, use `grep keyword myfile`.
`less`	Filter that enables you to display one screen of information at a time. For example, to see file `myfile` one page at a time, use `less myfile`.
`ls`	Lists files in a directory. If no directory is specified, the working directory is used. For example, to list files in a directory `/public/students`, use `ls /public/students`.
`mkdir`	Creates directories. The command `mkdir mydir` creates a directory called `mydir` in the current working directory.
`more`	Filter that enables you to display one screen of information at a time. For example, to see file `myfile` one page at a time, use `more myfile`.
`mv`	Moves or renames files. The command `mv myfile1 myfile2` renames `myfile1` to `myfile2`. The command `mv myfile /public/students` moves `myfile` to the `/public/students` directory.
`nano`	Nano is a simple, display-oriented text editor.
`pwd`	Displays the present working directory.
`rm`	Deletes a file or directory tree. To delete a file named `myfile`, use `rm myfile`. The `rm -r` command can be used to remove a directory and all files within the directory.
`rmdir`	Removes empty directories. If the directory has files in it (other than `.` and `..`), delete the files first and then use `rmdir` to remove the directory. The command `rmdir mydir` removes an empty directory called `mydir` from the current working directory.
`wc`	Displays a count of lines, words and characters in a file.

One other handy navigation shortcut is the tilde (∼). It is used as a shorthand representation of the user's home directory. To change to his home directory quickly, a user can use `cd ∼`. To create a directory called `mydir` in her home directory, a user could execute the `mkdir ∼/mydir` command, regardless of her working directory.

1.4.2 Input/Output Redirection and Piping

Input/Output redirection. Input/Output redirection is a powerful tool. It allows commands to read input from a file rather than the keyboard, or save output normally sent to the screen in a file. For example, `ls` prints a listing of files in a working directory to the screen, while `ls > myfile` saves the listing in the file named myfile. Input/Output redirection is based on the Unix standard I/O concept. Each command uses three I/O streams: standard input, standard output, and standard error. The standard I/O streams determine where input is read from and output is written to. By default, standard input, standard output, and standard error are set to keyboard, screen, and screen, respectively.

When programs require input from the user, it is usually entered from the keyboard. However, input can be read from a file by following the command with a less than sign

(<) followed by a filename: `command < filename`.

The output from a command can be saved in a file by following any command with a greater than sign (>) followed by a filename: `command > filename`. If a file named `filename` already exists, its contents will be replaced with the new data. To append the output from a command to a file (at the end of the file) use two greater than signs (>>) instead: `command >> filename`.

Piping. Piping directs the standard output of one command into the standard input of another command. Piping is done by separating two or more commands with a pipe character (|): `command1 | command2 | ...`. For example, `ls | more` sends output from the `ls` command (a directory listing of the current working directory) to the `more` command. This is useful if a directory listing scrolls of the screen.

1.4.3 Wildcard Characters

In some commands, such as the query commands, you can use wildcard characters to create a pattern-matching expression that specifies more than one object. Using wildcard characters makes it easier to tailor a command to your needs. The wildcard characters you use depend on the operating system from which you issue commands. For Linux and Solaris, you can use wildcard characters such as an asterisk (*) to match any (0 or more) characters, or you can use a question mark (?) to match exactly one character. For example, if you want to query all the files in your home directory (∼) whose names begin with `myfile`, type `find ∼ -name "myfile*"`. If you want to query the files `myfile1.txt` and `myfile2.txt`, you can enter `find ∼ -name "myfile?.txt"`.

1.4.4 File and Directory Permissions

The first step to control system access is physical. If someone can get to your machine, they can either read sensitive data or use brute force to destroy data. The second step is making sure that users have good passwords and change them frequently. The last step is setting up proper permissions, so the users who do log in to your network have the appropriate level of file access. This section covers the final step.

Like most other Unix-based operating systems, Linux and Solaris categorize users into three classes when it comes to resource access: the owner of the file (***owner*** or user), a member of the group (***group***), and everyone else (***other***). The basic set of Unix permissions allows for three distinct operations on a given file: ***read***, ***write***, and ***execute***. Each class and permission combination is considered separately. That is, the owner can have read, write, and execute permissions on a file, whereas everyone else (other) can be denied access to the same file. ***The owner of the file and the superuser are the only two persons who can modify permissions on a file.***

Before we look at displaying and changing permissions, let's clarify what is meant by read, write, and execute. The following table breaks down the permissions and how they affect what designated users can do to files and directories.

Permission	File Consequences	Directory Consequences
Read (`r`)	Can open and read the contents of the file.	Can list files in the directory.

Permission	File Consequences	Directory Consequences
Write (`w`)	Can modify or delete the file.	Can add or remove files in the directory.
Execute (`x`)	Can execute the file (program).	Can open and execute files in the directory.
None (`-`)	Cannot read, write, or execute the file.	Cannot list, add, remove, or execute files in the directory.

Permissions are displayed when you use `ls -l` to show a long directory listing. Here's an example to refresh your memory:

```
(stekt40)(markot)(183)(~/jtkdir) ls -l
total 4
drwxrwxr-x 2 markot tklab 512 Jul 7 15:27 dir
-rw-r--r-- 1 markot tklab  13 Jul 7 15:26 textfile.txt
```

Because we're discussing permissions, the only parts of the directory listing you need to be concerned with right now are the first 10 characters and the name of the file. The first character indicates that the file is either a special type of file or a regular file. The first file (`dir`) listed in our example is a directory, as indicated by the lower case `d`. The second file, named `textfile.txt`, is a regular file, as indicated by the dash (`-`) preceding the permissions. The following nine characters define permissions for that file. The first grouping of three letters (`rwx`) defines permissions for the owner, the second cluster is for group permissions, and the last tier is for other. In our example, the `dir` directory has read, write, and execute permissions set for the owner and group, and read and execute set for other. The `textfile.txt` file has read and write for the owner, and read-only for group and other.

The `chmod` command is used to set permissions on files. This command functions in two modes; ***symbolic mode*** and ***absolute mode***. When using `chmod` in symbolic mode, letters designate both whom the permissions will impact and the permissions to effect. The syntax for `chmod` in symbolic mode is as follows:

```
chmod who operator permission file
```

For example, the command `chmod u+w myfile` will add (`+`) write (`w`) permission to the owner (`u`) for `myfile`. To specify multiple groups, separate the `who`, `operator`, and `permission` grouping with commas, such as `chmod u+w,g+r myfile`.

There are four options for the `who` variable. To set owner (user) permissions, use `u`. Group permissions are set with `g`, other permissions with `o`, and all (owner, group, and other) permissions are set with `a`. The `operator` variable tells `chmod` what to do with the permissions specified. There are three options for the operator: a `+` tells `chmod` to add the permission to existing permissions, a `-` removes permissions, and an `=` sets the permissions to exactly what is specified. If the `=` is used and no permissions are specified, then all permissions for the specified `who` are cleared. There are 10 options for the `permission` variable. Only the three most commonly used options are discussed in this material: read permission (`r`), write permission (`w`), and execute permission (`x`).

Instead using letter designations to assign permissions, some users prefer to use `chmod` in absolute mode. Absolute mode enables you to assign permissions by using an octal

representation. You can perform the exact same permissions modifications with either method. Type `man chmod` to get more information about the `chmod` command.

1.5 Controlling Processes

When people think of processes, they generally think of applications running on a system. Although it's true that applications are processes, many other things are processes as well. Any running program, whether it's an application, a script, a daemon, or any other "running" component, even the user shell, is considered a process. Simply defined, a process is a single program running in its own memory space. Various applications run multiple processes concurrently to achieve full functionality.

In Unices, the system kernel manages processes. To manage processes you need to understand the commands used. The following table lists some commonly used process management commands. Use `man command` command to get more information about the commands.

Command	Description
`bg`	Run suspended job in background. To start job in background, append `&`. For example, to run program called `matlab` in background, use `matlab &`.
`fg`	Restart suspended job in foreground.
`jobs`	Lists current jobs.
`kill`	Sends a signal to process, or terminate a process (by PID).
`nice`	Changes the priority of a running process.
`pgrep`	Finds active processes based on name or other attribute.
`pkill`	Sends a signal to process, or terminate a process (by name).
`ps`	Reports the status of active processes, including process ID, terminal name, execution time, and command name.
`top`	Reports statistics of active processes. Sorts processes by CPU usage by default.
`[Ctrl+C]`	Interrupt a running process.
`[Ctrl+Z]`	Suspend a running process.

The most popular command used to list processes is `ps`. If you need information on running processes, `ps` can provide the status of the process, process ID (PID), parent process ID, user ID, scheduling class, priority, memory used, and processor time used. By default, `ps` provide the process ID, terminal from which it was executed, processor time taken, and the command that generated the process. With no arguments, the `ps` command displays only those processes that have the same effective user ID and the same controlling terminal as the person who ran the command. To see a full listing of all processes, you could use `ps -ef`. To see a listing of processes that belong to a user with username `username`, you could use `ps -fu username`.

The `pgrep` utility is also used a lot. It's a great search tool for processes; all you need to know is what you want to look for. The syntax for `pgrep` is `pgrep options pattern`. Therefore, if you want to find the process ID of `syslog`, use `pgrep syslog` and you will see the process ID returned. The `pgrep` utility has quite a number of switches because it's designed to give you a wide variety of options to search for processes. For more detailed information, type `man pgrep`.

Unices use signals to communicate with processes. Signals enable processes to inter-

act with each other. Signals are sent with the `kill` and `pkill` commands. Although the names of these commands imply that they're used exclusively to terminate processes, that's not necessarily the case. There are nearly 40 signals used. Some of the more important ones to know are listed in the following table.

No.	Signal	Description
2	`SIGINT`	Interrupt. Signal 2 can also be sent by pressing `[Ctrl+C]` or the `Delete` key.
9	`SIGKILL`	Forces the process to terminate unconditionally. This is the "sure kill" signal.
15	`SIGTERM`	Termination signal. Shuts down the process but gives the process a chance to terminate properly by cleaning up.

You can kill any process that you own. Used with no arguments, the `kill` command sends a signal `15` to the PID specified. Note that when you use `kill` command, you must specify the process' PID, not its name. If the process is still around after issuing a `kill` command, you can send it a `kill -9`, which should clean up the process without any questions. If you wanted to kill a process with PID `1814`, you would use `kill -9 1814`.

The other command used to send signals to processes is `pkill`. When using `pkill`, you can specify the PID or the name of the process you wish to signal. If no signal is specified, signal 15 (`SIGTERM`) is sent by default.

1.6 Managing Printing

Printing in Linux and Solaris is based around the ***Line Printer (LP) print service***. The LP print service provides a standard set of Unix-based print commands and enables you to manage printers both locally and through a name service. A local printer is physically attached to your machine, whereas a network printer is a printer that you can access but that is not located on your machine. The following table lists some common printing commands used by Solaris. Use `man command` command to get more information about the commands.

Command	Description
`cancel`	Cancels a print request.
`lp`	Sends print jobs to a printer.
`lpstat`	Shows the status of the LP print service.

Checking printer status. If you want to determine which printers are available or to determine characteristics of installed printers, you can do so with the `lpstat` command. The syntax of `lpstat` is as follows:

```
lpstat -d [-p printername -D -l] -t
```

For example, to show the status of a printer named `sthp1`, along with the printer's description, you would use `lpstat -p sthp1 -D`.

Printing. The `lp` command is used to send print jobs to a printer. The syntax of `lp` is as follows:

```
lp -d printername filename
```

For example, if you want to print a file named `mytext.txt` with the printer named `sthp1`, you would use `lp -d sthp1 mytext.txt`.

Canceling Print Requests. Print jobs can be canceled by the user who submitted the print request or by the root or lp users. The `cancel` command is used to cancel print jobs. The current print job can be canceled, and jobs can be canceled by ***request identification number (request ID)*** or by user who submitted the job. Note that the user can cancel a print job only from the machine from which he/she submitted the job. A print job's request ID is displayed when the job is submitted. However, if you've forgotten the request ID, you can see it by using the `lpstat -o printername` command. Request IDs consist of the printer's name, a dash (`-`), and the number of the print request. An example is `sthp1-19`. Here are some examples of canceling print requests. To cancel a print request for the user `markot`, use `cancel -u markot`. Canceling the current print job on the printer requires only the printer's name, such as `cancel sthp1` whereas canceling specific print jobs requires the request ID, as in `cancel sthp1-19`.

2 Networking

2.1 University's network

The university has an internal network of computers which function together to exchange information. The network of the department of electrical and information engineering is a part of the university's network. The computers you use are referred to as workstations and the machines they communicate with are servers. Each computer does not have its own printer; they are shared between all users of the network. You can print out your work on the various printers in the network.

The university network is connected to the ***Finnish University and Research Network (Funet)*** [1]. This fast data communication network serves the Finnish scientific community, interconnecting over 80 research organizations in Finland. Funet also provides its user organizations with data links to the global Internet network through the ***NORDUnet*** network. Visit `http://www.csc.fi/suomi/funet/` for more information.

2.2 E-mail

E-mail addresses in our department are of format `username@ee.oulu.fi`. You can read and write e-mails by using a program called ***alpine***. To open the program, issue the `alpine` command at a command prompt.

If you prefer to read your e-mail with a graphical user interface, ***Thunderbird*** is one good choice for that. The following settings should be used in order to read e-mail from the department servers:

Settings key	Value
`Server type`	IMAP
`Server name`	ee.oulu.fi
`Server port`	993
`Connection security`	SSL/TLS
`Secure authentication`	No

To send e-mail through the department servers, set up the outgoing server as follows:

Settings key	Value
`Server type`	SMTP
`Server name`	ee.oulu.fi
`Server port`	587
`Connection security`	STARTTLS
`Secure authentication`	No
`Use name and password`	Yes

2.3 The Internet and the World Wide Web

The material in this chapter is mainly borrowed from [2].

The *World Wide Web*, also called WWW or simply the web, is a subset of the Internet. It brings text, images, animation, video, sound and other multimedia content, all under one roof. The Web is changing continuously by absorbing new technologies and updating the old ones. The pages on the Web, appropriately called web pages, can contain multimedia objects such as images, sound, and video in addition to text. This greatly enhances the experience of the web surfer. Web pages are viewed using browsers. On the department's machines, you can find several web browsers that you can use to access the WWW. For example, to use the ***Firefox*** browser, issue `firefox` command at a command prompt.

2.4 Remote connection

SSH (Secure Shell) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to provide secure encrypted communications between two hosts over an insecure network.

You can take a remote connection to another machine by issuing a command `ssh username@machine`. For example, if the user with username `markot` wish to take a remote connection to a machine `stekt1`, he would use `ssh markot@stekt1.oulu.fi` command.

To display information about local and remote users, you can use the `finger` command. For example, if you type `finger markot` command at a command prompt, you will get information about the user with username `markot`. You'll get more information about the command by issuing `man finger` at a command prompt.

3 Rules for the Use of Computer Systems and Networks

`http://www.oulu.fi/tietohallinto/tietoturva/sisalto/politiikat/rules.html`
`http://www.oulu.fi/tietohallinto/tietoturva/sisalto/politiikat/rulesapp.html`

References

[1] *CSC, the Finnish IT center for science.*
http://www.csc.fi/suomi/info/.

[2] *Internet Fundamentals.*
http://www.webdevelopersnotes.com/basics/internet.php3.

[3] *Linux Online.*
http://www.linux.org/.

[4] *Linux (Wikipedia article).*
http://en.wikipedia.org/wiki/Linux.

[5] Q Docter. *Solaris 9 Sun Certified System Administrator Study Guide.* Sybex, ISBN: 0-7821-4181-1, 2003.

A Esitehtävät

1. Haluat tietää miten jotakin tiettyä komentoa käytetään. Millä komennolla saat esille kyseessä olevan komennon manuaalisivut?

2. Millainen on hyvä salasana? Millä komennolla voit vaihtaa salasanasi?

3. Mitä tarkoitetaan absoluuttisella ja suhteellisella polkunimellä?

4. Millä komennolla siirryt hakemistosta toiseen hakemistohierarkiassa? Millä komennolla näet nykyisen työskentelyhakemistosi?

5. Anna esimerkki ohjelman näyttötulostuksen ohjaamisesta (piping) toiselle komennolle.

6. Millä komennolla voit muuttaa tiedostojen ja hakemistojen käyttöoikeuksia? Anna esimerkki komennon käyttämisestä.

7. Millä komennolla saat tietoa ajossa olevista prosesseista? Millä komennolla voit lähettää signaalin prosessille?

8. Millä komennolla voit ottaa etäyhteyden johonkin toiseen työasemaan tai palvelimeen? Anna esimerkki komennon käyttämisestä.

B Questions

1. You want to find out how to use a certain command. Which command would you use to open the manual page for that command?
2. Describe a good password. What command is used to change the password?
3. What is meant by the *absolute* and *relative* pathnames?
4. What command is used to change the working directory? What command is used to show the current working directory?
5. Give an example of redirecting the output of a program to the input of another program (piping).
6. What command is used to set and change file permissions? Give an example.
7. What commands are used to list and signal processes running on a system, respectively?
8. How to make a remote connection to a server? Give an example.