
UNIX Survival Guide

Rafael Ostertag

\$Id: unixsurvivalguide.xml 1412 2009-01-09 19:36:16Z rafi \$

Copyright © 2008-2009 Rafael Ostertag

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The mark FreeBSD is a registered trademark of The FreeBSD Foundation.

UNIX® is a registered trademark of The Open Group.

Sun, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc.

The term “Linux” is a registered trademark of Linus Torvalds.

Important

This documentation is provided by Rafael Ostertag “As-Is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall Rafael Ostertag be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this documentation, even if advised of the possibility of such damage.

	Revision History
Revision 0.1	2009-01-09
	Initial release
Revision 0.2	2009-04-18
	Fixed spelling errors

Abstract

This article should help the non-UNIX savvy computer user to get along with an UNIX operating system. It mainly focuses on how to use the command line. Details not concerning the casual user have been omitted. It should be suited for Sun Solaris, FreeBSD, and Linux systems.

Send comments and suggestions to:

Rafael Ostertag <rafi@guengel.ch [mailto:rafi@guengel.ch]>

Get the latest copy online at guengel.ch [<http://www.guengel.ch/unixsurvivalguide/>].

Table of Contents

1. Typographical Conventions	2
2. The Terminal	3
2.1. The Shell	4
3. Users	4
3.1. User related Commands	5
4. Commands	6
4.1. Piping	9
4.2. Man Pages	11
4.3. Pagers	12
5. The File System	13
5.1. The Home Directory	15
5.2. The ls Command	15
5.3. File and Directory Commands	17
5.4. File and Directory Permissions	19
5.5. Special Stuff	21
6. Processes	22
6.1. Terminating Processes	24
6.2. Job Control	25
7. Miscellaneous	27
7.1. Reading Text Files	27
7.2. Editing Text Files	27
7.3. Printing	27
7.4. Search for Text in Files	28
7.5. File Manager	29
8. Command Glossary	31
9. References	43
9.1. FreeBSD	43
9.2. Sun Solaris	43
9.3. Linux	43
9.4. Miscellaneous	43
A. X Window System	43
B. GNU Free Documentation License	45

1. Typographical Conventions

\$Id: typography.xml 2330 2009-04-17 15:45:52Z rafi \$

Text you see on the screen appears in a shaded box like this.

Text on a line after the \$ symbol is what you would type in. Commands are in **bold letters**, and directory and file name are set in `this font`.

2. The Terminal

\$Id: terminal.xml 1404 2009-01-08 00:47:04Z rafi \$

The *terminal* or *console* is a text based interface to the operating system. It does not feature graphical widgets, so you need to type in the commands using the keyboard only. There is however a small difference between a terminal and a console. A console is directly attached to the computer, a terminal can be opened over the network or in a similar fashion. This difference does not affect the way you interact with the operating system. Therefore I will use those two terms interchangeably.

On FreeBSD and Linux you can switch from the graphical user interface, to the console by pressing simultaneously **Ctrl+Alt+Fx** where **Fx** is one of the **F1** to **F12** keys. Sun Solaris does not support switching from the graphical user interface to a console, so this key combination has no effect on Sun Solaris.

Example 1. Switching to the console on FreeBSD

You will be prompted to enter user name and password. After that, you're ready to use the console.

```
FreeBSD/i386 (penny.example.org) (ttyv0)
login:
```

If you do not want to leave your graphical user interface, you may open a terminal. It is likely that you will find an entry for a terminal somewhere in the menu of the graphical user interface.

It is possible to open a terminal on a remote UNIX system. Type the following command in a terminal

```
$ ssh username@hostname
```

Replace the `username` with your user name on the remote system and `hostname` with the host name of the remote host. For instance,

```
$ ssh joe@penny.example.org
Password:
```

As you can see, you will be prompted to enter the password for the user name you specified.

The above command will only work for remote system allowing *SSH Connections* (SSH stands for "Secure Shell"). In earlier days of UNIX, there were several other ways to connect to a remote system, telnet, rsh etc., to name only a few, but they were all insecure. They transmit your password unencrypted over the wire, so it is possible for anybody to read it.

2.1. The Shell

The *shell* is the program that lives inside a terminal and takes commands from you. As every where in the UNIX world, there is more than one way to do things, and so there exist many shells. The *bash* shell is the default shell on most Linux systems. FreeBSD favors the *tcsh* shell (*tcsh* stands for “Tenex C-Shell”). Sun Solaris uses *sh* (*sh* simply stands for “Shell”, on which *bash* is based on), and *ksh* (“Korn Shell”).

All shells provide a prompt (also known as *command line*). This is the place you enter commands. Depending on the shell and its configuration, the prompt may appear differently, but always serves the same purpose.

Example 2. Different prompts

A simple sh prompt

```
$
```

Default bash prompt

```
bash-3.00$
```

A customized tcsh prompt

```
jack@salma ~/unixsurvivalguide>
```

Bash, ksh, and tcsh shell have a nice feature called *file name completion*. File name completion allows you to enter some portion of the command or file name and make the shell complete the rest. By default, the bash shell invokes file completion if you press the **Tab** key. Tcsh needs you to press the **Ctrl+D**, and ksh **Alt+Esc** in order to invoke the file name completion. If the portion to complete is ambiguous, they will show a list of possible completions as shown in the following example

Example 3. File name completion

```
$ ema
emacs          emacs-chooser  emacsclient
emacs-athena-22.1  emacs-gtk-22.1
$ ema
```

To leave the terminal, you can type in **logout**, or if that doesn't work because you are not in a login shell (UNIX will tell you if you try **logout**) type **exit**. A short-cut is always using the key combination **Ctrl+D** (yes, it's the same key combination as used by the tcsh for command completion, so you have to be careful when using this combination in a tcsh shell).

3. Users

```
$Id: user.xml 2331 2009-04-18 12:59:52Z rafi $
```

UNIX is a multi-user operating system, meaning that it is possible for several users to work concurrently on the same system. Thus, UNIX needs a way to identify the user for login and security purposes. This is where your user name and password comes into play. Each user has a user name and a password which identify him or her on the system. The users are stored in `/etc/passwd`.

On regular UNIX systems, there exists a user called *root*. Root is the most powerful user, because he literally can do as he please. The file permissions do not apply to him, he can configure the system, create and delete users, change passwords, reboot and halt the system, you name it. I mention this user only for completeness.

Users can belong to groups, which are a way to manage access to files (see also Section 5.4, “File and Directory Permissions”). Groups are a way to give several users (the group members) access to files and directories in one whoop. Users can be listed in several groups. Last but not least, each user has its own home directory as described in Section 5.1, “The Home Directory”.

3.1. User related Commands

User related Commands

id Returns the “user identification”. On Sun Solaris the output looks like this

```
$ id
uid=1010(joe) gid=1010(joe)
```

On FreeBSD and Linux the output looks more confusing

```
$ id
uid=1001(joe) gid=1001(joe) groups=1001(joe),0(wheel),4(tty)
,92(gdm),99(video),700(cdrom),701(floppy),702(speaker),703(mou
nt),704(usb),1100(media),999(users)
```

The important thing with both outputs is the *uid*. The uid is the number internally assigned with your user name. UNIX does not think of users in terms of user names, but as uids. The user name is only provided for convenience, so you don't have to remember your uid.

groups To find out to which groups you belong, type in the command **groups**. It will display all the groups you belong to. Groups are used to give access to files and directories as explained in Section 5.4, “File and Directory Permissions”.

who am i If you need to find out as which user you are logged in (yes, this happens sometimes), type in the command **who am i**. On Linux it prints out something like this

```
$ who am i
joe pts/1 2009-01-01 14:05 (salma.example.org)
```

in the first column, you will see your user name.

If you type only **who**, you will see who is also currently using the computer, as shown below

```
$ who
jack      console    Jan  1 12:21    (:0)
jack      pts/7       Jan  1 13:21    (:0.0)
joe       pts/4       Jan  1 16:04    (penny.example.org)
averell   pts/8       Jan  1 16:04    (penny.example.org)
william   pts/9       Jan  1 16:05    (uma.example.org)
joe       pts/10      Jan  1 16:06    (ava.example.org)
```

su

The **su** command lets you switch to another user (switch user).

```
$ id
uid=1006(jack) gid=1006(jack)
$ su - william
Password:
$ id
uid=1002(william) gid=1002(william)
```

The dash is used to make the user switch behave like a full login. After the dash, type the name of the user you want to switch to. Of course, you need to type in the user's password, else security would be compromised.

passwd

This command lets you change your password. The usage is simple, as shown by Example 4, “Changing the user password” (please note, that the messages may vary from UNIX to UNIX). After using this command, the next login you'll do, you have to enter your newly set password.

Example 4. Changing the user password

```
$ passwd
Changing password for joe
(current) UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Important

UNIX is case-sensitive, so please make sure you remember the capitalization of the letters, else you will be locked out from your account.

4. Commands

\$Id: commands.xml 2330 2009-04-17 15:45:52Z rafi \$

You're always using commands when dealing with a terminal. You need to type them in at the shell prompt in order to do something. The shell then executes the command and shows you that it is done with it when displaying the prompt again. In order to execute the command,

which by the way is nothing more than a program, the shell needs to know where to search for commands. This is told the shell using the *PATH environment variable*. It holds the paths to search for commands by the shell. To see the contents of this variable, type in the command as shown by the example Example 5, “The PATH environment variable”.

Example 5. The PATH environment variable

The **echo** command can be used to print text and variables to the screen. To see the value of the PATH environment variable, type the command as shown below. Unfortunately, it is beyond the topic of this article to fully explain what environment variables are and how to deal with them.

```
$ echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin
$
```

As you can see, the paths are separated by colons.

If you type in a command in the shell, the shell searches for the command in the directories `/sbin`, `/bin`, `/usr/sbin`, and `/usr/bin` in exactly this order, given the PATH environment variable is set according to the example above. It starts the command first found. So, if you happen to have a command named **foo** in `/sbin` and `/bin` it will always start the one found in `/sbin` because of the order given.

Suppose you want to execute a command in a directory other than the ones specified in the PATH environment variable, you have to enter the full file name

Example 6. Executing a command not in your PATH environment variable

```
$ /home/joe/foo/bar
...
$
```

You mostly won't start commands by simply using their name on the command line, but you want to tell them how they should behave. You do this by supplying command line arguments, which are mostly referred to as options or switches. Every command takes different arguments, but the way you specify them is most of the time the same, as shown in the example Example 7, “Command line arguments”

Example 7. Command line arguments

```
$ somecommand [arg1] [arg2] ... [argN]
```

As you see by the example, you can specify more than one argument, but the key point is that they are separated by spaces. Another important thing to note is, that almost every command

line argument starts with a dash (-) if it consists of only one letter, or by two dashes (--) if the option is a term. Those are called *long options*. However, that is a rule of thumb, and the programmer of the command is free to expect the options in their own way. Therefore, each well programmed command has an option that will display a help text to the user, which is mostly invoked using either -h, or --help, as shown in the example Example 8, “The help switch”.

Example 8. The help switch

```
$ yapet -h
YAPET 0.2

yapet [-c] [-h] [-V] [<filename>]

-c, --copyright show copyright information

-h, --help      show this help text

-V, --version   show the version of YAPET

<filename>     open the specified file <filename>

YAPET stores passwords encrypted on disk using the blowfish algorithm.
The encryption key is computed from the master password using md5, sha1, and
ripemd-160 digest algorithms producing a 448 bit (56 bytes) key.
```

Most commands allow you to group single letter options together. As an example, I'll use the **ls** command, which is explained in more depth in Section 5.2, “The ls Command”

Example 9. Grouping command options

You can give each options separated with a space

```
$ ls -l -a -h
total 84K
drwxr-xr-x  2 joe joe   42 2009-01-03 00:34 .
drwx----- 80 joe joe  8.0K 2009-01-03 00:33 ..
-rw-r--r--  1 joe joe   40 2009-01-03 00:34 file1
-rw-r--r--  1 joe joe  123 2009-01-03 00:34 file2
-rw-r--r--  1 joe joe  62K 2009-01-03 00:35 file3
```

or you might group them together, as such

```
$ ls -lah
total 84K
drwxr-xr-x  2 joe joe   42 2009-01-03 00:34 .
drwx----- 80 joe joe  8.0K 2009-01-03 00:33 ..
-rw-r--r--  1 joe joe   40 2009-01-03 00:34 file1
-rw-r--r--  1 joe joe  123 2009-01-03 00:34 file2
-rw-r--r--  1 joe joe  62K 2009-01-03 00:35 file3
```

which produces the same result.

Sometimes you want to know where the command you want to start is located. UNIX provides the command **which** for this purpose. It displays the full path where the command is located. **Which** uses the `PATH` environment variable to locate the command in question. See the example below.

Example 10. The **which** command

```
$ which gtar
/usr/sfw/bin/gtar
```

If the command is not found, **which** prints an error message

Example 11. Error message of **which**

```
$ which dunno
dunno: Command not found.
```

4.1. Piping

UNIX allows you to feed the output of one command to another command for further processing. This is called *piping*. The output is redirected to another program by concatenating the two commands using the pipe symbol “|”. The next example should make this more clear.

Example 12. Redirecting the output using a pipe

Suppose you have a file containing an unordered list of names. You want to find all names containing a certain sequence of characters and print out all those names in a sorted fashion. You would need two commands to accomplish this, namely **grep** which searches for character strings in a file and **sort** which sorts the input it receives. Assuming your list is called `names.txt`. Start first with grepping the file for the sequence “in”, you would do

```
$ grep 'in' names.txt
```

which produces the output that is obviously unsorted

```
Mindy
Franklin
Erin
Cindy
Joaquin
Virginie
Josephine
Hermine
Rina
Colin
Nadine
Vince
$
```

if you do it again, but piping the output to **sort**, it looks more the way we want it

```
$ grep 'in' names.txt | sort
Cindy
Colin
Erin
Franklin
Hermine
Joaquin
Josephine
Mindy
Nadine
Rina
Vince
Virginie
$
```

You can pipe to any command that reads from the *standard input*. Refer to the man page (see Section 4.2, “Man Pages”) of the command in question to find out whether or not it supports piping. For example, the man page of **sort** states that

```
[...]
```

SYNOPSIS

```
/usr/bin/sort [-bcdfimMnru] [-k keydef] [-o output]
[-S kmem] [-t char] [-T directory] [-y [kmem]]
[-z recsz] [+pos1 [-pos2]] [file]...
```

```
[...]
OPERANDS
  The following operand is supported:

  file   A path name of a file to be sorted, merged or
         checked. If no file operands are specified, or if a
         file operand is -, the standard input is used.
[...]
```

The standard input is where the command reads the keystrokes when you're typing on the keyboard. It's counterpart is the *standard output*. The output of commands you see in a terminal is printed to standard output. The standard input is for reading and the standard output is for writing by the program.

You might wonder why they are called standard input and output. Well, if the program does not setup things differently, they are the default input and output “channels”. A program might read from a file, which is not a standard input. Same applies when writing to a file, which is not standard output..

4.2. Man Pages

UNIX provides a help system called “Man Pages”, which is short for “Manual Pages”. Most commands come along with a man page. You simply need to type in **man** and provide the command you want to know more about as argument as shown in Example 13, “Example of a man page” below.

Example 13. Example of a man page

```
$ man ls
LS(1)                                FreeBSD General Commands Manual          LS(1)

NAME
  ls -- list directory contents

SYNOPSIS
  ls [-ABCFGHILPRSTUWZabcdefghiklmnopqrstuwXl] [file ...]

DESCRIPTION
  For each operand that names a file of a type other than directory, ls
  displays its name as well as any requested, associated information. For
  each operand that names a file of type directory, ls displays the names
  of files contained within that directory, as well as any requested, asso-
  ciated information.
[...]
```

Man pages are divided in sections. If you don't specify the section when invoking the **man** command, **man** searches all the sections and picks the first manual page it finds. In fact, it is possible for commands to appear in more than one sections, as shown in the example below. The example uses the **whatis** command that displays a one-line summary about commands.

Example 14. Example of a command that appears in more than one man page section

```
$ whatis tr
tr          tr (1)          - translate characters
tr          tr (1b)         - translate characters
```

As you see by the example above, the **tr** command is listed in the section 1 and 1b (the sections are shown in parentheses). The Table 1, “The standard man page sections” lists the basic man page sections. To display a man page from a particular section, you enter the **man** command in the following manner under FreeBSD and Linux

```
$ man <section> <cmdname>
```

Under Sun Solaris you would enter it like this

```
$ man -s <section> <cmdname>
```

where you replace **<section>** by the section number and **<cmdname>** by the command name.

Table 1. The standard man page sections

Section #	Topic
1	Commands available to users
2	Unix and C system calls
3	C library routines for C programs
4	Special file names
5	File formats and conventions for files used by Unix
6	Games
7	Word processing packages
8	System administration commands and procedures

4.3. Pagers

Many times, the output of commands won't fit on your screen because it has more lines than your screen has. This is when pagers come into play. They can be used to “scroll” thru the output of commands.

The two most commonly used pagers are **more** and **less**. **More** is somewhat archaic, since you can scroll down only. **Less** is more user friendly. Both programs scroll down an entire screen when you press the **SPACE** key, and one line at a time when you press the **RETURN** key. They quit on the **q** key. When using **less** you can use your cursor keys to move up and down.

Both commands can be used to pipe output into them, as Example 15, “The usage of more or less” shows.

Example 15. The usage of more or less

```
$ ps ax | less
PID  TT  STAT      TIME COMMAND
  0  ??  DLs      0:00.05 [swapper]
  1  ??  SLs      0:00.02 /sbin/init --
  2  ??  DL       0:01.50 [g_event]
  3  ??  DL       0:01.02 [g_up]
  4  ??  DL       0:00.95 [g_down]
  5  ??  DL       0:00.00 [xpt_thr]
  6  ??  DL       0:00.00 [thread taskq]
  7  ??  DL       0:00.00 [kqueue taskq]
  8  ??  DL       0:00.00 [acpi_task_0]
:
```

For an explanation of the `ps` command, see Section 6, “Processes”. The important thing to grasp here, is that `less` takes the output from the `ps` command, and displays only a screen full of text and let you scroll thru the remaining text (`more` could have been used as well).

5. The File System

\$Id: filesystem.xml 2331 2009-04-18 12:59:52Z rafi \$

Important

File and directory names are case-sensitive in UNIX. For instance, the file names `figures.txt` and `Figures.txt` are different file names when it comes to UNIX.

The file system is the place where files are stored. Files can be grouped in directories (also known as “folders”). Directories are hierarchically organized, which can be thought of as a tree (see Figure 1, “An excerpt of the directory structure of the FreeBSD operating system”). The start point, or top-level directory of the file system is called *root* (not to be confused with the user also called “root”). Everything else is a sub-directory of the root directory. Sub-directories can have an arbitrary number of other sub-directories, and so on.

Figure 1. An excerpt of the directory structure of the FreeBSD operating system

```

/
|-- bin
|-- etc
|   |-- x11
|   |-- defaults
|   |...
|   `-- rc.d
...
|-- home
|   |-- william
|   `-- averell
|-- lib
|-- sbin
|-- usr
|   |-- X11R6
|   |-- bin
|   |-- doc
|   |-- include
|   |-- lib
|   |...
|   |-- local
|   |-- ports
|   |-- sbin
|   |-- share
|   `-- src
-- var
|   |-- account
|   |-- log
|   |-- mail
|   |...
|   |-- run
|   |-- spool
|   |-- tmp
|   `-- yp

```

To put it plain and simple, directories are like chapters in a book, sub-directories are sub-chapters, which can be even further divided in sub-sub-chapters etc. The root directory is the book itself, and files would be paragraphs of chapters.

To reference a directory, you use a *path*. There are two types of paths

absolute paths which always start at the root, for instance `/usr/sbin`

relative paths which are relative to the current directory, for instance `sbin`

The slash `/` is used to separate directory names in a path. The path `/usr/share/doc`, for instance, points to the root directory ("`/`"), in there to the `usr` directory. The next component points to the `share` directory which is a sub-directory of `/usr`. The last component eventually points to the `doc` directory in `/usr/share`.

As you have seen, the root directory is not named "root" but is only the single `/` character. As opposite to a Microsoft based operating system where for each drive a separate directory tree exists, UNIX has only one directory tree. Everything else is part of that directory tree. Thus,

you cannot be sure whether or not the directory you are currently in is really on your computer. But you do not need to bother either.

For you as user, the most important directories are `/bin`, `/usr/bin`, and `/usr/local/bin`. These are the directories most commands you work with are stored in. There may be other directories, but that depends on the operating system. Under Sun Solaris, for instance, the directories `/usr/sfw/bin` and `/opt/sfw/bin` hold some useful commands as well.

You are always in a directory. Sometimes it is not so obvious, sometimes it is more obvious. Thus there is an important term *current working directory* (`cwd`). The current working directory is what its name says, the directory you are working in.

In every directory, there are always two special directories. One is simply named `.` (one dot), and the other `..` (two dots). Two dots stands for the parent directory of the directory they are in. So, for example, if you are in the directory `/usr/bin`, the two dots point to `/usr` (see Example 16, “The two dots (`..`) directory”).

Example 16. The two dots (`..`) directory

The `cd` command is used to change directories, and `pwd` is used to show the *personal working directory* which is equivalent to `cwd` (see also Section 5.3, “File and Directory Commands” for an explanation of those two commands).

```
$ pwd
/home/jack/test
$ cd ..
$ pwd
/home/jack
$
```

5.1. The Home Directory

Each user has a *home directory*. If you do a fresh login, you will most likely find yourself in your home directory. Under Linux and FreeBSD the home directories are located under the `/home` directory. On Sun Solaris they are kept in the `/export/home` directory. Regardless of the system, they are called the same as your user name. If the user name is “joe”, the full path of the home directory would be `/home/joe` under FreeBSD and Linux and `/export/home/joe` under Sun Solaris.

The home directory is your directory, and you are by default the only person having access to it (of course, root has access, too). Programs store their settings in that directory, and you should use it to store your data, unless advised otherwise by your UNIX administrator.

5.2. The `ls` Command

If you want to know the content of a directory, you'll want to use the `ls`. It lists the content of the directory you specified, or the content of the current working directory if you don't specify a directory

UNIX Survival Guide

```
$ ls
AdobeFnt.lst      data          example.org    schemas.xml~
agent-info        Desktop      libwindow     SunStudioProjects
analysis1.tex     Documents    media         test
bin               files        pgadmin.log   unixsurvivalguide
cert              GNUstep     schemas.xml

$ ls /tmp
0_sun_studio_12      mc-jack
gpg-X0ygPY          ogl_select341
hsperfdata_noaccess opera-9.63-2474.gcc3-static-qt3
hsperfdata_root     opera-9.63.gcc3-static-qt3.tar.gz
kde-jack            ssh-YZysV664
ksocket-jack
```

Maybe you want some more information on the files, you could try the `-l` option, which produces something like that

```
$ ls -l
-rw-r--r--  1 jack   jack   494 Dec 28 22:05 abstract.xml
-rw-r--r--  1 jack   jack   115 Dec 28 22:00 commands.xml
drwxr-xr-x  4 jack   jack   27 Dec 28 22:04 docbook
-rw-r--r--  1 jack   jack  6496 Dec 28 22:18 filesystem.xml
-rw-r--r--  1 jack   jack   852 Dec 28 22:05 Makefile
-rw-r--r--  1 jack   jack   166 Dec 28 22:05 processes.xml
-rw-r--r--  1 jack   jack   123 Dec 28 22:00 terminal.xml
-rw-r--r--  1 jack   jack  59645 Dec 28 22:18 unixsurvivalguide.fo
-rw-r--r--  1 jack   jack   9566 Dec 28 22:18 unixsurvivalguide.html
-rw-r--r--  1 jack   jack   1195 Dec 28 22:05 unixsurvivalguide.xml
```

The table below explains the columns printed by the `ls` using the `-l` switch (from left to right).

Table 2. The output of `ls -l` explained.

Column #	Description
1	The permission of the directory or file as discussed in Section 5.4, “File and Directory Permissions”. However, the first character describes the type of the directory item. If it shows the letter <code>d</code> the entry describes a directory. Is a dash (<code>-</code>) printed, the entry in question is a regular file.
2	The meaning of the number in depends of the type of the entry. If the entry is a directory, that is the number of files (and only files) in the directory. Should the entry happen to be a file, that number represents a count of hard-links to that file (see also Section 5.5, “Special Stuff”).
3	Name of the user that owns that file or directory. For each directory or file, there can be only one owner. The owner can be changed by the owner himself or root, using the chown command (see Section 5.4.1, “Commands” for further details on chown).
4	The name of the group that has access to the file or directory. As you can see by the example above, it is possible for groups to have the same name as users. The system can distinguish between them. The group ownership can only be changed by the owner of the file or root using the command chgrp (see Section 5.4.1, “Commands” for further details on chgrp).

Column #	Description
5	The size in bytes of the file or directory. You might want to see that in a human readable fashion. Try <code>ls -lh</code> and the size will be printed nicely in units of bytes, kilo bytes, mega bytes, and giga bytes as appropriate.
6	The time and date of the last modification.
7	The name of the file or directory.

In order to display hidden directories and files (see Section 5.5, “Special Stuff”), call `ls` with the `-a` option. Each file or directory starting with a dot is hidden directory or file. Hidden directories and files are not displayed when invoking `ls` without the `-a` switch.

5.3. File and Directory Commands

Note

The commands explained below will work on the current working directory if you don't specify an absolute path.

pwd Returns the current working directory, for instance

```
$ pwd
/usr/bin
```

ls Refer to Section 5.2, “The ls Command”.

cd The most important command right next to `ls`. It allows to change the current directory.

```
$ pwd
/home/jack
$ cd /usr/bin
$ pwd
/usr/bin
```

You can use relative and absolute paths when invoking `cd`. For the following examples, we assume the current working directory is `/home/joe`. In `/home/joe` exist the two directories `tmp` and `bin`.

To change to the `bin` directory, there exist two ways: the longer way is to specify the absolute path

```
$ cd /home/joe/bin
```

and the probably preferred way using the relative path

```
$ cd bin
```

Both ways will change the directory to `/home/joe/bin`.

Suppose, you are still in the `/home/joe/bin` directory and you want to change to the `/home/joe/tmp` directory. Again, you have two choices: using the absolute path, you would enter

```
$ cd /home/joe/tmp
```

or the easy way

```
$ cd ../tmp
```

mv With the **mv** command you can move files or directories from one directory to another. It is also used to rename files or directories.

```
$ mv <source> <destination>
```

Replace `<source>` with the file or directory name to be moved. `<destination>` is the destination where the file is moved to. If it does not exist, **mv** assumes you want to rename the file or directory.

Warning

If the destination exists, it will be overwritten without confirmation.

cp The **cp** copies files. It has the same syntax as the **mv** command. If you want to copy a directory, use the `-r` switch as shown in the example below.

```
$ cp -r data/ /home/joe
```

Warning

If the destination exists, it will be overwritten without confirmation.

rm **rm** is used to delete files. You can only delete a file if you have write permission to the directory the file is in. Else you would see an error message like that

```
$ rm figures
rm: figures not removed: Permission denied
```

You can also delete directories with the **rm** providing the `-r` switch. See the example below.

```
$ rm -r olddata
```

Warning

You must be careful using the `-r` switch of **rm**. It deletes everything in the directory specified including sub-directories without any confirmation question, such as known by Microsoft operating systems. So you once fired on the wrong directory, you have little chance of stopping it. Especially since the **rm** command deletes the files for real, and does not move them to some recycle bin or trashcan, where you can salvage the files from.

mkdir **mkdir** creates a new directory.

```
$ mkdir <dirname>
```

Replace the <dirname> with the name of the new directory to be created.

rmdir With the **rmdir** you can delete an empty directory. The syntax is the same as the one of **mkdir**.

5.4. File and Directory Permissions

We look again at the output of the **ls** invoked with the **-l** option and pay only attention to the first column as shown in Example 17, “Permission bits”.

Example 17. Permission bits

```
-rw-r--r-- 1 joe users 494 Dec 28 22:05 somefile
drwxr-xr-x 4 joe users 27 Dec 28 22:04 docbook
```

The permission bits are displayed in the first column printed by **ls -l**.

Not taking the first character into account, the first column can be divided into three groups (*triplet*) of three characters each. The first triplet describes the permissions of the entry applying to the owner of the entry, the second the permission applying to the group owner. The last triplet applies to anybody not covered by the first two triplets, or the “other” or “world” in UNIX speak.

In the example above, the user **joe** has set the permission to **rw-**, the group **users** to **r--**, and the world to **r--**.

The first character of a triplet indicates whether or not the entry can be read. If a **r** appears, the bit is set and thus the entry can be read. A dash would indicate that the bit has not been set. The second character controls the write access to an entry. A **w** stands for write access granted, and the dash means the bit is not set. The third and last character of a triplet specifies whether or not the entry can be executed by the operating system. Depending of the type of the entry (file or directory) its meaning is different. If this bit is set for a file, it indicates that file can be executed by the operating system. For example, commands have this bit set. If the entry is a directory, the execution bit tells UNIX that users are allowed to change to this directory. But only changing is allowed. If only the execution bit is set on directory, you can do a **cd** to it, but you wont be able to read the contents of the directory.

See Example 18, “Permission bits revisited” for some examples of permission bits.

Example 18. Permission bits revisited

```
-rw-r--r-- 1 joe users
```

The user `joe` has read and write access to the file. The group `users` and world have read access to the file.

```
-rw-rw-r-- 1 root staff
```

Root and the group `staff` have read and write access to the entry. World has read-only access.

```
drwxr-xr-x 1 root users
```

Root has read and write access to the directory. The group `users` and world can read the directory. Root, `users`, and world are allowed to change to the directory.

```
-rwxr-xr-x 1 root bin
```

Root has read and write access to the file. The group `bin` and world have read-only access. Root, `bin`, and world can execute the file.

5.4.1. Commands

chmod **chmod** lets you change the permission bits of files and directories. I will explain only how to use the absolute-mode, where you specify a number representing the access bits. Refer to the man page of **chmod** for more details.

In absolute-mode, the number 1 is used to set the execution bit, 2 for write access, and 4 for read access. You can combine the mode by adding the numbers, e.g. 6 would mean read and write access, 5 read and execute permission.

With those numbers you construct a three digit number. The first digit represents the permissions for the owner, the second digit the permission for the group, and the third the permission for world. See Example 19, “`chmod` using absolute-mode”.

Example 19. `chmod` using absolute-mode

Issuing the command

```
$ chmod 774 somefile
```

would set the permission for the owner and the group of the file `somefile` to read, write, and execute. World would have read access only.

chown The **chown** lets you change the owner of a file. You are only allowed to do so, if you are the owner. You cannot take over the ownership of a file you don't already own.

The syntax of the **chown** command is simple:

```
$ chown <user> <filename>
```

Replace <user> with the name of the new owner and <filename> with the file or directory name of which the owner has to be changed.

chgrp **chgrp** is used in the same manner as **chown**. It will change the group owner of the file specified. You may only change the group owner if you are the owner of the file or directory.

5.5. Special Stuff

5.5.1. Hidden Files

It is a convention, that you can hide a file or directory under UNIX by prepending a dot in front the name. This is often used for configuration files in your home directory. You can list hidden files by invoking **ls** with the **-a** option as shown in Example 20, “List hidden files and directories”.

Example 20. List hidden files and directories

```
$ ls -a
.                .padminrc
..               .pgadmin3
.acrorc          .pgpass
.adobe           .printers
.alias           .purple
.aspell.en.prepl .qt
[...]
```

The entries with a dot in front of the name are hidden (except, of course, the first two entries in the left column).

But you cannot arbitrarily hide files by simply putting a dot in front of the name. The dot is part of the file name and thus you would change the file name. If you want to reference to a hidden file or directory, you have to include the leading dot as well.

5.5.2. Links

You can create links to files and directories. From a user's point of view, a link is no different to a file or directory, but they can be very convenient. You could, for example, create links to often used directories in your home directory. This would save you from always typing in the whole path.

UNIX knows two types of links, namely *hard-links* and *soft-links*. I won't discuss hard-links here, because they are somewhat limited.

You can create soft-links to files and directories as shown in Example 21, “Creating soft-links”. Make sure you don't omit the **-s** switch. Else you would create a hard-link.

Example 21. Creating soft-links

```
$ ln -s <sourcefile> <linkname>
```

Replace `<sourcefile>` with the file or directory name of the item the link has to point to. The name of the link is specified by `<linkname>`.

When using the `-l` with `ls`, you will spot soft-links easily, as shown in Example 22, “Spotting soft-links with `ls`”.

Example 22. Spotting soft-links with `ls`

```
$ ls -l
total 72
-rw-r--r--  1 jack   jack    6144 Jan  7 15:08 file1.txt
-rw-r--r--  1 jack   jack    1024 Jan  7 15:07 file12.txt
-rw-r--r--  1 jack   jack   15360 Jan  7 15:07 file13.txt
-rw-r--r--  1 jack   jack    5120 Jan  7 15:07 file2.txt
lrwxrwxrwx  1 jack   jack      9 Jan  7 15:08 justalink -> file1.txt
```

The last line of the output clearly shows that `justalink` is a soft-link to `file1.txt`.

Links can be removed using the `rm` command like regular files.

6. Processes

```
$Id: processes.xml 2331 2009-04-18 12:59:52Z rafi $
```

Every command that is started, is load into the memory by the *kernel*. The kernel is the thing that handles the hardware and executes programs. When the command is loaded it got a *process identification* (PID) assigned. The PID is unique within the operating system instance.

Most commands you will start, will only run for a certain time and then terminate, either because you requested so, or because their work is done. But there are processes that run in the background as long as the operating system runs. They are called *daemons*. Web servers, for instance, most like run as daemons.

To see the list of processes that are running on your system, use the command `ps`. On a FreeBSD box invoke it using the `ax` options as shown in Example 23, “Output of the `ps` command on FreeBSD”.

Example 23. Output of the ps command on FreeBSD

```
$ ps aux
  PID  TT  STAT      TIME COMMAND
    0  ??  DLs    0:00.04 [swapper]
    1  ??  SLs    0:00.02 /sbin/init --
    2  ??  DL     0:01.20 [g_event]
   41  ??  DL     0:00.04 [fdc0]
   42  ??  DL     0:00.05 [pfpurge]
   43  ??  DL     0:00.01 [pagedaemon]
   44  ??  DL     0:00.00 [vmdaemon]
   45  ??  DL     0:00.00 [pagezero]
  396  ??  Ss     0:02.52 /usr/sbin/moused -p /dev/ums0 -t auto -I /var/run/mou
  423  ??  Is     0:00.00 /usr/sbin/moused -p /dev/ums1 -t auto -I /var/run/mou
  471  ??  Is     0:00.00 /sbin/devd
  541  ??  DL     0:00.01 [accounting]
  628  ??  Ss     0:00.09 /usr/sbin/syslogd -s
  649  ??  Ss     0:00.02 /usr/sbin/rpcbind
  664  ??  Ss     0:00.05 /usr/sbin/ypbind
  692  ??  Ss     0:00.01 /usr/sbin/rpc.statd
  699  ??  Is     0:00.00 /usr/sbin/rpc.lockd
```

The first column shows the PID of the processes. Processes in square brackets are kernel processes (there is no command for them available, they were created directly by the kernel).

On Sun Solaris and Linux machines, you use the **ps** with the **-ef**, as shown in Example 24, “Output of the ps command on Linux and Sun Solaris machines”.

Example 24. Output of the ps command on Linux and Sun Solaris machines

```
$ ps -ef
  UID  PID  PPID  C   STIME TTY      TIME CMD
  root    0    0    0 10:12:53 ?        0:43 sched
  root    1    0    0 10:12:58 ?        0:00 /sbin/init
  root    2    0    0 10:12:58 ?        0:00 pageout
  root    3    0    0 10:12:58 ?        0:16 fsflush
  root   319   318  0 10:13:25 ?        0:00 /usr/sadm/lib/smc/bin/smcboot
  root    7    1    0 10:12:59 ?        0:03 /lib/svc/bin/svc.startd
  root    9    1    0 10:13:00 ?        0:10 /lib/svc/bin/svc.configd
  root   529   510  0 10:13:34 ?        0:00 /usr/openwin/bin/fbconsole -d
  root   82    1    0 10:13:11 ?        0:00 devfsadmd
  root   255    1    0 10:13:24 ?        0:00 /usr/lib/netsvc/yp/ypbind
  jack   668   653  0 10:17:40 ?        0:00 /bin/ksh /etc/dt/config/Xsessi
  root   275    7    0 10:13:25 ?        0:00 /usr/lib/saf/sac -t 300
  daemon 155    1    0 10:13:16 ?        0:01 /usr/lib/rcap/rcapd
```

The first column holds the user name of the user that has started the command. The second column shows the PID of the process.

It is possible to watch the processes in real time using the **top** command on FreeBSD and Linux. The **top** will also show their resource usage, such as cpu-time and memory consumption (see Example 25, “The top on FreeBSD and Linux”). In order to leave **top** press **q** on the keyboard.

Example 25. The top on FreeBSD and Linux

```
$ top
last pid: 13481; load averages: 0.14, 0.23, 0.18 up 0+02:53:49 15:45:03
3 processes: 1 running, 2 sleeping
CPU: 0.8% user, 0.0% nice, 2.6% system, 0.2% interrupt, 96.4% idle
Mem: 367M Active, 224M Inact, 98M Wired, 9564K Cache, 112M Buf, 800M Free
Swap: 3015M Total, 3015M Free

  PID USERNAME   THR PRI NICE   SIZE    RES STATE  C   TIME    WCPU COMMAND
13275 joe           1  20   0  5504K  2524K pause  1   0:00   0.00% tcsh
13281 joe           1  44   0  3496K  1752K CPU1   1   0:00   0.00% top
13273 joe           1  44   0  8332K  4032K select 0   0:00   0.00% sshd
```

Some Sun Solaris machines do not have the **top**. Use **prstat** instead as shown in Example 26, “The prstat command on Sun Solaris”. **Prstat** will also stay running unless you press the **q** key.

Example 26. The prstat command on Sun Solaris

```
$ prstat
  PID USERNAME   SIZE  RSS STATE  PRI NICE   TIME    CPU PROCESS/NLWP
19988 jack        31M  25M sleep  49  0   0:00:41 1.6% emacs-gtk-22.1/1
  511 jack        42M  80M run    58  0   0:07:03 1.3% Xorg/1
  688 jack        16M  10M sleep  49  0   0:01:46 0.6% gkrellm/1
  689 jack        16M  10M sleep  49  0   0:01:05 0.3% gkrellm/1
  742 jack       144M  77M sleep  59  0   0:03:54 0.3% firefox-bin/7
```

6.1. Terminating Processes

Sometimes it is necessary to terminate processes that got unresponsive or behave otherwise oddly. On all UNIXes, the **kill** command can be used to send processes a signal to terminate. Find out the PID of the process in question and use the kill command as shown below.

Example 27. The kill command

```
$ kill <pid>
```

Replace **<pid>** with the PID of the process. Sometimes you want to kill a process, you would use then the additional **-KILL** option.

Example 28. Using the -KILL option

```
$ kill -KILL <pid>
```

The difference between **kill** with and without the `-KILL` option, is that **kill** without that option lets terminate the process more gracefully, whereas **kill** with `-KILL` kills the process without mercy, so to speak.

If you happen to know the name of the command you want to terminate, you could use also the **killall** command on Linux, or **pkill** on FreeBSD and Sun Solaris. They both take the name of the process to terminate, and terminate *all* processes with that name (see Example 29, “The pkill and killall commands”).

Example 29. The pkill and killall commands

Use the **pkill** command on Sun Solaris and FreeBSD to kill processes with the same name.

```
$ pkill <procname>
```

On Linux, you use the **killall** command.

```
$ killall <procname>
```

Replace `<procname>` with the name of the process to terminate. You optionally could provide the `-KILL` switch, in the same manner as with **kill**.

If you simply want to terminate the command you just started on the command line, you might try pressing **Ctrl+C**. In most cases this will terminate the active command.

Please keep in mind, that you can only terminate or kill processes you launched. Processes from other users are protected from termination other than the “owner” of the process. As usual, root is the exception, it can terminate any process.

6.2. Job Control

The bash, ksh, and tcsh offer a feature called *job control*. Job control allows to start more than one command at a time from the command line. Job control is invoked using either the `&` sign, or by pressing **Ctrl+Z**. Using either way, you put the command in the background and you can fire up a new command. The difference is, that a command invoked with the `&` sign is running in the background, whereas a command is suspended in the background when using **Ctrl+Z**.

You use the `&` sign when starting a new command on the command line as shown in Example 30, “Starting a command in the background”. Simply put at the end of the line a `&`, and the command is started in the background.

Example 30. Starting a command in the background

```
$ cp -r hugedir copyofhugedir
```

If you want to put the active command in background, press **Ctrl+Z** as in Example 31, “Putting the active command in the background”.

Example 31. Putting the active command in the background

The example shows a file being edited in a text editor. By pressing **Ctrl+Z**, the text editor is put in the background and the prompt appears.

```
use warnings;
use List::Util 'shuffle';

# in minutes
my $SLEEP=3;
my $WMSETBG='/opt/csw/bin/wmsetbg';
my @DIRECTORIES= (
    '/mnt/media/background'
);
"chbg.pl" 45 lines, 865 characters

Suspended
$
```

Note the Suspended notice. The command is still there, but it does not run.

A command that has been suspended with **Ctrl+Z** can be run in the background calling **bg** right after pressing **Ctrl+Z** as shown in Example 32, “Running a suspended job in the background”. If you have more than one background processes, see first the output of the **jobs** command (Example 33, “Using the jobs command”) and invoke **bg** with the job number.

A command suspended or running in the background can be brought back to the foreground by calling **fg**. It uses the same syntax as **bg**.

Example 32. Running a suspended job in the background

```
^Z
Suspended
$ bg
[2]  cat &
$
```

Example 33. Using the jobs command

```
$ jobs
[1]  Suspended                vi chbg.pl
[2]  - Suspended (tty input)   cat
[3]  + Suspended                ls -R /
```

Use the number of the first column when you want to background or activate a command using **bg** or **fg** as shown below.

```
$ bg % <jobnum>
[... ]
$ fg % <jobnum>
```

Replace <jobnum> with the number obtained from **jobs**.

7. Miscellaneous

\$Id: misc.xml 2330 2009-04-17 15:45:52Z rafi \$

7.1. Reading Text Files

The easiest way to read text files is by using either **more** or **less**. Instead of piping the output from a command to them, you provide the file name of the file to be opened, as shown in Example 34, “Reading text files with more or less”.

Example 34. Reading text files with more or less

```
$ more <filename>
```

Replace `<filename>` with the file name of the file to be read. You could use **less** instead of **more** as well.

7.2. Editing Text Files

This is a somewhat difficult topic. There are many good editors out there for UNIX, but it heavily depends on the installation of the UNIX machine, which editors are available. No doubt, the most (in)famous editors are *Emacs* and *vi*. On almost every UNIX system you will find a sort of *vi*, but doing the most basic editing stuff can be a nuisance in *vi*.

The text editor I recommend for newbies is *jed*. It is easy to learn and has a text menu. You start *jed* by typing **jed** on the command line. You can optionally provide the file to open right away as argument to **jed**.

Jed is available for FreeBSD, Linux, and Sun Solaris, as far as I know.

7.3. Printing

This topic I will cover briefly. Ask your system administrator for further assistance with this topic.

7.3.1. Printing under FreeBSD

You submit a file to print to the default printer by calling

```
$ lpr <filename>
```

Replace `<filename>` with the file to print. Handling of different file types depends on the configuration. So it may not be possible to print PDF files, for instance. If you are in doubt, ask your system administrator for further help.

If you want to see which files are in the print queue, call **lpq**

```
$ lpq
lp@ava (originally hl5050) 0 jobs
```

To remove the currently active print job, type **lprm** on the command line. You can only remove jobs you sent to the printer, hence if you are the owner.

7.3.2. Printing under Linux and Sun Solaris

Submitting a file to the default printer is accomplished by issuing the following command

```
$ lp <filename>
```

Replace `<filename>` with the file to print. Handling of different file types depends on the configuration. So it may not be possible to print PDF files, for instance. If you are in doubt, ask your system administrator for further help.

If you want to see which files are in the print queue, call **lpstat**

```
$ lpstat
Printer: lp@ava 'hl5050'
Queue: no printable jobs in queue
Server: no server active
Status: job 'william@penny+624' saved at 18:21:15.497
Rank  Owner/ID          Pr/Class Job Files          Size Time
done  william@penny+624  B/penny.example.org 624  stdin          1502236 18:20:46
```

To remove the currently active print job, type **cancel** on the command line. You can only remove jobs you sent to the printer, and thus you are the owner.

7.4. Search for Text in Files

The **grep** command lets you search for text in files, or if you pipe output from other commands to it, within that output. **Grep** uses very powerful *regular expressions*, but this topic is not covered here. In its simplest form, it searches for just the text you specify as shown in Example 35, “Using grep the easy way”.

Example 35. Using grep the easy way

The `-n` option tells **grep** to print the line numbers the text has been found. If this option is omitted, it simply prints the line containing the text.

```
$ grep -n '<searchstring>' <filename>
```

Replace `<searchstring>` with the text **grep** has to look for (**grep** is by default case-sensitive, so searching for `Text` is not the same as `text`). `<filename>` is replaced by the file **grep** has to search in. See below for a concrete example.

```
$ grep -n 'sec' unixsurvialguide.xml
17:
```

Figure 2. The GNU Midnight Commander

```

Left      File      Command  Options  Right
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Name      | Size | MTime | Name      | Size | MTime |
|-----|-----|-----|-----|-----|-----|
|.cache    | 18 | Dec 16 12:18 | /etc      | 251 | Jan 7 10:13 |
|.config   | 69 | Dec 16 12:18 | /export   | 4   | Dec 3 21:16 |
|.dbus     | 24 | Apr 25 2008 | /home     | 2   | Jan 7 10:13 |
|.dbus-keyrings | 36 | Jan 7 15:51 | /kernel   | 19  | Dec 18 23:11 |
|.designer  | 6  | Dec 22 21:47 | /lib      | 241 | Dec 18 23:14 |
|.devsketch | 25 | Dec 22 22:17 | /mnt      | 4   | Jan 7 19:37 |
|.dia      | 111 | Dec 16 00:22 | /net      | 2   | Nov 29 23:53 |
|.dt       | 4096 | Dec 15 21:49 | /opt      | 20  | Dec 28 14:52 |
|.emacs.d  | 79 | Oct 23 21:17 | /platform | 5   | Nov 29 23:25 |
|.enlightenment | 4096 | Dec 15 21:48 | /proc     | 130496 | Jan 7 19:42 |
|.fluxbox  | 54 | Dec 15 21:49 | /rmdisk   | 2   | Jan 3 19:59 |
|.fontconfig | 8192 | Dec 7 10:39 | /rpool    | 4   | Dec 19 10:46 |
|.gaim     | 82 | Apr 22 2008 | /sbin     | 52  | Dec 4 22:59 |
|.gconf    | 44 | Jan 7 10:17 | /share    | 3   | Dec 3 22:38 |
|.gconfd   | 24 | Jan 7 15:52 | /system   | 4   | Nov 29 23:03 |
+-----+-----+-----+-----+-----+-----+
|.enlightenment | /proc
+-----+-----+-----+-----+-----+-----+
Hint: Want your plain shell? Press C-o, and get back to MC with C-o again.
jack@salma ~-> [^]
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir 8Delete 9PullDn 10Quit

```

If mc is installed, you may start it by typing **mc** on the command line. See Table 3, “Midnight Commander keys” for a short overview about the keys you can use with mc.

Table 3. Midnight Commander keys

Key	Action
F1	Bring up a help window.
F2	Show the user customizable menu.
F3	View the currently selected file.
F4	Edit currently selected file.
F5	Copy the selected file or directory to the directory active in the other pane.
F6	Move the selected file or directory to the directory active in the other pane. Can also be used to rename files.
F7	Create a new directory in the currently active directory.
F8	Delete the selected file or directory.
F9	Activate the menu.
F10	Quit Midnight Commander.
Tab	Switch to the other pane.
Insert	Mark the currently selected entry.

Key	Action
Ctrl+O	Switch to the shell. Press again to switch back to Midnight Commander.

8. Command Glossary

\$Id: cmdgloss.xml 2331 2009-04-18 12:59:52Z rafi \$

A glossary of the command presented in this article. For detailed information about the commands, refer to the manual pages of the respective command.

Switches enclosed in square brackets ([]) are optional. If not stated otherwise, the commands are available under FreeBSD, Linux, and Sun Solaris.

Glossary

bg Resumes a suspended command in the background. Available only in shells that are capable of job control.

Related commands: **fg**, **jobs**.

Refer to Section 6.2, “Job Control”.

Syntax.

```
bg [% jobnum]
```

Replace `jobnum` with the number of the suspended job to put in background. See also **jobs**. If you omit the number, the last suspended command will be put in the background.

cancel The **cancel** utility cancels print requests.

Related commands: **lp**, **lpq**, **lprm**, **lpstat**.

Refer to Section 7.3.2, “Printing under Linux and Sun Solaris”.

Availability. Linux, Sun Solaris

cat The **cat** utility reads files sequentially, writing them to the standard output.

Related commands: **grep**, **sort**.

Syntax.

```
cat file...
```

Cat processes the files specified by `file...`

cd The **cd** utility changes the current directory.

Related commands: **mkdir**, **rmdir**, **pwd**.

Refer to Section 5, “The File System”.

Syntax.

```
cd dirname
```

Replace `dirname` with either the absolute or relative path to the directory to change to.

chgrp

The **chgrp** command will set the group ID of the file named by each file operand to the group ID specified by the group. You are only allowed to change the group if you are the owner of the file or directory.

Related commands: **chown**, **chmod**.

Refer to Section 5.4.1, “Commands”.

Syntax.

```
chgrp [-R] groupname file...
```

Replace `groupname` by the name of the group. You can specify one or more files or directories by replacing `file...` Using the optional `-R` switch, it will recursively change the group of a directory.

chmod

The **chmod** command assigns or changes permissions of a file. You are only allowed to change permissions if you are the owner of the file or directory.

Related commands: **chgrp**, **chown**.

Refer to Section 5.4.1, “Commands”.

Syntax.

```
chmod [-R] mode file...
```

Replace `mode` with the numerical value as described in Section 5.4.1, “Commands”. `file...` can be replaced by one or more files or directories. Use the `-R` switch to change permissions recursively in directories.

chown

Sets the owner of a file or directory. You are only allowed to change the owner if you are the owner of the file or directory.

Related commands: **chgrp**, **chmod**.

Refer to Section 5.4.1, “Commands”.

Syntax.

```
chown [-R] username file...
```

Replace `username` with the name of the owner. `file...` can be replaced by one or more files or directories. Use the `-R` switch to change the owner recursively in directories.

cp

Copy files. Be careful, **cp** does not warn about overwriting existing target files.

Related commands: **ls**, **mv**, **rm**.

Refer to Section 5.3, “File and Directory Commands”.

Syntax.

```
cp [-R] src... dest
```

cp copies the files specified as `src...` to the directory or file specified by `dest`. If you want to copy entire directories, use the `-R` option.

echo

echo can be used to print a message or the content of a shell variable to the screen.

fg

Resumes a suspended command in the foreground, or brings a process running in the background to the foreground. Available only in shells that are capable of job control.

Related commands: **bg**, **jobs**.

Refer to Section 6.2, “Job Control”.

Syntax.

```
fg [% jobnum]
```

Replace `jobnum` with the number of the suspended or in the background running job to put in foreground. See also **jobs**. If you omit the number, the last suspended or to background sent command will be put to the foreground.

grep

Grep searches the named input files (or standard input if no files are named, or the file name `-` is given) for lines containing a match to the given `patter`. By default, `grep` prints the matching lines.

Related commands: **cat**, **less**, **more**, **sort**.

Refer to Section 4.1, “Piping”, Section 7.4, “Search for Text in Files”.

Syntax.

```
grep [-n] pattern file...
```

The `pattern` may be a regular expression (not covered here) or a simple string. The `-n` option tells **grep** to print out the line number of matching line. Replace `file...` by one or more files to search. If omitted, **grep** expects the input from the standard input.

groups

The command **groups** prints on standard output the groups to which you belong.

Related commands: **id**.

Refer to Section 3, “Users”.

Syntax.

```
groups [user...]
```

In addition to plainly call **groups** without option, you can specify user names, for which the group membership should be printed.

id

If no user name is provided, the **id** utility writes the user and group IDs and the corresponding user and group names of the invoking process to standard output.

Related commands: **id**.

Refer to Section 3, “Users”

Syntax.

```
id [user]
```

If you don't specify an user name, your user and group id are printed. You can alternatively provide a user name as the `[user]` option, in which case you see the user and group id of that user.

jobs

When job control is enabled, **jobs** reports all jobs that are stopped or executing in the background.

Related commands: **bg**, **fg**.

Refer to Section 6.2, “Job Control”.

Syntax.

```
jobs
```

kill The **kill** utility sends a signal to the process or processes specified by each PID operand. You can only send signals to processes you own.

Related commands: **killall**, **pkill**.

Refer to Section 6.1, “Terminating Processes”.

Syntax.

```
kill [-KILL] pid
```

The `pid` is the PID of the processes to which the terminate signal is sent. If the `-KILL` option is provided, the kill signal instead of the terminate signal is sent.

killall The **killall** utility kills processes selected by name, as opposed to the selection by PID as done by **kill**. You can only send signals to processes you own.

Related commands: **kill**, **pkill**.

Refer to Section 6.1, “Terminating Processes”.

Syntax.

```
killall [-KILL] procname
```

The syntax is the same as with **kill**, but it takes a process name instead of a PID.

Availability. Linux, FreeBSD

less **Less** is a program similar to **more**, but which allows backward movement in the file as well as forward movement.

Related commands: **cat**, **more**.

Refer to Section 4.3, “Pagers”.

Syntax.

```
less [filename]
```

If `filename` is omitted, **less** expects the input from the standard input.

ln **Ln** creates links to files and directories.

Related commands: **cp**.

Refer to Section 5.5.2, “Links”.

Syntax.

```
ln -s file linkname
```

Replace `file` with the file or directory name of which you want to create the link to. `linkname` is the name of the link, as, for instance, seen by `ls`.

logout

Terminate a login shell.

Refer to Section 2.1, “The Shell”

Syntax.

```
logout
```

lp

The `lp` utility submits print requests to a printer.

Related commands: **cancel**, **lpq**, **lpr**, **lprm**, **lpstat**.

Refer to Section 7.3.2, “Printing under Linux and Sun Solaris”.

Syntax.

```
lp [-d dest] [file...]
```

Replace `file...` with the files to print. If omitted, it expects the input from the standard input. The `-d dest` option can be used to specify a printer other than the standard printer. If omitted, the standard printer is used.

Availability. Linux, Sun Solaris

lpq

The `lpq` utility displays the information about the contents of a print queue.

Related commands: **cancel**, **lp**, **lpr**, **lprm**, **lpstat**.

Refer to Section 7.3.1, “Printing under FreeBSD”

Syntax.

```
lpq [-P queue]
```

Prints the jobs in the print queue. You can specify the printer queue to list with the `-P queue` option.

Availability. FreeBSD

lpr

The `lpr` utility submits print requests to a printer.

Related commands: **cancel**, **lp**, **lpq**, **lprm**, **lpstat**.

Refer to Section 7.3.1, “Printing under FreeBSD”.

Syntax.

```
lpr [-P printer] [file...]
```

Replace `file...` with the files to print. If omitted, it expects the input from the standard input. The `-P printer` option can be used to specify a printer other than the standard printer. If omitted, the standard printer is used.

Availability. FreeBSD

lprm

The **lprm** utility removes print job from the print queue.

Related commands: **cancel**, **lp**, **lpq**, **lpr**, **lpstat**.

Refer to Section 7.3.1, “Printing under FreeBSD”.

Syntax.

```
lpr [-P printer]
```

Removes your last submitted print job from the print queue of the default printer. You can specify the print queue with `-P printer` option.

Availability. FreeBSD

lpstat

The **lpstat** utility displays information about the current status of the printer queue.

Related commands: **cancel**, **lp**, **lpq**, **lpr**, **lprm**.

Refer to Section 7.3.2, “Printing under Linux and Sun Solaris”.

Syntax.

```
lpstat
```

Availability. Linux, Sun Solaris

ls

For each file that is a directory, **ls** lists the contents of the directory. For each file that is an ordinary file, **ls** repeats its name and any other information requested.

Related commands:

Refer to Section 5.2, “The ls Command”.

Syntax.

```
ls [-l] [-a] [-d] [-h] [file...]
```

Called with no `file...` argument, **ls** lists the content of the current working directory. The `-l` option generates a long list containing the permissions, owner, group owner, size, last modified date, and file name. The `-a` make **ls** show also hidden files. The `-h` displays the file sizes in human readable format. The `-d` option advises **ls** to not dive into directories, but instead print the information of the directory in question.

man

The **man** command displays information from the reference manuals. It displays complete manual pages that you select by name.

Related commands: **whatis**.

Refer to Section 4.2, “Man Pages”.

Syntax. On FreeBSD and Linux:

```
man [secnum] command
```

On Sun Solaris

```
man [-s secnum] command
```

You can request the manual page from a specific section by providing the `secnum` option.

mkdir

The **mkdir** command creates the named directory.

Related commands: **rmdir**.

Refer to Section 5, “The File System”.

Syntax.

```
mkdir dirname
```

You can create only new directories in directories you have write access to the parent directory.

more

The **more** utility is a filter that displays the contents of a text file on the terminal, one screenful at a time. It pauses after each screenful. **more** then prints `--More--`.

Related commands: **less**.

Refer to Section 7.1, “Reading Text Files”.

Syntax.

```
more [file...]
```

If you omit the `file...` option, **more** expects the input from the standard input.

mv

The **mv** utility moves the file named by the source operand to the destination specified by the target. Source and target file may not have the same name. If target does not exist, **mv** creates a file named target. If the target file exists, its contents are overwritten. If the target file is a directory, the source is moved to that directory.

Related commands: **cp**, **rm**.

Refer to Section 5, “The File System”.

Syntax.

```
mv source... destination
```

passwd

The **passwd** command changes the password associated with the user's login name.

Related commands:

Refer to Section 3, “Users”.

Syntax.

```
passwd
```

pkill

The **pkill** utility kills processes selected by name, as opposed to the selection by PID as done by **kill**. You can only send signals to processes you own.

Related commands: **kill**, **killall**.

Refer to Section 6.1, “Terminating Processes”

Syntax.

```
pkill [-KILL] procname
```

The syntax is the same as with **kill**, but it takes a process name instead of a PID.

Availability. Sun Solaris, FreeBSD

prstat

The **prstat** utility iteratively examines all active processes on the system and reports statistics.

Related commands: **ps**, **top**.

Refer to Section 6, “Processes”.

Syntax.

```
prstat
```

Availability. Sun Solaris

ps

The **ps** command prints information about active processes.

Related commands: **prstat**, **top**.

Refer to Section 6, “Processes”.

Syntax. On Sun Solaris and Linux

```
ps [-e] [-f]
```

On FreeBSD

```
ps [a][x]
```

pwd

The **pwd** utility writes an absolute path name of the current working directory to standard output.

Related commands: **cd**.

Refer to Section 5, “The File System”

Syntax.

```
pwd
```

rm

The **rm** utility removes the directory entry specified by each file argument. You can only remove files if you have write access to their parent directory.

Related commands: **cp**, **mv**, **rmdir**.

Refer to Section 5, “The File System”.

Syntax.

```
rm [-r] file...
```

Removes the file(s) specified by the `file...` argument. You can recursively delete files and directories by providing the `-r` option. *Warning!* You must be careful using the `-r` switch of **rm**. It deletes everything in the directory specified including sub-directories without any confirmation question, such as known by Microsoft operating systems. So you once fired on the wrong directory, you have little chance of stopping it. Especially since the **rm** command deletes the files for real, and does not move them to some recycle bin or trashcan, where you can salvage the files from.

rmdir

The **rmdir** command removes empty directories.

Related commands: **mkdir**.

Refer to Section 5, “The File System”.

Syntax.

```
rmdir dirname
```

sort

The **sort** command sorts lines of all the named files together and writes the result on the standard output.

Related commands: **cat**, **grep**.

Refer to Section 4.1, “Piping”.

Syntax.

```
sort [file...]
```

If you omit `file . . .` **sort** expects the input from the standard input.

ssh

ssh (Secure Shell) is a program for logging into a remote machine. It provides a secure encrypted communications between two untrusted hosts over an insecure network. X11 connections can also be forwarded over the secure channel.

Related commands:

Refer to Section 2, “The Terminal”, Appendix A, *X Window System*.

Syntax.

```
ssh [-X] username@host
```

Replace `username` with your user name on the remote host and `host` with the remote host name. If X forwarding is enabled on the remote host, the `-X` enables the client to allow them.

su

The **su** command allows one to become another user without logging off.

Related commands:

Refer to Section 3, “Users”.

Syntax.

```
su [-] [username]
```

If the `username` is omitted, the default user is `root`. The `-` makes the user switch behave like a login.

top

Top displays the processes on the system and periodically updates this information.

Related commands: **ps**, **prstat**

Refer to Section 6, “Processes”.

Syntax. On Linux

```
top [-d sec]
```

On FreeBSD

```
top [-s sec]
```

The `-d` or `-s` take the update frequency in seconds.

Availability. Linux, FreeBSD

whatis

Whatis looks up a given command and displays the header line from the manual section.

Related commands: **man**, **which**.

Refer to Section 4.2, “Man Pages”.

Syntax.

```
whatis command
```

which

Which takes a list of names and looks for the files which would be executed had these names been given as commands. Each argument is searched for along the user's `PATH` environment variable.

Related commands: **whatis**.

Refer to Section 4, “Commands”.

Syntax.

```
which file...
```

who

The **who** command lists the logged in users on the system.

Related commands:

Refer to Section 3, “Users”.

Syntax.

9. References

\$Id: references.xml 1413 2009-01-09 21:29:27Z rafi \$

9.1. FreeBSD

- FreeBSD web site [<http://www.freebsd.org>]
- The FreeBSD Handbook [<http://www.freebsd.org/doc/en/books/handbook/>]

9.2. Sun Solaris

- Sun Solaris web site [<http://www.sun.com/software/solaris/>]
- Solaris 10 User Collection [<http://docs.sun.com/app/docs/coll/8.45?l=en>]

9.3. Linux

- The Linux Documentation Project [<http://www.tldp.org>]
- Debian Linux distribution [<http://www.debian.org>]
- The Debian Reference [<http://www.debian.org/doc/manuals/reference/index.en.html>]
- Ubuntu Linux distribution [<http://www.ubuntu.com>].
- Ubuntu Help [<https://help.ubuntu.com/>].
- CentOS Linux distribution [<http://www.centos.org/>]

9.4. Miscellaneous

- Unix History [<http://www.levenez.com/unix/>]
- UNIX Tutorial for Beginners [<http://www.ee.surrey.ac.uk/Teaching/Unix/>]

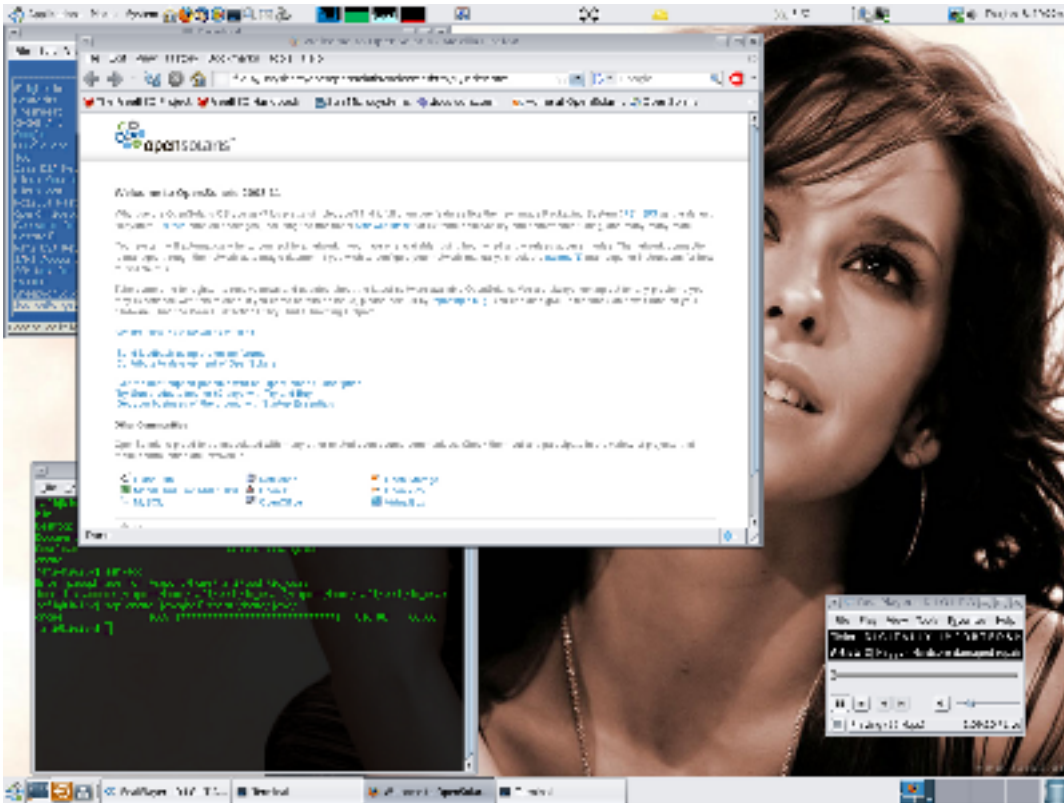
A. X Window System

The *X Window System* is the base of most graphical user interfaces (GUI) on UNIX systems. It provides the facilities used by *window managers* such as WindowMaker, Fvwm, etc.

The window manager is responsible for drawing window decorations and make windows resizable and movable. Without a window manager, you wouldn't be able to position your windows on the screen.

Desktop systems such as GNOME and KDE have their own window managers along with additional utilities and tools. See Figure A.1, “Screenshot of GNOME” for a screenshot of GNOME running on a OpenSolaris box (OpenSolaris is an offspring of Sun Solaris).

Figure A.1. Screenshot of GNOME



A screenshot of a GNOME session on a OpenSolaris box.

UNIX enables you to start an X application on a remote host and display the window of the application on your local host. The prerequisites for doing so are

1. the remote host has the application in question installed.
2. an ssh server is running on the remote host allowing X forwarding.
3. you are logged into an X session on your local host and have an ssh client installed.

Given the above prerequisites are met, you can use the `ssh` command as shown in Example A.1, “Using ssh to start X applications on a remote host”.

Example A.1. Using ssh to start X applications on a remote host

```
$ ssh -X joe@salma.example.org
Password:
$ xterm
```

In the above example, you would start an **xterm** (a terminal in a window). Replace `joe@salma.example.org` with your user name and the corresponding host. The `-X` tells the ssh client to accept X forwarding.

B. GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the

Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled

"Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive

"Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.