

Session #2 Part 2

The Manual Pages, Links, & Managing Directories and Files



Summary

1. Man Pages
 - 1.1. What are Man pages?
 - 1.2. Searching for a Command
2. Managing Directories and Files
 - 2.1. File Extensions
 - 2.2. Creating Directories
 - 2.3. Creating Files
 - 2.4. Renaming and Moving Files
 - 2.5. Copying Files
 - 2.6. Deleting Files and Directories
3. Links
 - 3.1. What are Links?
 - 3.2. Soft "Symbolic" Links
 - 3.3. Hard Links
 - 3.4. Deleting Links

Man Pages

What are Man Pages ?

It stands for **manual pages**.

They're a set of pages that explain what every command on the system does, what options are available, what arguments it can take, and shows you how to use them.

To open a man page type: `man [COMMAND NAME]`

Example: `man man`

This will show you the manual of the “man” command.

You can use the arrow keys to navigate the pages and you can hit **q** to quit or **h** for help.

```
MAN(1)                                Manual pager utils
MAN(1)
NAME
    man - an interface to the on-line reference manuals

SYNOPSIS
    man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-M system[,...]] [-M path]
    [-S list] [-e extension] [-i|-I]
    [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [
    --no-hyphenation] [--no-justifica-
    tion] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page ...] ...
    man -k [apropos options] regexp ...
    man -K [-w|-W] [-S list] [-t|-I] [--regex] [section] term ...
    man -f [whatIs options] page ...
    man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [
    -7] [-E encoding] [-p string] [-t]
    [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
    man -w|-W [-C file] [-d] [-D] page ...
    man -c [-C file] [-d] [-D] page ...
    man [-?V]

DESCRIPTION
    man is the system's manual pager. Each page argument given to man is normally the name of a program, uti-
    lity or function. The manual page
    associated with each of these arguments is then found and displayed. A section, if provided, will direc-
    t man to look only in that section
    of the manual. The default action is to search in all of the available sections following a pre-defined
    order ("1 n l 8 3 2 3posix 3pm
    3perl 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), and
    to show only the first page found,
    even if page exists in several sections.

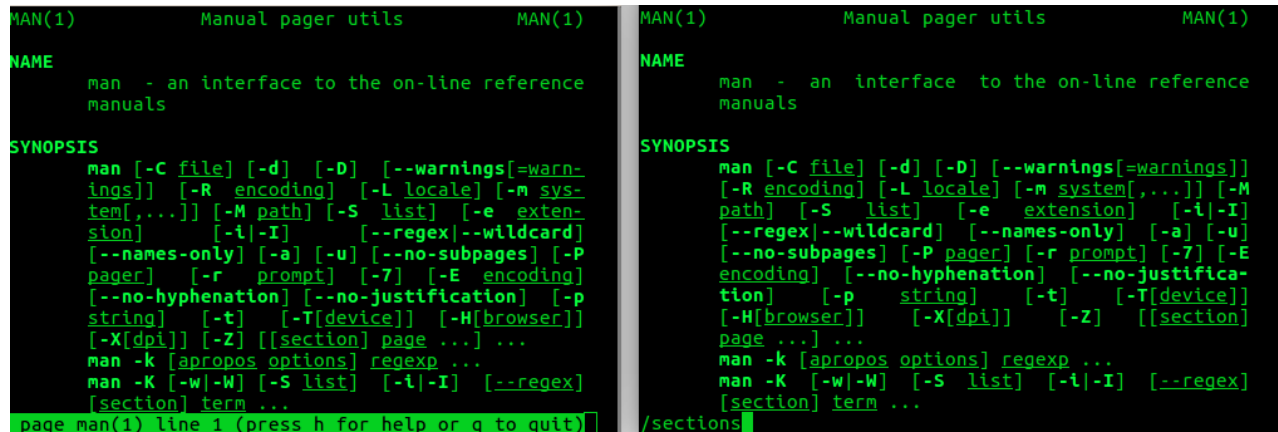
    The table below shows the section numbers of the manual followed by the types of pages they contain.

    1 Executable programs or shell commands
    2 System calls (functions provided by the kernel)
    3 Library calls (functions within program libraries)
    4 Special files (usually found in /dev)
    5 File formats and conventions eg /etc/passwd
    6 Games

Manual page man(1) line 1 (press h for help or q to quit)
```

If you want to search for a specific word you can press slash (/) then type in the keyword and then press enter.

Example: `/sections` will make you search for “sections”.



```
MAN(1) Manual pager utils MAN(1)
NAME
man - an interface to the on-line reference manuals

SYNOPSIS
man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...

page man(1) line 1 (press h for help or q to quit)

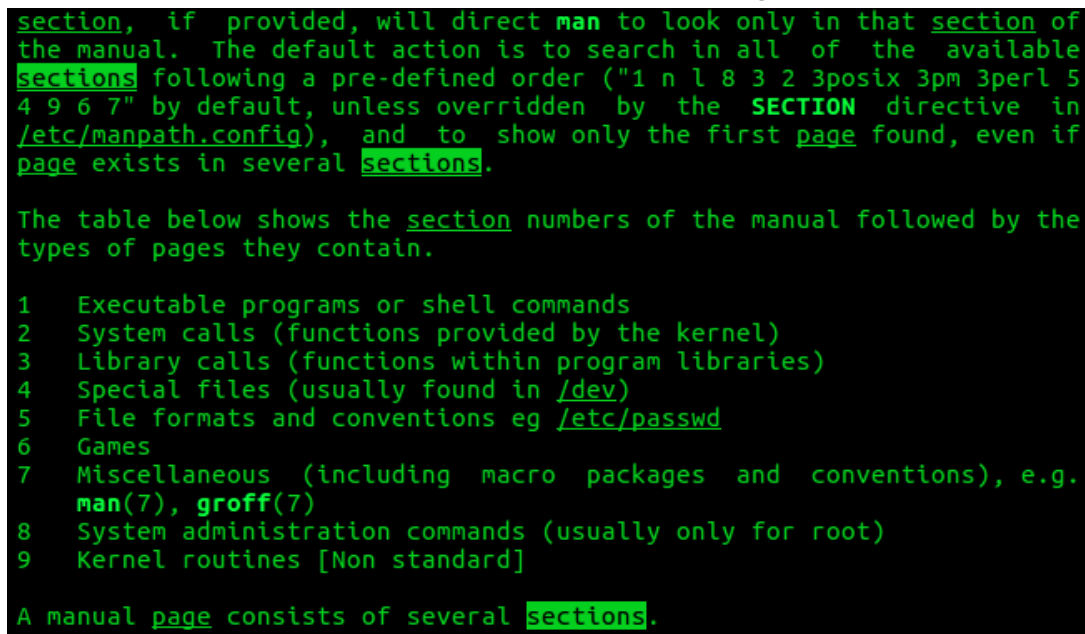
MAN(1) Manual pager utils MAN(1)
NAME
man - an interface to the on-line reference manuals

SYNOPSIS
man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...

/sections
```

Result:

This indeed does bring up the first occurrence of the word “sections”. You can learn more about sections from this image.



```
section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order ("1 n l 8 3 2 3posix 3pm 3perl 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), and to show only the first page found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

A manual page consists of several sections.
```

Note: You can also use `info` and `--help` to get information about a command.

Examples: `info ls`

`ls -help`

Searching for a Command

If you need to do a specific task but don't know which command can do it:

→ You can use **apropos** or **man -k** to search for that command by the keyword of the command.

Example:

You want to rename a file but you don't know what command does that:

```
osc@Ubuntu:~$ apropos "rename"
dpkg-name (1)      - rename Debian packages to full...
file-rename (1p)   - renames multiple files
File::Rename (3pm) - Perl extension for renaming mu...
git-mv (1)         - Move or rename a file, a direc...
gvfs-rename (1)    - Rename a file
mmove (1)          - move or rename an MSDOS file o...
mren (1)           - rename an existing MSDOS file
mv (1)             - move (rename) files
prename (1)        - renames multiple files
rename (1)         - renames multiple files
rename (2)         - change the name or location of...
rename.ul (1)      - rename files
renameat (2)       - change the name or location of...
renameat2 (2)      - change the name or location of...
XStoreName (3)     - set or read a window's WM_NAME...
XStoreNamedColor (3) - set colors
zipnote (1)        - write the comments in zipfile ...
osc@Ubuntu:~$

osc@Ubuntu:~$ man -k "rename"
dpkg-name (1)      - rename Debian packages to full ...
file-rename (1p)   - renames multiple files
File::Rename (3pm) - Perl extension for renaming mul...
git-mv (1)         - Move or rename a file, a direct...
gvfs-rename (1)    - Rename a file
mmove (1)          - move or rename an MSDOS file or...
mren (1)           - rename an existing MSDOS file
mv (1)             - move (rename) files
prename (1)        - renames multiple files
rename (1)         - renames multiple files
rename (2)         - change the name or location of ...
rename.ul (1)      - rename files
renameat (2)       - change the name or location of ...
renameat2 (2)      - change the name or location of ...
XStoreName (3)     - set or read a window's WM_NAME ...
XStoreNamedColor (3) - set colors
zipnote (1)        - write the comments in zipfile t...
osc@Ubuntu:~$
```

As you can see, all of these are commands that can rename something.

Which one you choose is up to you, you can use man pages to know more about these commands and select the suitable one for your case.

→ You can use **what is** or **man -f** to quickly see what a command does

Example:

```
satharus@Argon:~$ whatis ls
ls (1)      - list directory contents
ls (1p)     - list directory contents
satharus@Argon:~$ whatis mkdir
mkdir (2)   - create a directory
mkdir (1)   - make directories
mkdir (1p)  - make directories
mkdir (3p)  - make a directory relative to directory file descriptor
satharus@Argon:~$ man -f ls
ls (1)      - list directory contents
ls (1p)     - list directory contents
satharus@Argon:~$ man -f mkdir
mkdir (2)   - create a directory
mkdir (1)   - make directories
mkdir (1p)  - make directories
mkdir (3p)  - make a directory relative to directory file descriptor
```

Managing Directories and Files

File Extensions

A file extension is the ending of a file's name that helps the operating system and the user know the kind of file it is and what program should run to open this file.

By default Linux has 3 types of files:

1. Regular File (-):
 - Readable file (.txt, .cpp)
 - Binary file (.exe)
 - Image file (.png, .jpg)
 - Archive or "Compressed" file (.zip, .rar, .doc, .pdf)
2. Directory (d): A folder containing other files or folders
3. Special
 - Block File (b): Hardware files (Like some files under /dev/)
 - Soft "Symbolic" Link File (l): File pointing to another file (shortcut)

Creating Directories

You now know how to use man. Can you search for a command that creates a directory?

Solution: `man -k "make dir"` and the command is: `mkdir`

Example:

```
osc@Ubuntu:~$ mkdir linux
```

```
osc@Ubuntu:~$ mkdir "linux workshop"
```

If you want to create more than one directory at a time you can do the following:

Note: We use double quotes " " if the name of the directory has more than one word.
This is to avoid making the shell interpret the 2 words as 2 separate arguments

```
osc@Ubuntu:~$ mkdir one two three "i created three dirs"
```

Let's check:

```
osc@Ubuntu:~$ ls -l
total 28
-rw-r--r-- 1 osc osc 8980 فون 9 01:03 examples.desktop
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 i created three dirs
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 one
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 three
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 two
osc@Ubuntu:~$
```

Creating Files

You can use `touch` to create a file, like `mkdir` you can pass as many arguments to it and it'll create the files for you.

Example:

You want to create 2 files:

```
osc@Ubuntu:~$ touch file1 file2
osc@Ubuntu:~$ ls -l
total 28
-rw-r--r-- 1 osc osc 8980 فون 9 01:03 examples.desktop
-rw-rw-r-- 1 osc osc 0 فون 9 15:28 file1
-rw-rw-r-- 1 osc osc 0 فون 9 15:28 file2
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 i created three dirs
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 one
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 three
drwxrwxr-x 2 osc osc 4096 فون 9 15:18 two
osc@Ubuntu:~$
```

Copying Files

To copy files you can use `cp`

Example:

To copy a file, the following syntax can be used:

```
cp <original file> <copy file>
```

```
satharus@Argon:~/Workshop$ ls
Hello.txt
satharus@Argon:~/Workshop$ cat Hello.txt
Hello!
This is the second session

Good luck and have fun (^_^)
satharus@Argon:~/Workshop$ cp Hello.txt Copy.txt
satharus@Argon:~/Workshop$ ls
Copy.txt Hello.txt
satharus@Argon:~/Workshop$ cat Copy.txt
Hello!
This is the second session

Good luck and have fun (^_^)
satharus@Argon:~/Workshop$ |
```

Renaming and Moving Files

To rename a file use `mv`

Example:

Create a file that is called file1 and rename it to textfile.

```
osc@Ubuntu:~/one$ touch file1
osc@Ubuntu:~/one$ ls
file1
osc@Ubuntu:~/one$ mv file1 textfile
osc@Ubuntu:~/one$ ls
textfile
```

To move “cut” a file use `mv`

Example:

Moving “textfile” from ~/one) to ~/two.

```
osc@Ubuntu:~/one$ ls
textfile
osc@Ubuntu:~/one$ pwd
/home/osc/one
osc@Ubuntu:~/one$ mv textfile /home/osc/two/
osc@Ubuntu:~/one$ ls
osc@Ubuntu:~/one$ cd /home/osc/two/
osc@Ubuntu:~/two$ ls
textfile
osc@Ubuntu:~/two$
```

Deleting Files and Directories

You can delete a file or a directory using the `rm` command.

Try to figure it out from the man pages on your own before checking the next page.

Find how to do the following:

- Delete a file.
- Delete an empty directory.
- Delete a non empty directory.
- Delete but don't act until the user confirms.
- Delete by force and don't prompt the user

Solution:

```
✓ rm filename
✓ rm -r directory_name or rmdir directory_name
✓ rm -r directory_name
✓ rm -i filename
✓ rm -f filename
```

Links

What are Links?

A link in Linux is a file that points to another file/directory. Creating links is similar to creating shortcuts. A file can have multiple links linked to it. But a link can only be linked to (pointed to) one file.

There are two types of links:

1. Soft or Symbolic links.
2. Hard links.

Note: You can think of links like pointers in programming languages, if you're familiar with them.

These links behave differently when the source of the link (what is being linked to) is moved or removed.

- Symbolic links are not updated, and become “hanging links”.
- Hard links always refer to the source, even if moved or removed.

For example: We have a file A.txt. If we create a soft link and a hard link both pointing to it and then delete A.txt, the result is visible in the opposite figure.

We can simply say that a soft link is just a file that points to another (shortcut), while a hard link is a copy of a file that is always synchronised with it.

```
satharus@Argon: ~/Workshop$ ls
A.txt  hardlink  softlink
satharus@Argon: ~/Workshop$ cat A.txt
OSC
satharus@Argon: ~/Workshop$ cat hardlink
OSC
satharus@Argon: ~/Workshop$ cat softlink
OSC
satharus@Argon: ~/Workshop$ rm A.txt
satharus@Argon: ~/Workshop$ cat hardlink
OSC
satharus@Argon: ~/Workshop$ cat softlink
cat: softlink: No such file or directory
satharus@Argon: ~/Workshop$ ls
hardlink  softlink
satharus@Argon: ~/Workshop$ |
```


Soft “Symbolic” Links

- A soft link is similar to the file shortcut feature which is used in Windows operating systems.
- Soft links can be linked across different filesystems, although if the original file is deleted or moved, the soft link will not work correctly and is referred to as a “hanging link”.
- Soft links contain the path for original file but not the content.
- A soft link can link to a directory.

Example:

Make a link using the `ln` command.

The option `-s` makes the link a soft link and not a hard one.

`ln -s Filename Linkname` : Make a soft link

```
satharus@Argon: ~/Workshop$ touch File.txt
satharus@Argon: ~/Workshop$ ln -s File.txt S_Link
satharus@Argon: ~/Workshop$ ls -l
total 0
-rw-r--r-- 1 satharus users 0 Nov  9 19:40 File.txt
lrwxrwxrwx 1 satharus users 8 Nov  9 19:41 S_Link -> File.txt
satharus@Argon: ~/Workshop$ |
```

You can see that indeed, S_Link is pointing to File.txt.

Hard Links

- A hard link is similar to creating a copy that is always synced with the original file.
- Hard links can't be made across different file systems. But, if the original file is deleted or moved, the hard link will still work.
- A hard link can't link to a directory.

Example:

Make a hard link using the `ln` command.

`ln Filename Linkname`

In this example you can see that H_Link is treated as a normal file, it is just linked to File.txt. After editing File.txt, H_Link was edited too.

It is exactly like a synced copy of the file.

```
satharus@Argon: ~/Workshop$ ls
File.txt
satharus@Argon: ~/Workshop$ cat File.txt
Linking is cool!
satharus@Argon: ~/Workshop$ ln File.txt H_Link
satharus@Argon: ~/Workshop$ ls
File.txt  H_Link
satharus@Argon: ~/Workshop$ cat H_Link
Linking is cool!
satharus@Argon: ~/Workshop$ ls -l
total 8
-rw-r--r-- 2 satharus users 17 Nov  9 19:46 File.txt
-rw-r--r-- 2 satharus users 17 Nov  9 19:46 H_Link
satharus@Argon: ~/Workshop$ echo "Editing" >> File.txt
satharus@Argon: ~/Workshop$ cat File.txt
Linking is cool!
Editing
satharus@Argon: ~/Workshop$ cat H_Link
Linking is cool!
Editing
satharus@Argon: ~/Workshop$ |
```

Deleting Links

To delete a link you can use `unlink` or `rm`

Example:

`unlink linkname`

```
satharus@Argon: ~/Workshop$ ln File.txt H_Link
satharus@Argon: ~/Workshop$ ls
File.txt  H_Link
satharus@Argon: ~/Workshop$ unlink H_Link
satharus@Argon: ~/Workshop$ ls
File.txt
satharus@Argon: ~/Workshop$ |
```

And since a link is a file after all, you can also delete it with `rm`.

Example:

`rm linkname`

```
satharus@Argon: ~/Workshop$ ln File.txt H_Link
satharus@Argon: ~/Workshop$ ls
File.txt  H_Link
satharus@Argon: ~/Workshop$ rm H_Link
satharus@Argon: ~/Workshop$ ls
File.txt
satharus@Argon: ~/Workshop$ |
```