

LPIC-1 100-500 – Lesson 3

103.3 Perform basic file management



Introduction

- The UNIX philosophy:

”Everything is a file!”

- The above statement declares that all objects and structures in the UNIX world, exists in the form of files, or more correctly, file descriptors.



Copy files with `cp`

- `$ cp file1 file1.bak # backup file1`
- `$ cp ../file1 . # copy file1 from parent directory to current directory`

Options:

- `-a` # archive (preserve file attributes)
- `-f` # force overwrite destination file, if exists
- `-r, -R` # copy recursively including sub-directories
- `-i` # interactively confirm if you want the destination file to be overwritten

Create directories with `mkdir`

- `$ mkdir dir1 dir2 # create directories dir1 and dir2`

Options:

- `-p # create hierarchy of directories, e.g.:`
- `-m 750 # create directory with permission 750 (octal)`



Move/Rename with `mv`

- `$ mv name1 name2 # rename name1 to name2`
- `$ mv ../name1 . # move name1 from the parent directory to the current`
- `$ mv /tmp/name1 ~/name2 # move name1 to your home directory and rename to name2`

Options:

- `-f # force overwrite destination file, if exists`
- `-i # interactively confirm if you want the destination file to be overwritten`

Display files/directories with `ls`

- `$ ls` # Display files/directories in the current directory
 - `$ ls -la dir1` # detailed view of normal and hidden files and directories in the `dir1` directory
 - `$ ls -la .bashrc` # whatever starts with "." is a hidden file
- ```
-rw----- 1 theo theo 3353 2011-04-29 13:29 .bashrc
```



# Display files/directories with `ls`

- `$ ls -la .bashrc`

- `-rw----- 1 theo theo 3353 2011-04-29 13:29 .bashrc`

Annotations for the `ls` output:

- `^` points to each field in the output line.
- ``file type (d for directories) and permissions` points to the first field.
- ``number of hard links` points to the second field.
- ``owner` points to the third field.
- ``group` points to the fourth field.
- ``file size in bytes` points to the fifth field.
- ``datetime` points to the sixth field.
- ``filename` points to the seventh field.



# Display files/directories with `ls`

- `drwxr-xr-x 144 root root 12288 2011-08-22 17:21 etc`

^

**`the first character of the sequence, declares the file type`**

| Code | Description                                                                                                            |
|------|------------------------------------------------------------------------------------------------------------------------|
| -    | Regular file                                                                                                           |
| d    | Directory                                                                                                              |
| l    | Symbolic Link: a file pointing to another file                                                                         |
| p    | Named pipe: used in inter process communication                                                                        |
| s    | Socket: used in network inter-communication                                                                            |
| b    | Block Device: files that represent devices where data flows in blocks larger than a byte, e.g. Hard Disks, CD-ROM, etc |
| c    | Character Device: files that represent devices where data flows in one byte at a time, e.g. terminals, I/O ports, etc  |

# Display files/directories with `ls`

## Options:

- `-l` # display file in the long listing format
- `-a, --all` # display hidden files as well as normal
- `-R` # recursive listing of files/directories
- `-h` # display size in human readable format, e.g. 3K 24M, 2.3G
- `-d` # display information about directories instead of the content of directories

# Display files/directories with `ls`

- `-F` # display in this format:  
    `"*/=>@|"`
  - no symbol is for normal files
  - \* Executable
  - / Directory
  - @ Symbolic Link
  - = Socket
  - | Pipe



# Delete files with `rm`

- `$ rm file1 file2 # delete (definitively!)  
file file1 and file2`

## Options:

- `-d # delete directories when empty`
- `-f # enforced, non-interactive deletion of files and directories`
- `-i # interactively warn the user about the deleted files or directories`
- `-r, -R # recursively delete files or directories`

**WARNING! Never try this at home (or at work):**

**`rm -rf / # deletes everything!`**

# Delete directories with `rmdir`

- `$ rmdir dir1 # delete empty directory dir1`

## Options:

- `-p # delete parent and child directories, provided they are empty`



# Show file status with `stat`

- `$ stat .bash_history` # shows useful information for files

```
File: .bash_history
Size: 433956 Blocks: 848 IO Block:
4096 regular file
Device: fd01h/64769d Inode: 3932171 Links: 1
Access: (0600/-rw-----) Uid: (1000/theo)
Gid: (1000/theo)
Access: 2018-06-23 08:24:41.811736750 +0300
Modify: 2018-06-22 21:56:36.709083485 +0300
Change: 2018-06-23 08:24:41.811736750 +0300
```

# Access: Last Access time

# Modify: Last modification of file content

# Change: Last modification of file attributes

# Change timestamps of files with `touch`

- `$ touch .bash_history # change datetime with current. As a side-effect it creates an empty file if the filename does not exist`

## Options:

- `-a # change only the access time`
  - `-m # change only the modified time`
  - `-t 200302141625 # use different timestamp than current. The timestamp format is [[CC]YY]MMDDhhmm[.ss]`
- 

# Find file type with `file`

- `$ file /bin/bash # check the type of the bash file`
- `$ file /etc/fstab`
- `$ file /dev/cdrom`
- `$ file /dev/sr0`

*Note: the file extensions in Linux are optional and not indicative of the actual file type. The file type is determined by analyzing the file data.*



# Process raw data with `dd`

- `$ dd if=/dev/sda  
of=/media/external/disk.img # clone the  
disk /dev/sda to image file disk.img`

## Options:

- `conv=lower` # convert to lower case
- `bs=1024` # set block size to 1024 bytes
- `count=3000` # set the number of blocks the process should last

*Note: if you set the wrong outfile (of) you can permanently loose all data on the destination device. Use with caution!*



# Find files with `find`

- `$ find /etc # find all files under /etc`
- `$ find / -name fstab # find file fstab under the root directory "/"`
- `$ find /etc -name "*.conf" # find all ending in .conf under /etc`



# Find files with `find`

- `$ find /etc -size +4k # find files bigger than 4 kB`
- `$ find /usr -size -64M # find files smaller than 64 MB`
- `$ find /tmp -size +2k -size -4k # find files between 2kB and 4 kB`
- `$ find /usr -size 6k # find files between 5.1 kB and 6 kB`



# Find files with `find`

- `$ find /usr -type f # find all normal files under /usr`

## Options:

- `-type b # find block devices`
  - `-type c # find character devices`
  - `-type d # find directories`
  - `-type p # find named pipes`
  - `-type l # find symbolic links`
  - `-type s # find sockets`
- 

# Find files with `find`

- `$ find ~ -atime 3 # find files accessed 3 ago`
- `$ find ~ -mtime +3 # find files modified 4 or more days ago`
- `$ find ~ -ctime -3 # find files which status changed 4 or more days ago`



# Find files with `find`

- `$ find /tmp -size -4k -ls #`  
execute `ls -l` on all found files
- `$ find /usr/share -type f -exec`  
`file {} \;` # run the `file` command  
on all regular files
- `$ find /tmp -ctime +15 -delete #`  
delete files older than 15 days



# Compress/Decompress files with `gzip` and `gunzip`

- `$ gzip movie.mpg # create a compressed file movie.mpg.gz`
- `$ gunzip movie.mpg.gz # decompress the compressed file to movie.mpg`

## Options:

- `gzip -d # identical gunzip`
- `-r # recursive compression/decompression when dealing with directories`



# Compress/Decompress files with `bzip2` and `bunzip2`

- `bzip2` is considered a more efficient compression method than `gzip`
- `$ bzip2 movie.mpg # create compressed archive movie.mpg.bz2`
- `$ bunzip2 movie.mpg.bz2 # decompress to movie.mpg`

## Options:

- `bzip2 -d # identical to bunzip2`
- `-1 .. -9 # -1 compresses faster but less efficiently and -9 has a higher compression ration but slow. Default is -5`

# Compress/Decompress files with `xz` and `unxz`

- `xz` has an even higher compression ration than either `bzip2` or `gzip`
- `$ xz movie.mpg # create compressed archive movie.mpg.xz`
- `$ unxz movie.mpg.xz # decompress to movie.mpg`

## Options:

- `xz -d # identical to unxz`
- `-1 .. -9 # -1 compresses faster but less efficiently and -9 has a higher compression ration but slow. Default is -5`

# Archiving with `cpio`

- `$ ls . | cpio -ov > dir1.cpio # archive the contents of current directory to dir1.cpio`
- `$ find ~ -mtime +365 | cpio -o > old.cpio # archive files older than a year`
- `$ cpio -iv < dir1.cpio # extract data from the dir1.cpio to current directory`

## Options:

- `-o` # create archive
  - `-i` # extract from archive
  - `-v` # verbose display of data
- 

# Archiving with `tar`

- `$ tar cvf /media/external/backup.tar /home/user # archive home directory to backup.tar`
- `$ tar xvf archive.tar # extract data from archive.tar to current directory`
- `$ tar xvf archive.tar -C dir1 # extract data from archive.tar to directory dir1`



# Archiving with `tar`

- `$ tar cvzf /media/external/backup.tar.gz ~`  
# archive home directory and compress using `gzip` (`backup.tgz` is another alternative extension)
- `$ tar cvf /media/external/backup.tar ~ ;`  
`gzip backup.tar` # equivalent to the command above
- `$ tar cvjf /media/external/backup.tar.bz2 ~`  
# archiving and compression using `bzip2` (`backup.tbz2` is an alternative extension)
- `$ tar cvJf /media/external/backup.tar.xz ~`  
# archiving and compression using `xz` (`backup.txz` is an alternative extension)

# Archiving with `tar`

- `$ tar xvzf /media/external/backup.tar.gz #`  
extract and uncompress with `gzip` of `backup.tar.gz` to current directory
- `$ gunzip /media/external/backup.tar.gz ;`  
`tar xvf backup.tar # equivalent to above`
- `$ tar xvjf /media/external/backup.tar.bz2`  
`-C data # extract and uncompress with`  
`bzip2 of backup.tar.bz2 to the data`  
directory
- `$ tar xvJf /media/external/backup.tar.xz`  
`-C data # extract and uncompress with xz`  
of `backup.tar.xz` to the data directory

# Archiving with `tar`

- `$ tar tvzf backup.tar.gz # show contents of backup.tar.gz`

## Options (dashes are optional)

- `-c` # create archive
  - `-x` # extract archive `tar`
  - `-t` # display contents of archive
  - `-v` # verbose output
  - `-z` # use `gzip` to (de)compress
  - `-j` # use `bzip2` to (de)compress
  - `-J` # use `xz` to (de)compress
- 

# Backup to a tape drive with `tar`

- `$ tar --one-file-system cf /dev/st0 / #`  
backup the root directory to the magnetic tape drive `/dev/st0` without leaving the `"/"` filesystem
- `$ tar xf /dev/st0 -C / #` recover the data from the tape to the root directory

"Nobody cares if you can backup, only if you can restore"  
~ Ancient UNIX Proverb ~

"Only wimps use tape backup: *real* men just upload their important stuff on ftp, and let the rest of the world mirror it!"  
~ Linus Torvalds ~



# File Globbing

- The Shell has the option of matching File Names using **wildcards**
  - If we want to use the wildcard characters literally they have to be embraced in " " or ' ' or be **'escaped'** using '\'
  - The difference between double quotes (" ") and single quotes (' ') is that double quotes return the value of shell/environment variables while single quotes interpret those literally
- 

# File Globbing

| Wildcard                             | Description                                                                    |
|--------------------------------------|--------------------------------------------------------------------------------|
| <b>*</b>                             | Match 0 or more characters                                                     |
| <b>?</b>                             | Match exactly one character                                                    |
| <b>[char]</b>                        | Match exactly one character, to the characters embraced in square brackets     |
| <b>[!char]</b>                       | Match exactly one character, to the characters NOT embraced in square brackets |
| <b>[a-z]</b>                         | Match exactly one character, to the characters from a to z (lower case)        |
| <b>[!a-z]</b>                        | Match exactly one character, NOT to the characters from a to z (lower case)    |
| <b>{string1,string2,string3,...}</b> | Match a string with one of the strings embraced in curly brackets              |

# File Globbing

- `$ ls /etc/*.co*` # matches files names containing `.co`
- `$ cp /etc/*.c? .` # copy all files which their extension start with `c` and followed by any single character e.g. `/etc/bogofilter.cf`
- `$ ls -ld .??*` # display all hidden files with at least two characters in their name



# File Globbing

- `$ mkdir dir with space #` this will create three different directories
- `$ rmdir dir with space #` remove three different directories
- `$ mkdir "dir with space" #` create a directory with spaces in its name
- `$ rmdir dir\ with\ space #` backslash “\” “escapes” and so the whole expression references the directory
- `$ rmdir 'dir with space' #` remove a directory with spaces in its name

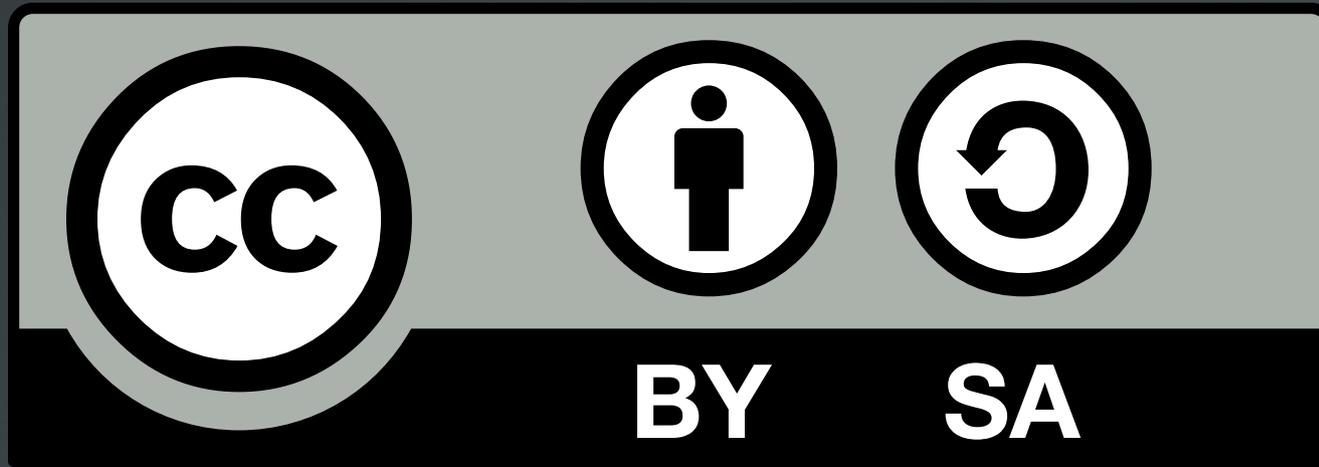


# File Globbing

- `$ touch semicolon\; # create file semicolon; (the ';' is part of the name)`
- `$ rm semicolon\; # delete file semicolon;`
- `$ touch backslash\\ # create file backslash\`
- `$ rm backslash\\ # delete file backslash\`
- `$ echo "my home is $HOME" # print my home is /home/user`
- `$ echo 'my home is $HOME' # print my home is $HOME`



# License



The work titled "LPIC-1 101-500 – Lesson 3" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

