Overview

- Introduction – Linux philosophy
- Command line basics – getting help
- The shell revisited: some features
- Navigating the file system
- File manipulation
  - Text editing
  - Various commands
  - Archiving
  - Groups, users, security
- Process control

Text editing

Text editor

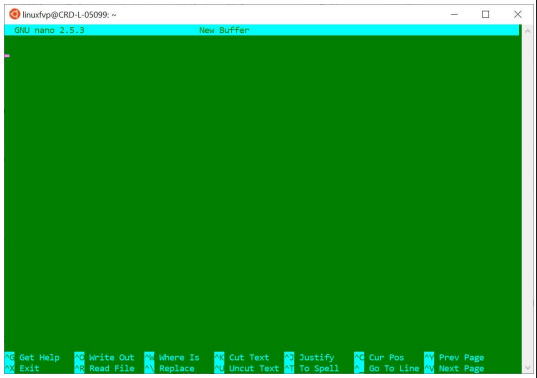
- Tool to create and edit files.
- There is no best text editor; it depends on personal taste.
- Text-only text editors
  - Simplicity first:
    - `nano`
  - With a steep learning curve: (needed for sysadmins and great for power users)
    - `vi, vim`
    - `emacs`
- Graphical text editors
  - `gedit`: general purpose GUI based text editor

Primitive text editing

- Combine Redirection and Viewing
- use `cat` to direct stdin to a text file
  - `$ cat > my_text.txt`
  - Enter text.
  - To end the text input, press Ctrl-D.
- Check with `cat my_text.txt`
- Try adding another line of text to the existing file
  - `$ cat >> my_text.txt`

nano

- Entering text: nano is a "modeless" editor. This means that all keystrokes, with the exception of Control and Meta sequences, enter text into the file being edited.
- Commands (lower part) are given by using the Control key (Ctrl, shown as ^) or the Meta key (Alt or Cmd, shown as M-).
  - A control-key sequence is entered by holding down the Ctrl key and pressing the desired key.
  - A meta-key sequence is entered by holding down the Meta key (normally the Alt key) and pressing the desired key.
- Manual: https://www.nano-editor.org/dist/v4/nano.pdf
- Cheat sheet: https://www.nano-editor.org/dist/latest/cheatsheet.html



vi

- Text-mode text editor available in all Linux systems.
- Created before computers with mice appeared.
- Very productive for power users.
- Check the web for tutorials:
  - https://upload.wikimedia.org/wikipedia/commons/d/d2/Learning_the_vi_Editor.pdf
  - ftp://ftp.vim.org/pub/vim/doc/book/vimbook-OPL.pdf

vi

- 2 basic modes of operation:
  - *command mode* and *editing mode*.
  - Within Command Mode, signals from the terminal are interpreted as editing commands.
  - Editing mode: letters typed at the keyboard are inserted into the editing buffer.
- Pressing **Esc** on the keyboard activates command mode.

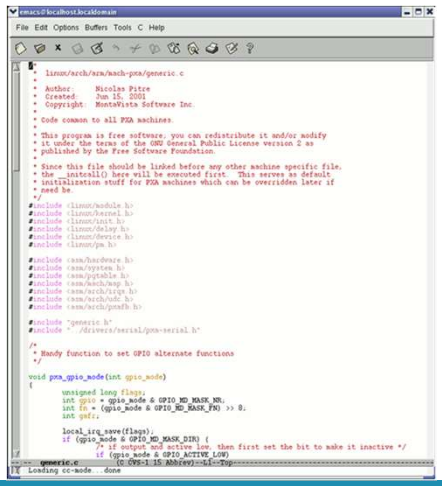
Key(s)	Function	Key(s)	Function
:w	Save	A	Append text after
:x	Save and exit	r	Replace text before cursor
:q	Quit	R	Replace text after cursor
I	Insert text after	i	Insert text before
P	Paste copied text	yy	Copy current line
a	Append text before	/[TEXT]	Search for the specified text

vi

- 2 modes
  - Input mode
    - ESC to back to cmd mode
  - Command mode
    - Cursor movement
      - h (left), j (down), k (up), l (right)
      - ^f (page down)
      - ^b (page up)
      - ^ (first char.)
      - $ (last char.)
      - G (bottom page)
      - :1 (goto first line)
    - Switch to input mode
      - a (append)
      - i (insert)
      - o (insert line after
      - O (insert line before)
- Delete
  - dd (delete a line)
  - d10d (delete 10 lines)
  - d$ (delete till end of line)
  - dG (delete till end of file)
  - x (current char.)
- Paste
  - p (paste after)
  - P (paste before)
- Undo
  - u
- Search
  - /
- Save/Quit
  - :w (write)
  - :q (quit)
  - :wq (write and quit)
  - :q! (give up changes)

Emacs

- Extremely powerful text editor features
- Great for power users
- Non standard shortcuts
- Much more than a text editor (games, e-mail, shell, browser).
- Some power commands have to be learnt.



Emacs

- $ emacs
- Cursor movement
  - ^f (forward one char.)
  - ^b (backward one char.)
  - ^a (begin of line)
  - ^e (end of line)
  - ^n (next line)
  - ^p (prev. line)
  - ^v (page up)
  - alt-v (page down)
- Deletion
  - ^d (delete one char)
  - alt-d (delete one word)
  - ^k (delete line)
- Paste
  - ^y (yank)
- Undo
  - ^/
- Load file
  - ^x^f
- Cancel
  - ^g
- Save/Quit
  - ^x^c (quit w/out saving)
  - ^x^s (save)
  - ^x^w (write to a new file)

Various commands

wget

- Instead of downloading files from your browser, just copy and paste their URL and download them with `wget`
  `wget https://github.com/franklbvp/linuxintro/blob/master/docs/LinuxDev.zip`
- main features
  - http and ftp support
  - Can resume interrupted downloads
  - Can download entire sites or at least check for bad links
  - Very useful in scripts or when no graphics are available (system administration, embedded systems)

•`$ wget -c http://microsoft.com/customers/dogs/winxp4dogs.zip`
Continues an interrupted download.

•`$ wget -r -np http://www.xml.com/ldd/chapter/book/`
Recursively downloads an on-line book for off-line access.
-np: "no-parent". Only follows links in the current directory.

time

- Helpful command for doing simple benchmarking
- Run your scripts along with **time** command, and compare the execution time.
- Example:

```
$ time ls
```

real 0m2.304s (actual elapsed time)
user 0m0.449s (CPU time running program code)
sys 0m0.106s (CPU time running system calls)

time

- real = user + sys + waiting
  waiting = I/O waiting time + idle time (running other tasks)
- real or total or elapsed (wall clock time)
  - is the time from start to finish of the call. It is the time from the moment you hit the Enter key until the moment the command is completed.
- user
  - amount of CPU time spent in user mode.
- system or sys
  - amount of CPU time spent in kernel mode.

information about users

- `$ who`
  Lists all the users logged on the system.
- `$ whoami`
  Tells what user I am logged as.
- `$ groups`
  Tells which groups I belong to.

Measuring disk usage

- `$ du -h <file>`
  -h: returns size on disk of the given file, in human readable format: K (kilobytes), M (megabytes) or G (gigabytes),
  Without -h, du returns the raw number of disk blocks used by the file (hard to read).
  Note that the -h option only exists in GNU du.
- `$ du -sh <dir>`
  -s: returns the sum of disk usage of all the files in the given directory.

Measuring disk space

- `$ df -h <dir>`
  Returns disk usage and free space for the filesystem containing the given directory.
  Similarly, the -h option only exists in GNU df.
- Example:
  `$ df -h .`

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/hda5	9.2G	7.1G	1.8G	81%	/

- `$ df -h`
  Returns disk space information for all filesystems available in the system. When errors happen, useful to look for full filesystems.

How much space do I have?

- `quota`: command to see all quotas for your directories are, if any

```
3.10.0-957.27.2.el7.x86_64
bash-4.2$ echo $SHELL
/bin/bash
bash-4.2$ ls
Desktop                      Videos                mytest.m
Documents                    core.23219            openmp
Downloads                    inbox                 output.txt
MATLABDesktopCreateError.log intel                 pbsnodes-list
Matlab_and_Worker_p2.pdf     java.log.14915        result10hpc.txt
Music                        matlab                simple_script_1
Pictures                     matlabtest            test
Public                       mex_sum_openmp.c      test_mex_openmp.m
Templates                    mex_sum_openmp.mexa64 testwo
bash-4.2$ pwd
/vsc-hard-mounts/leuven-user/300/vsc30051
bash-4.2$ quota
quota: error while getting quota from nfshome.usr.hydra.brussel.vsc:/apps/brussel for vsc30051 (id 2530051): Operatio
quota: error while getting quota from nfsdata.usr.hydra.brussel.vsc:/data/brussel for vsc30051 (id 2530051): Operatio
quota: error while getting quota from nfshome.usr.hydra.brussel.vsc:/user/brussel for vsc30051 (id 2530051): Operatio
quota: error while getting quota from nfsdata.gastly.gent.vsc:/user/data/gent for vsc30051 (id 2530051): Operation no
quota: error while getting quota from nfsapps.gastly.gent.vsc:/apps/gent for vsc30051 (id 2530051): Operation not per
quota: error while getting quota from nfshome.gastly.gent.vsc:/user/home/gent for vsc30051 (id 2530051): Operation no
Disk quotas for user vsc30051 (uid 2530051):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
10.118.240.67:/user
               2650468  2831156 3145728          61675   90000  100000
10.118.240.67:/data
                 34919020  76546048 78643200           4930  9000000 10000000
bash-4.2$
```

Comparing files and directories

- `$ diff file1 file2`
  Reports the differences between 2 files, or nothing if the files are identical.
- `$ diff -r dir1/ dir2/`
  Reports all the differences between files with the same name in the 2 directories.
- These differences can be saved in a file using the redirection, and then later re-applied.
- https://linuxacademy.com/blog/linux/introduction-using-diff-and-patch/

The grep command

- **G**lobal **r**egular **e**xpression **p**rint
- Grep is used to search text files with **regular expressions** (**regex**).
  - It prints the lines matching the given pattern in a text file.
  - If no file is given, grep will recursively search the given pattern in the files in current directory

The grep command

- `$ grep <pattern> <files>`
  Scans the given files and displays the lines which match the given pattern.
- `$ grep error *.log`
  Displays all the lines containing error in the *.log files
- `$ grep -i error *.log`
  Same, but case insensitive
- `$ grep -ri error .`
  Same, but recursively in all the files in the current directory and its subdirectories
- `$ grep -v info *.log`
  Outputs all the lines in the files except those containing info.
- http://www.thegeekstuff.com/2009/03/15-practical-unix-grep-command-examples/

More commands

- `$ sleep 60`
  Waits for 60 seconds
  (doesn't consume system resources).
- `$ wc report.txt`
  word count
  Counts the number of lines, words and characters in a file or from standard input.
- `$ date`
  Returns the current date. Useful in scripts to record when commands started or completed.
- Before you run a command, `which` tells you where it is located
  `$ which ls`

More commands

- `touch`
  - changes a file's modification timestamp without editing the contents of the file.
  - It is also useful for creating an empty file when the filename given does not exist.
- `echo` – displays a line of text
- `sort` – sorts a file of lines alfabetically
- `tr` – translates between characters (e.g. tr a-z A-Z)
- `whereis` - locate the binary, source, and manual page files for a command

Archiving

File Archiving: tar

- File and Directory Compression
- Files or directories can be stored as a "tarball" (.tar file) as well as compressed further using other programs.
- Saves and restores multiple files to/from a single file. Directories are added recursively.
- Format:
  - `$ tar [options] [options_values] [files]`

  - c – create a new archive
  - v – verbosely list files which are processed.
  - f – following is the archive file name
  - z – filter the archive through gzip (compress)
  - x – extract files from archive
  - C - specified directory
  - j - filter the archive through bzip (compress)

File Archiving: tar

- Examples:
  - `$ tar -cvf [FILE] [ITEMS]` Backup the specified item(s)
    - `$ tar -cvf /tmp/backup.tar ~/data ~/test`
  - `$ tar -czvf [FILE] [ITEMS]` Compress the archive to save space
  - `$ tar -xvf [FILE] [ITEMS]` Restore the specified item(s)
    - `$tar -xvf backup.tar`
  - `$ tar -tf [FILE]` List all files in the specified archive
    - `$ tar -tf backup.tar`
- http://www.thegeekstuff.com/2010/04/unix-tar-command-examples/

File Compression: gzip

- Compressing files: `gzip filename` or `bzip2 filename`
- Gzip compresses only single files and creates a compressed file for each given file. By convention, the name of a file compressed with gzip should end with either .gz or .z.
- `gzip` will create a file filename.gz and delete the original file.
  - `$ gzip backup.tar`
  - `$ bzip2 backup.tar`

  The resulted file is backup.tar.gz/ backup.tar.bz2
- Uncompressing files: `gzip -d filename.gz` or `bzip2 -d filename.bz2`
  - `$ gzip -d backup.tar.gz`
  - `$ bzip2 -d backup.tar.bz2`

  The uncompressed file is backup.tar
- `unzip` for zip-files

demo

Users, groups, security

File access rights

Linux File Access permissions

- Linux is a multiuser system, the files of all users are stored in a single file structure
- Mechanism is required to restrict one user to access the files of another user, if the user is not supposed to
- User can impose access permission to each file to restrict its access

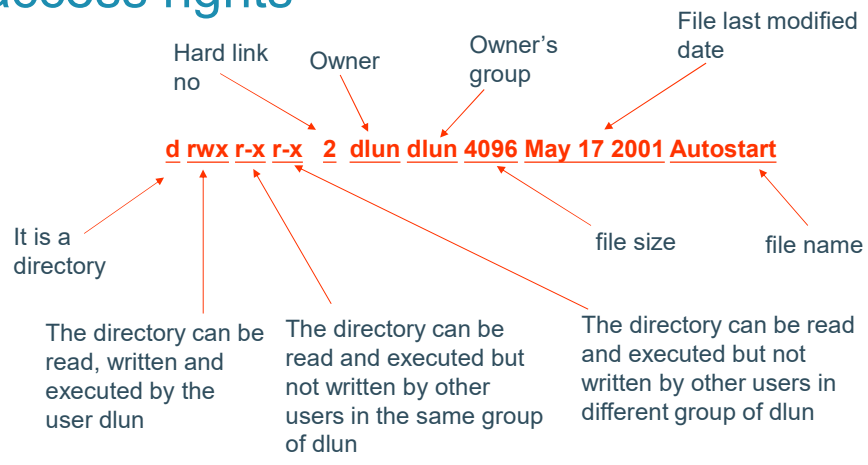
File access rights

3 types of **access rights**

- Read access (r)
  - reading, opening, viewing, and copying the file is allowed
- Write access (w)
  - writing, changing, deleting, and saving the file is allowed
- Execute rights (x)
  - executing and invoking the file is allowed. This is required for directories to allow searching and access.

Use `ls -l` to check file access rights

File access rights

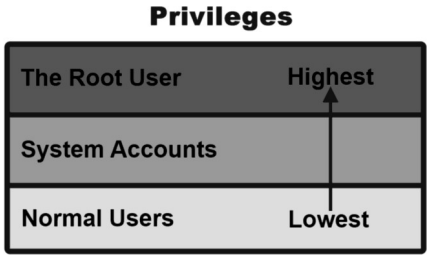


The group of a user is assigned by the administrator when a user is added to the system

File access rights

3 types of **access levels**

- User (u): for the owner of the file
- Group (g): each file also has a “group” attribute, corresponding to a given list of users
- Others (o): for all other users



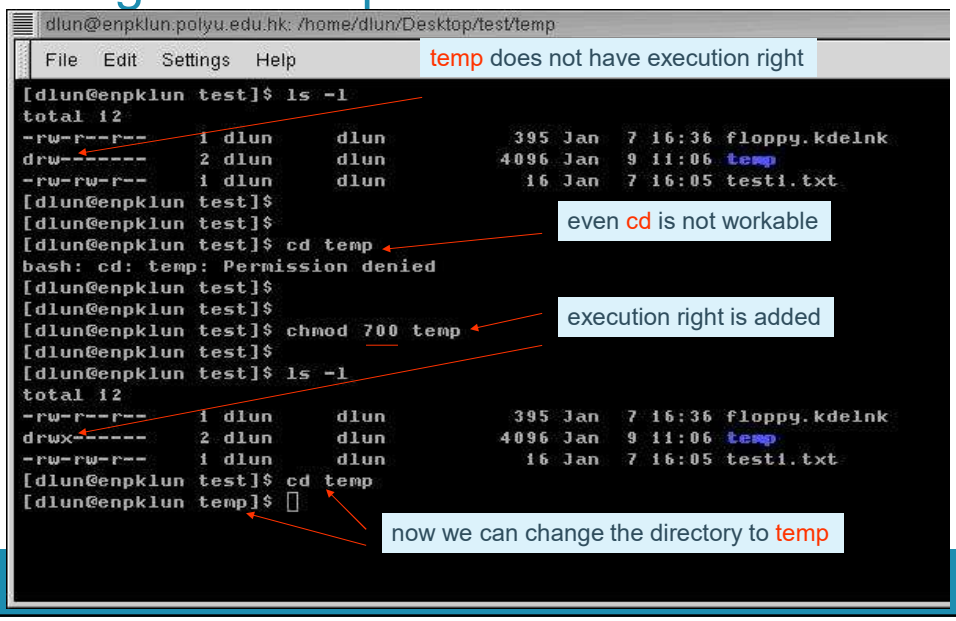
File access rights

- Access permission can also be assigned to a directory
- Directory is also a file that contains the attributes of the files inside it
- If read permission is not given to a directory
  - cannot show the structure of this directory
  - e.g. cannot use ls
- If write permission is not given to a directory
  - cannot modify anything of the directory structure
  - e.g. cannot copy a file into this directory since it will modify the directory structure by adding one more file
- If execute permission is not given to a directory
  - nearly nothing can be done with this directory, even cd

Access rights examples

- `-rw-r--r--`
  Readable and writable for file owner, only readable for others
- `-rw-r-----`
  Readable and writable for file owner, only readable for users belonging to the file group.
- `drwx------`
  Directory only accessible by its owner
- `-------r-x`
  File executable by others but neither by your friends nor by yourself.

Access rights examples



chmod: changing permissions

- Permissions allow you to share files or directories or to lock them down to be private.
- `$ chmod` (change mode)
- `$ chmod <permissions> <files>`
  2 formats for permissions:
  - octal format (3 digit octal form)
  - symbolic format

chmod: changing permissions

- octal format (abc):
  a,b,c = r*4+w*2+x*1 (r, w, x: booleans)
  - 0 none ---
  - 1 execute-only --x
  - 2 write -w-
  - 3 execute and write -wx
  - 4 read-only r–
  - 5 read and execute r-x
  - 6 read and write rw-
  - 7 read, write, and execute rwx

- `$ chmod 644 <file>`
  (rw for u, r for g and o)

660 : 110 110 000
⇒ rw- rw- ---
545 : 101 100 101
⇒ r-x r-- r-x

chmod: changing permissions

- symbolic format:
  `$ chmod go+r`: add read permissions to group and others.
  `$ chmod u-w`: remove write permissions from user.
  `$ chmod a-x`: (a: all) remove execute permission from all.
- Tip: WSL
  - Is not working for the windows partion
  - Works for the linux part
  - Check in home directory

Hands-on 4