Outline

- Introduction - history
- Command line basics – getting help
- File system
- Working with files and directories
- ➢More file handling
- The shell revisited
- Monitoring resources

Outline details

- Finding files
- Archiving
- Links
- File permissions

Finding files and more

More on files

- Search for files and directories
  - The **`find`** command performs a raw search on a file system to locate the specified items.
  - `$ find  location –name some-name`

    `($ find / -name matrix.c)`
  - You can also specify more than one location to search,
- Search the locate database for files and directories
  - The `locate` command displays the location of files that match the specified name.
  - Faster than find but lacks the ability to search for advanced characteristics such as file owner, size, and modification time.

More on files

- Display extended information about a file system, file, or directory
- What does a file contain?
  - Determine a file's type: **`file`**
  - will print a brief description of the file's contents
  - `$ file  filename`
- The `stat` command displays extended information about files. It includes helpful information not available when using the `ls` command
  - Shows different "timestamps":
    - Access - the last time the file was read
    - Modify - the last time the file was modified (content has been modified)
    - Change - the last time meta data of the file was changed (e.g. permissions)

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/

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
    e.g. `$ 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 backup.tar`
  - `$ bzip2 backup.tar`

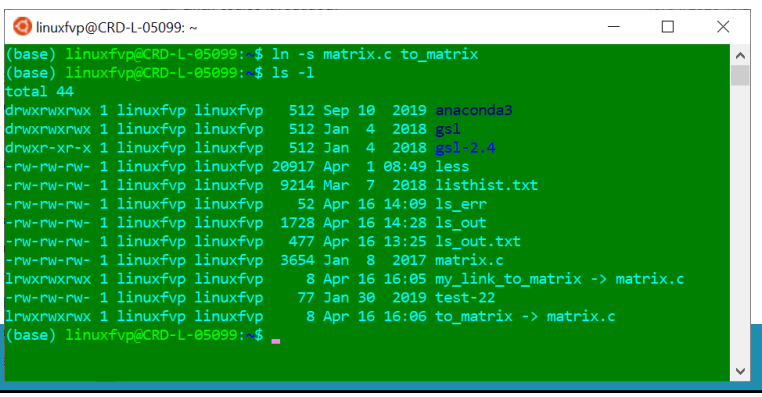
  The resulted file is backup.tar.gz/ backup.tar.bz2
- Uncompressing files: `gzip -d filename.gz` or `bzip2 -d filename.bz2`
  - `$ gzip -d backup.tar.gz`
  - `$ bzip2 -d backup.tar.bz2`

  The uncompressed file is backup.tar

Links

Create links

- **Soft link**: similar to a shortcut in Windows. It is an indirect pointer to a file or directory; can point to a file or a directory on a different filesystem or partition.
- Symbolic links are created when using the `-s` option with the `ln` command.
  `ln -s [OPTIONS] FILE LINK`
- Check with `ls -l`
- The first character "l", indicates that the file is a symlink.
- The "->" symbol shows the file the symlink points to.



Create links

- Editing a symbolic link file is the same as editing the source file
- Deleting the symbolic link does not delete the source file.
- Deleting the source file leaves a dangling link
- `$ ln -s file_v5.doc file_final.doc`
  creates a symbolic link called file_final.doc that points to file_v5.doc
- `$ ln -s /home/demo/dir1/dir2/dir3 /home/demo/jump2dir`
  creates a symbolic link called jump2dir that points to a deep directory (allows for quicker access)

File permissions

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
- User can impose access permission to each file to restrict its access

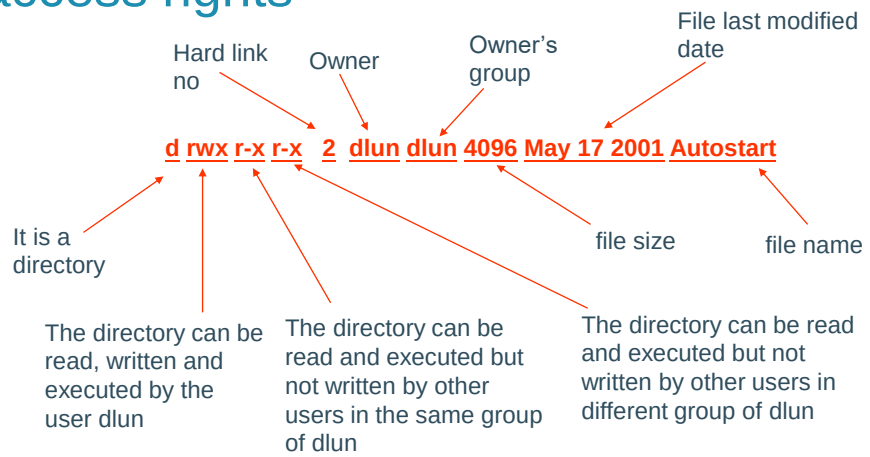
File access rights

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

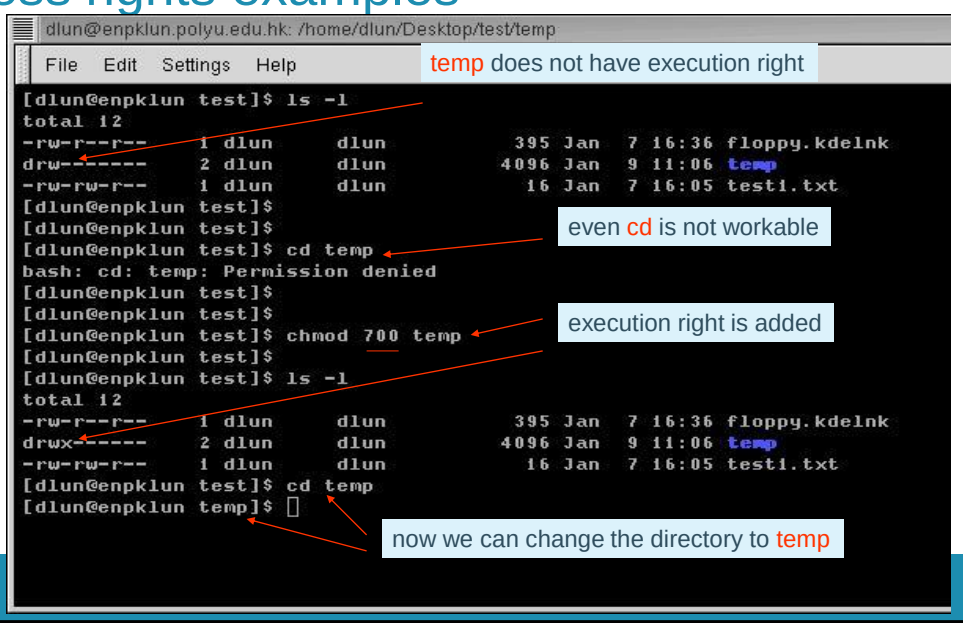
File access rights

- 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 group 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: u(user) group(g) others(o) all(a)
  - `$ 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.