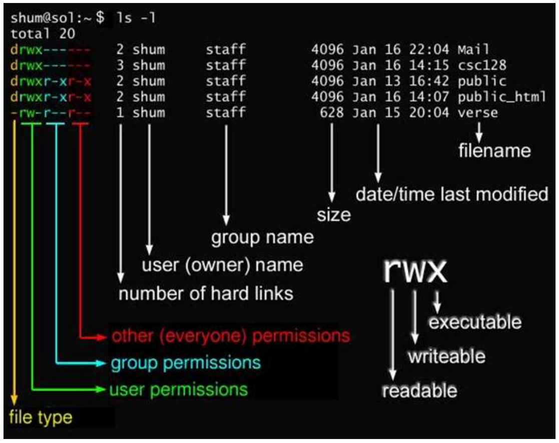
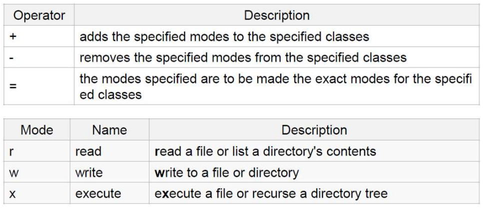
- Shell
    - tab auto completet
    - !! replace with previous command
      ![str]: refer to previous command with str
      ^[str]: replace with command referred to as str
- pwd print working directory
- mv cp rm -r mkdir rmdir
- ls
      d: list only directories
      a: list all files including hidden ones
      l: show long listing including permission info
      s: show size of each file, in blocks
    - 

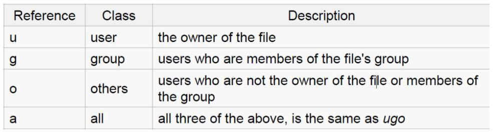
```
shum@sol:~$ ls -l
total 20
drwx------  2 shum    staff       4096 Jan 16 22:04 Mail
drwx------  3 shum    staff       4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum    staff       4096 Jan 13 16:42 public
drwxr-xr-x  2 shum    staff       4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum    staff        628 Jan 15 20:04 verse
```

- ln
    - hard link
        - Symbolic links link to a path name. This can be anywhere in a system's file tree, and doesn't even have to exist when the link is created. The target path can be relative or absolute.
          Hard links are additional pointers to an inode, meaning they can exist only on the same volume as the target. Additional hard links to a file are indistinguishable from the "original" name used to reference a file.
        - Hard link point to the file content. while Soft link points to the file name.
        - while size of hard link is the size of the content while soft link is having the file name size.
        - Hard links share the same inode. Soft links do not.
        - Hard links can't cross file systems. Soft links do.
        - you know immediately where a symbolic link points to while with hard links, you need to explore the whole file system to find files sharing the same inode.
        - hard-links cannot point to directories.
    - -s soft link
          However, symlinks can be broken for many reasons,
          including:
          ☜♓⧫♒♏❒ ⧫♒♏ □❒♓♑♓■♋● ♐♓●♏ □❒ ⧫♒♏ ⧫♋❒♑♏⧫ □♐
          ⧫♒♏ ●♓■& ♎♓♎ ■□⧫ ♏⌧♓⬧⧫ ⬥♒♏■ ⧫♒♏ ●♓■& ⬥♋⬧
          ♍❒♏♋⧫♏♎ ☎♋⬧
          in Listing 5).
          ❄♒♏ ⧫♋❒♑♏⧫ □♐ ♋ ●♓■& ♓⬧ ♎♏●♏⧫♏♎ □❒
          ❒♏■♋❍♏♎📪
          💧□❍♏ ♏●♏❍♏■⧫ ♓■ ⧫♒♏ □♋⧫♒ ⧫□ ⧫♒♏ ⧫♋❒♑♏⧫ ♓⬧
          ❒♏❍□❖♏♎ □❒ ❒♏■♋❍♏♎📪
- touch -t 201101311759.30 filename
- chmod

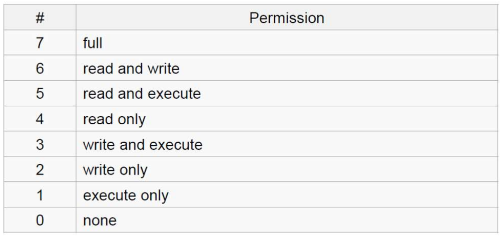


Operator	Description
+	adds the specified modes to the specified classes
-	removes the specified modes from the specified classes
=	the modes specified are to be made the exact modes for the specified classes

Mode	Name	Description
r	read	read a file or list a directory's contents
w	write	write to a file or directory
x	execute	execute a file or recurse a directory tree



Reference	Class	Description
u	user	the owner of the file
g	group	users who are members of the file's group
o	others	users who are not the owner of the file or members of the group
a	all	all three of the above, is the same as *ugo*



#	Permission
7	full
6	read and write
5	read and execute
4	read only
3	write and execute
2	write only
1	execute only
0	none

- Usage

– chmod [''references''][''operator''][''modes''] ''file1'' ...
Example: **chmod** ug+rw mydir, **chmod** a-w myfile,
Example: **chmod** ug=rx mydir, **chmod** 664 myfile

- sticky bit +t
  - it locks files within the directory from being modified/deleted by users other than the file creator, owner of the directory, or root, even if others have write permissions.
- special permission

  The setuid (set user id) is a permission bit, that allows the users to exec a program with the permissions of its owner. The setgid (set group id) is a bit that allows the user to exec a program with the permissions of the group owner.
- chmod -r recursive ssh:chmod -r . 700

- find

  -type: type of a file (e.g: directory, symbolic link)
  -perm: permission of a file
  -name: name of a file
  -user: owner of a file
  -maxdepth: how many levels to search
  find . -name my*
  find . -name my* -type f
  find / -type f -name myfile
  find -mtime 21
- man

  man section command_name
  1 User Commands 2 System Calls 3 C Library Functions 4 Devices and Special Files 5 File Formats and Conventions 6 Games et. al. 7 Miscellanea 8 System Administration tools and Daemons
- whatis(name section of man page) whereis(locate)
- apropos
  - takes its name from the English word with the same spelling (and the same pronunciation) that means relevant. It is particularly useful when searching for commands without knowing their exact names.
- diff
  - diff original new
    - The "c" tells patch to replace the content of the lines. Two other characters with a meaning exist: "a" and "d", with "a" meaning "add" or "append" and "d" meaning "delete".
  - diff -u ori new(unified format)
    - change to context
    - The "<" means that patch should remove the characters after this sign, and the ">" means that the characters after this sign should be added.
- patch

- ○ diff format
    - --- path/to/original_file
    - +++ path/to/modified_file
    - @@ -l,s +l,s @@
    - ■
        - – @@: beginning and end of a hunk
        - – l: beginning line number
        - – s: number of lines the change hunk applies to for each file
        - – A line with a:
            - - sign was deleted from the original
            - + sign was added in the new file
            - ' ' stayed the same
- ○ patch -p0 < patchfile
    - ■ #after p
        - □ diff -ru old_version new_version >some.patch

          Then the patch contains file names, given in header lines like diff -ru old_version/dir/file new_version/dir/file. You need to tell patch to strip the prefix (old_version or new_version) from the file name. That's what -p1 means: strip one level of directory. Sometimes, the header lines in the patch contain the file name directly with no lead-up. This is common with version control systems; for example cvs diff produces header lines that look like diff -r1.42 foo. Then there is no prefix to strip, so you must specify -p0.

          In the special case when there are no subdirectories in the trees that you're comparing, no -p option is necessary: patch will discard all the directory part of the file names. But most of the time, you do need either -p0 or -p1, depending on how the patch was produced.

          /u/howard/src/blurfl/blurfl.c

          setting -p0 gives the entire file name unmodified, -p1 gives

          u/howard/src/blurfl/blurfl.c

          without the leading slash, -p4 gives

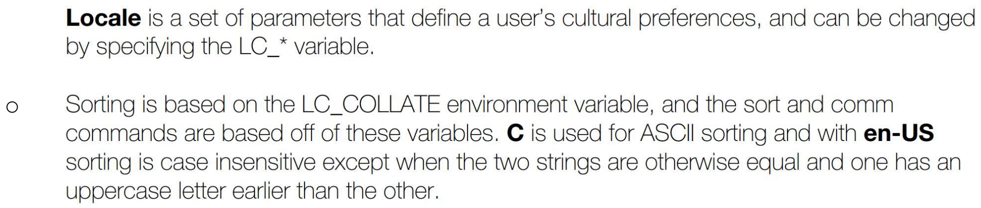
          blurfl/blurfl.c

          and not specifying -p at all just gives you blurfl.c. Whatever you end up with is looked for either in the current directory, cor the directory specified by the -d option.
        - □ The -p options tells patch how many slashes (including what's before them, usually directories) it should strip away before the filename (note that, when using the option -p0, patch looks for the files to patch in both originaldirectory and updateddirectory, in our case)
        - □ (...) *** /home/username/sources/program/originaldirectory/file1 2007-02-04 16:17:57.000000000 +0100 --- /home/username/sources/program/updateddirectory/file1 2007-02-04 16:18:33.000000000 +0100 (...)

          You could just count the slashes (/ (1) home/ (2) username/ (3) sources/ (4) program/ (5)) and give that value with the -p option. If you're using -p5, patch would look for both originaldirectory/file1 and updateddirectory/file1. Please do note that patch considers two slashes next to each other (like in /home/username//sources) as a single slash. This is because scripts sometimes (accidently or not) put an extra slash between directories.
- ○ patch -R <patchfile to reverse
- ○ create a patch
    - ■ diff -u originalfile updatedfile > patchfile.patch
    - ■ emacs C-X 4 a

- process
    - ○ ps list process
    - ○ kill kill process
    - ○ fg resume
- tar
    - ○ -cf create
    - ○ -xf extract
    - ○ -z gzip
    - ○ -v verbose
    - ○ zip: tar -czvf <output_archive.tar.gz> <file1> <directory2> < file3>

- gcc
    - -W turn on warnings
        - -Wall turn all warnings on
    - the rest see dynamic linking
- strace
    - strace: intercepts and prints out system calls.
    - $ strace –o strace_output ./tr2b ‘AB’ ‘XY’ < input.txt
    - $ strace –o strace_output2 ./tr2u ‘AB’ ‘XY’ < input.txt
- time
- sort
    - sort [OP] [FILE]



Locale is a set of parameters that define a user’s cultural preferences, and can be changed by specifying the LC_* variable.

- Sorting is based on the LC_COLLATE environment variable, and the sort and comm commands are based off of these variables. **C** is used for ASCII sorting and with **en-US** sorting is case insensitive except when the two strings are otherwise equal and one has an uppercase letter earlier than the other.

- comm
    - have to be sorted, line by line
    - compare -123 file1 file2(compress
- cmp
    - compare bytes by bytes
- tr
    - tr [OPTION] SET1 SET2
- grep
    - grep - Default as basic, egrep is extended and fgrep is fixed (doesn't understand REGEX)
- sed
    - sed -n 1~2p
        - p print
        - first~step (1~2p print all odd # lines)
    - s/ substitution
    - /g make changes to every occurrence
      sed 's/*Regex to find things to replace*/*replace with*/'
        - Example: Find comments from bash script and remove them, but don't remove the shabang line!
          sed 's/^#[^!].*//' = find # at start, not ! and any character
        - Example: From a file of phone numbers, replace digit only form. Find (xxx) xxx-xxxx and make xxxxxxxxxx. Note the group () stores what’s inside it for back referencing, but remember sed is basic regex so you have to escape things.
            sed 's/(\([0-9]\{3\}\))\([0-9]\)\{3\}-\([0-9]\{4\}\)/\1\2\3/' < contacts.txt
            Note that sed does in place replacement!
            sed 's/[()-]//g'
- head / tail
    - default 10 lines
    - head --bytes=5000000 /dev/urandom > input
- od
    - dump the file
    - od -An -f -N 40000000 < /dev/urandom | tr -s ' ' '\n' > random.txt
- lld
    - List dynamically linked library that used at run time
- make
    - configure: create makefile
        - ./configure --prefix=/u/eng/class/classhen/35l/hw3/redi/coreutils-7.6
    - make: compile the code & create executables into temp dir
        - use macros assigned by BASH variables
            - □ e.g. CFLAGS= -O -systype bsd43
    - make install: make utility searches for a label named install within the Makefile, and executes only

that section of it. executables are copied into the final directories (system directories)

- make clean - should clean things up. Get rid of the executables, any temporary files, object files, etc.