cat (takes files (or stdin) as input and copies them, sending result to stdout

cat >filename (creates a file)

cat filename (shows you what is in file)

cat file1 file2 >file3 (creates file3 and puts contents of file1 and 2 into it)

mkdir x (make directory named x)

rmdir x (removes directory named x)

rm f (removes files named f)

more (display files page by page)

date (display current date)

id (display information about current user)

# (comment 🡪 comp will ignore anything after it

man (get info from Unix manual 🡪 man x (output first manual pg found that is named x)

man –k yy (output the names of pgs whose terse description contains yy

script (record a terminal session in a file named \_\_\_) 🡪 script yyy names the file yyy

mkdir –P x/y/z

touch x/y/z/z (last z is a file created) – real application is to update a file (time)

rm –r r (erases everything (files + dir); -r is recursive cmd; r at end is name of top lvl dir

ls –R (recursive listing); how to go to top of tree structure and look down

ls –a (or –A) see hidden files 🡪 ls –A (more useful because it suppresses the . and ..)

-C (forces to list everything out in a row/line)

-1 (forces everything to be listed own)

ls | more (pager.. can scroll though pages)

who (who else is online)

id (who am i)

wc (word count 🡪 count lines (wc -l), words (wc -w), and bytes (wc -c). default is –lwc

ls | more (lists down in a column and you can scroll down using space bar)

ls –C | more (lists down in rows and columns and you can scroll down using space) 🡪-C flag forces multiple columns

ls -1 (lists down in a column all the way to the bottom and allows you to enter next cmd)

rmbs < typescript > typescript1

sftp (secure file transfer protocol) 🡪 sftp [user@]host connect to host as user

sftp cmds:

ls, pwd, mkdir, cd operate on remote system

use 1 prefix to each cmd to operate on the local system (eg. 1pwd)

get x (transfers file x from current directory on the remote system to the current dir on local system)

put x (transfers file x from current dir on local system to current dir on remote system)

put – transfer file x from the current dir on the local system to the current dir on the remote system

get – t/f file x from the current dir on the remote system to the current dir on the local system

If I am on Linux; want to copy script from current dir to hills; placing it in ~ (use put)

$ sftp hills.ccsf.edu’

Pw

Sftp> ls (remote) vs lls (local)

Sftp> put typescript

Msg

Sftp> pwd (remote) vs lpwd (local)

If on hills; T/F file to linux

$sftp IPaddress

Pw

Sftp>pwd

Sftp> (from dir your in)

Sftp> put (filename)

scp (secure copy program. Execute a single copy cmd between two hosts

scp xxx [user@]hostx:[path] copy xxx from the current host to hostx, logging in as user, and placing the result at path. If user is missing, the current login is used. If path is missing, the user’s home dir is used. \*\*\*\*

$scp

Copy local 🡪 remote

$scp (path local) user@host:(path remote – relative to home dir)

Eg. scp typescript hills.ccsf.edu:typescript1

Copy from remote 🡪 local

$scp user@host:(path remote) (path local)

If copying a dir: (from local to remote)

$scp –r (path local) user@host:(path remote)

Eg. scp –r (dir) (remote):dir

If copying a dir: from remote to local

$scp –r user@host:(path remote) (path local)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A\* capital A followed by 0 or more char

? single char

[] single character that is one of the characters in the brackets.

[a-z][A-Z][0-9]

[[:alpha:]] upper, lower, digit, alnum, blank, space(whitespace), punct

[!a-z] a single character whose value is not greater than or equal to a and less than or equal to z

? and \* will not match a leading . in a name (indicating a hidden file)

NOTE: you can’t copy a file on top of a dir or or a dir on top of a file

cp file1 file2 🡪 copy file1 to file2. File2 is overwritten if it exists.

cp file1 file2 file3 fileN dir 🡪 copy one of more files into an existing dir

cp –r dir1 dir2 🡪 copy dir1 and all of its contents to dir2. If dir2 does not exit, the new copy of dir1 is named dir2. If dir2 exists, the copy of dir1 is placed in dir2.

cp –r dir1….dirN dir2 🡪 copy one or more directories and their contents into the existing dir2

mv file1 file2 🡪 file2 is deleted if it exists. Then file1 is renamed file2

mv file or dir1….fileordirN dir 🡪 move one or more existing files or dir into an existing dir. If a dir is moved, everything is moved, everything beneath it goes with it.

mv dir1 dir2 🡪 if dir2 does not exist, rename dir1 to dir2. If dir2 exists, dir1 is moved into dir2

Variables: (to create a variable: $variable= ) 🡪 to use type $ $variable?

$echo $PWD (absolute path to current dir) 🡪 ie. /students/vmalchik

$echo $SHELL (your shell) /usr/bin/bash

$echo $LOGNAME (your login) 🡪 vmalchik

$echo $HOME (absolute pathway to home) 🡪 /students/vmalchik

$echo $PATH (a list of dir searched to locate cmds)

$echo $TERM 🡪? vt100 (vt100 is language used by terminal)

$echo $HISTSIZE (size of history – set to 500)

Permissions:

Owners – each piece of data on unix has one owner (whomever created it)

Groups – users on the system are organized into groups

* Most users in may groups (each piece of data is in one group)
* One of my groups is my default group (the one that data goes into)

u - if you are the owner

g - I am not the owner but a member of the data’s group

o -I am not the owner nor do I belong to a data’s group

-(file or d for dir)|r –x (u)|rwx (g)|r - - (o) 1 (ownder) (group #) ….. file1

R – read (ie cat the file)

W – write (ie change file content)

X – execute (ie. attempt to execute it as a program)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Remember: to delete a file you do not need its permissions but only permissions to the directory

Every dir can find itself . and its parent ..

|  |  |  |
| --- | --- | --- |
| Operation | Perm to dir | Perm to file |
| Ls dir | R |  |
| Ls –l dir | Rx |  |
| Ls –l dir/file | X |  |
| Cat dir/file | X | R |
| Create new file in dir | Wx |  |
| Cat > dir/file | X | W |
| Rm dir/file | wx |  |
| Mv dir/file | Wx |  |

Step1: check owner, check if in group, if not those then you are in other!

Use ls –l 🡨 will list permissions

Cp

1. Does file exist?

Yes.

|  |  |  |  |
| --- | --- | --- | --- |
| Dir | File | Dir | File1 |
| X | R | X | w |

No.

|  |  |  |  |
| --- | --- | --- | --- |
| Dir | File | Dir | File1 |
| X | R | XW |  |

$chmod to change permissions; $chgrp to change group $chown to change owner

Symbolic Mode

Chmod [ugo][+ - =][rwx] filename or dir

Eg. u+r,g-w; a=r; a= ; gu+w; o+r \*; go-w \*

Absolute Mode

Rwx = 421

When you create data: permissions, ownder, group set to:

The owner is who created it; group is your default group

Check your info do $id

$umask (only takes away permissions)

Find cmd

$find dir/file –name “ “ –type[d or f]

$find . (everything recursively in the dir); use “ “ so it can read wildcards

-The r permission for dir governs if you can list the objects in it; you need r to have access to the names of objects

-to get more info you will also need to search the dir, which requires the (x) permission; also needed if you want to use options (ie. –l cmd or other flags)

-w permission for dir governs if you can update the dir (create, delete, rename/mv, objects)

-if you do not have the (x) perm for dir you cannot pass through it, making its subdirs inaccessible; you also need (x) perm to access the attributes (owner, group, perm) of an object in the dir

\*\*\*\*\*\*\*\*\*

$nl (adds line # but not to non-blanks) –ba (adds # to blanks)

$cat –n (all lines numbered)

$head –n  # (output the first # numbers)

$tail –n # (output the last # numbers of the file) eg. $tail –n 10 file

$tail –n +# (start from line #) eg. $tail –n +90 (starts at line 90 and goes to end of file)

--$fold  -w # (wrap at # columns) –s (wrap at nearest space) eg. $fold –s –w 60 filename

$wc (has –l, -w, -c)  by default has –lwc  (lines, words, bytes)

Note: $wc –l < filename (gives u number of lines in file without file name)  or $wc –c \*

--$cut –d’ ‘ –f#, f-f (eg $cut –d’#’ –f1,6-4 >  tmp/file) \*\* default is tab \*\* note: -f5,9- the – gets filenames with blanks but has a problem with symlinks

--$cut –c#,#-# filename (eg $cut –c4,6-10 file1 > temp/file) note; if you don’t close the field it goes to end-c #-

$paste –d’ ‘ filefrom fileto (white space is default delimiter)

(Note: for $tr you need to use < > ) \*\* substitute / delete / squeeze

$tr ‘string’ ‘string2’ *(substitute*) filename (eg. ‘[a-z]’ ‘[A-Z]’ change to uppe

$tr –d ‘string’ < file (each char is *deleted*) > file1

Str –s ‘sting’ file (each successive duplicate is removed/squeezed) eg –s’ ‘ (removes spaces)

--$grep “pattern”  filename \*\*\*find pattern\*\*\*

-i (ignore case); -v (don’t contain pattern); -c (count number of lines that contain pattern);

-l (output the name of the files that contain at least one match)

- \*\*or \*\* -e (grep –e “pattern” –e “pattern” … (file name)

\*\*and\*\* grep “pattern” file | grep “pattern2”

^start end$

$ln –s path nameOflink (eg. $ln –s ~/tmp/file file1;  $ln –s ./resume resume1)  can point to dir or file

$ln abspath name (hard link)  can only point to file

$ls –l (see where the symlink points to) $ls –lL see the permissions of the symlink (which is the file/ dir it points to)

$ls –i (inode number)

Abs link can be anywhere for it will lead you to the place where as relative link has to be in the same dir\*

$sort (-n for numeric; -r for reverse; ) \*\* white space is default delimiter (organize)

$sort -f (ignore case)

-o outfile into outfile (ie. sort –o fileN –t’#’ –k2,2 –k1 fileN !!

-t ‘delimiter’ -k’field’ (ie. k1,1 or 1.2nr)

-u (sorts and removes duplicate fields)

$sort file | uniq = sort –u note u can also $uniq < file

$wc (has –l, -w, -c)  by default has –lwc  (lines, words, bytes)

RE (grep)

[ ] character set

[^ ] negated char set

^ $ anchors

. any single char

.\* anything or nothing

Extended Regular Expressions (grep –E)

+ 1 or more of preceding RE

? 0 or 1 of preceding RE

| or

( ) group a new RE

{n} exactly n of preceding RE

{n,m} at least n, not more than m of preceding RE

{n,} at least n of preceding RE

Phone:

((1-?)[0-9]{3}[ .-]? |\([0-9]{3}\)[ -]?)? (area code)

‘^[0-9]{3}[. -][0-9]{4}$’ (phone)

*Only numbers and dashes*

*Start c blank or tab*

*Contain a string*

Contain a .

At least 3 digits

Number with exactly 3 digits

The word cow (separate)

A number with at least 3 digits

A number with exactly 3 digits

A proper name

Dollar and cents

Only letters and blanks

Containing more than 9 characters

Containing less than 9 characters

DeptsID : DeptName : EMpId : Empname

DeptId that begins with an E

DeptId has exactly two digits

DeptsName starts with M

DeptName is more than one [alphabetic word]

The EmpId is 3 digits