ICB2017 - Unix cheat sheet

Below is a running list of Unix commands and common arguments as used in Intro to Biocomputation Fall 2017.

`man` – format and display the on-line manual pages

- `man` expects a function/command name as an argument

`ls` – list directory contents

- `ls` expects a directory path as an argument, but if none is provided `.` is used
- `-a`: list all files, including hidden files
- `-F`: display slashes behind directories
- `-l`: include information about date of creation and file size
- `-h`: when used with `-l` this makes the file sizes human readable with varied units
- `-r`: reverse the sorted order of the files listed
- `-t`: sort by time modified

`cd` – change directory

- `cd` expects a path for the directory you would like to change to your present workind directory

`pwd` – present working directory

- `pwd` expects no argument and returns the path to your current working directory

`mkdir` – make directory

- `mkdir` expects a directory name in order to create it
- `-p`: creates intermediate directories if a directory within other non-existent directories is specified

`rm` – remove

- `rm` expects a path to a file or directory to be removed
- `-r`: remove recursively, removes directories and their contents
- `-i`: request confirmation before attempting to remove each file

`sort` – sort lines of text files

- `-d`: dictionary sort
- `-k`: which key (field or column) to sort lines by; this flag should be followed by a number; white space delineates columns
- `-n`: numeric sort
- `-r`: sort in reverse order
- `-u`: remove duplicate entries and return a sorted set of lines

`cut` – cut out selected portions of each line of a file

- `-d`: use the provided delimiter for fields/columns
- `-f`: which fields to return; note that this can be more than one field separated by commas

`cat` – concatenate and print files

- `cat` expects one or more file names as an argument
- `-b`: print line numbers for non-blank lines
- `-n`: number all output lines

`echo` – write arguments to standard out

- `-n`: do not print a trialing new line character

'less – text viewing utility

- `less` allows scrolling through a file, but no editting of the file
- `-N`: display line numbers
- `-S`: do not wrap lines

`wc` – word, line, character, and byte count

- `wc` expects one or more input file(s)
- `-c`: output the number of bytes
- `-l`: output the number of lines
- `-m`: output the number of characters
- `-w`: output the number of words

`uniq` – filter out sequential repeated lines in a file

- NOTE: the input should be sorted first if you want to remove all duplicates, not only sequential duplicates
- `-c`: precede each output line with the count of the number of times the line occurred in the input
- `-d`: only output lines that are repeated in the input
- `-u`: only output lines that are not repeated in the input
- `-i`: case insensitive comparison

`grep` – file pattern searcher

- `-c`: only a count of selected lines is written to stdout
- `--colour`: mark up the matching text with a color
- `-E`: interpret the pattern as an extended regular expression
- `-F`: interpret the pattern as a fixed string; don't use the special meaning of special characters or wildcards
- `-i`: ignore case
- `-m`: stop reading the file after the specified number of matches
- `-n`: each output is preceded by the line number in the original file
- `-v`: invert match; return non-matching lines
- `-w`: only match the pattern if it is a complete word

`find` – walk and search a file hierarchy

- `-type`: type of object to search for; can be `d` directory, `f` file, etc.
- `-name`: a pattern can be specified after this argument to only find files matching that pattern

`sed` – stream editor (find and replace)

- `-E`: interpret patterns as extended regular expressions
- the most commonly used forms of commands for sed are:
  - `'s/pattern/replacement/'`
    - this command can be followed by g - global or N - replace the Nth match
  - `'y/abc/xyz/'`

`tr` – translate characters

- `tr` expects two arguments *string1* and *string2*
- `-d`: delete characters in string1 from the input

`history` – return a history of commands entered at the prompt

- `history` does not expect any arguments

`chmod` – change files modes or access control lists

- `chmod` expects arguments of the format *who op permission* `filename`
  - *who* comes first: user (u), group (g), other (o), or all (a)
  - *op* comes next: this indicates if we are adding (+) or removing (-) permissions
  - *permission* is last: read (r), write (w), and execute (x) are the most common

Common Git commands

`git config` – get and set repository or global options

- `--global`: for writing to the ~/.gitconfig file; these will apply to all uses of Git
- `--local`: for writing to the .git/config file and setting options for a specific repository
- `-e`: will open a specified config file (global or local) in a text editor and allow you to manually edit options
- `user.name`: specify user name in config file
- `user.email`: specify user email in config file
- `color.ui`: specify coloring option in config file
- `core.editor`: specify default text editor to be used

`git init` – create an empty Git repository or reinitialize an existing one

- `git init` can take a directory as an argument, but by default it uses `.`

`git add` – add file contents to the index or "staging area"

- `git add` expects at least one file name as an argument
- `-A`: add all directory contents to the staging area

`git commit` – Record changes in files to the repository

- `-m`: use the quoted string that follows as the commit message

`git status` – show the working tree (staging area) status

- `-s`: give the output in a short format

`git log` – show the commit log

- arguments limiting the commit list returned from the log are numerous, including filtering by author, date range, regular expression, etc.; see `git log --help` for details.

`git checkout` – switch branches or restore working tree files

`git revert` – revert some existing commits