Code Commands

Richard L. Trotta III

May 1, 2019

Contents

1 Chrome

- new tab, C-t
- open closed tab, C+shift-t
- reload page, C-r
- next tabs, C-TAB
- previous tabs, C+shift-TAB
- go to search bar, C-l
- close current tab, C-w
- open downloads, C-j

- open history, C-h
- open link in new tab, C-CLICK
  - Select link with TAB then C-ENTER to open in new tab
- search page, C-f (shift+enter to cycle options)
- go to previous/next page, M-(left/right arrows)

2 Linux

- suspend current activity, C-z
- reset desktop enviroment, M-F2 r
- to switch desktop enviroments, logout then select enviroment
- see all running jobs in terminal, jobs
- resume suspended activity in foreground, fg %# (where # is job number)
- resume suspended activity in background, bg %# (where # is job number)
- kill a job, kill %# (where # is job number)
- find file in directory, find -name "<filename>"
- make python or shell script work, chmod u+x <script>
- run python script, python <script>.py
- see size of directory, du -h
- see max size of directory, df -h
- see size of file, ls -ltrh
- see path of directories, ls -la
- see amount of data processes, top -u <username>
- see last commands, history
- to uncompress a gzip tar file (.tgz or .tar.gz), tar xzf file.tar.gz

- to uncompress a bzip2 tar file (.tbz or .tar.bz2) to extract the contents., tar xjf file.tar.bz2
- to uncompressed tar file (.tar), tar xf file.tar
- to uncompress tar file (.tar) to another directory, tar xC /var/tmp -f file.tar
- give anyone permission to edit home directory, chmod o+rw /home/<username> (works for individual directories and files)
- take away permission to edit home directory, chmod o-rw /home/<username> (works for individual directories and files)
- to give permission to members of group to read home directory, drwxr-x
- by default jlab has most secure permissions on home directory, drwx
- procedures (ie source root version) automatically on login, emacs ~/.login
- to check ram usage use the htop program, htop (hit q to quit)
- rsync
  - basic syntax, rsync options source destination
  - common options...
    - verbose (show file sync info), -v
    - copies data recursively, -r
    - archive mode (recursive but also preserves timestamps and permissions), -a
    - compress file data, -z
    - human-readable, -h
    - specify a protocol, -e
    - show progress, -P (or −−progress)
    - sync a <dest> and <source> so that they match (file or directory exists in <dest> but not <source> so we delete the ones in <dest>), −−delete
  - sync a single file on a local machine, rsync -zvh <source> <dest>
  - sync a directory on local computer, rsync -avzh <source> <dest>

- sync files and directory over ssh, rsync -avzhe ssh <root>@<ip>:<source> <dest>
- specify file specific parameters, below is an example where we include files starting with R and exclude all else
  - rsync -avze ssh −−include 'R*' −−exclude '*' <root>@<ip>:<source> <dest>
- set max file size to be transferred, rsync -avzhe −−max-size='200k' <source> <dest>
- automatically delete source files after successful transfer, rsync −−remove-source-files <anyoptions> <source> <dest>
- do a test (dry) run to make sure it works properly, rsync −−dry-run <anyoptions> <source> <dest>
- set bandwidth limit and transfer file, rsync −−bwlimit=100 -avzhe ssh <source> <dest>

3 Emacs

- undo, C-x u (or simply C-/)
- redo, C-g C-_
- save, C-x C-s
- Save buffer as different file, C-x C-w
- Save all open buffers, C-x s
- Insert another file's content into current one, C-x i
- exit (no save), C-x C-c
- load .emacs file, M-x load-file
- next line, C-n
- previous line, C-p
- Move one character forward, C-f
- Move one word forward, M-f
- Move one word backward, M-b

- Move to start of a line, C-a
- Move to end of a line, C-e
- Move to start of a sentence, M-a
- Move to end of a sentence, M-e
- Move one page down, C-v (pgDn)
- Move one page up, M-v (pgUp)
- Move to beginning of file, M-<
- Move to end of file, M->
- Jump to the beginning of the current function, M-C-a
- Jump to the end of the current function, M-C-e
- Jump to the end of braces, M-C-f
- Jump to the beginning of braces, M-C-b
- Mark (highlight) text, C-space (C-@)
- Select all, C-x h
- Select paragraph, M-h
- copy, M-w
- paste, C-y
- cut, C-w
- delete word, M-d or C-BACK
- delete line, C-k or SHIFT+C-BACK
- delete sentence, M-k
- search (forward), C-s (C-s to see next instance)
- search (backward), C-r (C-r to see next instance)
- replace word, M-% (press '!' to replace all)

- spell check, M-x (type ispell in mini-buffer)
  - a, correct
  - r, replace
- center line, M-o M-s
- change mode (ie c++, java, etc.), M-x (then type; c-mode, java-mode, etc.)
- bold, M-o b
- italic, M-o i
- underline, M-o u
- default, M-o d
- tab, C-q TAB
- keep indentation, C-j
- indent multiple lines, C-u <TAB>
- Find difference between two files, M-x diff (then enter names of files)
- Switch buffer, C-x b (TAB then type buffer name from list of avaliable)
- Kill buffer, C-x k (TAB then type buffer name from list of avaliable)
- See all open buffers, C-x C-b
- Open different file in current buffer, C-x C-f
- Open buffer in new frame, C-x 5 (type in file name)
- Open split window horizontal, C-x 2
- Open split window vertical, C-x 3
- Close all split windows, C-x 1
- Open newly opened file in main buffer, C-x 0
- Select next split window, C-x o
- Clear bufffers not used in a while, M-x clean-buffer-list

- Switch between buffers more easily, M-x ido-mode (to temporarily disable, C-f)
- Open terminal in emacs, M-x ansi-term (then hit ENTER)
  - to use limited C-x commands, use C-c <singlecharacter> (e.g. C-c o == C-x o)
- Use mouse in -nw, M-x xterm
- Update buffer if changes occur, C-x C-v (then hit ENTER)
- Auto update buffer if changes occur, M-x (then type global-auto-revert-mode)
- Customize emacs, M-x customize
- Customize emacs with search, M-x customize-group
- ~/.emacs is the file with custom settings
- See and download packages, M-x list-packages
- Enter dired (directory) mode, C-x C-f ENTER
- In dired mode...
  - to delete a file...
  - d (which marks for deletion)
  - x (deletes marked items)
  - to create a directory, t
  - to create a file, C-x C-f (then save)
  - refresh buffer, g
  - run shell command on file, select file then ! (will be prompted to shell command)
  - to copy files, S-c
  - rename file, S-r
  - to mark files, m (then can run multiple shell commands if you want)
  - to unmark files, u
  - to unmark all files, S-u

  - to mark/unmark inverse files, t
  - mark all directories, -/
  - mark all files, -/ then t
  - search for expression, S-a (go to next with M-,)
  - change sorting of directory, s (will cycle time of edit and alphabetical)
  - make dired editable, C-x C-q
  - to exit, C-c C-c
  - to abort changes, C-c ESC
  - M-% is usable here
  - Replace across multiple files (in dired mode)
    * mark all files, t
    * start a grep session to mark files, Q
    * accept all changes, !

- You can save the current desktop, M-x desktop-save
- reload one saved in another directory, M-x desktop-change-dir
- reverts to the desktop previously reloaded, M-x desktop-revert
- See buffer list, C-x C-b (similar to dired)
- Search buffer for expression, M-x occur (in buffer list)
- Make names more distinct with uniqify
- Use -scratch- to edit files and such, it is erased upon leaving emacs
- Find a word in any file
  - recersively, M-x rgrep
  - just current directory, M-x lgrep
- Begin macro, C-x (
- End macro, C-x)
- Run macro, C-x e
- Macro editor, C-x C-k e

- Comment out selected area, M-;
- Align lines of code, M-x align or M-x align-regexp (then enter what to align, e.g. // to align comments)
- Page up/down in other buffer, M-pg(Up/Down)
- Open calender, C-c C-d

3.1 Org Mode

- Used with emacs to create lists and some other cool features
- convert document, C-c C-e
- open links(i.e. left mouse click), C-c C-o
- move the order of item list, M-(up/down)
- move indentation, M-(left/right)
- mark item todo, S-(right)
- mark item done, S-(left)
- set deadline to item, C-c C-d
- tag item, C-c C-c (while cursor on item)
- collaspe bullet, TAB
- collaspe/open all bullets, S-TAB
- bullet on next line, M-ENTER
- reset org to fix issues, C-u M-x org-reload

4 Batch Job

- run batchscript, jsub <batchscript>
- find where files about batch are found (e.g. -.err), ls ~/.farm_out/
- see job info, jobinfo <jobindex#>
- cancel job, jkill <jobindex#>
- cancel all jobs, jkill 0

5 Python

6 GitHub

- add name to git, git config −−global user.name '<name>'
- add email to git, git config −−global user.email '<email>'
- change editor used for git comments, git config −−global core.editor "emacs"
- see global configuration, git config −−list −−global
- clone a remote repo (https) to your local repo, git clone <remoteRepoWebAddress>
- clone a remote repo (https) to your local repo with desired directory name, git clone <remoteRepoWebAddress> <directoryName>
- clone one specific branch, git clone −−single-branch −−branch <branchname> <repo>
- see changes to local repo, git status
- pull all submodules, git submodule update −−init −−recursive
- to clone a repo with submodules,
  - check that the repo submodule links in github work
  - git clone <repo with submodules>
    - if only certain branch submodule links work you can clone one specific branch, git clone −−single-branch −−branch <branchname> <repo>
  - git submodule update −−init −−recursive
  - if that does not work check .gitmodules to make sure submodule is properly listed. The form should be
    - [submodule "<submodulename>"] path = <submodulename> url = "https://github.com/<username>/<submodulename>" branch = <branch>
  - git submodule update −−recursive −−remote
    - if HEAD detached from commit...

        - git branch -a (should see HEAD detached)
        - check if the head is really detached, git symbolic-ref HEAD (should result in *fatal: ref HEAD is not a symbolic ref*)
        - git remote update
        - change branch to master, git checkout master
        - git pull
        - git branch -a (HEAD detached should disappear but you won't be able to switch back to other branch)
        - git checkout <originalBranch> (should be fixed)
        - git rebase master
        - git add <any conflicts>
        - git rebase master (should be good then)

- bring up window to see all commits, gitk
- see differences from previous version of file, git diff <file>
- to ignore file from git...
    - open .gitignore
    - add file name to this
    - this works for directories as well (add /directory to .gitignore)
- prepare change for commit, git add <file>
- discard all local commits on this branch, git reset –hard -u
- pull one file from one branch to another, git checkout <branch-with-file> <file> (run from branch you want file)
- add all deleted files not tracked yet, git add .
- remove file from tracked list, git rm −−cached <file>
- reset modified file to unmerged path (ie no longer ready for commt), git reset HEAD <file> (do a git add after this then, may have to do a few times)
- discard change from commit, git checkout <file>
- commit all added items to local repo, git commit -author "Richard-Trotta <trotta@cua.edu>" -m "<some message>"

- check where remote repo is and name of repo, git remote -v
- remove all files that are untracked, git clean -f
- remove tracked/untracked file, git checkout −− <file>
- how to push local repo to remote repo,
  - git status
  - git add -all (for all changes)
  - git commit (do commit procedure above)
  - git pull origin <branch>
  - git push origin <branch>
- create branch from local repo, git branch <newbranch>
- delete local branch from local repo, git branch -d <branch> (-D forces)
- see all branches, git branch -avv
- change branch, git checkout <differentBranch>
- if branches of repo aren't showing up, git fetch <repo>
- go to remote branch version of local repo, git checkout −−track origin/<branch>
- delete remote branch, git push branch origin -delete <branch>
- specify a new remote repo (ie upstream), git remote add upstream <remoteRepo>
- set up upstream where push will default, git push -set-upstream origin <branch>
- block push to a remote repo, git remote set-url -push <remoterepo> <messagereminder>
- replace remote repo (ie upstream), git remote set-url upstream <URL-forRemoteRepo>
- rename current branch, git branch -m <newbranchname>
- how to create new branch in local repo and push to remote repo,

  - create new branch on github.com
  - git branch <newbranch>
  - git fetch
  - git checkout <newbranch>
  - git pull origin <newbranch>
  - git push origin <newbranch>

- look at project history, git log -oneline
- see what is different between repo and open submodule, git diff -cached -submodule
- when copying a directory (ie submodule) into your main directory and this submodule is already part of a different repo do the following,
  - git submodule status (to see if any submodules heads are not your repo)
  - cd <submodule>
  - git remote -v (to see which repo submodule is in)
  - git remote set-url origin https://github.com/<username>/<repo> (will point submodule to your repo)
  - git remote -v (you should see origin now assigned to your repo)
  - cd ../<outofsubmodule>
  - git rm -cached <submodule>
  - git status (check to make sure your submodule is untracked)
  - git commit
  - git push
  - git submodule status (your submodule should no longer be on here because it is no longer in your repo, only locally accessible)
  - git add <submodule>
  - git commit
  - git push
  - git submodule status (double check the submodule is properly in your repo now)
- to list the file types taking up the most space in your repository, git lfs migrate info (Note: you need the lfs program)

- git has a strict 100mb limit so to convert some file types to LFS (i.e. so they can be pushed), git lfs migrate import −−include="<filetype>"
- check for large files in your local master branch, git lfs migrate info –include-ref=master
- check for large files in every branch, git lfs migrate info −−everything