Code Commands

Richard L. Trotta III

April 2, 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 file, du -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

- 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)

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->
- 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
- delete line, C-k
- 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
- 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

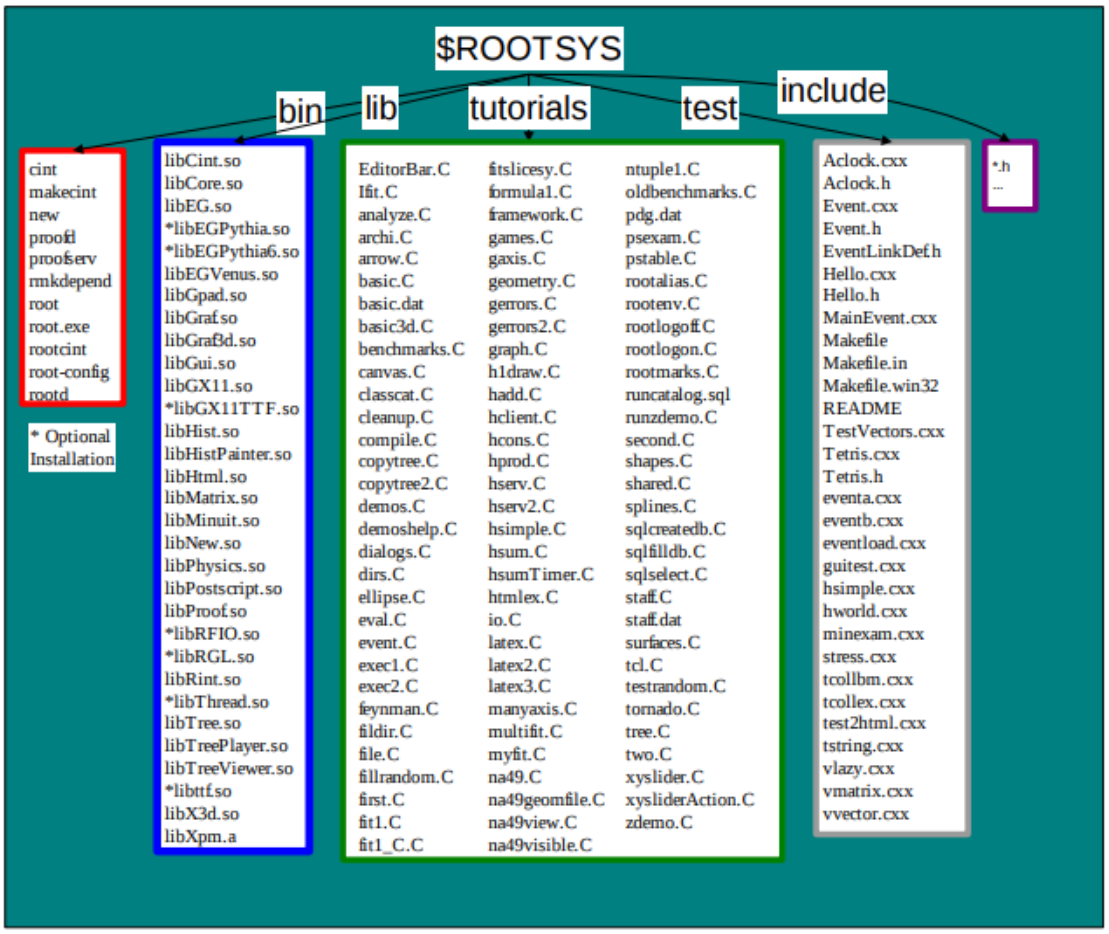
3.1 Org Mode

- 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 Root

- Object Oriented Concepts
  - Members: a "has a" relationship to the class.
  - Inheritance: an "is a" relationship to the class.
  - Class: the description of a "thing" in the system
  - Object: instance of a class
  - Methods: functions for a class
- The Framework Organization



- After starting root (i.e. >root)
  - to quit, .q
  - to load file, .L macro.C
  - to load and excecute file, .x macro.C
  - to compile and excecute file, .x macro.C++
  - to open browser to see histograms, TBrowser b
    - Left click
      - select, drag, resize objects
    - Right click
      - context menu, class::name, methods

        * Middle click
            · activate canvas, freezes event status bar

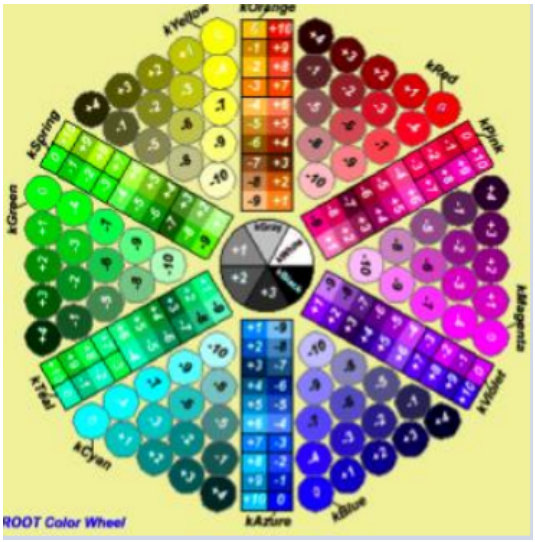
- Reading & Storing Data in Root
  - Data can be read from files/database/network
  - Data is generally stored as a TTree/TNtuple (similar to a table with rows and columns)
  - Each row represents an event
  - Each column represents a quantity
  - Trees can be created from ASCII files.
- TH1F *name = new TH1F("name","Title", Bins, lowest bin, highest bin)
  - Fill with x data, h1->Fill(x)
  - Draw histogram, h1->Draw()
  - Draw another histogram on same plot, h1->Draw("same")
  - Mean, h1.GetMean()
  - Root of Variance, h1.GetRMS()
  - Maximum bin content, h1.GetMaximum()
  - Location of maximum, h1.GetMaximumBin(int bin_number)
  - Center of bin, h1.GetBinCenter(int bin_number)
  - Content of bin, h1.GetBinContent(int bin_number)
  - Histogram cosmetics

h1.SetMarkerStyle()

h1.SetFillColor()

- Canvas: an area mapped to a window

h1->SetLineStyle()



h1.SetLineColor()

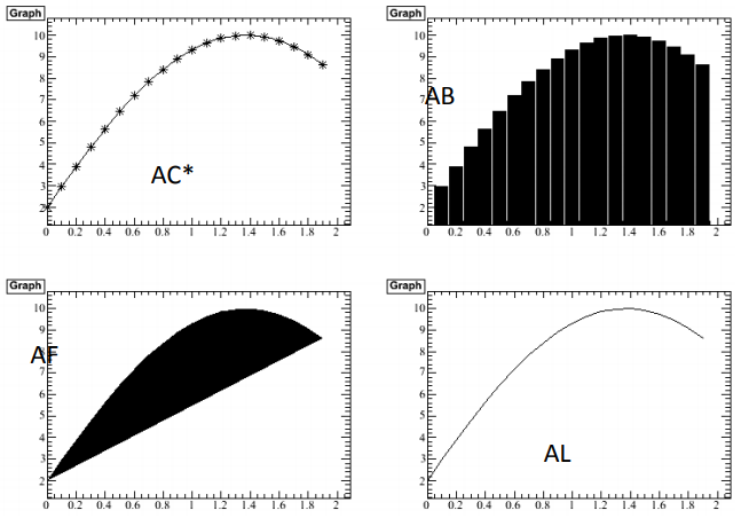
  - Creates a new canvas with width equal to w number of pixels and height equal to h number of pixels, c1 = new TCanvas(“c1”,”Title, w, h)
  - Divides the canvas to 4 pads, c1->Divide(2,2)
  - Select the 3rd Pad, c1->cd(3)
  - You can set grid along x and y axis, c1->SetGridx() or c1->SetGridy()
  - You can also set log scale plots, c1->SetLogy()

- TH2F *name = new TH2F(”name",”Title”, xBins, low xbin, up xbin, yBins, low ybin, up y bin)
  - Fill with x and y data, h12->Fill(x,y)
  - Draw histogram, h12->Draw()

- TH3F *name = new TH3F(”name",”Title”, xBins, low xbin, up xbin, yBins, low ybin, up ybin, zBins, low zbin, up zbin)
  - Fill with x, y and z data, h123->Fill(x,y,z)
  - Draw histogram, h123->Draw()

- Histogram Drawing Options
  - " SAME": Superimpose on previous picture in the same pad.
  - " CYL": Use cylindrical coordinates.
  - " POL": Use polar coordinates.
  - " SPH": Use spherical coordinates.
  - " PSR": Use pseudo-rapidity/phi coordinates.
  - " LEGO": Draw a lego plot with hidden line removal.
  - " LEGO1": Draw a lego plot with hidden surface removal.
  - " LEGO2": Draw a lego plot using colors to show the cell contents.
  - " SURF": Draw a surface plot with hidden line removal.
  - " SURF1": Draw a surface plot with hidden surface removal.
  - " SURF2": Draw a surface plot using colors to show the cell contents.
  - " SURF3": Same as SURF with a contour view on the top.
  - " SURF4": Draw a surface plot using Gouraud shading.
  - “ SURF5”: Same as SURF3 but only the colored contour is drawn.

- Graph Drawing Options



- Graph with error bars, gr = new TGraphErrors(n,x,y,errorx,errory)
- Graph polar plot, TGraphPolar * grP1 = new TGraphPolar(1000,r,theta)
- ROOT Tree
  - Store large quantities of same-class objects
  - TTree class is optimized to reduce disk space and enhance access speed
  - TTree can hold all kind of data
  - TNtuple is a TTree that is limited to only hold floating-point numbers
  - If we do not use TTree, we need to
    - read each event in its entirety into memory
    - extract the parameters from the event
    - Compute quantities from the same
    - fill a histogram
  - Prints the content of the tree, T->Print()
  - Scans the rows and columns, T->Scan()
  - Draw a branch of tree, T->Draw("x")
  - Apply cuts

        * Draw “x” when “x>0”, T->Draw(“x”,”x>0”)
        * Draw “x” when both x >0 and y >0, T->Draw(“x”,”x>0 && y>0”)
    - Superimpose “y” on “x”, T->Draw(“y”,” “,”same”)
    - Make “y vs x” 2d scatter plot, T->Draw(“y:x”)
    - Make “z:y:x” 3d plot, T->Draw(“z:y:x”)
    - Plot calculated quantity, T->Draw(“sqrt(x*x+y*y)”)
    - Dump a root branch to a histogram, T->Draw(“x»h1”)
- To deal with number of large Root files with same trees
    - name of the tree is T, TChain chain("T")
    - To chain root file1, chain.Add("file1.root")
    - To chain root file1, chain.Add("file1.root")
    - You can draw “x” from all the files in the chain at the same time, chain.Draw("x")
- Fitting with RooFit
    - RooFit packages provide a toolkit for modeling the expected distribution of events in a physics analysis
    - Models can be used to perform likelihood fits, produce plots, and generate "toy Monte Carlo" samples for various studies
- TSelector
    - root [0] TFile *f = TFile::Open("treefile.root")
    - root [1] TTree *t = (TTree *) f->Get("T")
    - root [2] t->MakeSelector("MySelector")
    - root [3] .!ls MySelector*
    - MySelector.C MySelector.h

5 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

6 Python

- Pyroot import root file

```
#!/usr/bin/env python

#import ROOT as R
from ROOT import TCanvas, TPad, TFile, TPaveLabel, TPaveText
from ROOT import gROOT
from rootpy.interactive import wait
import numpy as np

c1 = TCanvas( 'c1', 'Histogram Drawing Options', 200, 10, 700, 900 )

pad1 = TPad( 'pad1', 'The pad with the function',  0.03, 0.62, 0.50, 0.92, 21 )
pad2 = TPad( 'pad2', 'The pad with the histogram', 0.51, 0.62, 0.98, 0.92, 21 )
pad3 = TPad( 'pad3', 'The pad with the histogram', 0.03, 0.02, 0.97, 0.57, 21 )
pad1.Draw()
pad2.Draw()
pad3.Draw()

f = TFile.Open("TDISpion.root","read")
f.ls()

title = TPaveLabel( 0.1, 0.94, 0.9, 0.98,
     'Drawing options for one dimensional histograms' )
title.SetFillColor( 16 )
title.SetTextFont( 52 )
title.Draw()

#
# Draw histogram hpx in first pad with the default option.
pad1.cd()
pad1.GetFrame().SetFillColor( 18 )
hpx = gROOT.FindObject( 'hypi' )
hpx.SetFillColor( 45 )
```

```
hpx.DrawCopy()
label1 = TPaveLabel( -3.5, 700, -1, 800, 'Default option' )
label1.SetFillColor( 42 )
label1.Draw()
#
# Draw hpx as a lego. Clicking on the lego area will show
# a "transparent cube" to guide you rotating the lego in real time.
pad2.cd()
hpx.DrawCopy( 'lego1' )
label2 = TPaveLabel( -0.72, 0.74, -0.22, 0.88, 'option Lego1' )
label2.SetFillColor( 42 )
label2.Draw()
label2a = TPaveLabel( -0.93, -1.08, 0.25, -0.92, 'Click on lego to rotate' )
label2a.SetFillColor( 42 )
label2a.Draw()
#
# Draw hpx with its errors and a marker.
pad3.cd()
pad3.SetGridx()
pad3.SetGridy()
pad3.GetFrame().SetFillColor( 18 )
hpx.SetMarkerStyle( 21 )
hpx.Draw( 'e1p' )
label3 = TPaveLabel( 2, 600, 3.5, 650, 'option e1p' )
label3.SetFillColor( 42 )
label3.Draw()
#
# The following illustrates how to add comments using a PaveText.
# Attributes of text/lines/boxes added to a PaveText can be modified.
# The AddText function returns a pointer to the added object.
pave = TPaveText( -3.78, 500, -1.2, 750 )
pave.SetFillColor( 42 )
t1 = pave.AddText( 'You can move' )
t1.SetTextColor( 4 )
t1.SetTextSize( 0.05 )
pave.AddText( 'Title and Stats pads' )
pave.AddText( 'X and Y axis' )
pave.AddText( 'You can modify bin contents' )
pave.Draw()
c1.Update()
```

```
#leaf = f.Get("hypi")

#print(leaf)
#print(f)

#print(c1)

wait()
```

7 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
  - 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>
- 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,
  - git branch <newbranch>
  - 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