UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento di matematica e geoscienze

The DMG Manuals

Terminal: a mini survival guide

For Unix newbies...

Document name: DMGTerminal.pdf
Document Version: 1.0
Date Issued: September 26, 2017 3:22 PM
Author: Franco Vaccari

University of Trieste
Department of Mathematic and Geosciences
Group of Seismology
via Weiss, 4
34128 Trieste, Italy

Table of Contents

Surviving the terminal

The DSTX environment

DSTX is a Unix environment (Mac OS X actually). This is a mini survival kit for those unfamiliar with Unix.

The Terminal application

It's the application you will use to run the computational jobs, taking advantage of the Unix underpinnings of Mac OS X. You better get familiar with the Unix system: spending some time learning the basics will vastly improve your working experience with the computational software installed on the DSTX computers. When you launch the Terminal application you can start enjoying the power of Unix.

The Terminal application can be launched by clicking its icon in the Dock. Frequently used keyboard equivalent actions are:

cmd-N	open a new terminal window
cmd-T	open a new terminal in a new tab of the current terminal window
cmd-W	close the frontmost terminal
cmd-`	cycle through open terminal windows

The prompt

The characters that appear before the cursor when you open a Terminal window is named "prompt". The content of the prompt is highly customizable. By default in DSTX the prompt looks like:

[is01:/XDST/user] user%

where

is01	is the name of the machine you are logged in
/XDST/user	is the path to the working directory
user	is the username

The home directory

The home directory for each user resides on a network disk:

/Network/Servers/ps01.dstx.units.it/Volumes/xh01/NetUsers/<username>

that is better reached using the standard Unix name

~

Relevant subdirectories in ~ are:

~/Desktop corresponds to the desktop as seen when sitting in front of a Mac computer

~/Documents/DigiFiles/Polygons where .pof files used for modelling 2D profiles are stored

~/Downloads the standard location for files downloaded from Internet

This is **NOT** the directory where your computations should be carried on.

Directories to be used for computational tasks

For new computational tasks, users should use the directory:

/tmpXDST/<username>

that can be reached with alias `cdt, or`

/bkXDST/<username>

that can be reached with alias `cdk`

Long time users will still find their old data in directory

/XDST/<username>

that can be reached with alias `cdx`

Working with directories and files

In the simplest case, when you issue commands in a Terminal, you act on the files located in the so called “working directory”, i.e. the directory that appears in the DSTX prompt. Main commands to act on directories are given below, mostly by example.

A suggested convention about filenames

Filenames are case sensitive, so fil1 and Fil1 are two different items. It is suggested that directories are named with the first letter uppercase, and files are named starting with a lowercase letter. In such a way, when listing items, directories will be grouped, and listed before the files, that is:

Adir

Dir1

Dir2

MyDir

afile

file1

file2

myFile

This is because the ASCII code of Uppercase letters is smaller than the ASCII code of lowercase letters, and by default listed items are sorted by the ASCII code.

Create a new directory: mkdir

`mkdir NewDir`	create directory NewDir in the current working directory
`mkdir Dir1 Dir2`	create directories Dir1 and Dir2 in the working directory
`mkdir ~/NewDir`	create directory NewDir in the home directory
`mkdir -p Dir3/NewDir`	create directory NewDir in directory Dir3 (creates also Dir3 if it doesn’t exist
`mkdir /tmpXDST/user/Dir`	create directory Dir specifying its full path, starting from root directory /

Select the working directory: cd

`cd NewDir`	move to NewDir, located in the current directory
`cd ~/NewDir`	move to NewDir, located in the home directory
`cd`	move to the home directory (same as cd ~)
`cd ..`	move to the parent of the current directory (go one level above)
`cd ../..`	move two levels above the current directory
`cd ../Dir3`	move to Dir3, found one level above the current directory

List: ls

`ls`	list the files in the current directory
`ls Dir`	list the files in directory Dir
`ls ..`	list the files in the parent directory
`ls -l`	list the files in the current directory, in long format (showing extra info about type, size, etc)

Copy: cp

`cp a b`	make a copy of file a and name it b
`cp a Dir`	make a copy of file a into existing directory Dir (new file will still be named a)
`cp a Dir/b`	make a copy of file a into existing directory Dir (new file will still be named b)
`cp ../Dir3/a b`	make a copy of file a found one level above in Dir3 into current directory, and name it b
`cp -r Dir Dir1`	make a copy of all files and directories of directory Dir into existing directory Dir1, recursively

Rename (move): mv

`mv fil1 fil2`	rename file fil1 into fil2 (both fil1 and fil2 are files)
`mv fil1 ../Dir1`	move fil1 a into directory Dir1, that exists already one level above
`mv ../fil1 .`	move file fil1 from one level above into current directory
`mv Dir1 Dir2`	(Dir2 exists already) move the directory Dir1 into Dir2
`mv Dir1 Dir2`	(Dir2 doesn't exist yet) rename directory Dir1 into Dir2
`mv a b c Dir1`	move files a, b and c into directory Dir1

Delete: rm

`rm a`	delete file a
`rm a b`	delete files a and b
`rm -r Dir1`	delete directory Dir1 with all its content, recursively

Show the content of a file: cat

`cat myFile`	copy the full content of file myFile to the terminal

Be careful! Don't do the above if myFile is a huge file (or worse, if it's not a plain text file).

Show it better: less

less myFile	show the content of myFile, with paging

To scroll through the pages:

arrows	move cursor up, down, left, right
space	scroll forward one page
b	scroll back one page
g	go to the top of the manual
G	go to the bottom of the manual
/this	search the occurrence of “this” in the manual
n	search the next occurrence
N	search the previous occurrence
q	quit (exit the manual)

See the first part of a file: head

head myFile	show the first 10 lines of myFile
head -1 myFile	show the first line of myFile

See the last part of a file: tail

tail myFile	show the last 10 lines of myFile
tail -n1 myFile	show the last line of myFile

Edit a file: vi (and others, and TextWrangler...)

vi myFile	edit myFile, with little hope of success until you practice...

Learning even the basics of vi can be somehow frightening, but it may be worth trying. There is an interactive tutorial install, that you can use to practice. Try it by typing:

vilearn

and follow the instructions you are given.

Until you get familiar with vi (at DSTX it is actually vim - vi improved), you better start using a more user friendly editor, like TextWrangler. But don’t give up learning vi, if you plan to continue with your Unix experience. You may also consider other simpler editors like pico and nano, or powerful and complicated like emacs.

Completing commands and filenames: <tab> key

When issuing commands on files, both commands and filenames can be completed by pressing the <tab> key. The command or filename will be fully completed if there is a unique completion.

ta<tab>	will show all commands starting with ta
ls a<tab>	will show all possibilities of files to be listed, that start with “a”

Repeating commands: arrows, <ctrl-a>, <ctrl-e>

Up and down arrows

A history of the commands issued during a Terminal session is kept, and you can recall previously used commands using the up and down arrows. To scroll only through a subset of commands, type the first command character(s), and then the up or down arrow.

Right and left arrows, <ctrl-a>, <ctrl-e>

Once you have recalled a command using the up and down arrows, you can change part of it moving the cursor with the left and right arrows, and then modify the command at your wish.

To quickly move to the beginning or the end of the command, type <ctrl-a> or <ctrl-e>, respectively.

Wildcards: working with multiple items at a time

*	match any character(s)
?	match a single character
[abc]	match a single character, either a, b or c

Examples

`ls *`	list all items in the current directory
`ls a*`	list all items whose name starts with “a”
`ls *.par`	list all items with extension .par
`ls ?.dat` Dir1	list all files with extension .dat whose name has a single char before the “.”
`ls Dir1`	list the content of directory Dir1
`ls -R Dir1`	list recursively the content of directory Dir2
`ls *.[rtz]ac`	list all files with extension .rac, .tac and .zac
`cp *.par Dir1`	copy all files with extension .par into existing directory Dir1
`mv *.dat *.f Dir1`	move all files with extension .dat and .f into existing directory Dir1
`rm a*.par`	delete all files with extension .par whose name starts with a

Command options

Most of Unix command can be invoked with options that alter the command’s standard behavior. We have seen examples of this already for command ls:

`ls`	list items
`ls -R`	list items recursively (directories and their content)
`ls -l`	list items providing extra info about them

Other commonly used options for ls

`ls -t`	list items sorted by time
`ls -1`	list items in a single column
`ls -a`	list also hidden items (those starting with “.”)

Options can be combined:

ls -lRa	list items recursively, with long format, including hidden files

Help about commands and their options: man

To see the available options for a specific Unix command, and details about its usage, you can invoke the man command:

man ls show the manual for command ls

man mv show the manual for command mv

These manual pages are actually shown using the less file viewer, so you should know already how to move around them. To know more about moving through the man pages, you can obviously try

man less and to know more about man: *man man*

The shell

The default shell configured for DSTX users is tcsh. The user configuration file is located in ~/.tcshrc and for all the DSTX users is set by default to properly configure the shell environment so that all DSTX software is made available for execution. So, NOTHING should be removed from that file, and eventually more settings can be defined, in addition to the default ones.

You can use other shells, like csh, ksh, bash, zsh, if you wish. But you'll have to configure properly the environment to reproduce the configuration of tcsh.

Aliases

Aliases are user-defined, easy-to-remember equivalents to other fairly complicated commands. A set of aliases is already defined at DSTX, and can be seen with command alias. Some predefined alias examples of the DSTX environment are shown below:

```
cdb     cd /XDST/$USER/bin/Intel
cdr     cd /XDST/$USER
cdt     cd /tmpXDST/$USER
gc2d    gcc -O2 !$ -o \/XDST\/bin\/Intel\/!$:s/.c//
```

where $USER is an environmental variable that keeps the user's username. You can define your own aliases if you need them: *alias hazard 'cd $DST_ROOT/$USER/Git/Hazard'*

Surviving vi editor

Launch the editor

command	purpose
vi *filename*	Create or Edit *filename*
vi -r *filename*	Recover *filename* that was being edited when system crashed

Move the cursor around

command	purpose
Arrow keys	Move the cursor in the direction of the arrow
0 (zero)	Move the cursor to the beginning of current line
$	Move the cursor to the end of current line
G	Move the cursor to the last line
:5	Move the cursor to line n.5
k	Moves the cursor up one line
***N*k**	Moves the cursor up N lines
j	Moves the cursor down one line
***N*j**	Moves the cursor down *N* lines
h	Moves the cursor to the left one character position
l	Moves the cursor to the right one character position
ctrl-f	Scroll one page forward
ctrl-b	Scroll one page backward

Edit the text

command	purpose
i	Insert text before cursor
I	Insert text before the first character of current line
a	Add text after cursor
A	Add text to end of current line

o	Creates a new line for text entry below the cursor location
O	Creates a new line for text entry above the cursor location
u	Undo
r	Replaces with a single character the character under the cursor. vi returns to the command mode after the replacement is entered
R	Overwrites multiple characters beginning with the character currently under the cursor. You must use Esc to stop the overwriting
s	Replaces the current character with the character you type. Afterward, you are left in the insert mode.
S	Deletes the line the cursor is on and replaces it with the new text. After the new text is entered, vi remains in the insert mode
c$	Change (replace) the characters in the current line, beginning with character under cursor
cw	Change (replace) the characters in the current word, beginning with character under cursor
x	Delete single character under cursor
X	Delete single character before cursor
Nx	Delete N characters, starting with character under cursor
dw	Delete the single word beginning with character under cursor
dNw	Delete N words beginning with character under cursor
D	Delete the remainder of the line, starting with current cursor position
dd	Delete entire current line
Ndd	Delete N lines beginning with the current line
yy ***or*** **Y**	Copy the current line
Nyy ***or*** **NY**	Copy the next N lines, including the current line
p	Put (paste) the lines into the text after the current line
P	Put (paste) the lines into the text above the current line
.	Repeat the last editing command, at the cursor position

:r *filename*	Read file named *filename* and insert after current line
<Esc> (Escape) key	Exit the Edit mode

Search and replace text

command	purpose
/string	search for occurrence of string in text
n	search down for next occurrence
N	search up for next occurrence
:1,$s/old/new/g	Change “old" into “new" from first (1) to last ($) line

Save/discard changes and exit the editor

command	purpose
:q!	Quit without saving the changes
:wq *or* ZZ	Quit and save the changes
:w *filename2*	Save the file you were working on as another filename called *filename2*