Tip of the Week

Tuesday, October 21st, 2014

October 15, 2014

Dotfiles

Huh?

Dotfile

- Linux hides files beginning with a “.”

Dotfile

- Linux hides files beginning with a “.”
  - `ls -l` vs `ls -la`

Dotfile

- Linux hides files beginning with a “.”
  - `ls -l` vs `ls -la`
- Refers to configuration files in your home directory

Dotfile

- Linux hides files beginning with a “.”
  - `ls -l` vs `ls -la`
- Refers to configuration files in your home directory
  - .bashrc

Dotfile

- Linux hides files beginning with a “.”
  - `ls -l` vs `ls -la`
- Refers to configuration files in your home directory
  - .bashrc
  - .bash_aliases

Dotfile

- Linux hides files beginning with a “.”
  - `ls -l` vs `ls -la`
- Refers to configuration files in your home directory
  - .bashrc
  - .bash_aliases
  - .vimrc

.bashrc vs .bash_profile

- ~/.bashrc executed when using interactive shell that is **not** a *login shell*

.bashrc vs .bash_profile

- ~/.bashrc executed when using interactive shell that is **not** a *login shell*
- ~/.bash_profile executed only during *login shell*

.bashrc vs .bash_profile

- ~/.bashrc executed when using interactive shell that is **not** a *login shell*
- ~/.bash_profile executed only during *login shell*
- `http://hacktux.com/bash/bashrc/bash_profile`

.bashrc

- Usually sources system default (`/etc/bashrc`)

.bashrc

- Usually sources system default (`/etc/bashrc`)
- Used to setup:

.bashrc

- Usually sources system default (`/etc/bashrc`)
- Used to setup:
  - aliases

.bashrc

- Usually sources system default (`/etc/bashrc`)
- Used to setup:
  - aliases
  - environmental variables (e.g., PATH)

.bashrc

- Usually sources system default (`/etc/bashrc`)
- Used to setup:
  - aliases
  - environmental variables (e.g., PATH)
  - system commands (e.g., `ls` –> `ls -h --color=auto`)

.bashrc

- Usually sources system default (`/etc/bashrc`)
- Used to setup:
  - aliases
  - environmental variables (e.g., PATH)
  - system commands (e.g., `ls` –> `ls -h --color=auto`)
  - shell prompts

.bashrc

- Usually sources system default (`/etc/bashrc`)
- Used to setup:
  - aliases
  - environmental variables (e.g., PATH)
  - system commands (e.g., `ls` –> `ls -h --color=auto`)
  - shell prompts
  - functions

Tips

Aliases

Aliases

System Commands

- `alias ls='ls -h --color=auto'`

Aliases

System Commands

- `alias ls='ls -h --color=auto'`
- `alias topu='top -u $(id -un)'`

Aliases

System Commands

- `alias ls='ls -h --color=auto'`
- `alias topu='top -u $(id -un)'`
- check out default SCC/GEO alias for `rm`

Aliases

Sun Grid Engine

- `alias qstatu='qstat -u $(id -un)'`

Aliases

Sun Grid Engine

- `alias qstatu='qstat -u $(id -un)'`
- `alias qshv='qsh -V -l h_rt=24:00:00'`

Aliases

Movement

- `alias chris='cd /projectnb/landsat/users/ceholden'`

Aliases

Movement

- `alias chris='cd /projectnb/landsat/users/ceholden'`
- `alias landsat='cd /projectnb/landsat'`

Aliases

Movement

- `alias chris='cd /projectnb/landsat/users/ceholden'`
- `alias landsat='cd /projectnb/landsat'`
- `alias cms='cd /projectnb/landsat/projects/CMS/'`

Modules

Modules

Not necessarily recommended, but...

```
host=$(hostname)
if [[ "$host" == "geo" || "$host" == "scc1" || "$host" == "
    . /usr/local/Modules/default/init/bash
    source ~/.module
fi
```

Modules

And ~/.module:

```
module load python/2.7.5
module load gdal/1.10.0
module load R_earth/3.1.0

module load CCDCTools/_beta
```

PATH

PATH

Call your own scripts or programs:

```
if [ -d "$HOME/bin" ]; then
    PATH="$PATH:$HOME/bin"
fi
```

Shell Prompts

Shell Prompts

Current prompt:

```
ceholden@ceholden-lhome:~/Documents/ceholden.github.io/pres
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\u@\h
```

Shell Prompts

```
export PS1="\[$(tput setaf 2)\]\u@\h:\W\\$ \[$(tput sgr0)\]
```

Shell Prompts

See `http://bashrcgenerator.com/` or
`https://www.kirsle.net/wizards/ps1.html` for generators.

Functions

Functions

"abusers"

```
function abusers() {
    qstat | awk 'NR > 2 { if ($5 == "r") print $4 " " $9 }'
        awk '{ sums[$1] += $2} END \
        { for (i in sums) printf("%s %s\n", i, sums[i])}' |
        sort -k2 -n -r
}
export -f abusers
```

Functions

Cluster job ID matching pattern...

```
function jidof() {

    pattern=$1

    jid=""
    for j in $(qstat -u $USER | grep $pattern | awk '{ prin
        jid="$jid $j"
    done
    echo "$jid"
}
export -f jidof
```

Functions

GDAL - from https://github.com/dwtkns/gdal-cheat-sheet

```
function gdal_extent() {
    if [ -z "$1" ]; then
        echo "Missing arguments. Syntax:"
        echo "  gdal_extent <input_raster>"
        return
    fi
    EXTENT=$(gdalinfo $1 |\
        grep "Upper Left\|Lower Right" |\
        sed "s/Upper Left  //g;s/Lower Right //g;s/).*//g"
        tr "\n" " " |\
        sed 's/ *$//g' |\
        tr -d "[(]" | tr "," " ")
    echo -n "$EXTENT"
}
export -f gdal_extent
```

Git

Git

After doing so much work, why not back up your dotfiles?

Git

Very popular practice - `http://dotfiles.github.io/`

- Steal/borrow/get inspired by others

Git

Very popular practice - `http://dotfiles.github.io/`

- Steal/borrow/get inspired by others
- Sync across computers

Git

Very popular practice - `http://dotfiles.github.io/`

- Steal/borrow/get inspired by others
- Sync across computers
- Backup and track

Git

Very popular practice - `http://dotfiles.github.io/`

- Steal/borrow/get inspired by others
- Sync across computers
- Backup and track
- Share with others

Git

In general,

1. Create repository and clone to GEO

Git

In general,

1. Create repository and clone to GEO
2. Move all dotfiles into repo folder

Git

In general,

1. Create repository and clone to GEO
2. Move all dotfiles into repo folder
3. Symlink (`ln -s`) the files to previous locations

Git

In general,

1. Create repository and clone to GEO
2. Move all dotfiles into repo folder
3. Symlink (`ln -s`) the files to previous locations
4. Git add, commit, and push

Git

"Homeshick" - makes symbolic linking of dotfiles easy!

`https://github.com/andsens/homeshick`

Dotfiles

Mine

`https://github.com/ceholden/dotfiles`