UNIX

Basic commands:

ls # list dir content
mkdir *folder_name* # create directory called "folder_name"
rmdir *folder_name* # remove an empty directory as long as it is empty
rm -r *folder_name* # remove dir that is not empty, "r" stands for recursive
cd # change dir
../ # two dots represents parent dir
. # single dot represents current workingdir
cd ~/*projects* # concatenate with forward slashes
cd ../.. # change to two parent layer beyond
cd - # whatever dir you were before
cd # return to the home dir
pwd # retorna o diretório atual
mv ***path-to-file path-to-destination-directory*** # move uma pasta de local
rm ***filename-1 filename-2 filename-3*** *...* # remove um arquivo permanentemente
cp # copia um arquivo em vez de movê-lo
less ***nome-do-arquivo*** # visualiza um arquivo
- **q** para sair do less

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2* if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command* | grep *pattern – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

Installation

Install from source:
./configure
make
make install
dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

* use with extreme caution.

Manipulação de diretórios:

mkdir cria um diretório, exemplo: mkdir docs

rmdir exclui um diretório (se estiver vazio)

rm -rf exclui um diretório e todo o seu conteúdo

cd entra num diretório (exemplo: cd docs) ou retorna para *HOME*

cd ~ vai direto para o diretório home do usuário autenticado.

cd - volta ao último diretório acessado

pwd exibe o local do diretório atual

ls listar o conteúdo do diretório

ls -alh mostra o conteúdo detalhado do diretório

ls -ltr mostra os arquivos no formado longo(l) em ordem inversa(r) de data (t)

du -msh mostra o tamanho do diretório em Megabytes

whereis mostra onde se encontra determinado arquivo (binários), exemplo: whereis ls

which mostra qual arquivo binário é chamado pelo shell quando chamado via linha de comando

Comandos manipulação de arquivos:

cat mostra o conteúdo de um arquivo binário ou texto

tac semelhante ao cat, mas inverte a ordem

tail mostra as últimas 10 linhas de um arquivo (util para ler logs)

head mostra as primeiras 10 linhas de um arquivo

less mostra o conteúdo de um arquivo de texto com controle

vi editor de texto

vim versão melhorada do editor vi

nano editor de texto

rm remoção de arquivos (também remove diretórios)

cp copia diretórios, exemplo: 'cp -r' copia recursivamente

mv move ou renomeia arquivos e diretórios

chmod altera as permissões de arquivos ou diretórios

chown altera o dono de arquivos ou diretórios

cmd > txt cria um novo arquivo (txt) com o resultado do comando (cmd)

cmd >> txt adiciona o resultado do comando (cmd) ao fim do arquivo (txt)

touch touch foo.txt - cria um arquivo foo.txt vazio; também altera data e hora de modificação para **agora**

> arquivo.txt mais rápido que o touch para criação de arquivos

split divide um arquivo

recode recodifica um arquivo ex: recode iso-8859-15..utf8 file_to_change.txt

[mc] gerenciador de arquivos em modo texto

Comandos para administração:

man mostra informações sobre um comando

adduser adiciona usuários

addgroup adiciona grupos

apropos realiza pesquisa por palavra ou string

df reporta o uso do espaço em disco do sistema de arquivos

dmesg exibe as mensagens da inicialização(log)

du exibe estado de ocupação dos discos/partições

find comando de busca ex: find ~/ -cmin -3

userdel remove usuários

chfn altera informação relativa a um utilizador

who informa quem está logado no sistema

whoami informa com qual usuário você está logado

passwd modifica senha (password) de usuários

umask define padrões de criação de arquivos e diretórios

ps mostra os processos correntes

ps -aux mostra todos os processos correntes no sistema

kill manda um sinal para um processo. Os sinais SIGTERM e SIGKILL encerram o processo.

killall manda um sinal para todos os processos.

su troca para o super-usuário root (é exigida a senha)

su user troca para o usuário especificado em 'user' (é exigida a senha)

chown altera a propriedade de arquivos e pastas (dono)

env mostra variáveis do sistema

ntsysv exibe e configura os processos de inicialização

Comandos para manipulação de rede:

ifconfig mostra as interfaces de redes ativas e as informações relacionadas a cada uma delas

route mostra as informações referentes as rotas

mtr mostra rota até determinado IP

nmap lista as portas de sistemas remotos/locais atras de portas abertas. Pode verificar sistema operacional em execução no host remoto.

netstat exibe as portas e protocolos abertos no sistema.

iptraf analisador de trafego da rede com interface gráfica baseada em diálogos

tcpdump sniffer muito popular. Sniffer é uma ferramenta que "ouve" os pacotes que estão passando pela rede.

traceroute traça uma rota do host local até o destino mostrando os roteadores intermediários

nslookup consultas a serviços DNS

dig testa a configuração do servidor DNS

REPRODUCIBLE REPORTS

-Usados para gerar relatórios ao final de um código.

RMARKDOWN

-Com ele, podemos gerar relatórios automáticos que adicionam os gráficos ao documento sem a necessidade de colocá-los um a um.
markdowntutorial.com
Código para relatório simples:
https://raw.githubusercontent.com/rairizarry/murders/master/report.Rmd

Key points:
-R Markdown is a format for *literate programming* documents. Literate programming weaves instructions, documentation and detailed comments in between machine executable code, producing a document that describes the program that is best for human understanding.
-Start an R markdown document by clicking on File > New File > the R Markdown
-The output could be HTML, PDF, or Microsoft Word, which can be changed in the -header output, e.g. pdf_document / html_document/p>

Code:

a sample code chunk
```{r}
summary(pressure)
```
When echo=FALSE, code will be hided in output file
```{r echo=FALSE}
summary(pressure)
```
use a descriptive name for each chunk for debugging purpose
```{r pressure-summary}
summary(pressure)

```

KNITR

-Usado para compilar os documentos, podendo escolher o formato do arquivo.
-Knitr basics
-Knitr website

Code:

output: html_document
output: pdf_document
output: word_document
output: github_document

GIT & GITHUB

-curso git codecademy
-github guide

- Recap: there are four stages: working directory, staging area, local repository, and upstream repository
- Clone an existing upstream repository (copy repo url from clone button, and type "**git clone <url>**"), and all three local stages are the same as upstream remote.
- The working directory is the same as the working directory in Rstudio. When we edit files we only change the files in this place.
- **git status**: tells how the files in the working directory are related to the files in other stages
- edits in the staging area are not tracked by the version control system by default - we add a file to the staging area by **git add command**

- **git commit**: to commit files from the staging area to local repository, we need to add a message stating what we are doing by git commit -m "something"
- **git log**: keeps track of all the changes we have made to the local repository
- **git push**: allows moving from the local repository to upstream repository, only if you have the permission (e.g. if it is yours)
- **git fetch**: update local repository to be like the upstream repository, from upstream to local
- **git merge**: make the updated local sync with the working directory and staging area
- To change everything in one shot (from upstream to working dir), use **git pull** (equivalent to combining git fetch + git merge)
- Make a local git repository: On the local machine, in the project directory, use **git init**. Now git starts tracking everything in the local repo.
- Now we need to start moving files into our local repo and connect local repo to the upstream remote by **git remote add origin <url>**
- **Note:** The first time you push to a new repository, you may also need to use these git push options: **git push --set-upstream origin master**. If you need to run these arguments but forget to do so, you will get an error with a reminder.

UNIX ARGUMENTS

- Arguments typically are defined using a dash (**-**) or two dashes (**--**) followed by a letter of a word.
- **r**: recursive. For example, rm -r <directory-name>: remove all files, subdirectories, files in subdirectories, subdirectories in subdirectories, etc.
- Combine arguments: rm -rf directory-name
- **ls -a**: Shows all files in the directories including hidden files (e.g. .git file when initializing using git init) (a for all).
- **ls -l**: Returns more information about the files (i.e. l for long).
- **ls -t**: Shows files in chronological order.
- **ls -r**: Reverses the order of how files are shown.
- **ls -lart**: Shows more information for all files in reverse chronological order.

TRICKS

- ***** means any number of any combination of characters. Specifically, to list all html files: ls *****.html and to remove all html files in a directory: rm *****.html.
- **?** means any single character. For example, to erase all files in the form file-001.html with the numbers going from 1 to 999: rm file-**???**.html.
- Combined wild cards: rm file-001.* to remove all files of the name file-001 regardless of suffix.
- **Warning**: Combining rm with the * wild card can be dangerous. There are combinations of these commands that will erase your entire file system without asking you for confirmation. Make sure you understand how it works before using this wild card with the rm command.

ENVIRONMENT VARIABLES AND SHELLS

- In Unix, variables are distinguished from other entities by adding a **$** in front. For example, the home directory is stored in **$**HOME.
- See home directory: echo **$**HOME
- See them all: **env**
- See what shell is being used: (most common shell is bash)
- Change environmental variables: (*Don't actually run this command though!*) export PATH = /usr/bin/

EXECUTABLES, PERMISSIONS, AND FILE TYPES

- In Unix, all programs are files. They are called executables. So, ls, mv, and git are all files.
- To find where these program files are, use which. For example, which git would return /usr/bin/git.
- Type ls /usr/bin to see several executable files. There are other directories that hold program files (e.g. Application directory for Mac or Program Files directory in Windows).
- Type echo $PATH to see a list of directories separated by ":".
- Type the full path to run the user-created executables (e.g ./my-ls).
- Regular file -, directory d, executable x.