Introducción a BASH

Lic. Marcos Mazzini - CCAD / UNC
CPA CONICET

Módulo 1: Primer Contacto

¿Qué es Shell?

- Es un **programa** especial que permite controlar un equipo mediante texto
- También se conoce como **consola**, **terminal** o **CLI** (command-line-interface)
- Se llama shell porque vendría a ser la última capa alrededor del kernel
- Es un intérprete de comandos REPL
  - Read: leer el comando
  - Eval: evaluar, interpretar, verificar, ejecutar
  - Print: imprimir el resultado (o el error)
  - Loop: repetir

Permite

- Administrar **archivos** (copiar, mover, borrar,editar)
- Administrar **procesos** (ejecutar y cerrar aplicaciones)
- Ejecutar **scripts** (secuencias de comandos guardados en un archivo)
- Monitorear y configurar el Sistema Operativo

Es la forma estándar de interactuar con servidores, sistemas remotos o sin interfaz gráfica

Shells

- La gran mayoría de los SO ofrece una terminal (o varias)
- Hay varias terminales para LINUX, **BASH** es la más extendida

Bash (Bourne-again Shell)

- Es una evolución de **Bourne shell** de UNIX (sh)
- Bash es compatible con sh, cumple con la estandarización POSIX y agrega extensiones

Emuladores de terminal

Como la mayoría de los usuarios usa una interfaz gráfica lo que se termina utilizando es un **emulador de terminal**

- Windows: PuTTY, MoabXterm
- Linux: Terminal, Konsole, XTerm, Terminator
- Mac OS X: Terminal (default), iTerm 2

Iniciando una terminal: El Prompt

Cada vez que abrimos una terminal o nos conectamos a un sistema remoto veremos el motd (message of the day) y el **prompt**

El **prompt** es personalizable, como todo en linux, así que podría no verse igual en todos lados.

El Prompt

```
usuario@localhost:~$
```

- **usuario**: Nombre de usuario del usuario actual
- **localhost**: nombre del equipo al cual ingresamos
- **~**: carpeta actual, ~ es un atajo a **/home/usuario**
- **$**: fin del prompt, inicio del input

Prompt del usuario “root”, logueado al servidor “server” en la carpeta “/var/log”:

```
root@server:/var/log#
```

Los Comandos

- Se ejecutan tipeando el nombre del archivo ejecutable seguido de ENTER
- Puede ser un **binario** o un **script**
- Al comando corriendo se lo conoce como **proceso**
- Por defecto un proceso corre en **foreground** (primer plano), esto implica que hay que esperar a que termine para poder ejecutar el siguiente comando (**Ctrl-c** envía señal de finalización)

Comandos: sintaxis para argumentos

Sin argumento	**$ls**
Con argumento	**$ls /usr/bin**
Con opciones (flags)	**$ls -l** (corta) **$ls --all** (larga) **$ls -la** (combinada ls -l -a)
Con argumento y opciones	**$ls -la /home**

comandos: ls, echo

```
[usuario@localhost ~]$ ls
curso1  curso2
[usuario@localhost ~]$ ls -l
drwxrwxr-x. 2 usuario usuario 4096 abr 23 10:57 curso1
drwxrwxr-x. 3 usuario usuario 4096 abr 23 11:06 curso2
[usuario@localhost ~]$ ls -la /home
total 12
drwxr-xr-x.  3 root   root 4096 ene 24 11:59 .
dr-xr-xr-x. 19 root   root 4096 abr 23 09:38 ..
drwx------. 29 usuario usuario 4096 abr 23 09:38 usuario
[usuario@localhost ~]$echo una   linea   con   muchos  espacios
una linea con muchos espacios
```

los paths (rutas)

- Se usa **/** (shift-7) como separador, al revés que en windows
- Sirve para referenciar archivos o carpetas en una estructura de árbol
- La raíz es /
- Puede ser **absoluto** (completo) o **relativo** (a partir de la posición actual)

comandos: pwd, cd, cp, ls

```
[usuario@localhost ~]$ pwd
/home/usuario
[usuario@localhost ~]$ cd curso1
[usuario@localhost curso1]$ cd ../curso2
[usuario@localhost curso2]$ cp ../curso1/texto1.txt .
[usuario@localhost curso2]$ ls
texto1.txt
```

Historial y Autocompletado (tab)

Cada línea ingresada queda registrada y es navegable con las flechitas.

Se puede revisar todo con el comando **history** y repetir alguno sin volver a tipear

La tecla <tab> autocompleta de forma similar al predictivo del celular para:

- comandos
- nombres de archivos
- variables

comandos: history, !

```
[usuario@localhost ~]$history
1000  ls
1001  ls /usr/bin
1002  ls -l
1003  ls --all
1004  ls -la
1005  ls -la /home
1006  history
[usuario@localhost ~]$ !1005
ls -la /home
total 12
drwxr-xr-x.  3 root   root 4096 ene 24 11:59 .
dr-xr-xr-x. 19 root       root 4096 abr 23 09:38 ..
drwx------. 29 usuario usuario 4096 abr 23 09:38 usuario
```

comandos

```
[usuario@localhost ~]$ ls /usr/<tab><tab>
bin/     games/   lib/     libexec/ sbin/    src/
etc/     include/ lib64/   local/   share/   tmp/
[usuario@localhost ~]$ ls /usr/lib<tab><tab>
lib/     lib64/   libexec/
[usuario@localhost ~]$ ls /usr/lib
```

links

Son atajos o accesos directos y se muestran como origen->destino

Lo que más se utiliza es el **soft-link** que puede ser a un archivo o a una carpeta, si el destino se borra el link queda roto

El **hard-link** solo puede ser a un archivo y referencia el contenido de este (inodo), es como un nombre alternativo

comandos: ln, rm

```
[usuario@localhost ~]$ cd curso1
[usuario@localhost curso1]$ ln -s ../curso2 c2
[usuario@localhost curso1]$ ll
total 0
lrwxrwxrwx. 1 usuario usuario 9 abr 23 18:16 c2 -> ../curso2
-rw-rw-r--. 1 usuario usuario 0 abr 23 10:35 texto1.txt
-rw-rw-r--. 1 usuario usuario 0 abr 23 10:36 texto2.txt
[usuario@localhost curso1]$ rm c2
[usuario@localhost curso1]$ ll
total 0

-rw-rw-r--. 1 usuario usuario 0 abr 23 10:35 texto1.txt
-rw-rw-r--. 1 usuario usuario 0 abr 23 10:36 texto2.txt
```

Archivos

En LINUX todo se abstrae como un archivo y consecuentemente existen varios comandos para manejar líneas de texto.

El **editor de texto** básico es **nano**.

vi y **emacs** son editores avanzados que una vez que se domina alguno ofrece muchas ventajas

comandos: cat, nano

```
[usuario@localhost curso1]$ cat texto1.txt
linea1
linea2
linea 3
[usuario@localhost curso1]$ cat texto1.txt texto2.txt
linea1
linea2
linea3
linea1
linea2
linea 3
[usuario@localhost curso1]$nano texto1.txt
```

comandos: head, tail, less

```
$ less /usr/share/dict/words
1080
10-point
10th
11-point
12-point
16-point
18-point
1st
2
20-point
2,4,5-t
2,4-d
:
```

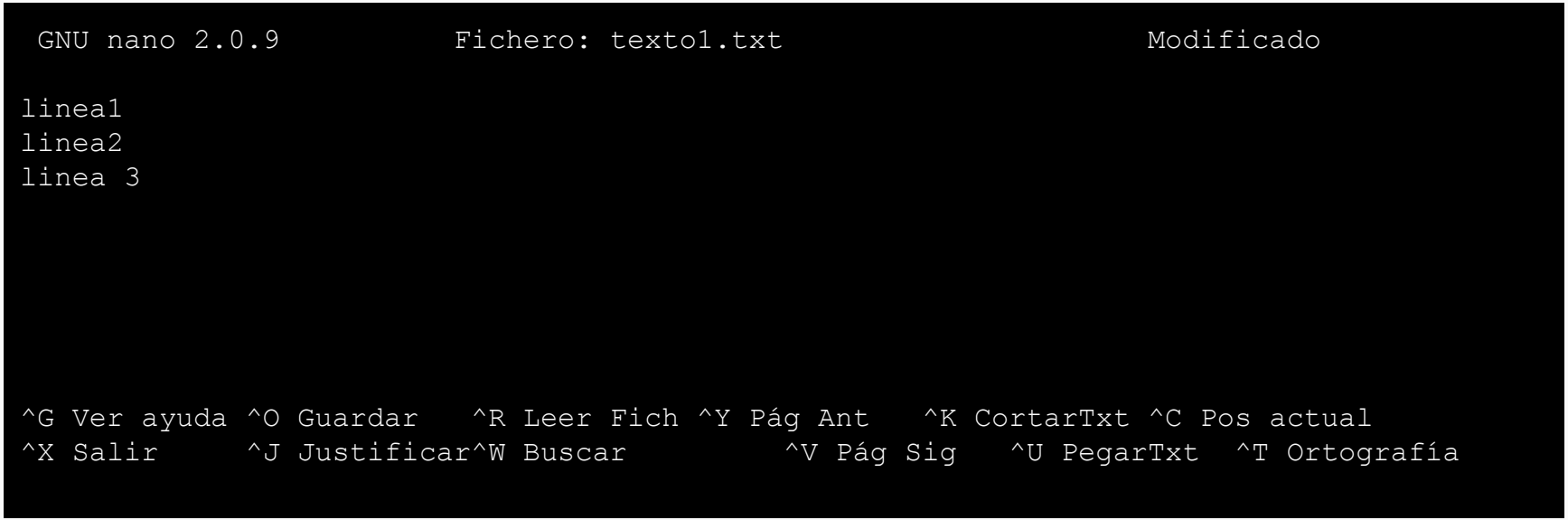
<arriba>, <abajo>,pgup,pgdwn, q

```
[usuario@localhost ~]$ head -5 /usr/share/dict/words
1080
10-point
10th
11-point
12-point
[usuario@localhost ~]$ tail -5 /usr/share/dict/words
zyzzyvas
ZZ
Zz
zZt
ZZZ
```

```
$ <comando_salida_extensa> | less
```

comandos : nano

^<tecla> = ctrl-<tecla>



comandos: grep, *(glob)

```
[usuario@localhost curso1]$ grep ea2 texto1.txt
linea2
[usuario@localhost curso1]$ grep "linea 3" texto*
texto1.txt:linea 3
texto2.txt:linea 3
[usuario@localhost curso1]$
```

Sistema de archivos

Las distribuciones LINUX tienen algunas diferencias entre sí pero en general tienen **las mismas carpetas**. Es porque tratan de cumplir con la estandarización POSIX que define los usos para cada una.

A continuación veremos qué finalidad tiene cada carpeta:

comandos

```
[usuario@localhost /]$ ls -1
bin
boot
dev
etc
home
lib
lib64
lost+found
media
```

```
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

/ (raiz-root)	C: y otras particiones también. No hay nada fuera del raiz
/bin	Binarios esenciales del usario: tienen que estar si o si aun en recovery mode.En /usr/bin van el resto. En /sbin van los de admin
/boot	Archivos necesarios para iniciar el sistema
/cdrom - /media	Punto de montaje para medios removibles (/cdrom es historico)
/dev	Representan dispositivos, particiones (/dev/sda1) o dispositivos virtuales (/dev/random o /dev/null)
/etc	Archivos de configuración de todo el sistema. Son modo texto editables con cualquier editor. Las configuraciones por usuario van en el home de cada uno
/home	Hay una carpeta para cada usuario. Es el único lugar donde cada usuario puede escribir (a no ser que sean root)
/lib, /lib64	Librerías esenciales para los binarios esenciales (/bin). El resto está en /usr/lib

lost+found	Archivos perdidos por un crash del sistema (favorece el recovery)
/mnt	Punto de montaje temporal (igual se puede montar en cualquier lugar)
/proc	Similar a /dev representa información del sistema y de los procesos
/root	Es el home del usuario root. No confundir con /
/run	Archivos temporales, a diferencia de /tmp estos no se borran
/srv	Datos para los servicios provistos por el sistema. Ej. httpd o ftp
/tmp	Archivos temporales, podrían ser borrados en cualquier momento
/usr	Binaros de usuario y datos de solo lectura (/usr/share y /usr/local contienen los que fueron compilados por el usuario)
/var	Es la contraparte escritura de /usr. Los logs van en /var/log

comandos: mv, mkdir, touch, rm, rmdir

```
[usuario@localhost ~]$ cd curso1
[usuario@localhost curso1]$ cp texto1.txt texto_copia.txt
[usuario@localhost curso1]$ mv texto_copia.txt texto1_copia.txt
[usuario@localhost curso1]$ mkdir extra
[usuario@localhost curso1]$ touch extra/algo
[usuario@localhost curso1]$ ls extra/
total 0
-rw-rw-r--. 1 usuario usuario 0 abr 23 18:57 algo
[usuario@localhost curso1]$ rm extra/algo
[usuario@localhost curso1]$ rmdir extra
```

Usuarios y grupos

Linux es un sistema **multiusuario**, el sistema de por sí cuenta con varios

Cada usuario tiene un nombre y **número** único (UID)

También pertenece a uno o más **grupos** (GID)

root es el nombre de usuario para administración, es superusuario y tiene acceso a todo el sistema (lo tienen también usuarios miembros del grupo wheel o sudoers)

comandos: id, w, su

```
[usuario@localhost ~]$ id usuario
uid=1000(usuario) gid=1000(usuario)
grupos=1000(usuario),10(wheel),984(vboxusers)
[usuario@localhost ~]$ w
 19:01:22 up  9:23,  3 users,  load average: 0,35, 0,57, 0,57
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
usuario  tty1     :0               09:38    9:23m  3:41   1.22s mate-session
usuario  pts/0    :0               10:12    7:32m  0.04s  0.04s /bin/bash
usuario  pts/1    :0               17:15    2.00s  0.19s  0.00s w
[usuario@localhost ~]$ su
Contraseña:
[root@localhost usuario]# cd
[root@localhost ~]# pwd
/root
```

Permisos

Cada archivo y carpeta pertenece a **un usuario** y a **un grupo**. Sobre éstos se disponen permisos de **lectura**, **escritura** y **ejecución** para:

- el propietario (owner)
- el grupo
- otros

Permisos

Hay 3 cosas que se pueden hacer un archivo:

read **w**rite e**x**ecute

```
$ls -l archivo.txt
rw-rw-r--  usuario   grupo    …
```

- el usuario puede leer y escribir
- el grupo puede leer y escribir
- los otros solo puede leer

Permisos (carpetas)

```
$ ls -l
drwxr-xr-x    usuario  usuario  carpeta
```

- **r**: listar
- **w**: crear archivos
- **x**: acceder y modificar archivos

También se pueden ver los permisos en formato binario, permitido (1/0)

rw-	r- -	r - -
110	100	100
6	4	4

comandos: chmod, chown, sudo, exit

```
[usuario@localhost curso2]$ touch nada
[usuario@localhost curso2]$ ll
-rw-rw-r--. 1 usuario usuario 0 abr 23 20:41 nada
[usuario@localhost curso2]$ chmod go-rw nada
[usuario@localhost curso2]$ ll
-rw-------. 1 usuario usuario 0 abr 23 20:41 nada
[usuario@localhost curso2]$ chown root.root nada
chown: cambiando el propietario de «nada»: Operación no permitida
[usuario@localhost curso2]$ sudo chown root.root nada
[usuario@localhost curso2]$ ll
-rw-------. 1 root root 0 abr 23 20:41 nada
[usuario@localhost curso2]$ cat nada
cat: nada: Permiso denegado
[usuario@localhost curso2]$ su
Contraseña:
[root@localhost curso2]# cat nada
[root@localhost curso2]# exit
```

Procesos

Como dijimos antes un comando ejecutándose se convierte en un **proceso** al que se le asigna un identificador de proceso **PID.**

Un proceso puede crear otros procesos hijos, cuando termina el padre terminan todos los hijos.

Hay procesos que se están ejecutando todo el tiempo **en segundo plano** a los que se los conoce como **demonios** o **daemons.**

Los demonios proveen servicios a los otros procesos.

comandos: ps, pgrep, kill, pkill

```
[usuario@localhost ~]$ ps
  PID TTY          TIME CMD
 4182 pts/1  00:00:00 bash
 5171 pts/1  00:00:00 ps
[usuario@localhost ~]$ pgrep -u usuario
.... <todos los del usuario> ...
[usuario@localhost ~]$ pgrep firefox
2010
[usuario@localhost ~]$ kill -9 2010
[usuario@localhost ~]$ pkill firefox
```

comandos: top

```
[usuario@localhost ~]$ top
top - 19:18:53 up  9:41,  3 users,  load average: 0,41, 0,41, 0,48
Tasks: 155 total,   1 running, 154 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3,2 us,  0,8 sy,  0,0 ni, 94,8 id,  1,1 wa,  0,0 hi,  0,1 si,  0,0 st
KiB Mem :  3323796 total,   494604 free,  2451908 used,   377284 buff/cache
KiB Swap:  7812092 total,  7480368 free,   331724 used.   465384 avail Mem

  PID USER      PR  NI    VIRT  RES   SHR S  %CPU %MEM     TIME+ COMMAND
 2131 usuario   20   0 3447472 1,261g  93932 S   6,0 39,8  85:51.02 Web Content
 1425 root      20   0  355440  83152  68584 S   3,7  2,5   4:06.71 X
 2010 usuario   20   0 9198812 518668 104732 S   2,3 15,6  23:31.93 firefox
 4173 usuario   20   0  643660  14252   6068 S   1,3  0,4   0:07.54 mate-termi+
 5203 usuario   20   0  157716   2228   1544 R   0,3  0,1   0:00.03 top
    1 root      20   0  128164   2320    688 S   0,0  0,1   0:02.16 systemd
    2 root      20   0       0          0     0 S   0,0  0,0   0:00.01 kthreadd
    3 root      20   0       0          0     0 S   0,0  0,0   0:00.34 ksoftirqd/0
    5 root       0 -20       0          0     0 S   0,0  0,0   0:00.00 kworker/0:+
```

comandos: ¿cuántos más quedan?

- builtins
  - cd, echo, exit, pwd, history,...
- paquete coreutils (~105) - POSIX
  - https://www.gnu.org/software/coreutils/manual/html_node/index.html
- instalados por la distribución, el admin o el usuario

investigar: find, wget, tar, gunzip

Cómo obtener ayuda

- `$<comando> --help`
- `$<comando> -h` (si -h no se usa para otra cosa)
- `$help <comando>` (para bultins)
- `$man <comando>`
- `$man -k <palabra clave>`
- `$apropos <palabra clave>`

- explainshell
- stackoverflow

SSH

Secure Shell

Es un protocolo de red criptográfico que permite realizar tareas privadas sobre una red insegura

Es la forma más extendida de conectarse a una terminal remota

Al conectarse por ssh se abre un shell y todos los comandos que introducimos se transmiten hacia el servidor por un túnel encriptado

SSH

La conexión SSH está implementada con un modelo **cliente-servidor**

Para establecer la conexión el **servidor** debe estar ejecutando un proceso (sshd) que escuche por conexiones en un puerto (22 por default). El servidor se encarga de **autenticar la conexión** y **lanzar el entorno** cuando el usuario provee las credenciales correctas.

El **cliente** (máquina local) debe ejecutar un **cliente ssh**, que se encarga de enviar el nombre de usuario, las credenciales y cada comando que se va ingresando.

SSH

El **cliente SSH** para línea de comandos viene instalado por defecto en la amplia mayoría de las distribuciones LINUX

En otros sistemas se puede utilizar algún emulador de terminal de los listados antes.

Para windows recomendamos **moabXterm**

SSH

Para conectarse hay que tener una cuenta de usuario **en el servidor remoto** (la cuenta local podría ser distinta).

Los mecanismos más extendidos para autenticación son **contraseña** y **llave publico-privada**

comandos: ssh, w

```
[usuario@localhost ~]$ ssh mmazzini@mendieta.ccad.unc.edu.ar
Last login: Mon Apr 23 11:16:47 2018 from 152.168.141.237
…<motd>
Pedidos al soporte del CCAD soporte@ccad.unc.edu.ar.
**************************
[mmazzini@mendieta ~]$ w
 11:30:36 up 14 days,  3:47, 15 users,  load average: 0,22, 0,27, 0,43
USER     TTY        FROM             LOGIN@     IDLE   JCPU   PCPU WHAT
jborioni pts/1      200.12.138.65    Fri14      2days  0.66s  0.66s -bash
bc       pts/8      :50              16Apr18    4days  0.01s  0.01s bash
gsoldano pts/9      200.16.18.156    10Apr18    6days  1.16s  1.16s -bash
dmasone  pts/10     200.12.137.60    10:11      1:18m  0.05s  0.05s -bash
mmazzini pts/13     152.168.141.237  11:30      0.00s  0.02s  0.01s w
jolmos   pts/15     200.16.18.224    10:48      0.00s  0.29s  0.29s -bash
[mmazzini@mendieta ~]$ exit
logout
Connection to mendieta.ccad.unc.edu.ar closed.
```

SSH

El login por **contraseña** es vulnerable a ataques de fuerza bruta y es más engorroso de administrar.

El login por **llave público-privada** es más cómodo de usar porque permite conectarse sin ingresar la contraseña.

SSH - Llave público-privada

Consiste en un **par de archivos** criptográficos:

1. El correspondiente a la **llave pública** que puede ser distribuido libremente sin preocuparse.
2. El correspondiente a la **llave privada** que debe resguardarse y **nunca exponerlo** a nadie.

SSH - LLave público-privada

Generar el par en la máquina cliente:

$ssh-keygen -t rsa

la llave privada podría a su vez protegerse con una contraseña

En la máquina cliente tendremos:

/home/<usuario>/.ssh/id_rsa.pub -> llave **pública**

/home/<usuario>/.ssh/id_rsa -> llave **privada (RESGUARDAR)**

comandos: ssh-keygen

```
[usuario@localhost ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/usuario/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/usuario/.ssh/id_rsa.
Your public key has been saved in /home/usuario/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:7qNHM6kHjiQJrKerrTstugFopiOgyJNtTkGiYAiazuY usuario@localhost
```

comandos

```
The key's randomart image is:
+---[RSA 2048]----+
|.                |
|+.               |
|*o .             |
|O.o              |
|*B o    S.       |
|& * o ..=        |
|BE * o +.o       |
|=+* . o.+        |
|XB..  .+..       |
+----[SHA256]-----+
```

comandos

```
[usuario@localhost ~]$ ls -l .ssh
total 28
-rw-------. 1 usuario usuario 1007 abr 14 07:54 authorized_keys
-rw-------. 1 usuario usuario   98 abr 14 07:54 config
-rw-------  1 usuario usuario 1675 abr 23 11:25 id_rsa
-rw-r--r--  1 usuario usuario  414 abr 23 11:25 id_rsa.pub
-rw-r--r--  1 usuario usuario  895 abr 16 20:34 known_hosts
```

SSH - LLave público-privada

Para autenticarse el usuario debe tener **el par** en su computadora **local** y la **llave publica** debe estar copiada en el **servidor remoto** para la cuenta remota en ~/.ssh/authorized_keys

```
[usuario@localhost ~]$ssh-copy-id <usuario_remoto>@<servidor>
```

Hay sistemas en los que el método de contraseña directamente no se utiliza

En estos casos “ssh-copy-id” no funciona y se ofrece una **interfaz web** para subir la llave pública o enviarla por **mail al soporte**.

comandos

```
[mmazzini@mendieta ~]$ cat .ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDPXJ7oVYamyjf2f2CZC5UbFMgCUY4lgpC1l3m09ElNdcAfGEQXZFMmx8kG
ZBSDHn+/7HcpOEGeBGH6UmFJWkezfbgneQRRFfrNSqlLn6sDWZn0cdLOFa7CNByDk6tAy4aQFQEVxn69r4KNsvoo
Vis5TrSCJH7Yi1BHP5lrSm/xR+EGxr/wxrDh0TqBcoIVeeQDmYZgFBKxBcKywHeQPv1qXB9xBZ+4W4mm1jote3vo
52UB4rv6Tw0npaYFTmiLAnWORbURo/pxx7rENnm6e+7OhUG0pCfQiT+MZ7JU0I9gMfhEQscBHzy7rYZtFMY3HzjN
KwVHtGeVRU8n2aCB+Til marcos@notebook
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCtxt0y9oenaAlOKIl+1wSr1LtScWWGzh+119YoOUMZ9hsjq2D7aqGIx4EF
H+Y4v86i8YoF7sPmWjwbXVUdg4PDWf9UZAL/dvymaUZ5DsEF+1mKA2nmk1MnUDg6gA+NwuNo8sW94ZJuSi77qnjd
UgwNdrv4l69THFpEvM5xs+MqAvy9dFqf1wFNET3g9tT5ealYLj+oVhuyYdJHQ4N0vG9QGnFiCfByeo4zk42lcmxz
oA4WUz2qTKpeJ3GKfEk5xh4u/tNPlgGcTycwGycVGiGtlkrLh1tL4+6Vt8oajH3g4oZNlC3JeBeIi8iGwkTqNFdT
lX1Fzj7ggJe/TsaEZSyd celu
```

más comandos

Comandos

```
#poweroff

#shutdown -r now  (reiniciar ahora)

#shutdown -h +23:59 (apagar antes de medianoche)
```

Notas finales

Practicar comandos básicos resolviendo el misterio

- https://github.com/veltman/clmystery

Desafíos incrementales online para mejorar las habilidades

- https://cmdchallenge.com/

Consola gratuita para usuarios con cuenta google

- https://console.cloud.google.com/cloudshell

Consola Bash nativa para Windows 10

- https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10

Practiquen para la clase que viene! :)

CCAD - UNC

Centro de Computación de Alto Desempeño

Contamos con varios cluster con variadas aplicaciones para simulaciones computacionales:

http://ccad.unc.edu.ar

Solicitar acceso en

servicios -> pedidos de cuentas

Si van a instalar LINUX:

Elegir una distribución amigable con los principiantes

- Ubuntu
- Mint
- Fedora

FLISOL 2018

Festival Latinoamericano de Instalación de Software Libre

- Sábado 28 de abril de 10:00hs a 18:00hs
- Biblioteca de la Facultad de Ciencias Económicas - UNC
- https://librebase.org.ar/inscripciones
- https://flisol.info/FLISOL2018/Argentina/Cordoba

¿Preguntas?