

```

5.5%] 4[|||||
0.0%] 5[
9.2%] 6[||
0.0%] 7[

7.3%] 8[|]
0.0%] 9[|]
2.6%] 10[|]
0.0%] 11[|]

2.6%] 12[
0.7%] 13[|]
1.3%] 14[|]
0.6%] 15[

0.0%]
1.3%]
0.6%]
0.0%]

3.27G/15.3G] Tasks: 128, 1486 thr, 217 kthr; 1 running
678M/977M] Load average: 0.73 1.08 1.02
Uptime: 3 days, 14:34:49

```

RT	RES	SHR	S	CPU%	MEM%	TIME+	Command
5M	198M	126M	S	7.2	1.3	40:10.17	/usr/bin/kwin_wayland --wayland-fd 7 --socket wayland-0 --xwayland-fd 8 --xwayland-fd 9 --xwayland-display :1 --xwayland-xauthority /run/user/1000/xauth_zVDchs --xwaylan
7G	704M	250M	S	4.6	4.5	49:21.78	/usr/lib/firefox-esr/firefox-esr
6M	383M	88320	S	3.9	2.4	3h04:31	/usr/bin/plasmashell --no-respawn
0M	166M	112M	S	3.3	1.1	0:00.25	/usr/bin/spectacle
0M	159M	96308	S	2.6	1.0	15:10.83	/usr/lib/firefox-esr/firefox-esr -contentproc -childID 4 -isForBrowser -prefsLen 36809 -prefMapSize 219042 -jsInitLen 277276 -parentBuildID 20230724114731 -appDir /usr/l
84	7472	3488	R	2.6	0.0	0:00.42	htop
5M	198M	126M	S	2.0	1.3	2:04.55	/usr/bin/kwin_wayland --wayland-fd 7 --socket wayland-0 --xwayland-fd 8 --xwayland-fd 9 --xwayland-display :1 --xwayland-xauthority /run/user/1000/xauth_zVDchs --xwaylan
5M	19568	8440	S	1.3	0.1	39:39.80	/usr/bin/ksystemstats
7G	704M	250M	S	1.3	4.5	5:38.27	/usr/lib/firefox-esr/firefox-esr
7G	704M	250M	S	1.3	4.5	4:29.75	/usr/lib/firefox-esr/firefox-esr
7G	704M	250M	S	1.3	4.5	0:09.06	/usr/lib/firefox-esr/firefox-esr
1M	181M	98M	S	1.3	1.2	2:54.91	/usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefsLen 36809 -prefMapSize 219042 -jsInitLen 277276 -parentBuildID 20230724114731 -appDir /usr/l
0M	159M	96308	S	1.3	1.0	4:28.05	/usr/lib/firefox-esr/firefox-esr -contentproc -childID
0M	11844	8168	S	0.7	0.1	2:32.34	/usr/lib/x86_64-linux-gnu/libexec/kf5/kio_http_cache_cl
7G	704M	250M	S	0.7	4.5	0:42.13	/usr/lib/firefox-esr/firefox-esr
7G	704M	250M	S	0.7	4.5	3:31.73	/usr/lib/firefox-esr/firefox-esr
4M	152M	112M	S	0.7	1.0	2:52.90	/usr/lib/firefox-esr/firefox-esr -contentproc -childID
5M	203M	105M	S	0.7	1.3	0:37.10	/usr/lib/firefox-esr/firefox-esr -contentproc -childID
3M	209M	102M	S	0.7	1.3	6:07.68	/usr/lib/firefox-esr/firefox-esr -contentproc -childID
1M	171M	98344	S	0.7	1.1	0:22.27	/usr/lib/firefox-esr/firefox-esr -contentproc -childID
9M	180M	111M	S	0.7	1.2	0:02.48	/usr/lib/firefox-esr/firefox-esr -contentproc -childID
0M	166M	112M	S	0.7	1.1	0:00.01	/usr/bin/spectacle
4M	8688	5132	S	0.0	0.1	5:13.55	/sbin/init
5M	43724	42384	S	0.0	0.3	3:02.01	/lib/systemd/systemd-journald
72	4884	2464	S	0.0	0.0	0:15.47	/lib/systemd/systemd-udev
2M	3488	220	S	0.0	0.0	0:37.75	/usr/libexec/accounts-daemon
68	1036	588	S	0.0	0.0	0:09.72	avahi-daemon: running [muca10-t14.local]
88	3864	3044	S	0.0	0.0	1:45.62	/usr/libexec/bluetooth/bluetoothd
08	1112	860	S	0.0	0.0	0:15.20	/usr/sbin/cron -f
28	3776	1572	S	0.0	0.0	5:03.35	/usr/bin/dbus-daemon --system --address=systemd: --nof
88	304	0	S	0.0	0.0	0:00.00	avahi-daemon: chroot helper
3M	5216	1312	S	0.0	0.0	0:12.40	/usr/lib/polkit-1/polkitd --no-debug
88	2244	1200	S	0.0	0.0	0:01.53	/usr/sbin/smartd -n
20	3752	2472	S	0.0	0.0	0:21.85	/lib/systemd/systemd-logind
5M	5060	1536	S	0.0	0.0	0:07.48	/usr/libexec/udisks2/udisksd
2M	3488	220	S	0.0	0.0	0:37.54	/usr/libexec/accounts-daemon
60	272	0	S	0.0	0.0	0:00.00	/usr/bin/nvidia-persistenced --user nvpd
3M	5216	1312	S	0.0	0.0	0:00.00	/usr/lib/polkit-1/polkitd --no-debug
						0:00.00	/usr/libexec/udisks2/udisksd
						0:05.10	/usr/lib/polkit-1/polkitd --no-debug
						0:00.60	/usr/libexec/udisks2/udisksd
						0:00.05	/usr/libexec/accounts-daemon
						0:00.34	/usr/sbin/ModemManager
						2:40.42	/usr/sbin/NetworkManager --no-daemon

Introducción a Linux y bash terminal

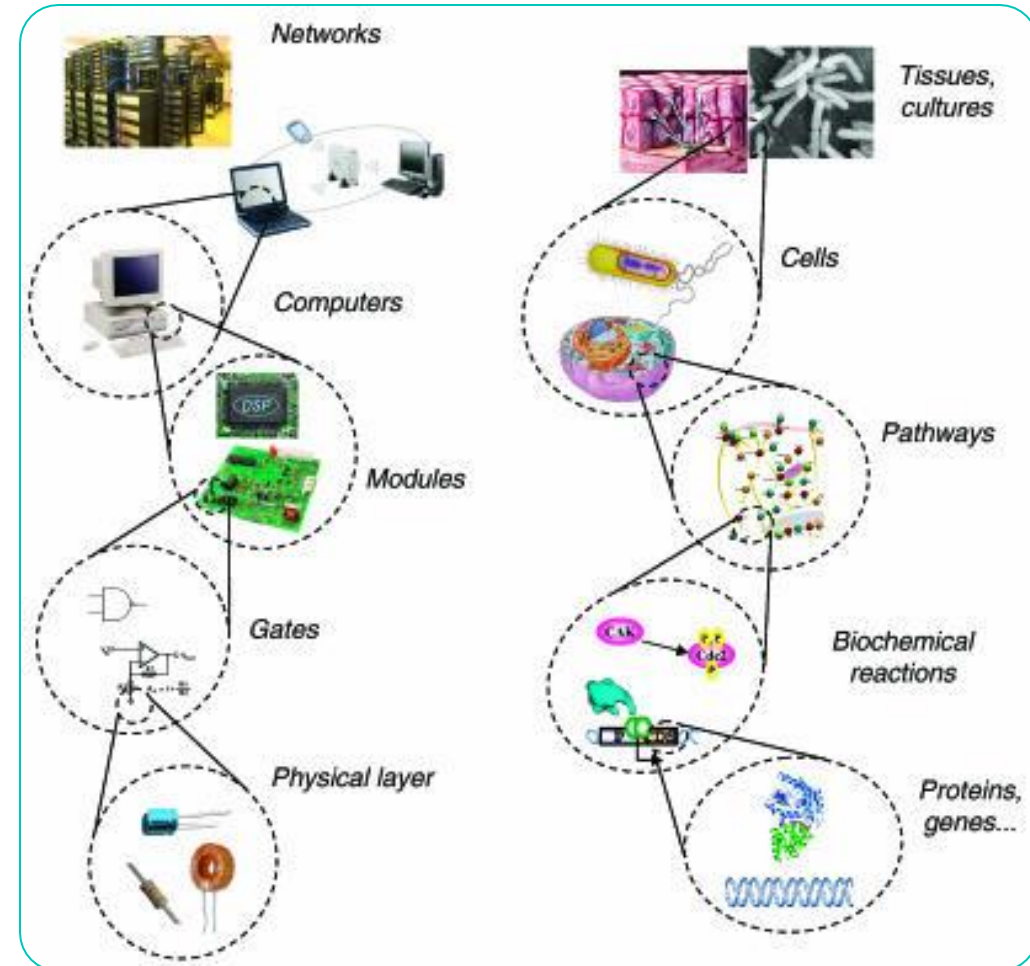
Murilo Cassiano, M.Sc.

Mensajes fundamentales

- No tengas miedo, ejecutando comandos con cuidado no puede pasar nada malo
- La shell es una herramienta que será útil para todo tipo de trabajo computacional
- Se necesita tiempo para adquirir "fluidez" en este tipo de entorno y se mejora con el tiempo.
- En esta formación intentaré darte las bases para que puedas desarrollarte por tu cuenta.

Niveles de abstracción

- También en ciencias de la vida elegimos un nivel para especializarnos
- ¡No existe un "científico completo"!
 - Tony Stark, Dr. House, etc...
- Siempre elegimos un nivel de comprensión para trabajar.



Procesador (CPU): es el cerebro del ordenador. Realiza cálculos y ejecuta instrucciones de los programas, permitiendo que el sistema funcione y realice tareas específicas.



RAM (Memoria RAM): es la memoria temporal de acceso aleatorio del ordenador. Almacena datos y programas en uso para que el procesador pueda acceder rápidamente a ellos, mejorando así el rendimiento general del sistema.



Tarjeta de Video (GPU): un componente que procesa datos relacionados con la imagen y se encarga de mostrar imágenes en el monitor. Es esencial para juegos, aplicaciones de diseño gráfico y reproducción de video en alta calidad, aliviando la carga de trabajo del procesador principal.



Disco Duro (HDD): es un dispositivo de almacenamiento permanente utilizado para guardar datos a largo plazo, como el sistema operativo, programas, archivos y documentos. Es más lento que las unidades de estado sólido (SSD), pero ofrece una gran capacidad de almacenamiento a un costo más bajo.

Placa Base (Motherboard): es el componente principal de un ordenador, al cual se conectan todos los otros dispositivos, como el procesador, la memoria RAM, las tarjetas de expansión y otros periféricos. Actúa como el "esqueleto" del sistema.

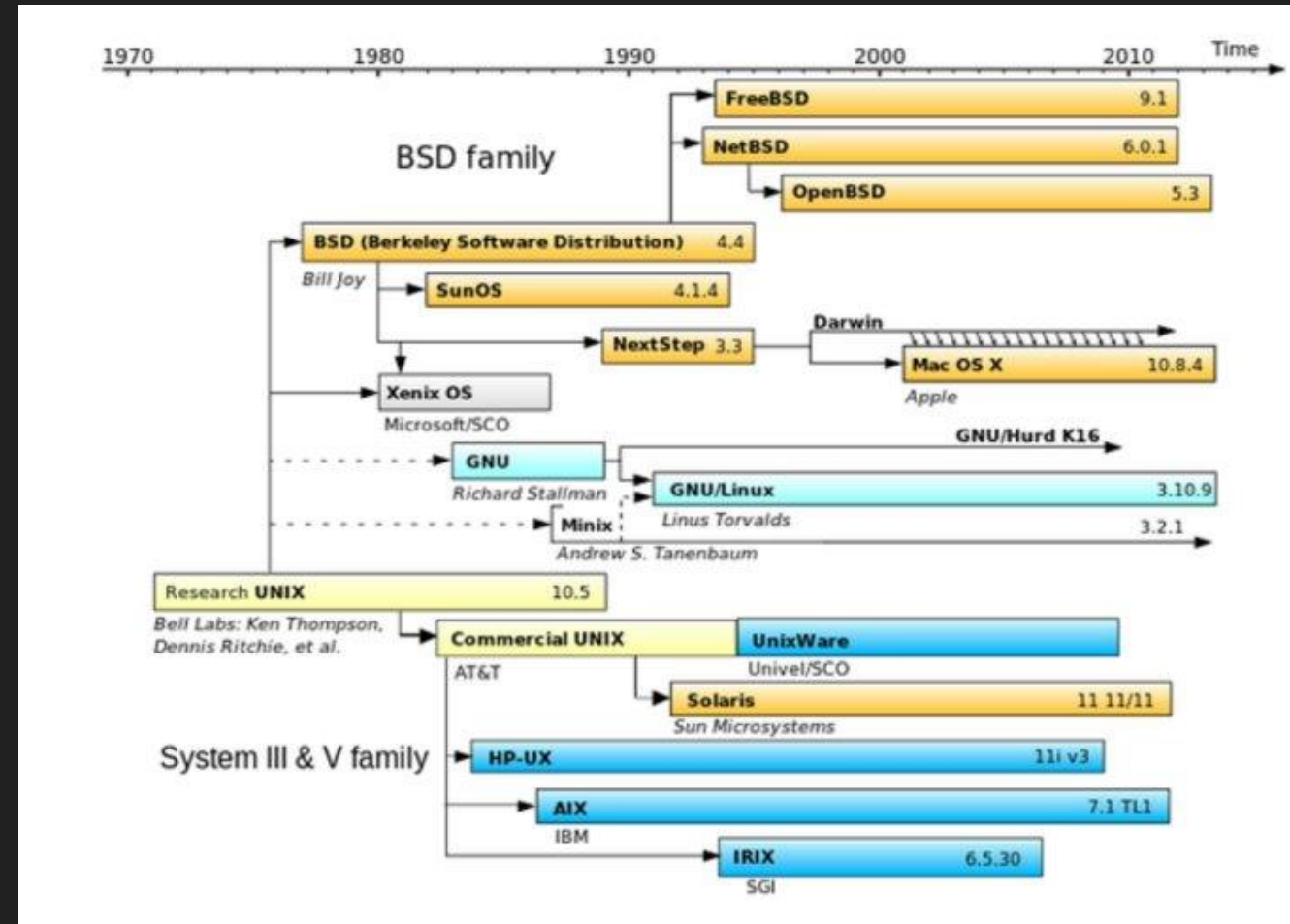
Sistemas Operativos



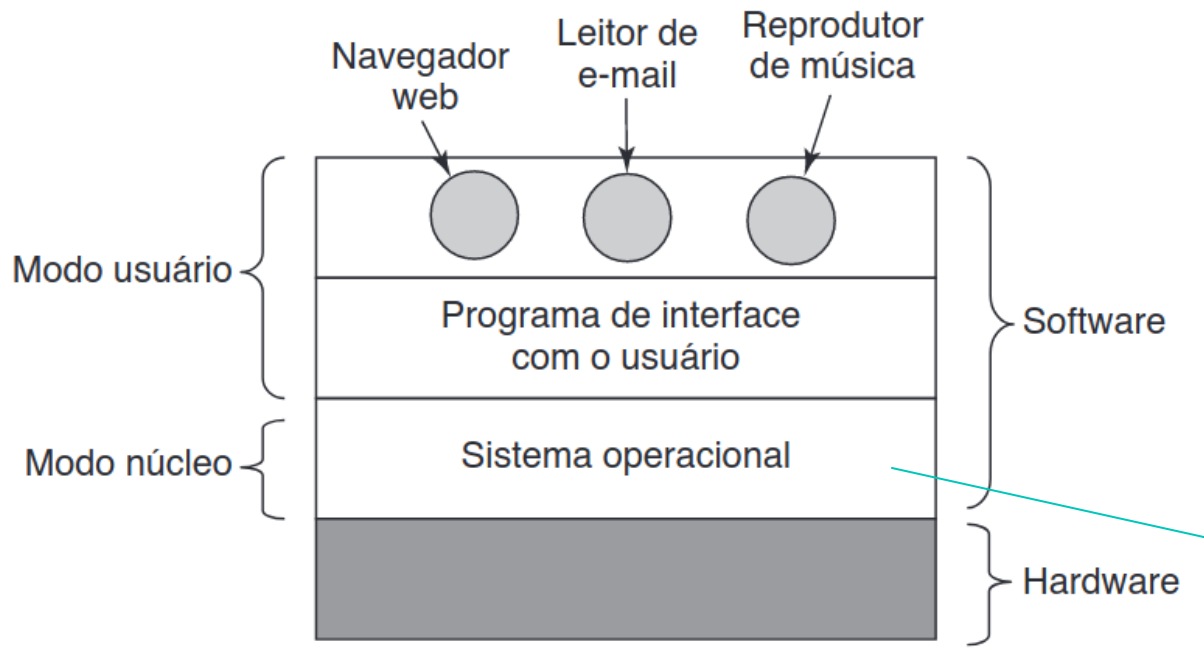
Un sistema operativo puede verse fundamentalmente como algo que proporciona abstracciones para programas de aplicación (de arriba hacia abajo)

Una visión alternativa, de abajo hacia arriba, sostiene que el sistema operativo está ahí para administrar todas las partes de un sistema complejo.

Los sistemas operativos modernos permiten que haya varios programas en la memoria y se ejecuten al mismo tiempo.

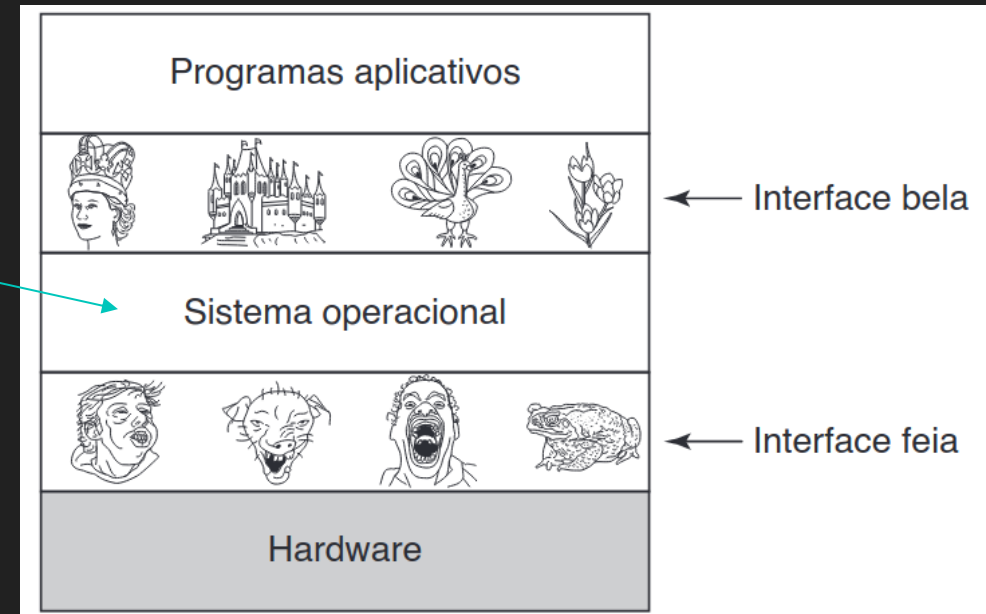


Sistemas Operativos

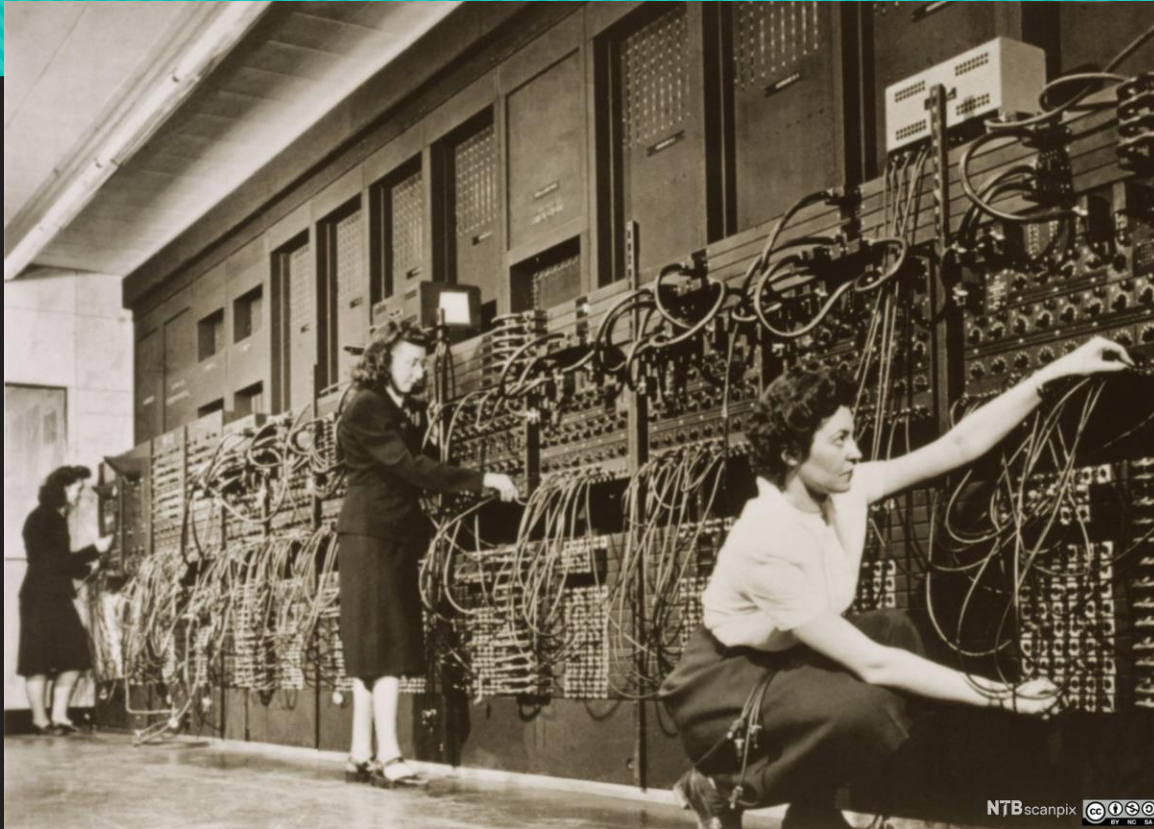


```

00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 2z.....yy...
00000012 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 .....@.....
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000036 00 00 00 00 00 00 00 01 00 00 0e 1f ba 0e 00 b4 09 cd .....o...f
00000048 21 b8 01 4c cd 21 54 68 69 73 20 70 72 6f 67 72 61 6d !.Lf!This program
0000005a 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e 20 69 6e 20 cannot be run in
0000006c 44 4f 53 20 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 DOS mode....$....
0000007e 00 00 30 01 92 29 74 60 fc 7a 74 60 fc 7a 74 60 fc 7a ..0.')}t'úzt'úzt'úz
00000090 71 6c a1 7a 76 60 fc 7a 71 6c 9c 7a 75 60 fc 7a 71 6c qlizv`úzql.zu`úzql
000000a2 f3 7a 75 60 fc 7a 71 6c a3 7a 79 60 fc 7a 67 68 a1 7a ózu`úzqlízy`úzhiz
000000b4 76 60 fc 7a f7 68 a1 7a 70 60 fc 7a 8e 43 e5 7a 71 60 v`úz=hjzp`úz.Cázq`
000000c6 fc 7a 74 60 fd 7a 4c 60 fc 7a fa 77 9c 7a 77 60 fc 7a úzt`ýzL`úzuw.zw`úz
000000d8 98 6b a2 7a 75 60 fc 7a fa 77 a6 7a 75 60 fc 7a 52 69 .kczu`úzuw|zu`úzi
000000ea 63 68 74 60 fc 7a 00 00 00 00 00 00 00 00 00 00 00 00 cht`úz.....
000000fc 00 00 00 00 50 45 00 00 4c 01 04 00 41 7e db 49 00 00 ....PE..L...A~OI..
0000010e 00 00 00 00 00 00 e0 00 0f 01 0b 01 07 0a 00 10 00 00 .....à.....
00000120 00 c0 00 00 00 00 00 00 02 1b 00 00 00 10 00 00 20 .A.....
00000132 00 00 00 00 40 00 00 10 00 00 00 10 00 00 04 00 00 00 .....@.....
00000144 00 00 00 00 04 00 00 00 00 00 00 00 00 e0 00 00 00 10 .....à.....
00000156 00 00 00 00 00 00 02 00 00 00 00 00 10 00 00 10 00 00 .....
00000168 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 .....
0000017a 00 00 00 00 00 00 24 24 00 00 8c 00 00 00 40 00 00 00 .....$$.....@..
0000018c 30 9a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0.....
0000019e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001c2 00 00 00 00 00 00 c8 23 00 00 40 00 00 00 00 00 00 00 .....È#.....
000001d4 00 00 00 00 00 20 00 00 f8 00 00 00 00 00 00 00 00 00 .....ø
    
```



Regreso al pasado



La calculadora electrónica ENIAC medía 30 metros de largo y casi 3 metros de alto. El ordenador de la imagen se construyó en Pensilvania entre 1943 y 1946.



Desde 1905 hasta 1955, las tarjetas perforadas fueron el primer medio para el ingreso y almacenamiento de datos, y el procesamiento en computación institucional

De vuelta al futuro



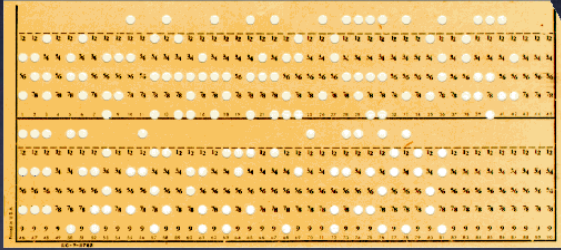
Steven P Jobs (izquierda) y John Sculley presentan los primeros ordenadores Macintosh con interfaz gráfica en 1984.



Noviembre de 2014. Alexa de Amazon interactuando mediante comandos de voz.

Puntos comunes: la interfaz

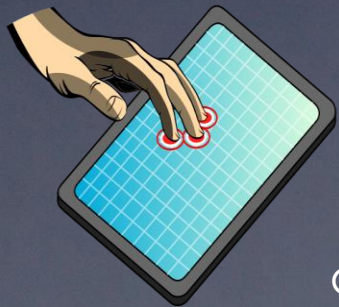
tarjetas perforadas



tubo de vacío



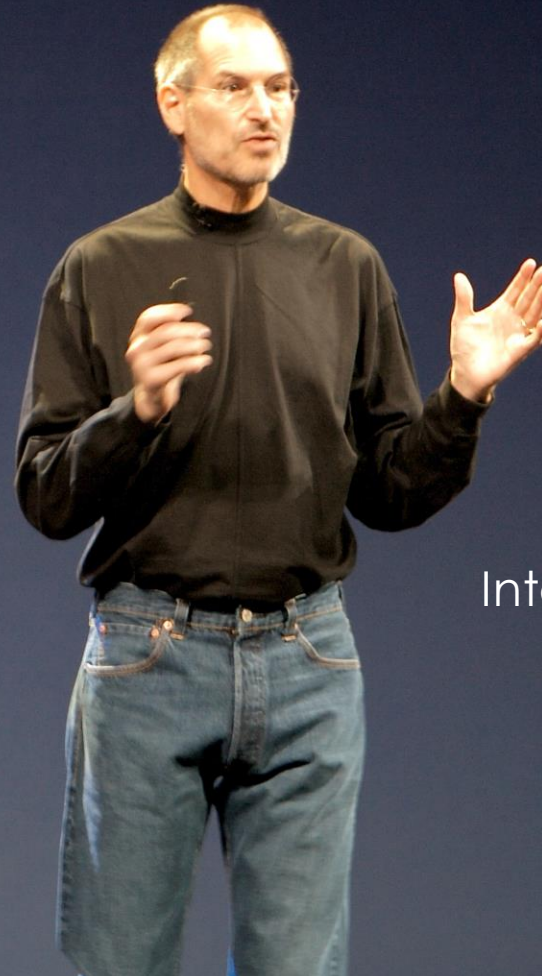
pantalla táctil



detectores de movimiento



Usuario



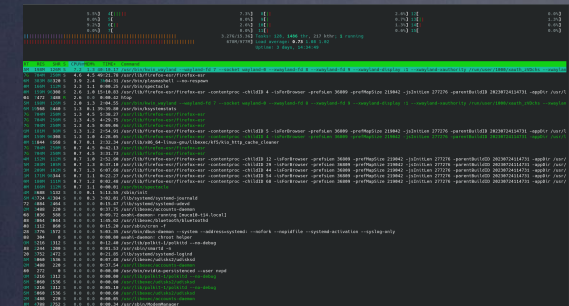
Interfaces gráficas de usuario



Interfaz de voz



Interfaz textual



¿Qué es la shell?

- La shell es una interfaz de usuario que facilita el acceso a los servicios del sistema operativo.
- Las shells permiten ejecutar programas, proporcionarles entrada de datos y examinar su salida de manera semiestructurada.
- Puedes escribir scripts o códigos, similares a un lenguaje de programación, para realizar tareas de forma automatizada.



Ventajas: ¿Por qué utilizamos la (¿antigua?) interfaz de terminal?

- **No** tienen **restricciones**: no puedes pulsar un botón de la interfaz que no existe ni dar una instrucción de voz que no esté programada.
- Las shells suelen ser altamente **personalizables**. Podemos personalizar el aspecto del shell, configurar 'alias' para comandos largos y crear funciones personalizadas para adaptar el entorno a sus necesidades específicas
- **¡Es más rápido!** Básicamente, es la forma en que el sistema operativo lleva a cabo sus tareas, lo que agiliza la ejecución de comandos.
- Es extremadamente **estandarizable**, lo que facilita la normalización de las tareas o protocolos que se ejecutan repetidamente.
- **Acceso Remoto**: Puedes acceder a sistemas remotos a través de la línea de comandos utilizando SSH (Secure Shell) u otros protocolos, lo que facilita la administración de servidores y sistemas distribuidos (como **superordenadores** o **clústeres**).

¡Empecemos!

- Abra una terminal e intente seguir algunos comandos (usemos su memoria muscular: D).
- En Ubuntu Ctrl+Alt+T o botón Win, busque terminal (o consola).
- En Windows, busque WLS: subsistema de Windows para Linux.

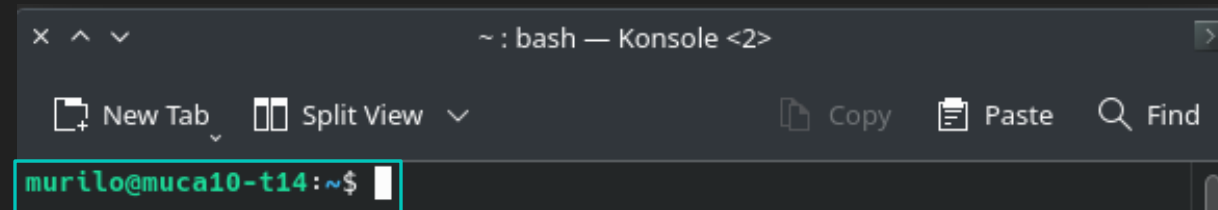
Usando la shell

username@computer-name:directorio-atual \$

↓
"usuario x" en la "computadora y"

↙
\$ = usuario sin poder administrativo

→
Siempre ejecutamos el comando que sea, estando en una carpeta de nuestro sistema de archivos.



The screenshot shows a terminal window titled '~ : bash — Konsole <2>'. The prompt is 'murilo@muca10-t14:~\$'. The window has a menu bar with 'New Tab', 'Split View', 'Copy', 'Paste', and 'Find'.

Comandos simples: date, echo, top, df

```
murilo@muca10-t14:~$ date
Mon Oct 9 02:39:42 PM CEST 2023
murilo@muca10-t14:~$ echo "Hola mundo!"
Hola mundo!
```

¿Existe alguna diferencia entre df y df -h?

```
murilo@muca10-t14:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.7G   0    7.7G   0% /dev
tmpfs           1.6G  2.1M   1.6G   1% /run
/dev/nvme0n1p2  28G   16G   11G   61% /
tmpfs           7.7G   17M   7.7G   1% /dev/shm
tmpfs           5.0M   8.0K   5.0M   1% /run/lock
/dev/nvme0n1p4  909G  242G  621G  29% /home
/dev/nvme0n1p1  511M   28M  484M   6% /boot/efi
tmpfs           1.6G   16M   1.6G   1% /run/user/1000
/dev/sdb1       13T   9.4T   3.5T  74% /media/murilo/NGS_capsid
```

```
murilo@muca10-t14:~$ top
```

```
top - 14:44:33 up 4 days, 14:48, 4 users, load average: 0.50, 0.81,
Tasks: 368 total, 1 running, 367 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.9 us, 1.1 sy, 0.0 ni, 96.8 id, 0.0 wa, 0.0 hi, 0.1 s
MiB Mem : 15691.2 total, 1865.4 free, 9942.8 used, 7296.3 buff
MiB Swap: 977.0 total, 5.8 free, 971.2 used. 5748.4 avai
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
1442	murilo	20	0	3576944	262640	181236	S	12.3	1.6
146112	murilo	20	0	46.5g	986.3m	233272	S	5.6	6.3
164987	murilo	20	0	3729144	1.0g	116264	S	4.7	6.4
198286	murilo	20	0	2831344	180456	116276	S	4.0	1.1
1577	murilo	20	0	9497408	418440	91784	S	3.3	2.6
801	root	20	0	13588	3700	2880	S	3.0	0.0
197176	murilo	20	0	1769716	156448	116232	S	3.0	1.0
153015	murilo	20	0	2790520	181664	70532	S	2.7	1.1
1654	murilo	20	0	611124	20656	8444	S	1.7	0.1
198006	root	20	0	0	0	0	I	1.7	0.0
186326	murilo	20	0	3891676	177288	97992	S	1.3	1.1
186494	murilo	20	0	67.7g	387832	71964	S	1.0	2.4

man: manual

¡tú no estás solo!

La comunidad que usa Linux tiene varias formas de ayudar: man describe los **parámetros** y **opciones** de los programas en el sistema

El **parámetro** es una opción que tiene valor.
Ejemplo:

```
--block-size=1,048,576
```

La "**opción**" funciona más como un estado de presencia o ausencia, activado o desactivado. Ejemplos:

```
-a o -all  
-h --human-readable  
-H
```

```
DF(1)                                User Commands                                DF(1)

NAME
    df - report file system space usage

SYNOPSIS
    df [OPTION]... [FILE]...

DESCRIPTION
    This manual page documents the GNU version of df. df displays the
    amount of space available on the file system containing each file name
    argument. If no file name is given, the space available on all cur-
    rently mounted file systems is shown. Space is shown in 1K blocks by
    default, unless the environment variable POSIXLY_CORRECT is set, in
    which case 512-byte blocks are used.

    If an argument is the absolute file name of a device node containing a
    mounted file system, df shows the space available on that file system
    rather than on the file system containing the device node. This ver-
    sion of df cannot show the space available on unmounted file systems,
    because on most kinds of systems doing so requires very nonportable in-
    timate knowledge of file system structures.

OPTIONS
    Show information about the file system on which each FILE resides, or
    all file systems by default.

    Mandatory arguments to long options are mandatory for short options
    too.

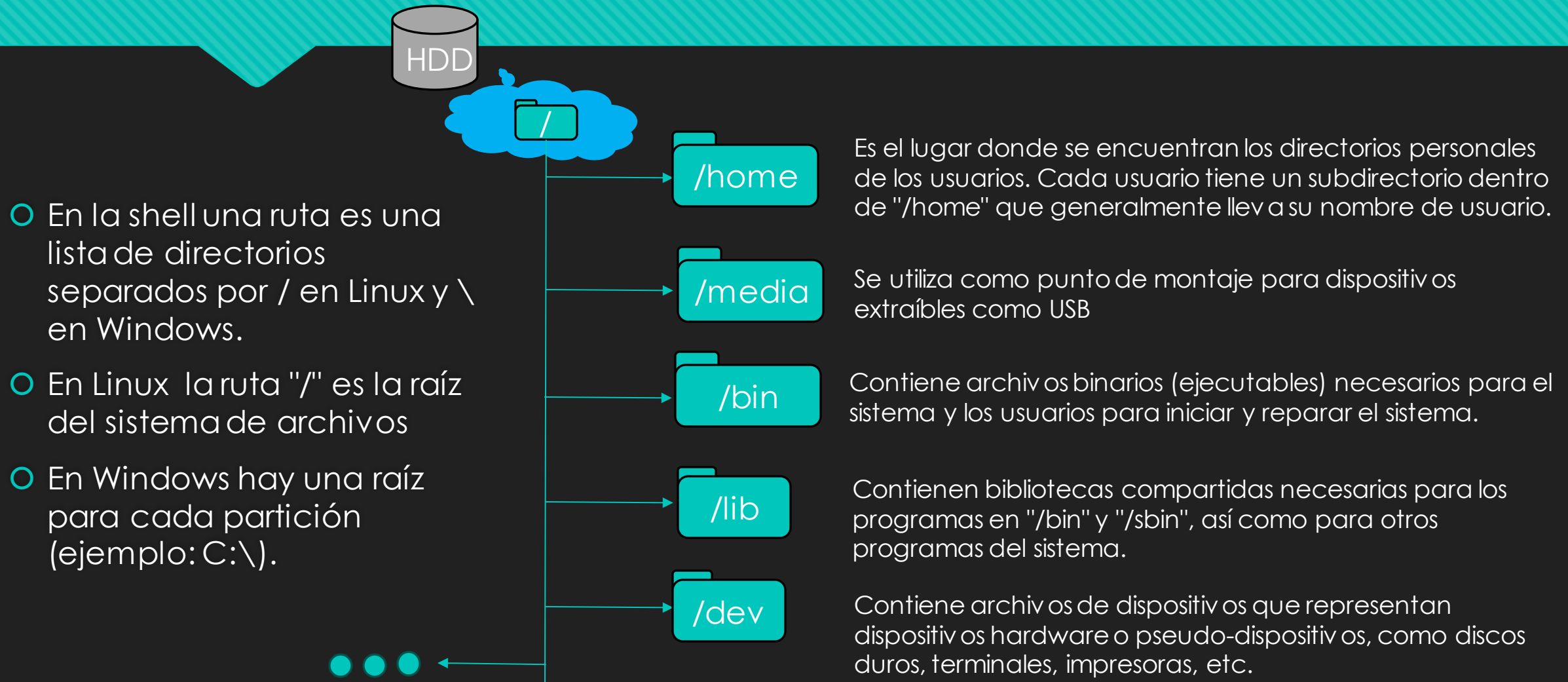
    -a, --all
        include pseudo, duplicate, inaccessible file systems

    -B, --block-size=SIZE
        scale sizes by SIZE before printing them; e.g., '-BM' prints
        sizes in units of 1,048,576 bytes; see SIZE format below

    -h, --human-readable
        print sizes in powers of 1024 (e.g., 1023M)

    -H, --si
        print sizes in powers of 1000 (e.g., 1.1G)
```

Si estamos en un sistema de archivos, necesitamos saber cómo navegar.



Tipos de rutas

Una ruta que empieza con / es llamada una ruta absoluta.

```
murilo@muca10-t14:~$ /home/murilo/Desktop
bash: /home/murilo/Desktop: Is a directory
murilo@muca10-t14:~$ /home/murilo/software
bash: /home/murilo/software: Is a directory
murilo@muca10-t14:~$ /home/murilo/software/IGV_Linux_2.16.1_WithJava/IGV_Linux_2.16.1/
bash: /home/murilo/software/IGV_Linux_2.16.1_WithJava/IGV_Linux_2.16.1/: Is a directory
```

Todas las otras son rutas relativas. ¡Porque dependen de la referencia actual!

```
murilo@muca10-t14:~/software$ pwd
/home/murilo/software
murilo@muca10-t14:~/software$ IGV_Linux_2.16.1_WithJava/IGV_Linux_2.16.1/
bash: IGV_Linux_2.16.1_WithJava/IGV_Linux_2.16.1/: Is a directory
```

Podemos consultar el directorio de trabajo actual con el comando **pwd**.

Punto "." Y dos puntos ".."

Formas de encontrar el archivo
"genoma.fasta":

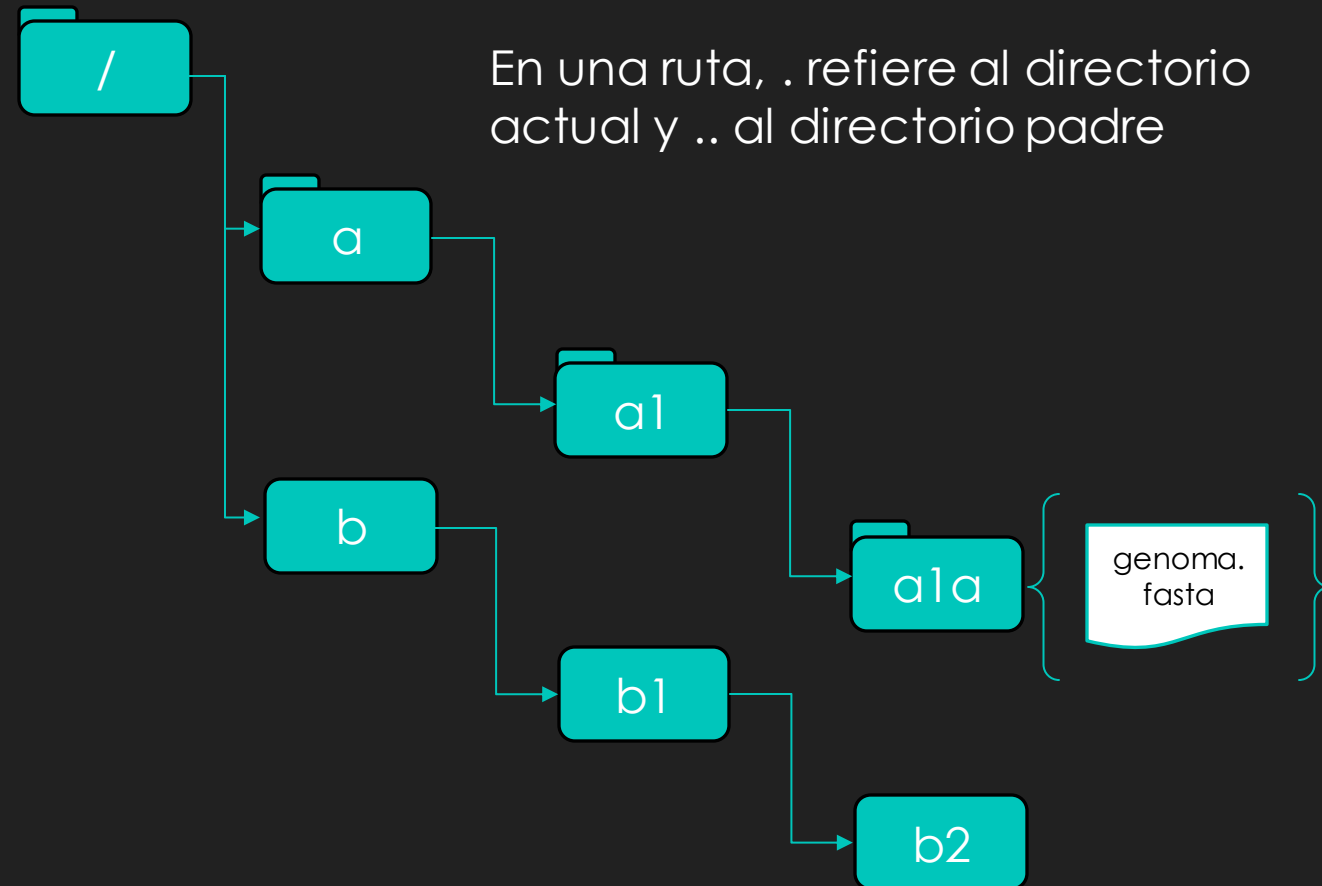
Absoluto

```
me@pc: ~ $ /a/a1/a1a/genoma.fasta
```

Relativo

```
me@pc: /b $ ../a/a1/a1a/genoma.fasta
```

```
me@pc: /b2 $ ../../a/a1/a1a/genoma.fasta
```



¿Pero cómo navegar entre directorios?

Usando pwd, cd y ls.

```
murilo@muca10-t14:~$ pwd
/home/murilo
murilo@muca10-t14:~$ cd /home
murilo@muca10-t14:/home$ pwd
/home
murilo@muca10-t14:/home$ cd ..
murilo@muca10-t14:/$ pwd
/
murilo@muca10-t14:/$ cd ./home
murilo@muca10-t14:/home$ ls
lost+found  murilo
murilo@muca10-t14:/home$ cd murilo
murilo@muca10-t14:~$ cd Desktop/
murilo@muca10-t14:~/Desktop$ pwd
/home/murilo/Desktop
murilo@muca10-t14:~/Desktop$ ls
1.ParaMurillo_KAZ26          core-pa
2023-05-04_refseqViral_2.0.15 DENV
230405_virusDiscoveryPipelineTestData german
2307_camila_bovCoV          HupafIDB
anhangavirus                ictv_blast_db
buffer                      institute_seminar_231009
buffer_filter_reads.sh      L_eddied_algn.fasta
bugado                     L_eddied.fasta
car-hpc.md                 mrBayes
contigs.fasta              new_gen
murilo@muca10-t14:~/Desktop$
```

Opciones de ls :

```
murilo@muca10-t14:~/Desktop$ ls
1.ParaMurillo_KAZ26
2023-05-04_refseqViral_2.0.15
230405_virusDiscoveryPipelineTestData
2307_camila_bovCoV
anhangavirus
```

```
murilo@muca10-t14:~/Desktop$ ls -l
total 232
drwxr-xr-x  2 murilo murilo  4096 Jul 12 13:32 1.ParaMurillo_KAZ26
drwxr-xr-x  5 murilo murilo  4096 Sep 26 12:14 2023-05-04_refseqViral_2.0.15
drwxr-xr-x  2 murilo murilo  4096 May 22 13:32 230405_virusDiscoveryPipelineTestData
drwxr-xr-x  2 murilo murilo  4096 Aug  7 12:21 2307_camila_bovCoV
drwxr-xr-x 21 murilo murilo  4096 Sep  1 18:45 anhangavirus
```

```
murilo@muca10-t14:~/Desktop$ ls -lh
total 232K
drwxr-xr-x  2 murilo murilo 4.0K Jul 12 13:32 1.ParaMurillo_KAZ26
drwxr-xr-x  5 murilo murilo 4.0K Sep 26 12:14 2023-05-04_refseqViral_2.0.15
drwxr-xr-x  2 murilo murilo 4.0K May 22 13:32 230405_virusDiscoveryPipelineTestData
drwxr-xr-x  2 murilo murilo 4.0K Aug  7 12:21 2307_camila_bovCoV
drwxr-xr-x 21 murilo murilo 4.0K Sep  1 18:45 anhangavirus
```

Con "ls -l" y derivados podemos conocer los permisos de un archivo

¿Es o no es un directorio?

```
murilo@muca10-t14:~/Desktop$ ls -lh
total 232K
drwxr-xr-x  2 murilo murilo 4.0K Jul 12 13:32 1.ParaMurillo_KAZ26
drwxr-xr-x  5 murilo murilo 4.0K Sep 26 12:14 2023-05-04_refseqViral_2.0.15
drwxr-xr-x  2 murilo murilo 4.0K May 22 13:32 230405_virusDiscoveryPipelineTestData
drwxr-xr-x  2 murilo murilo 4.0K Aug  7 12:21 2307_camila_bovCoV
drwxr-xr-x 21 murilo murilo 4.0K Sep  1 18:45 anhangavirus
-rw-r--r--  1 murilo murilo  407 Oct  9 17:40 institute_seminar_231009
-rw-r--r--  1 murilo murilo  13K Sep  1 14:44 L_eddie_d_algn.fasta
-rw-r--r--  1 murilo murilo  13K Sep  1 14:43 L_eddie_d.fasta
```

| d | rwx | rwx | rwx |

Permisos
del
dueño

Permisos
del grupo
del
dueño

Permisos
de otros

d: directory
r: read
w: write
x: execute
-: absence

d: directorio
r: leer
w: escribir
x: ejecutar
-: ausencia

El carácter "espacio" tiene significado

- La shell divide el comando por espacios y ejecuta el programa indicado en la primera palabra, proporcionando las siguientes palabras como argumentos.
- Para incluir espacios u otros caracteres especiales en un argumento, puedes usar comillas simples ('mis secuencias') o dobles ("mis secuencias") o escapar el carácter con \ (mis\ secuencias).

```
murilo@muca10-t14:/tmp/giz_bolivia$ pwd
/tmp/giz_bolivia
murilo@muca10-t14:/tmp/giz_bolivia$ touch hola.txt
murilo@muca10-t14:/tmp/giz_bolivia$ touch 'mis secuencias.fasta'
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh
total 0
-rw-r--r-- 1 murilo murilo 0 Oct 17 12:10 hola.txt
-rw-r--r-- 1 murilo murilo 0 Oct 17 12:10 'mis secuencias.fasta'
```

**Personalmente, para
estandarizar y evitar
errores, ¡evite usar espacio!**

touch, mkdir, cp, mv, rm

touch: comando para crear nuevos archivos vacíos. Si el archivo no existe, lo creará. Si ya existe, actualizará su fecha de modificación.

```
murilo@muca10-t14:/tmp/giz_bolivia$ pwd
/tmp/giz_bolivia
murilo@muca10-t14:/tmp/giz_bolivia$ touch hola.txt
murilo@muca10-t14:/tmp/giz_bolivia$ touch mis secuencias.fasta
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh
total 0
-rw-r--r-- 1 murilo murilo 0 Oct 17 12:08 hola.txt
-rw-r--r-- 1 murilo murilo 0 Oct 17 12:08 mis
-rw-r--r-- 1 murilo murilo 0 Oct 17 12:08 secuencias.fasta
```

touch, mkdir, cp, mv, rm

mkdir: Esta comando se utiliza para crear un nuevo directorio (carpeta).

```
murilo@muca10-t14:/tmp/giz_bolivia$ ls
hola.txt 'mis secuencias.fasta'
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir 2023_lib_denv
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir 2023_lib_denv/reads/raw_reads
mkdir: cannot create directory '2023_lib_denv/reads/raw_reads': No such file or directory
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir -p 2023_lib_denv/reads/raw_reads
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh
total 4.0K
drwxr-xr-x 3 murilo murilo 4.0K Oct 17 12:19 2023_lib_denv
-rw-r--r-- 1 murilo murilo  0 Oct 17 12:10 hola.txt
-rw-r--r-- 1 murilo murilo  0 Oct 17 12:10 'mis secuencias.fasta'
murilo@muca10-t14:/tmp/giz_bolivia$ cd 2023_lib_denv/
murilo@muca10-t14:/tmp/giz_bolivia/2023_lib_denv$ ls
reads
murilo@muca10-t14:/tmp/giz_bolivia/2023_lib_denv$ cd reads/
murilo@muca10-t14:/tmp/giz_bolivia/2023_lib_denv/reads$ ls
raw_reads
```

La opción -p crea el directorio principal si no existe

touch, mkdir, cp, mv, rm

mkdir: Esta comando se utiliza para crear un nuevo directorio (carpeta).

```
murilo@muca10-t14:/tmp/giz_bolivia$ ls
hola.txt  'mis secuencias.fasta'
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir 2023_lib_denv
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir 2023_lib_denv/reads/raw_reads
mkdir: cannot create directory '2023_lib_denv/reads/raw_reads': No such file or directory
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir -p 2023_lib_denv/reads/raw_reads
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh
total 4.0K
drwxr-xr-x 3 murilo murilo 4.0K Oct 17 12:19 2023_lib_denv
-rw-r--r-- 1 murilo murilo  0 Oct 17 12:10 hola.txt
-rw-r--r-- 1 murilo murilo  0 Oct 17 12:10 'mis secuencias.fasta'
murilo@muca10-t14:/tmp/giz_bolivia$ cd 2023_lib_denv/
murilo@muca10-t14:/tmp/giz_bolivia/2023_lib_denv$ ls
reads
murilo@muca10-t14:/tmp/giz_bolivia/2023_lib_denv$ cd reads/
murilo@muca10-t14:/tmp/giz_bolivia/2023_lib_denv/reads$ ls
raw_reads
```

La opción -p crea el directorio principal si no existe

touch, mkdir, cp, mv, rm

mkdir: Esta comando se utiliza para crear un nuevo directorio (carpeta).

Un buen paso hacia la automatización es el uso de "conjuntos" como en {iten1,iten2,...,iten3}

```
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir -p directorio_principal/{subdirectorio1,subdirectorio2,subdirectorio3}
murilo@muca10-t14:/tmp/giz_bolivia$ cd directorio_principal/
murilo@muca10-t14:/tmp/giz_bolivia/directorio_principal$ ls -lh
total 12K
drwxr-xr-x 2 murilo murilo 4.0K Oct 17 12:49 subdirectorio1
drwxr-xr-x 2 murilo murilo 4.0K Oct 17 12:49 subdirectorio2
drwxr-xr-x 2 murilo murilo 4.0K Oct 17 12:49 subdirectorio3
```

touch, mkdir, cp, mv, rm

cp: La función de este comando es copiar archivos o directorios.

```
murilo@muca10-t14:/tmp/giz_bolivia$ touch raw_reads/sample_{R1,R2,UP}.fastq
murilo@muca10-t14:/tmp/giz_bolivia$ ls -l raw_reads/
total 0
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:43 sample_R1.fastq
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:43 sample_R2.fastq
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:43 sample_UP.fastq
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir trimmed
murilo@muca10-t14:/tmp/giz_bolivia$ cp raw_reads/sample_R1.fastq trimmed/
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh trimmed/
total 0
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:45 sample_R1.fastq
```

El símbolo "*" puede usarse para representar "cualquier cadena de caracteres" y debe usarse con precaución.

```
murilo@muca10-t14:/tmp/giz_bolivia$ touch raw_reads/sample_{R1,R2,UP}.fastq
murilo@muca10-t14:/tmp/giz_bolivia$ ls -l raw_reads/
total 0
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:43 sample_R1.fastq
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:43 sample_R2.fastq
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:43 sample_UP.fastq
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir trimmed
murilo@muca10-t14:/tmp/giz_bolivia$ cp raw_reads/sample_R1.fastq trimmed/
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh trimmed/
total 0
-rw-r--r-- 1 murilo murilo 0 Oct 17 14:45 sample_R1.fastq
```

touch, mkdir, cp, mv, rm

mv: Este comando se usa para mover archivos o cambiar el nombre de archivos/directorios.

Cambiando el nombre

```
murilo@muca10-t14:/tmp/giz_bolivia$ touch archivo_viejo.txt
murilo@muca10-t14:/tmp/giz_bolivia$ mv archivo_viejo.txt archivo_nuevo.txt
murilo@muca10-t14:/tmp/giz_bolivia$ ls
archivo_nuevo.txt  raw_reads  trimmed_
```

Moviendo un archivo

```
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir nuevo_directorio
murilo@muca10-t14:/tmp/giz_bolivia$ mv archivo_nuevo.txt nuevo_directorio/
murilo@muca10-t14:/tmp/giz_bolivia$ ls nuevo_directorio/
archivo_nuevo.txt
```


touch, mkdir, cp, mv, rm

rm: Este comando se utiliza para eliminar archivos o directorios. Ten cuidado al usar **rm** ya que los archivos eliminados no se envían a la papelera de reciclaje **y se eliminan permanentemente**.

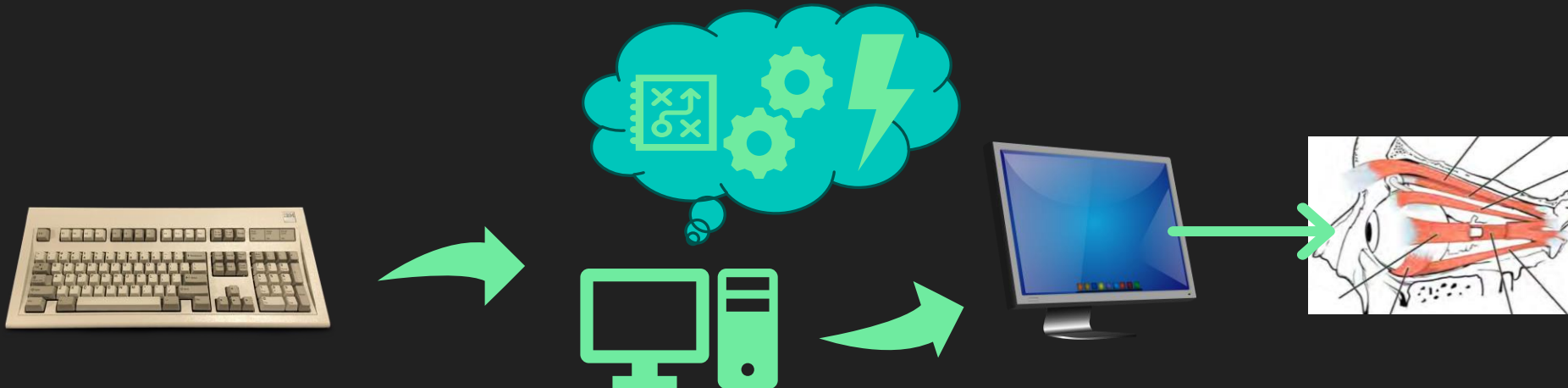
```
murilo@muca10-t14:/tmp/giz_bolivia$ touch archivo.txt
murilo@muca10-t14:/tmp/giz_bolivia$ ls
archivo.txt  nuevo_direc  raw_reads  trimmed
murilo@muca10-t14:/tmp/giz_bolivia$ rm archivo.txt
murilo@muca10-t14:/tmp/giz_bolivia$ ls
nuevo_direc  raw_reads  trimmed
murilo@muca10-t14:/tmp/giz_bolivia$ rm nuevo_direc/
rm: cannot remove 'nuevo_direc/': Is a directory
murilo@muca10-t14:/tmp/giz_bolivia$ rm -r nuevo_direc/
murilo@muca10-t14:/tmp/giz_bolivia$ ls
raw_reads  trimmed
murilo@muca10-t14:/tmp/giz_bolivia$
```

Conectando programas

En el shell, los programas tienen flujos de entrada y salida, además de tener opcionalmente opciones y parámetros.

Generalmente, la entrada proviene del teclado y la salida se muestra en la pantalla del terminal.

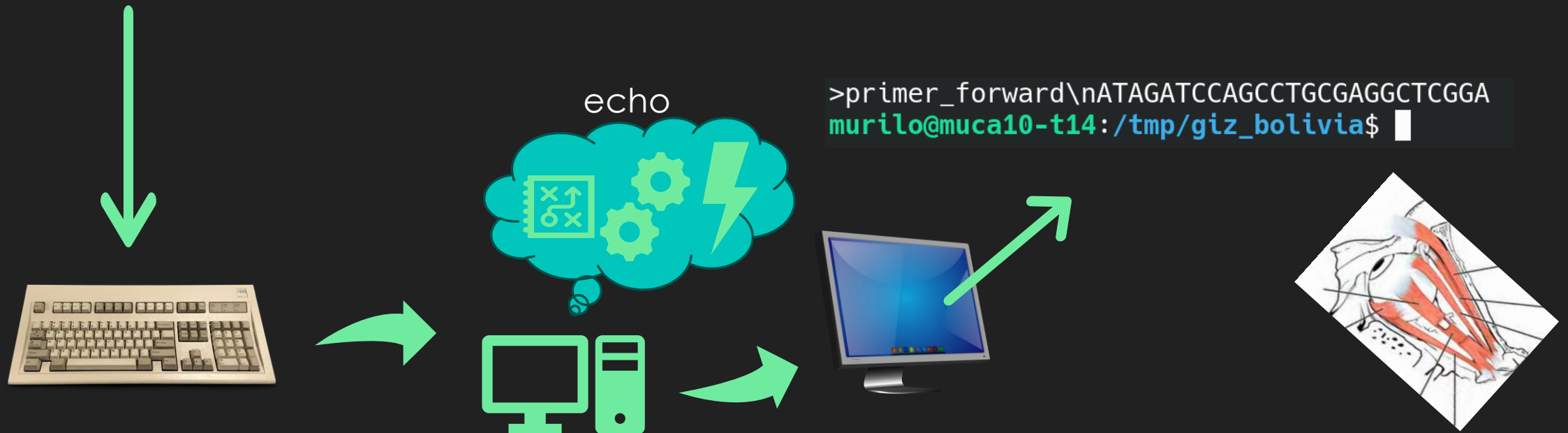
Estos flujos pueden ser redirigidos según sea necesario.



Conectando programas

Ejemplo simple:


```
murilo@muca10-t14:/tmp/giz_bolivia$ echo ">primer_forward\nATAGATCCAGCCTGCGAGGCTCGGA"
```



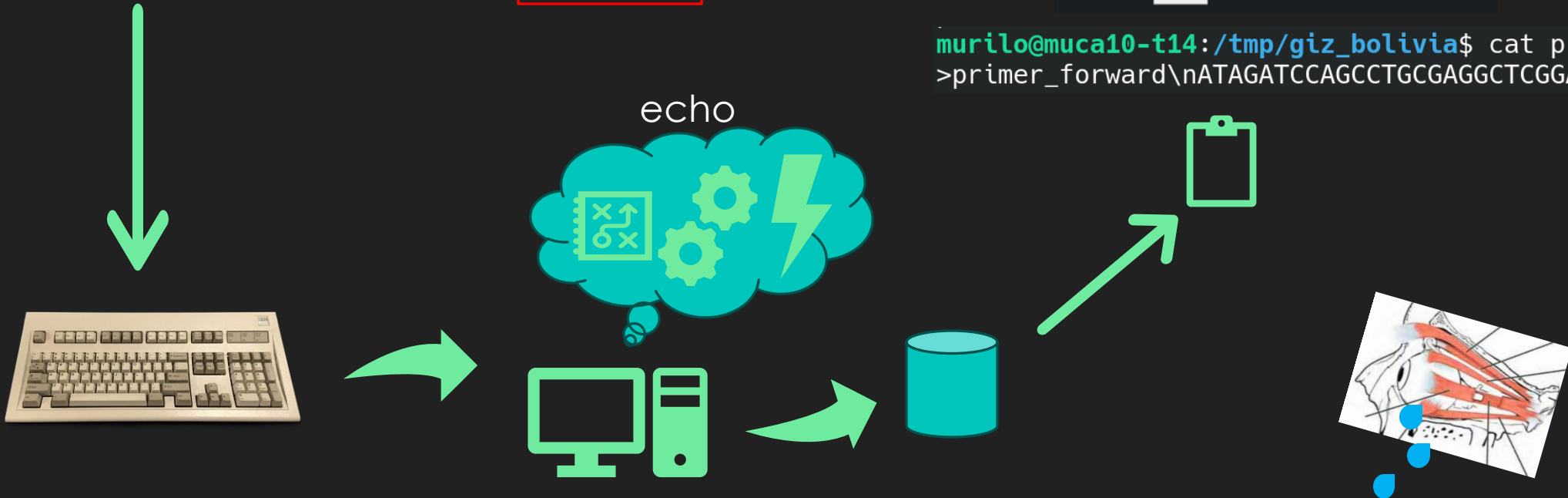
Redirigiendo el flujo: usando >

Ejemplo simple:

```
echo ">primer_forward\\nATAGATCCAGCCTGCGAGGCTCGGA" > primer.txt
```

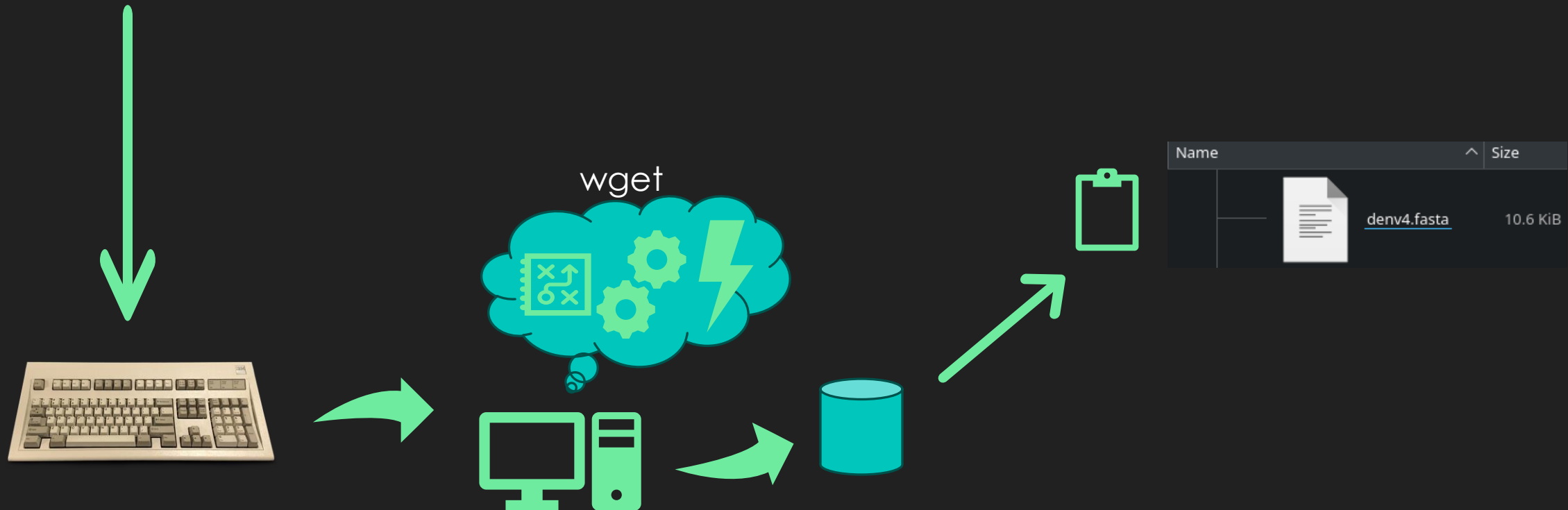
Name	Size	Modified
 primer.txt	43 B	9 minutes ago

```
murilo@muca10-t14:/tmp/giz_bolivia$ cat primer.txt  
>primer_forward\\nATAGATCCAGCCTGCGAGGCTCGGA
```



Redirigiendo el flujo: usando >

```
wget -q -O - "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id=NC_002640.1&rettype=fasta" > env4.fasta
```



Redirigiendo el flujo: usando >, >> y <

```
echo ">primerF" > primer_set.fasta
echo "AGCTCGACTACGACTAGACTA" >> primer_set.fasta
echo ">primerR" >> primer_set.fasta
echo "GGCATCAGCCGATGCGACAC" >> primer_set.fasta
cat primer
```

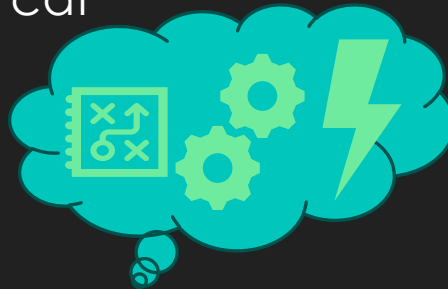
```
>primerF
AGCTCGACTACGACTAGACTA
>primerR
GGCATCAGCCGATGCGACAC
```

```
murilo@muca10-t14:/tmp/giz_bolivia$ cat < primer_set.fasta
>primerF
AGCTCGACTACGACTAGACTA
>primerR
GGCATCAGCCGATGCGACAC
murilo@muca10-t14:/tmp/giz_bolivia$ cat < primer_set.fasta > primer_set2.fasta
murilo@muca10-t14:/tmp/giz_bolivia$ cat primer_set2.fasta
>primerF
AGCTCGACTACGACTAGACTA
>primerR
GGCATCAGCCGATGCGACAC
```

primer_set.fasta

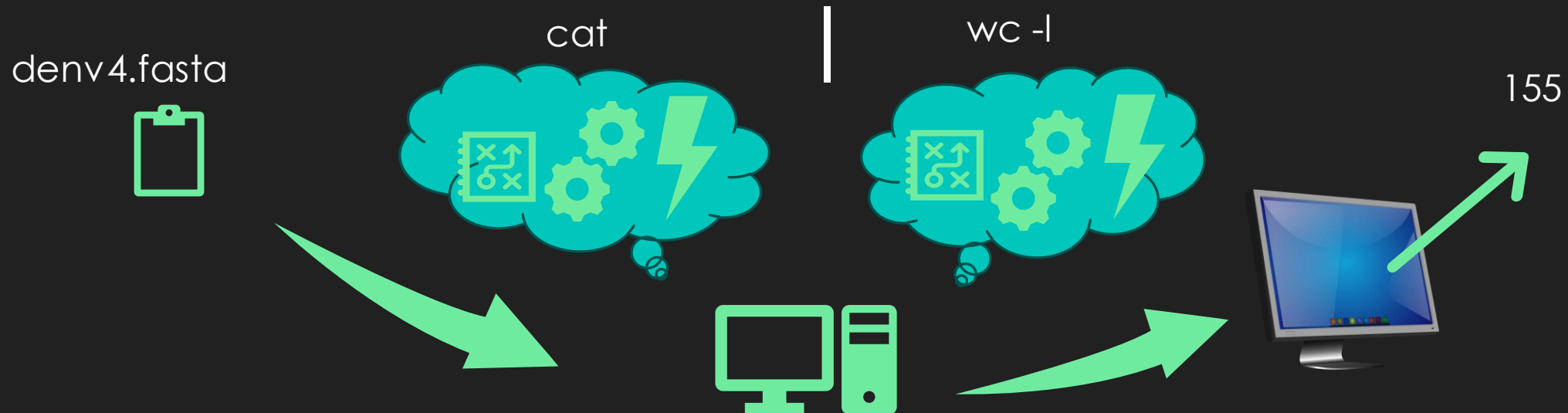
cat

primer_set2.fasta



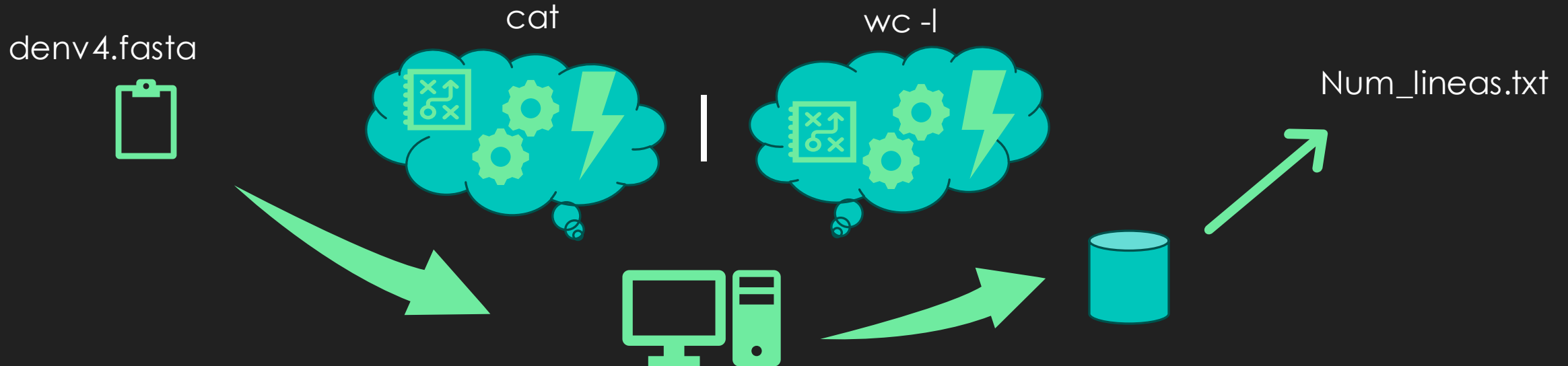
Conectando programas: el operador | (pipe)

```
murilo@muca10-t14:/tmp/giz_bolivia$ cat denv4.fasta | wc -l  
155
```



Conectando programas: el operador | (pipe)

```
murilo@muca10-t14:/tmp/giz_bolivia$ cat env4.fasta | wc -l > num_lineas.txt  
murilo@muca10-t14:/tmp/giz_bolivia$ cat < num_lineas.txt  
155
```



¿Cómo logra la shell encontrar los programas?

Hay una **variable de entorno** llamada \$PATH, que lista los directorios en que buscar programas al recibir un comando:

```
murilo@muca10-t14:~$ echo $PATH
/home/murilo/miniconda3/condabin:./usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:.
murilo@muca10-t14:~$ which echo
/usr/bin/echo
murilo@muca10-t14:~$ /usr/bin/echo $PATH
/home/murilo/miniconda3/condabin:./usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:.
```

```
missing:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
missing:~$ which echo
/bin/echo
missing:~$ /bin/echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

¡También podemos agregar una dirección a la variable de ruta para que se reconozca una carpeta de programa personal!

Ejercicio después

Los directorios están separados por dos puntos (:)

La\$ variable\$ de entorno

Usamos \$ para acceder al valor de una variable.
¡Inténtalo!

Las variables de entorno en Linux Bash son muy útiles para personalizar el comportamiento del sistema y de las aplicaciones. Algunas variables de entorno comunes y útiles:

- **HOME:** almacena la ruta del directorio principal del usuario actual.
- **USER:** Almacenan el nombre del usuario que ha iniciado sesión.
- **SHELL:** Indica el intérprete de comandos que está utilizando el usuario actualmente. Por ejemplo, /bin/bash para Bash.
- **PWD:** Almacena la ruta del directorio de trabajo actual (Present Working Directory).
- **LANG** y **LC_*:** Controlan la configuración regional y el idioma del sistema. Por ejemplo, LANG=en_US.UTF-8 establece el idioma a inglés estadounidense con codificación UTF-8.
- **TZ:** Define la zona horaria del sistema.
- **TERM:** Especifica el tipo de terminal que estás utilizando, lo cual es importante para las aplicaciones que dependen de la apariencia del terminal.

También podemos definir nuestras propias variables.

También usamos \$ para acceder al valor de estas variables.

de entorno

```
export MI_VARIABLE="Hola, Mundo!"  
export mi_apps="/home/murilo/Documents/software"  
export mi_codigo=$HOME/illumina_pipeline.sh
```

Accesible por procesos secundarios y para compartir información entre diferentes scripts o comandos, llamados por la misma sesión de shell.



locales

```
mi_variable=16  
mi_texto="CTAGCTGTAGCTAGAT"
```

Las variables locales son útiles para almacenar valores temporales dentro de un script o una función sin afectar otras partes del sistema.



Reglas para definiciones de variables

- Asignar: `variable=valor`
 - Nota que `var = valor` no funcionará desde que es interpretado como llamar el programa `foo` con el argumento `=` y `bar`
- Definición de cadena de caracteres
 - Con `'`: cadenas literales
 - Con `"`: cadenas que pueden tener variables

```
foo=bar
echo "$foo"
# imprime bar
echo '$foo'
# imprime $foo
```

- Utilice `{ }` para concatenar cadenas con variables

```
murilo@muca10-t14:/tmp/giz_bolivia$ var=muestra_1
murilo@muca10-t14:/tmp/giz_bolivia$ echo $var
muestra_1
murilo@muca10-t14:/tmp/giz_bolivia$ echo $var.fastq.gz
muestra_1.fastq.gz
murilo@muca10-t14:/tmp/giz_bolivia$ echo $var/raw
muestra_1/raw
murilo@muca10-t14:/tmp/giz_bolivia$ echo ${var}.fastq.gz
muestra_1.fastq.gz
murilo@muca10-t14:/tmp/giz_bolivia$ echo $varpersona_A
murilo@muca10-t14:/tmp/giz_bolivia$ echo ${var}persona_A
muestra_1persona_A
```


Reglas para definiciones de variables

- *Sustitución de comandos:* obtener la salida de un comando como una variable.
 - Cuando se usa `$(CMD)` ejecutará CMD, obtiene la salida del comando y la sustituye en el lugar.

```
murilo@muca10-t14:/tmp/giz_bolivia$ touch sample_{01..06}
murilo@muca10-t14:/tmp/giz_bolivia$ ls
sample_01 sample_02 sample_03 sample_04 sample_05 sample_06
murilo@muca10-t14:/tmp/giz_bolivia$ mi_var=ls
murilo@muca10-t14:/tmp/giz_bolivia$ $mi_var
sample_01 sample_02 sample_03 sample_04 sample_05 sample_06
murilo@muca10-t14:/tmp/giz_bolivia$ cd ..
murilo@muca10-t14:/tmp$ $mi_var
closeditems
giz_bolivia
plasma-csd-generator.FebwTs
sddm-:0-njxURV
sddm-auth170edae6-c5a8-41e7-856f-8c256f09ab21
systemd-private-7e4f95cbd954490084f5567a532047bd-bluetooth.service-7pcHQ5
systemd-private-7e4f95cbd954490084f5567a532047bd-bolt.service-LQkWbD
systemd-private-7e4f95cbd954490084f5567a532047bd-colord.service-bSS5HJ
murilo@muca10-t14:/tmp$ cd giz_bolivia/
```

```
murilo@muca10-t14:/tmp/giz_bolivia$ mi_var=$(ls)
murilo@muca10-t14:/tmp/giz_bolivia$ echo $mi_var
sample_01 sample_02 sample_03 sample_04 sample_05 sample_06
murilo@muca10-t14:/tmp/giz_bolivia$ cd ..
murilo@muca10-t14:/tmp$ echo $mi_var
sample_01 sample_02 sample_03 sample_04 sample_05 sample_06
```

Ahora sabemos muchas cosas sobre la shell (no te preocupes, seguiremos practicando).

Sabemos cómo:

- navegar por una estructura de directorios
- crear carpetas y archivos
- combinar el flujo de ejecución de programas
- tratar con variables de entorno y crear nuestras propias variables.

Ahora demos dos pasos hacia la automatización de procesos.



El bucle 'for'



scripts

For (para cada elemento "i" de la lista "l" haga...)

Un bucle for en Bash se utiliza para iterar sobre una lista de elementos y ejecutar un conjunto de comandos para cada elemento en la lista.

```
for variable in lista_de_elementos
do
    # Comandos que se ejecutarán para cada elemento en la lista
    # Usar $variable para acceder al elemento actual
done
```

- variable es el nombre de la variable que almacenará cada elemento de la lista en cada iteración.
- lista_de_elementos es una lista separada por espacios que queremos recorrer.

For (para cada elemento "i" de la lista "l" haga...)

Para cada nun en la lista 1, 2, 3, 4, 5

```
for num in 1 2 3 4 5
do
    echo $num
done
```

```
inicio=1
fin=5

for num in $(seq $inicio $fin)
do
    echo $num
done
```

```
murilo@muca10-t14:/tmp/giz_bolivia$ for num in {1..5}
> do
> echo $num
> done
1
2
3
4
5
```

El uso más útil para nosotros sería iterar sobre una lista de archivos.

```
murilo@muca10-t14:/tmp/giz_bolivia$ mkdir sample_{01..05}
murilo@muca10-t14:/tmp/giz_bolivia$ for sample in $(ls)
> do
> touch $sample/${sample}_R{1,2}.fastq
> done
murilo@muca10-t14:/tmp/giz_bolivia$ ls
sample_01 sample_02 sample_03 sample_04 sample_05
murilo@muca10-t14:/tmp/giz_bolivia$ ls sample_01
sample_01_R1.fastq sample_01_R2.fastq
```

O podría ser algo como
*.csv o incluso *.fasta

```
# Iterar sobre la lista de archivos en el directorio actual
for archivo in *
do

    # Verificar si el elemento es un archivo regular
    if [ -f "$archivo" ]; then
        echo "Archivo: $archivo"
    fi
done
```


Scripts

- Hasta ahora hemos visto como ejecutar comandos en la terminal y encadenarlos juntos. Sin embargo, en muchos escenarios vas a querer ejecutar una serie de comandos y hacer uso de expresiones de flujos de comandos como ciclos.
- Los scripts de terminal son el siguiente paso en complejidad.
- **Podemos escribir un protocolo de forma estandarizada, automatizada y con fácil acceso a los parámetros y programas utilizados durante el procesamiento de nuestros datos.**

Scripts

- Para hacer esto, los comandos deben estar organizados como código dentro de un archivo de texto.
- Generalmente usamos la extensión .sh para identificar archivos de shell.

```
#!/bin/bash

echo "Iniciando el programa a las $(date)"

echo "Ejecutando el programa $0 con $# argumentos y el identificador de proceso (pid) $$"

for parametro in "$@"; do
    echo $parametro
done
```

Scripts

- Es necesario comprobar los permisos para ejecutar. Luego usamos "./" seguido del nombre del script y cada uno de los argumentos separados por espacios.

```
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh script.sh
-rw-r--r-- 1 murilo murilo 253 Oct 21 02:21 script.sh
murilo@muca10-t14:/tmp/giz_bolivia$ chmod 744 script.sh
murilo@muca10-t14:/tmp/giz_bolivia$ ls -lh script.sh
-rwxr--r-- 1 murilo murilo 253 Oct 21 02:21 script.sh
```

```
murilo@muca10-t14:/tmp/giz_bolivia$ ./script.sh par1 par2 parA parB parC
Iniciando el programa a las Sat Oct 21 02:25:45 AM CEST 2023
Ejecutando el programa ./script.sh con 5 argumentos y el identificador de proceso (pid) 71042
par1
par2
parA
parB
parC
```

Cuando ejecutamos mediante script, hay algunas variables especiales.

- \$0 es el nombre del script
- \$1 a \$9 - Argumentos del script. \$1 es el primer argumento y así sucesivamente.
- \$@ - Todos los argumentos del script. Usado para iterar sobre todos los argumentos.
- \$# - Número de argumentos que se pasaron al script.
- \$\$ - Número de proceso asignado por el sistema operativo.

Para saber mas:

[./missing-semester](#) | [lectures](#) | [about](#)

The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

○ <https://missing.csail.mit.edu/>



○ <https://blogs.upm.es/estudiaciencia/#>

¡Vamos a practicar!

