Introducción a la Bioinformática

Unidad 1: Introducción a UNIX / LINUX

Facultad de Ciencias - UNAM / Instituto de Ecología - UNAM

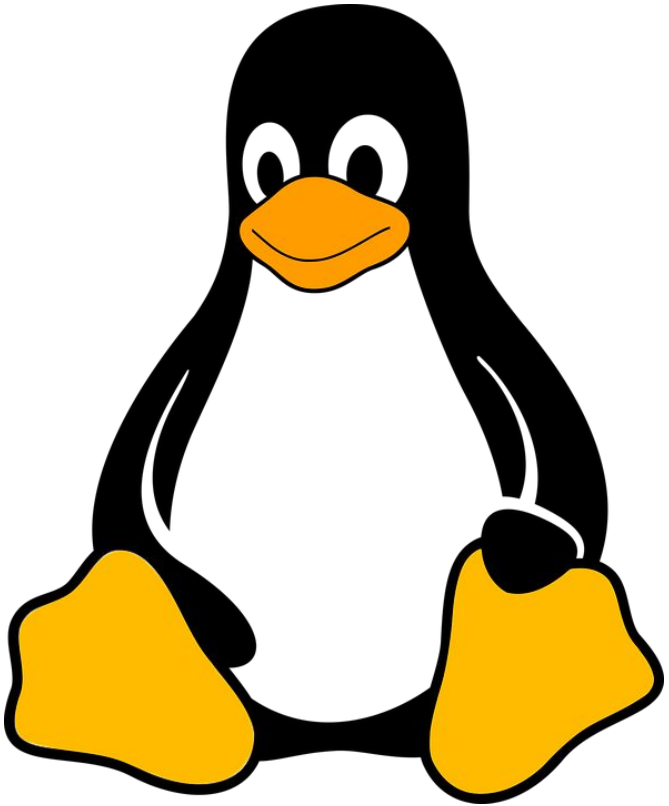
Marisol Navarro Miranda

21 de enero de 2018

Contenido del curso

Objetivos
Justificación

Módulo 1: Introducción a Unix
Módulo 2: Comandos básicos
Módulo 3: Manipulación de archivos
Módulo 4: Herramientas útiles



Objetivos

- Se abordarán los fundamentos de UNIX.

- El alumno será capaz de manejar comandos esenciales para el uso de la terminal.

- Se enseñarán herramientas para la manipulación de datos en la terminal.

Introducción

El usuario generalmente está acostumbrado a interactuar con interfaces gráficas (ventanas, íconos, etc). Esta forma de interactuar permite hacer tareas simples.

Justificación

1. La mayoría de las herramientas en Bioinformática están desarrolladas para funcionar en sistemas operativos tipo Linux.

2. Si se quiere darle instrucciones más complejas a una computadora, se debe acudir a aplicaciones **(shell)** que facilitan la ejecución de estas tareas.

3. Facilita la manipulación de datos masivos.

>>>>> Easter egg I <<<<<

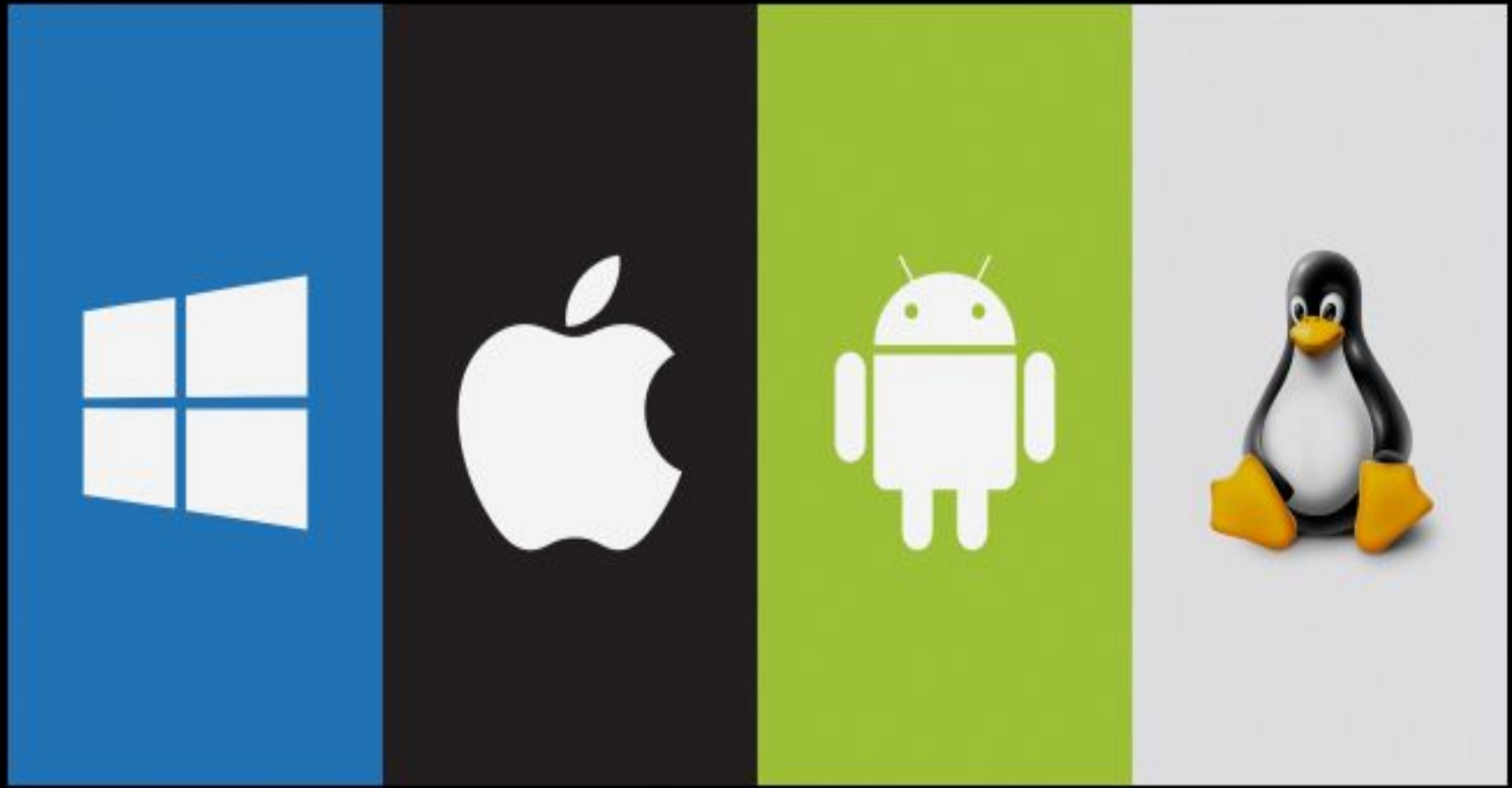
In computer software and media, an Easter egg is an intentional inside joke, hidden message or image, or secret feature of a work.

Módulo 1: Introducción a Unix

> El sistema operativo

- Es un programa que controla otras partes de la computadora tanto hardware como software.

- Permite además al usuario acceder a las facilidades que ofrece el sistema.



Módulo 1: Introducción a Unix

1969: Unix se comienza a desarrollar (hace 50 años !).
1973: Ken Thompson y Dennis Ritchie escribieron el primer artículo sobre UNIX, mientras trabajaban en los Laboratorios Bell de AT&T.
1977: Sale versión comercial.

Escrito en lenguaje C, fue desarrollado con los siguientes objetivos:

- Programas en palabras para una sola tarea.
- Que colaboren entre sí.
- Manejar flujo de texto que pudiera funcionar en una interfaz universal.
- Pequeño y eficiente en el uso de la memoria.
- Fácil de mantener.

“hacer una sola cosa y hacerla bien” = **OS exitoso**

>>>>> Easter egg II.I <<<<<

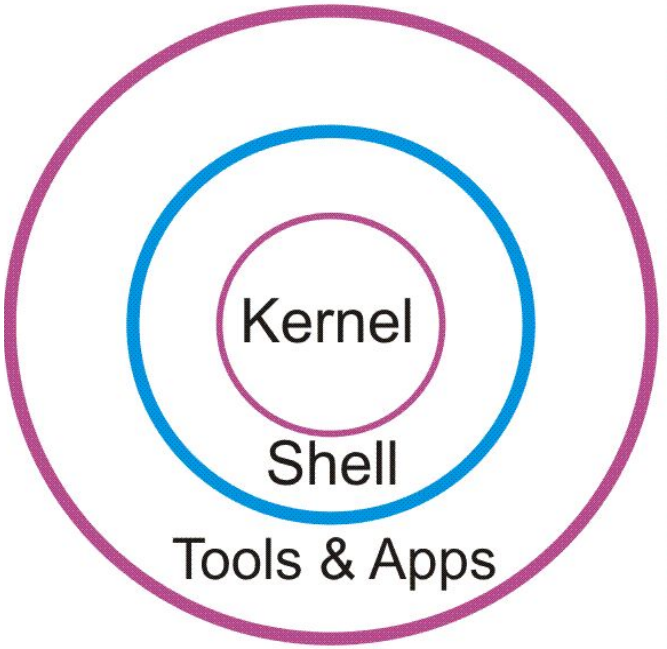
Módulo 1: Introducción a Unix

Partes del sistema operativo UNIX:

-> Kernell

-> Shell

-> UNIX file system



Módulo 1: Introducción a Unix

> Kernell

Puede considerarse como el núcleo del sistema operativo y es leído cada vez que se inicializa el sistema.

Realiza una serie de tareas básicas como son:

- Controlar la memoria de la máquina y asignar una parte a cada proceso.
- Distribuir el trabajo realizado por la CPU de forma que sea lo más eficiente posible.
- Organizar la transferencia de datos entre las distintas partes del sistema.
- Aceptar las instrucciones del shell.
- Hacer cumplir los permisos especificados en el sistema de archivos.

Módulo 1: Introducción a Unix

> Shell

Es un programa como cualquier otro. Permite la interacción vía el kernel, “leyendo” los comandos tecleados por el usuario para ser ejecutados.

El shell puede servir de base para escribir scripts, que son series de instrucciones para realizar tareas más complejas.

El shell de Unix más popular es Bash (Bourne Again Shell), llamado así porque se deriva de un shell escrito por Stephen Bourne.

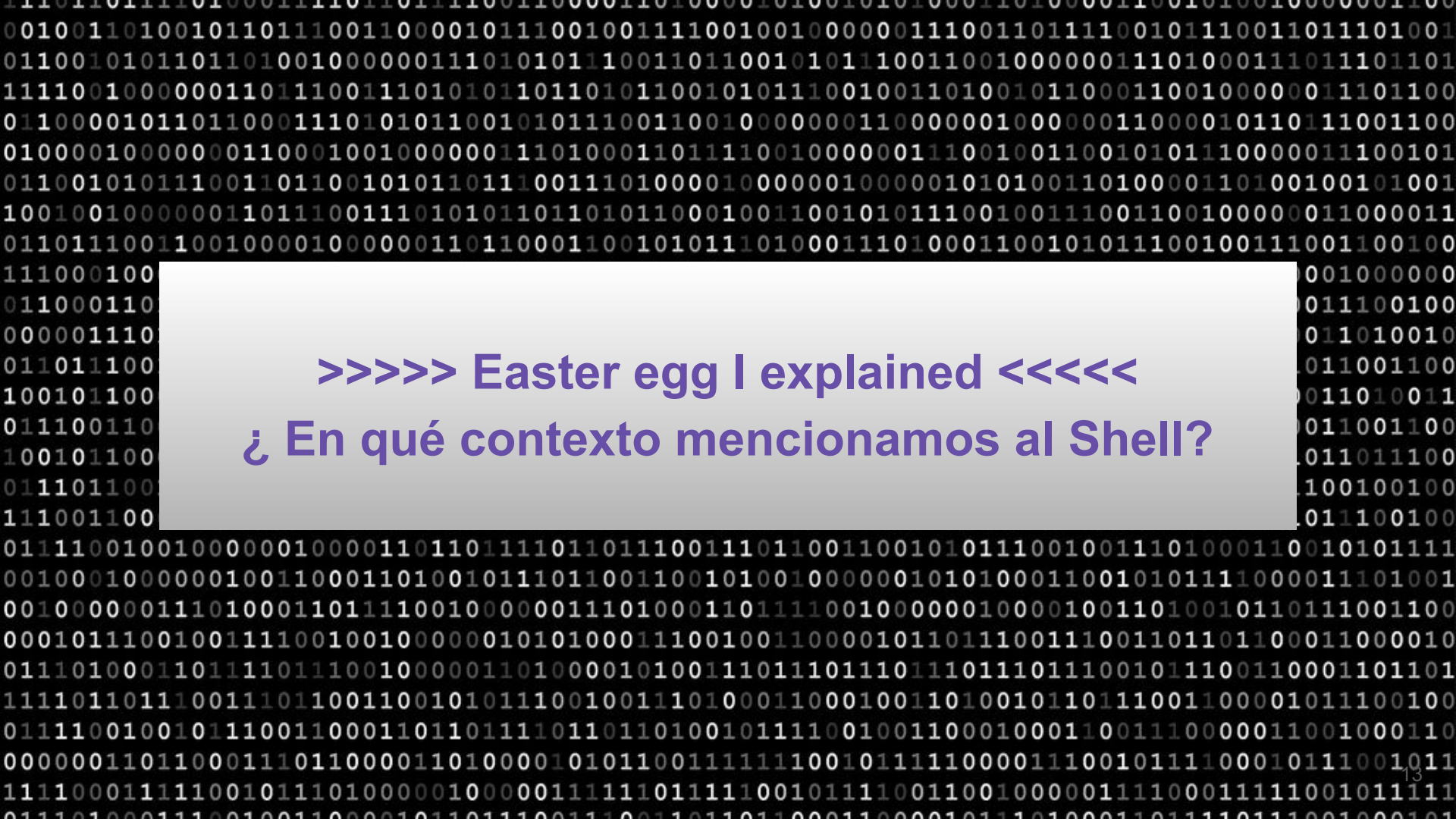
Módulo 1: Introducción a Unix

> Tipos de Shell:

- Bourne shell (sh)
- C shell (csh)
- TC shell (tcsh)
- Korn shell (ksh)
- Bourne Again Shell **(bash)**

Los sistemas operativos gráficos permiten tener una interface opcional al shell para ejecutar programas por medio de ventanas con botones que realizan o ejecutan comandos.

>>>>> Easter egg III <<<<<


>>>>> Easter egg I explained <<<<<
¿ En qué contexto mencionamos al Shell?

Módulo 1: Introducción a Unix

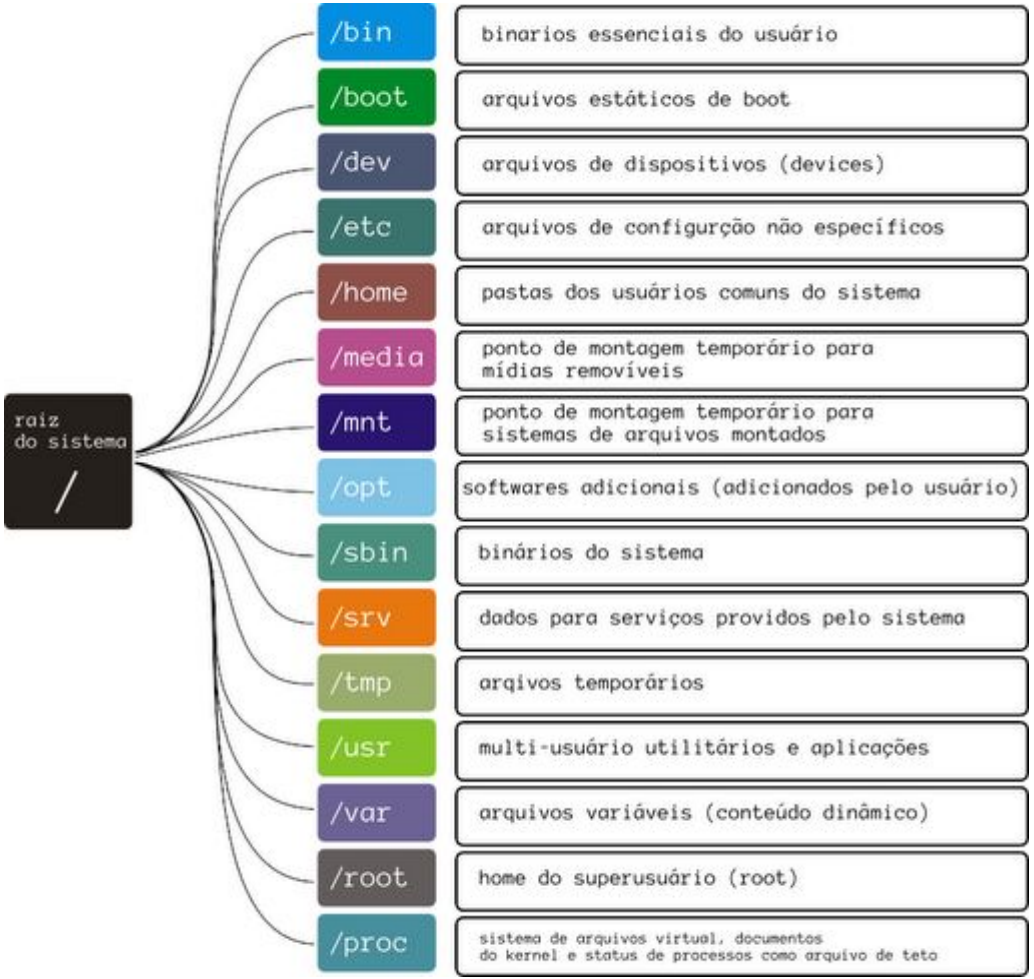
> UNIX file system

Es la forma que tiene el sistema operativo de organizar los datos en una estructura o colección de archivos.

UNIX considera como archivos no sólo a los archivos normales (en los que guardamos datos, programas, etc) sino también a los directorios y los dispositivos
conectados al sistema.

Está organizado en una estructura jerárquica de directorios que comienza en el directorio root representado por /.

Para efectos prácticos, cualquier directorio o archivo, será referenciado a partir de la raíz /.



/bin: contiene comandos y utilidades, son archivos ejecutables.

/dev: contiene los archivos que representan a los dispositivos conectados al sistema.

/etc: contiene comandos y archivos usados en la administración del sistema.

/home: contiene los archivos home de cada usuario del sistema.

/lib: contiene librerías utilizadas por diferentes programas y lenguajes.

/tmp: es el directorio donde se guardan los archivos temporales.

/usr: contiene archivos del sistema que son comunes a los usuarios como programas o documentación.

Módulo 1: Introducción a Unix

UNIX estaba reservado para los “mainframes” o supercomputadoras.

1991: Linus Torvald ingeniero finlandés que necesitaba un sistema operativo maleable y económico, en comparación con las licencias costosas de UNIX en aquel entonces.

- **Inició la creación del kernel Linux**, para implementar un equivalente a UNIX en las PC.

Richard Stallman: intentó su propia versión de UNIX, pero prefirió unirse al proyecto Linux con lo que ahora conocemos como GNU/Linux o Linux.

Linux: Es un sistema operativo basado en UNIX pero reescrito desde cero.

Ejemplos de versiones Linux: Ubuntu, Debian, OpenSUSE, Fedora, etc.

>>>>> Easter egg II.II <<<<<

Módulo 1: Introducción a Unix

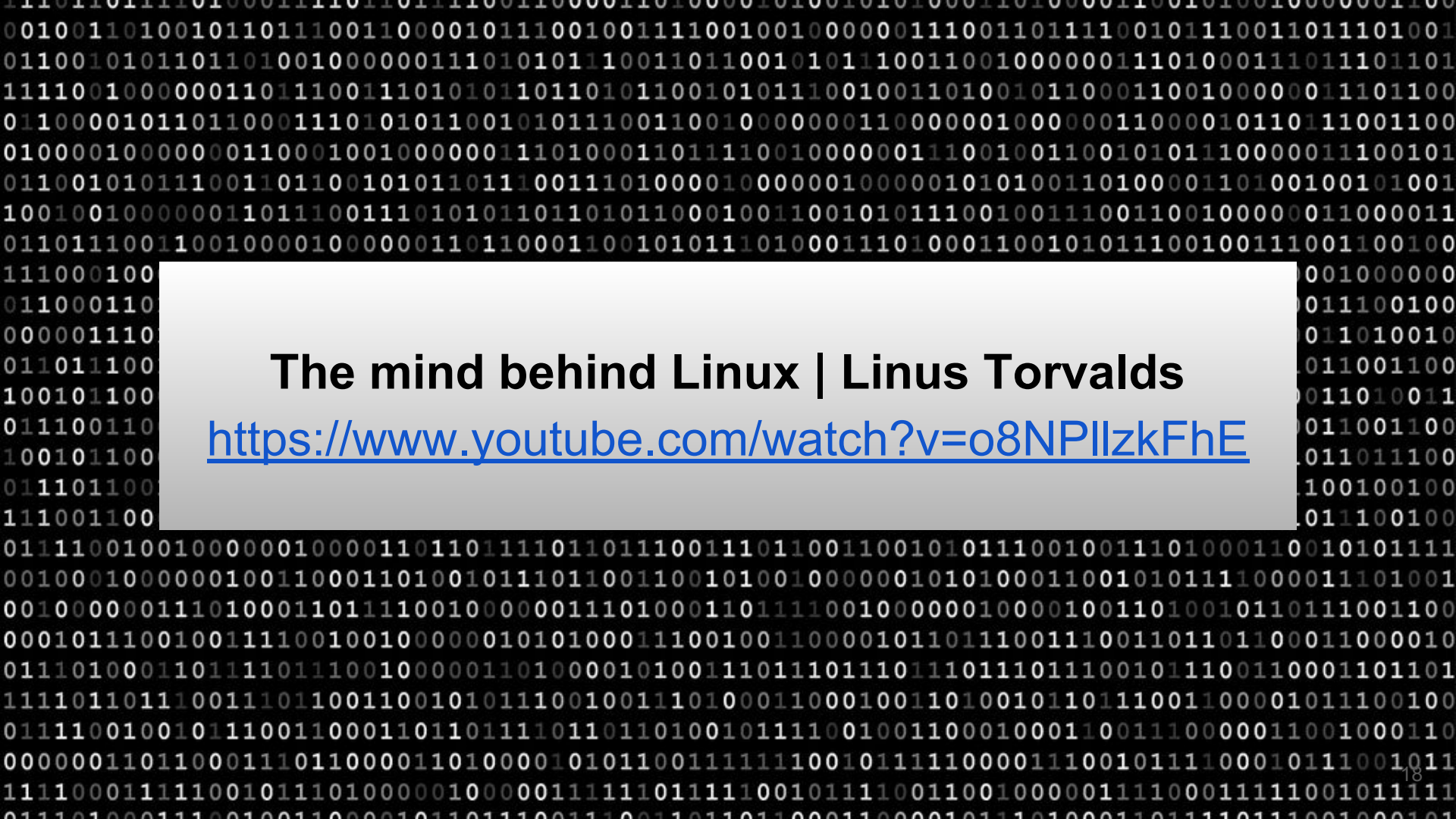
- Una de las características o propiedades con la que fue creado Linux fue el “libre acceso”.

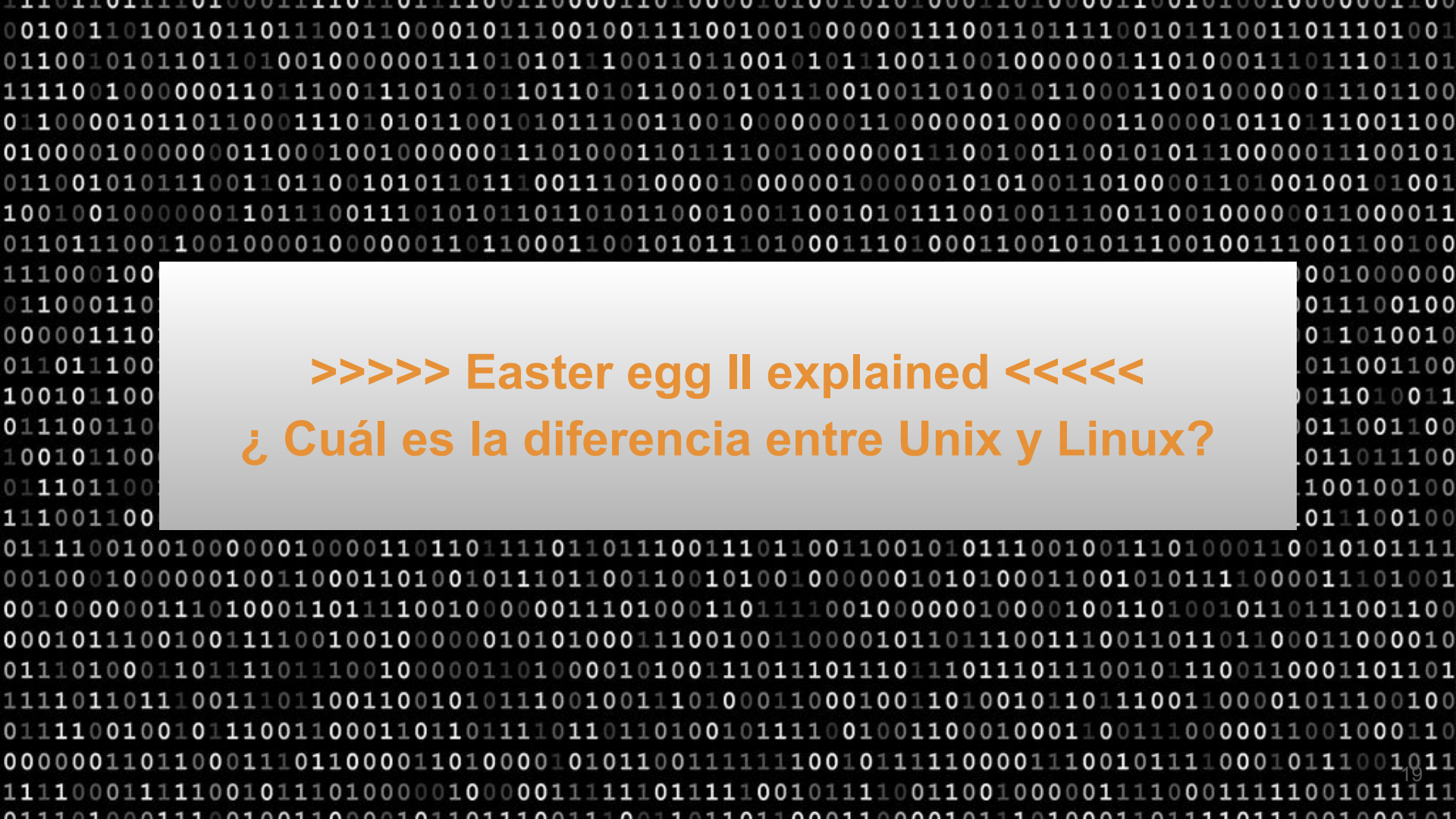
  > Software libre

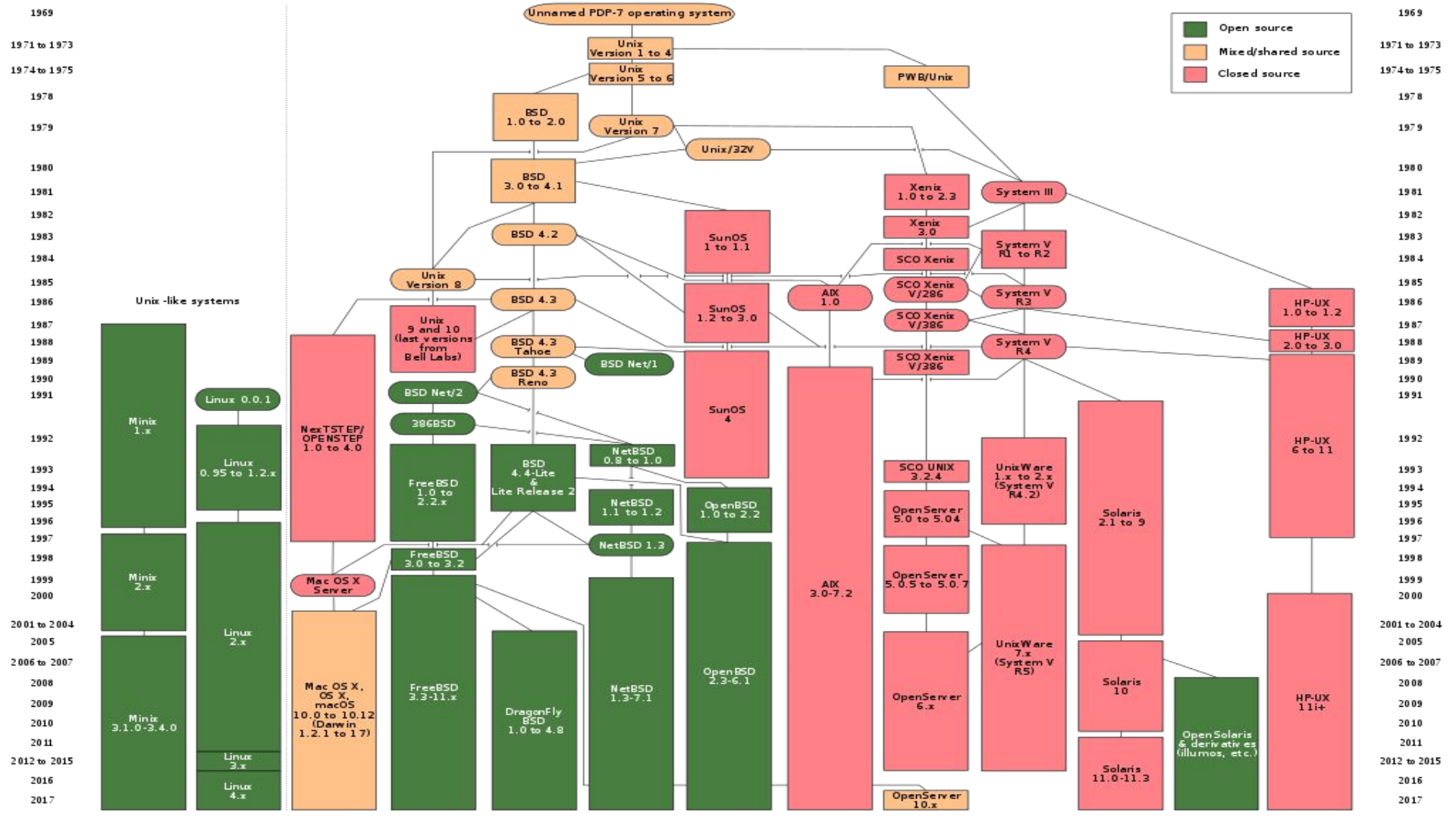
  - Cualquiera puede editarlo y adaptarlo a sus propias necesidades.
  - Algo que con el software propietario es difícil cuando no imposible por motivos legales.

- El nombre correcto del sistema operativo es GNU/Linux donde GNU es un acrónimo recursivo que significa “GNU is Not Unix!”.

  - Haciendo alusión de que Linux es software libre y no contiene código propietario de UNIX.


The mind behind Linux | Linus Torvalds
https://www.youtube.com/watch?v=o8NPllzkFhE


>>>>> Easter egg II explained <<<<<
¿ Cuál es la diferencia entre Unix y Linux?


Open source
Mixed/shared source
Closed source
Unix-like systems
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
AIX 1.0
SCO Xenix V/286
System V R3
BSD 4.3
SunOS 1.2 to 3.0
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
SCO Xenix V/386
HP-UX 2.0 to 3.0
BSD 4.3 Tahoe
BSD Net/1
System V R4
BSD 4.3 Reno
SCO Xenix V/386
Minix 1.x
Linux 0.0.1
BSD Net/2
SunOS 4
NeXTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
HP-UX 6 to 11
Linux 0.95 to 1.2.x
NetBSD 0.8 to 1.0
BSD 4.4-Lite & Lite Release 2
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
FreeBSD 1.0 to 2.2.x
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.0.4
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
Minix 2.x
NetBSD 1.3
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
Linux 2.x
Mac OS X Server
AIX 3.0-7.2
UnixWare 7.x (System V R5)
OpenServer 6.x
Solaris 10
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Minix 3.1.0-3.4.0
Linux 3.x
Solaris 11.0-11.3
Linux 4.x
OpenServer 10.x

Booting Pure DarwinXmas in vmware fusion!

https://www.youtube.com/watch?time_continue=48&v=_Z-Mfg9wTtc

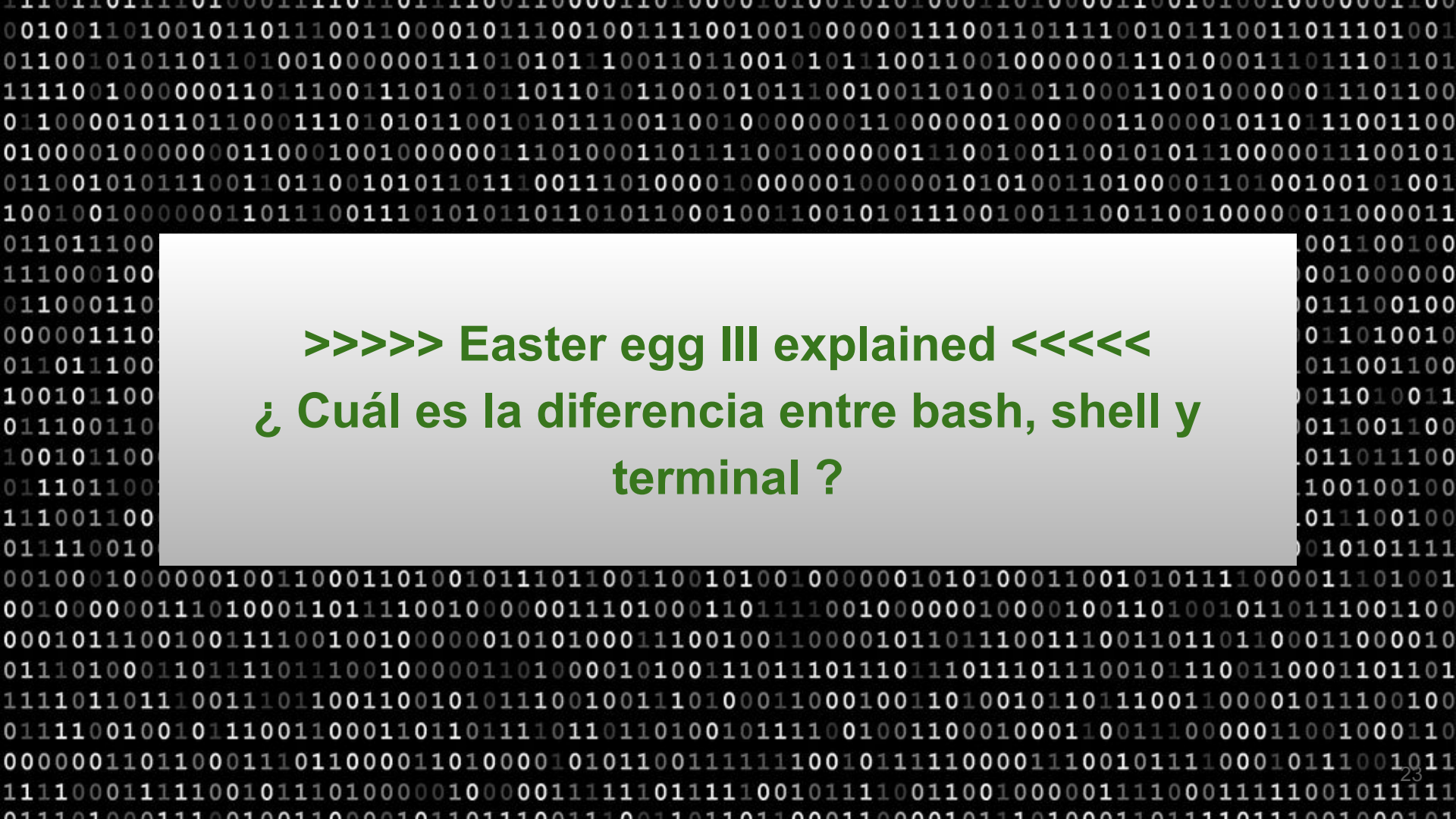
Módulo 1: Introducción a Unix

> Terminal

Es un programa llamado emulador de terminal. Abre una ventana y te permite interactuar con el shell.
Hay muchos tipos de emuladores, algunas de las distribuciones que maneja Linux son:

- gnome-terminal
- konsole
- xterm
- nxterm
- eterm

>>>>> Easter egg III <<<<<


>>>>> Easter egg III explained <<<<<
¿ Cuál es la diferencia entre bash, shell y terminal ?

```
[12:10:53] Finished 'default' after 965 ms
[12:10:53] Starting 'default'...
[12:10:53] Starting 'scripts'...

Fetching Scripts Source Files...
   - resources/assets/js/admin/jquery.min.js
   - resources/assets/js/admin/bootstrap.min.js
   - resources/assets/js/admin/nicescroll/jquery.nicescroll.min.js
   - resources/assets/js/admin/icheck/icheck.min.js
   - resources/assets/js/admin/sweetalert.js
   - resources/assets/js/admin/lity.js
   - resources/assets/js/admin/custom.js


Saving To...
   - ./public/admin/js/main.js

[12:10:53] Finished 'default' after 543 ms
[12:10:56] gulp-notify: [Laravel Elixir] Scripts Merged!
[12:10:56] Finished 'scripts' after 2.85 s
[12:10:56] Starting 'styles'...

Fetching Styles Source Files...
   - resources/assets/css/admin/bootstrap.min.css
   - resources/assets/css/admin/fonts/css/font-awesome.min.css
   - resources/assets/css/admin/animate.min.css
   - resources/assets/css/admin/custom.css
   - resources/assets/css/admin/icheck/flat/green.css
   - resources/assets/css/admin/sweetalert.css
   - resources/assets/css/admin/lity.css
```

Módulo 1: Introducción a Unix

> Sintaxis de los Comandos

Disciplina lingüística que estudia el orden y la relación de las palabras o sintagmas en la oración, así como las funciones que cumplen.

$ comando [opciones] [argumentos]

(1) **Comando:** el nombre de un programa guardado en la computadora.
(2) **Opciones:** Banderas '-' ó '--' que cambian el funcionamiento original del comando.
(3) **Argumentos:** información que necesita el comando que se usa para realizar su tarea.

Módulo 1: Introducción a Unix

> Prompt

Al inicio de la línea de comando en una terminal, se puede observar un texto similar al siguiente:

usuario@computadora:~$

A este texto se le llama Prompt y en este caso, nos indica el **nombre del usuario**, el nombre de la máquina al que nos hemos conectado y **nuestra ubicación** en dicho servidor.

Módulo 2: Comandos básicos

> Nota: Red button

Cuando un comando se empieza a salir del control o el usuario se equivocó de comando, esto tiene un resultado desastroso (como borrar todos sus archivos).

En esos casos, la única solución es el uso combinado de las teclas “[Ctrl] c” es decir, Control (Ctrl) abajo y teclear la letra c minúscula.

Módulo 2: Comandos básicos

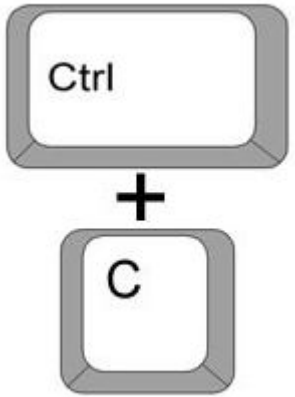
> Sleep

Usemos como ejemplo **sleep** para simular un comando que nos hace entrar en pánico:

usuario@computadora:~$ sleep 3600

Nos damos cuenta que escribimos el comando erróneo, y no observamos ningún resultado. Tecleamos **[Ctrl] c**.

usuario@computadora:~$ sleep 3600
^C
usuario@computadora:~$



Módulo 2: Comandos básicos

> Manuales

El comando **man** se utiliza para mostrar los detalles del comando que deseamos consultar.

Una vez en la página de ayuda para cierto comando, podemos navegar en el manual con las **flechas ↑ o ↓**.

Para salir del comando man, simplemente tecleamos la **tecla q.**

usuario@computadora:~$ man sleep

Módulo 2: Comandos básicos

> Buscar en manuales

Op1. Usar la diagonal **"/"** una vez **dentro del manual de un comando específico.**

Op2. El comando man tiene una opción para buscar por “keyword” o ₁."apropos" **dentro de toda la lista de los manuales para comandos.**

Ejercicio 1

1. Busca la opción para buscar por "palabra" en el manual del manual.
2. Cuando la encuentres, úsala con la palabra "count".
   Recuerda poner en orden las instrucciones: [Comando] [Opciones] [Argumentos].

1.apropos: *used to introduce something that is related to or connected with something that has just been said.*

Solución 1

a. **usuario@computadora:~$** man man
b. La opción del comando man es “-k”. Para buscar el comando que permite contar líneas en un archivo.

usuario@computadora:~$ man -k count

acct (2) - switch process accounting on or off
acct (5) - process accounting file
userdel (8) - delete a user account and related files
usermod (8) - modify a user account
wc (1) - print newline, word, and byte counts for each file

Módulo 2: Comandos básicos

> Datos del usuario

Un usuario tiene un nombre y un número de usuario asignado que se conoce como “UID”. Cada usuario pertenece al menos a un grupo, con número que a su vez tiene un identificador o “GID”.

Esa información se puede obtener con los comandos **whoami** e **id.**

usuario@computadora:~$ whoami
usuario
usuario@computadora:~$ id
uid=501(usuario) gid=20(staff) groups=20(staff)...

Módulo 2: Comandos básicos

> Usuario root

- Todas las versiones de UNIX tienen un super-usuario el cual tiene TODOS los derechos y privilegios sobre el sistema operativo y los archivos dentro del mismo.
- Este usuario es **root** y pertenece al grupo del mismo nombre, siendo su UID y GID 0.
- El usuario root se dedica a la administración del sistema.
- La noción de “sudo” que permite a un usuarios realizar tareas de administración como root, con el comando **sudo**.

usuario@computadora:~$ sudo [comando]

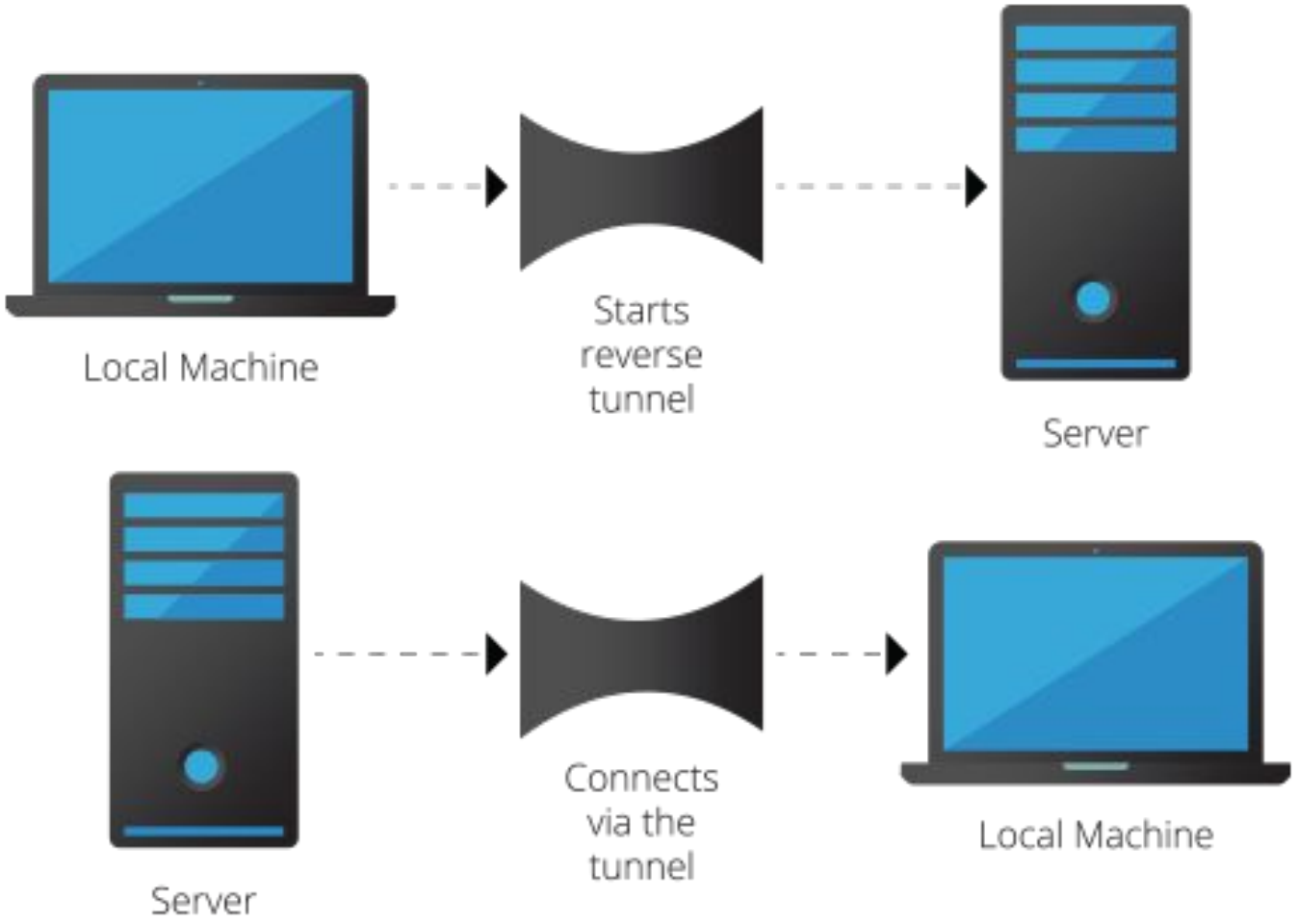
Módulo 2: Comandos básicos

> Conexión remota

SSH™ (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente.

usuario@computadora:~$ ssh -l mnavarro 132.248.32.3
usuario@computadora:~$ ssh mnavarro@132.248.32.3

En el ejemplo anterior la bandera -l modifica el programa de tal manera que usa el argumento mnavarro como el nombre del usuario para el comando ssh.


Local Machine
Starts reverse tunnel
Server
Server
Connects via the tunnel
Local Machine

Módulo 2: Comandos básicos

> Para salir

usuario@computadora:~$ logout
usuario@computadora:~$ exit

> Conexión con la interfaz gráfica

Existe el sistema cliente/servidor llamado “X Window System” que nos permite interactuar por medio de elementos gráficos incluso en una máquina a la cual nos conectamos remotamente.

Es decir, uno puede utilizar un programa gráfico que esté corriendo en otra máquina, pero utilizando los dispositivos de interfaz de nuestra computadora (monitor, teclado y ratón).

Ejercicio 2

1. Buscar en el manual de ssh, la opción para “Enables X11 forwarding” que nos permite importar el ambiente gráfico de una computadora a la cual nos conectamos remotamente, para poder desplegarlo en nuestra computadora.
2. Entra con el comando **ssh** añadiendo la opción de gráficos.
   Recuerda que así habíamos entrado:
   usuario@computadora:~$ ssh mnavarro@132.248.32.3
3. Ejecuta el comando **xclock.**

Si crees que es muy sencillo, detén el proceso y ahora ejecuta el comando **firefox** que te abrirá esta aplicación en el ambiente gráfico.

Solución 2

a. -X
b. **usuario@computadora:~$** ssh -X mnavarro@132.248.32.3
c. **usuario@computadora:~$** xclock
d. **usuario@computadora:~$** ^C
   usuario@computadora:~$ firefox

Módulo 2: Comandos básicos

> Nota: Ampersand simple "&"

Cuando ejecutamos un proceso en la computadora pero queremos seguir trabajando en la terminal, acudimos al ampersand.

El ampersand nos permite seguir trabajando, mientras el proceso queda en el "background" o "segundo plano".

usuario@computadora:~$ xclock &
[enter]
usuario@computadora:~$

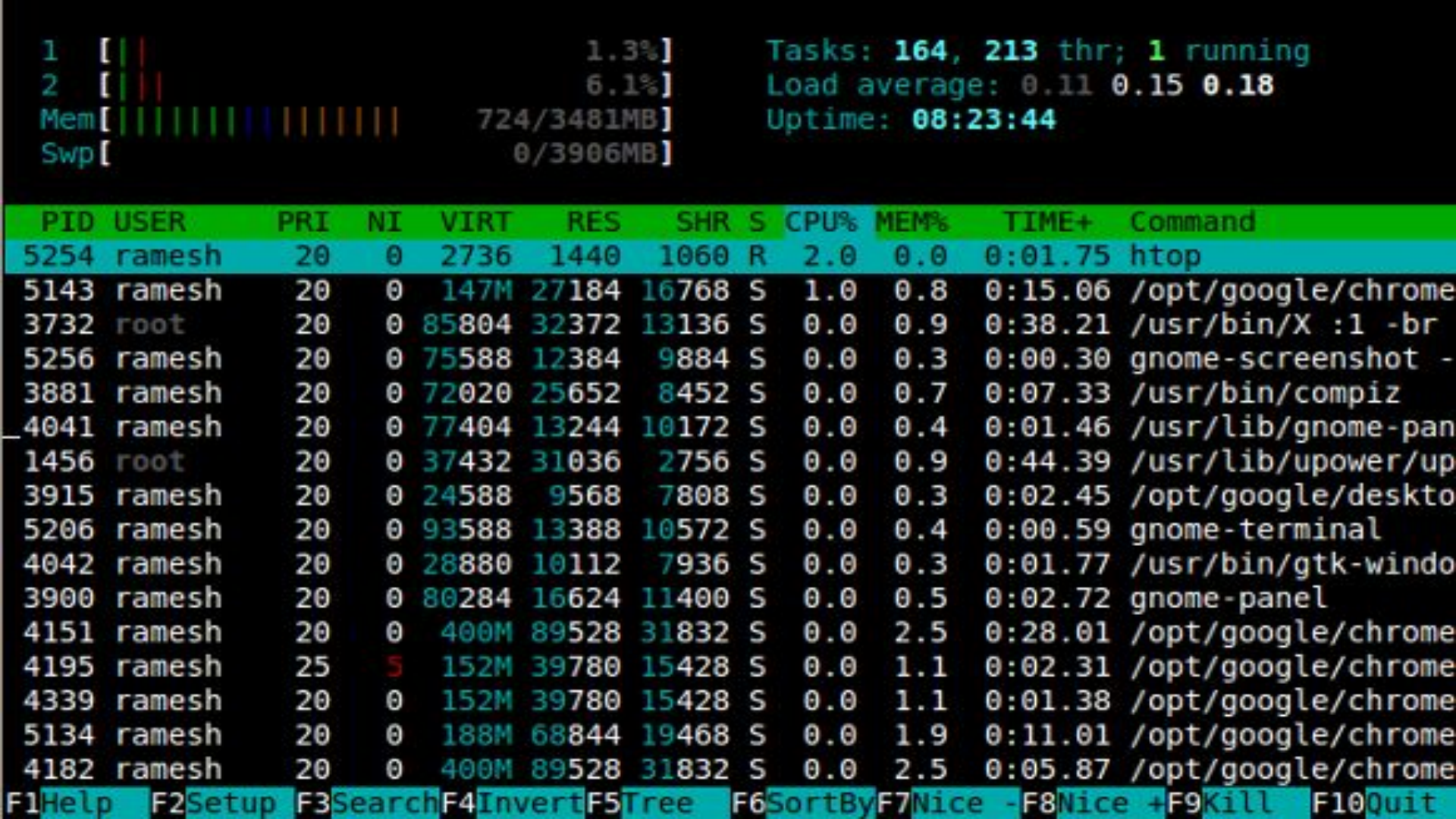
Módulo 2: Comandos básicos

> Nota: Top

Es una herramienta de la línea de comandos para monitorear procesos en tiempo real en sistemas Unix / Linux.

Viene preinstalada en la mayoría, si no en todas las distribuciones de Linux, y muestra un resumen útil de la información del sistema, incluido **el tiempo de actividad, la cantidad total de procesos y procesos en ejecución, inactivos, uso de CPU-RAM y una lista de procesos o subprocesos actualmente administrados por el kernel.**

usuario@computadora:~$ top


1 [|| 1.3%]
2 [||| 6.1%]
Mem[||||||||||||||| 724/3481MB]
Swp[0/3906MB]
Tasks: 164, 213 thr; 1 running
Load average: 0.11 0.15 0.18
Uptime: 08:23:44
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
5254 ramesh 20 0 2736 1440 1060 R 2.0 0.0 0:01.75 htop
5143 ramesh 20 0 147M 27184 16768 S 1.0 0.8 0:15.06 /opt/google/chrome
3732 root 20 0 85804 32372 13136 S 0.0 0.9 0:38.21 /usr/bin/X :1 -br
5256 ramesh 20 0 75588 12384 9884 S 0.0 0.3 0:00.30 gnome-screenshot -
3881 ramesh 20 0 72020 25652 8452 S 0.0 0.7 0:07.33 /usr/bin/compiz
4041 ramesh 20 0 77404 13244 10172 S 0.0 0.4 0:01.46 /usr/lib/gnome-pan
1456 root 20 0 37432 31036 2756 S 0.0 0.9 0:44.39 /usr/lib/upower/up
3915 ramesh 20 0 24588 9568 7808 S 0.0 0.3 0:02.45 /opt/google/deskto
5206 ramesh 20 0 93588 13388 10572 S 0.0 0.4 0:00.59 gnome-terminal
4042 ramesh 20 0 28880 10112 7936 S 0.0 0.3 0:01.77 /usr/bin/gtk-windo
3900 ramesh 20 0 80284 16624 11400 S 0.0 0.5 0:02.72 gnome-panel
4151 ramesh 20 0 400M 89528 31832 S 0.0 2.5 0:28.01 /opt/google/chrome
4195 ramesh 25 5 152M 39780 15428 S 0.0 1.1 0:02.31 /opt/google/chrome
4339 ramesh 20 0 152M 39780 15428 S 0.0 1.1 0:01.38 /opt/google/chrome
5134 ramesh 20 0 188M 68844 19468 S 0.0 1.9 0:11.01 /opt/google/chrome
4182 ramesh 20 0 400M 89528 31832 S 0.0 2.5 0:05.87 /opt/google/chrome
F1Help F2Setup F3Search F4Invert F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit

Módulo 2: Comandos básicos

> Nota: How to kill a mockingbird

Por el nombre del proceso:
usuario@computadora:~$ killall [nombre_del_proceso]

Por el PID:
usuario@computadora:~$ kill -9 [PID_del_proceso]

Arrastrando el cursor:
usuario@computadora:~$ xkill

Módulo 2: Comandos básicos

> Directorio HOME

Este directorio, como su nombre lo sugiere, es el directorio de cada usuario y que por default es el primero al que tenemos acceso al ingresar a la computadora.

El comando pwd nos permite determinar en cual directorio nos encontramos:

usuario@computadora:~$ pwd

Este directorio es uno de los pocos donde tienen derecho absoluto para crear, modificar o borrar archivos.

Módulo 2: Comandos básicos

> Comando ls

¿Cómo podemos conocer los archivos y directorios existentes en el mismo?
Para esto, empleamos el comando **ls**

usuario@computadora:~$ ls

Ejercicio 3

1. Prueba los siguientes comandos y explica para qué sirven. También puedes revisar el manual.

   usuario@computadora:~$ ls -F
   usuario@computadora:~$ ls -l
   usuario@computadora:~$ ls -lh

2. Encuentra la opción que permite ver "archivos escondidos".

Solución 3

a. Permite diferenciar entre directorios y archivos
b. Da opción de ver en formato largo
c. La opción muestra en términos "humanos" la memoria que ocupa el archivo
d. **usuario@computadora:~$** ls -a

Módulo 2: Comandos básicos

> Nota: extensiones

En Linux, no es obligatorio el uso de extensión (.mp3, .pdf, .jpg, .txt, .fasta). Un archivo de texto plano (por ej. mi-canción.txt) no será un archivo de audio simplemente por cambiar la extensión a .mp3.

Se usa comúnmente porque ayuda a reconocer los tipos de archivos, pero no afecta sus propiedades.

Usamos el comando **file** para saber que tipo de archivos son:

usuario@computadora:~$ file archivo_01.txt
usuario@computadora:~$ file archivo_02

Módulo 2: Comandos básicos

> Comando cd

El comando cd significa Change Directory. Si no se agrega una opción o destino en particular, este comando nos regresa al HOME de nuestro usuario.

usuario@computadora:~$ cd ..
usuario@computadora:~$ cd

Ejercicio 4

Verifica tu ubicación inicial
Entra a: follow/rabbit/
Una vez ahí entra a los directorios: path_a y path_b.

1. **Responde:** ¿Cuál de los directorios contiene **2 archivos**?

   Regresa a: rabbit/

2. **Responde:** ¿Cuántos archivos hay? ¿Cuántos directorios hay?
3. Con un comando regresa a tu ubicación principal.
4. Verifica que estés en tu ubicación principal.

Solución 4

a. path_a
b. 2 directorios; 3 archivos
c. cd
d. pwd

Módulo 2: Comandos básicos

> Destino absoluto y relativo

destino = dirección = referencia = camino = path = ruta

Un **destino absoluto** de un directorio nos indica todo el camino que hay que recorrer para acceder a él.

Ej. Destino absoluto de Ejercicio 5
home/unix/modulo2/ejercicio5
~/unix/modulo2/ejercicio5

Un **destino relativo** apunta a una referencia pero depende de la posición particular en la que estemos.

Ej. Estando en modulo2 y queriendo ir al directorio the/
../follow/the/

Módulo 2: Comandos básicos

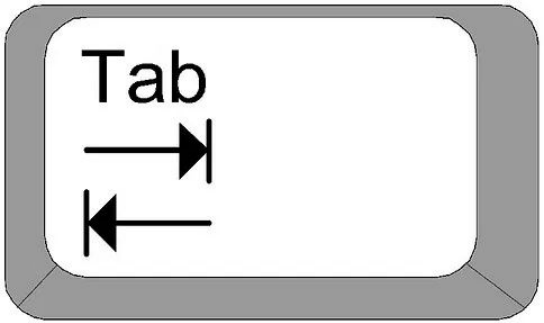
> Nota: tab

La tecla tab puede completar el texto si hay un archivo o directorio existente.

> Nota: principio y final de la línea

Principio: Ctr + a

Final: Ctr + e



> Nota: limpiar pantalla

No se pierde el seguimiento de los comandos anteriores.

Ctr + l

Clear

Módulo 3: Manipulación de archivos

> Creación de directorios y archivos

Un directorio nos permite organizar los datos y propiamente es un archivo especial que puede contener archivos y/o directorios (sub-directorios).

La **creación de un directorio** se hace con el comando **mkdir**.
La **supresión de un directorio** se hace con el comando **rmdir**.

La **creación de un archivo** se hace con el comando **touch**.
La **supresión de un archivo** se hace con el comando **rm**.

Módulo 3: Manipulación de archivos

> Nota: Ampersand doble (&&)

Estos caracteres sirven para ejecutar un comando seguido de otro(s).
Un ejemplo común es:

1. Crear un directorio
2. Entrar a ese nuevo directorio

Esto se podría hacer en dos pasos independientes:

usuario@computadora:~$ mkdir nuevo
usuario@computadora:~$ cd nuevo

Módulo 3: Manipulación de archivos

O de la siguiente manera:

usuario@computadora:~$ mkdir nuevo && cd nuevo

Borramos el directorio que acabamos de crear

usuario@computadora:~$ rmdir nuevo

Ejercicio 5

Con la idea de preparar unos análisis, vamos a crear una serie de directorios.

1. Hay que crear un directorio con el nombre FASTA dentro del HOME.
2. Dentro del directorio FASTA hay que crear dos directorios con los nombres Especie1 y Especie2.
3. Dentro de Especie1 hay que crear un directorio llamado Resultados.
4. Traten de suprimir el directorio original FASTA y todo su contenido en una sola línea de comando.
5. ¿Se puede usar el comando **rmdir** en este caso?

Solución 5

a. **usuario@computadora:~$** mkdir FASTA
b. **usuario@computadora:~$** cd FASTA
   usuario@computadora:~$ mkdir Especie1 && mkdir Especie2
c. **usuario@computadora:~$** cd Especie1
   usuario@computadora:~$ mkdir Resultados
d. **usuario@computadora:~$** cd
   usuario@computadora:~$ rmdir FASTA
e. **usuario@computadora:~$** man -k remove
   usuario@computadora:~$ man rm
   usuario@computadora:~$ rm -r FASTA

Módulo 3: Manipulación de archivos

> Copia de archivos

El comando **cp** se utiliza para copiar archivos.
usuario@computadora:~$ cp nuc.fna esp1

los datos de nuc.fna se duplican al crear el archivo esp1 que tendrá el mismo contenido.

Copiando el archivo desde ubicación del archivo a un directorio.
usuario@computadora:~$ cp nuc.fna mis_datos/

Copiando un archivo desde el directorio destino.
usuario@computadora:~/mis_datos$ cp ../nuc.fna .

Módulo 3: Manipulación de archivos

> Mover archivos y directorios

Con el comando mv se puede mover ya sea archivos o directorios. Es decir, cambia el nombre y/o la ubicación de un archivo.

usuario@computadora:~$ mv nuc.fna mis_datos/
usuario@computadora:~$ mv nuc.fna gene.fna

Módulo 3: Manipulación de archivos

> Transferencia de archivos

1. Para transferir un archivo de la maquina local a una maquina remota, el formato es:

   usuario@comp:~$ scp <path_local>/<archivo> usuario@ip:/path_destino/

2. Para transferir un archivo de la maquina remota a la maquina local, el formato es:

   usuario@comp:~$ scp usuario@ip:<path_destino>/<archivo> path_local

Ejercicio 6

1. Crea un directorio raw_data y entra al directorio en un solo paso.
2. Copia el archivo que esta en ~/mis_datos/contigs.fastq desde raw_data.
3. Renombra el archivo contigs.fastq y llámalo ecoli.fastq.
4. Regresa a HOME con un solo comando.
5. Crea una copia en HOME de nuc.fna y prot.faa que se llamen ecoliXX.fna y ecoliXX.faa.
6. Mueve a nuc.fna y prot.faa al directorio mis_datos.

7. Transfiere tus archivos ecoli.fna y ecoli.faa a tu cuenta en el servidor.
8. Transfiere tu directorio raw_data también.

Solución 6

a. **usuario@computadora:~$** mkdir raw_data && cd raw_data
b. **usuario@computadora:~/raw_data$** cp ~/mis_datos/contigs.fastq .
c. **usuario@computadora:~/raw_data$** mv contigs.fastq ecoli.fastq
d. **usuario@computadora:~/raw_data$** cd
e. **usuario@computadora:~$** cp nuc.fna ecoli.fna && cp prot.faa ecoli.faa
f. **usuario@computadora:~$** mv nuc.fna mis_datos/ && mv prot.faa mis_datos

g. **usuario@computadora:~$** scp ~/ecoliXX.fna usuario@ip:~/
h. **usuario@computadora:~$** scp ~/ecoliXX.faa usuario@ip:~/
i. **usuario@computadora:~$** scp -r ~/raw_data <path_local>/<archivo> usuario@ip:~/

Módulo 3: Manipulación de archivos

El comando **ln** crea una liga entre 2 archivos. Esto es similar a lo que harían otros sistemas (como MS Windows) al crear un acceso directo o “shortcut”.

usuario@computadora:~$ ln ~/mis_datos/contigs.fasta .

En este caso, se llama un “hard link”.

Cuando no se tiene propiedad sobre el archivo fuente o si está en otro sistema de archivos, hay que usar una liga ”soft”. Para esto, se usa la **opción -s**.

Ejercicio 7

1. Hacemos un directorio que se llame Resultados, y dentro los subdirectorios Especie1 y Especie2 .
2. Copiamos en ~/Resultados/Especie1 el archivo ~/mis_datos/especie01.fasta
3. Ahora dentro del subdirectorio Especie2, hacemos una "liga soft" con el archivo ~/mis_datos/especie02.fasta
4. Cambiamos el nombre de la liga ~/Resultados/Especie2/especie02.fasta por ecoli.fasta . ¿Cuál puede ser una razón de hacer una liga en lugar de una copia real?
5. Movemos ~/mis_datos/especie02.fasta al HOME
6. Revisamos nuevamente ~/Resultados/Especie2/Ecoli.fasta. ¿Qué sucedió?

Solución 6

usuario@computadora:~$ mkdir Resultados && cd Resultados
usuario@computadora:~/Resultados$ mkdir Especie1 && mkdir Especie2
usuario@computadora:~/Resultados$ cd Especie1
usuario@computadora:~/Resultados/Especie1$ cp ~/mis_datos/especie01.fasta .
usuario@computadora:~/Resultados$ cd ../Especie2
usuario@computadora:~/Resultados/Especie2$ ln -s ~/mis_datos/especie02.fasta .
usuario@computadora:~/Resultados/Especie2$ mv especie02.fasta ecoli.fasta
usuario@computadora:~/Resultados/Especie2$ cd
usuario@computadora:~$ cd mis_datos
usuario@computadora:~/mis_datos$ mv especie02.fasta ../

Módulo 3: Manipulación de archivos

> Permisos

Los sistemas UNIX tienen un enfoque particular en la seguridad de los datos.
Hasta ahora, hemos hablado de varios comandos y de su ejecución, así como de archivos, pero no hemos mencionado un detalle importante en UNIX que es el concepto de permisos:

- Para leer un archivo, debemos tener derecho de lectura = R
- Para escribir en un archivo, debemos tener derecho de escritura = W
- Para ejecutar un comando, debemos tener derecho de ejecución = X

Para visualizar los derechos de los archivos se puede usar el comando ls -l.

Módulo 3: Manipulación de archivos

> Cambios de derechos

El comando **chmod** permite cambiar los derechos de los archivos. Se pueden usar las tres letras, o la versión numérica (octal) correspondiente, con estas reglas: r=4, w=2 y x=1.

Para añadir derecho de ejecución al propietario (“u” = user) de un archivo:
usuario@computadora:~$ chmod u+x archivo

Para añadir derecho de escritura al grupo (“g” = group) de un archivo:
usuario@computadora:~$ chmod g+w archivo

Módulo 3: Manipulación de archivos

Para quitar derecho a los demás (“o” = others) de ejecución de un archivo:
usuario@computadora:~$ chmod o-x archivo

Para dar derecho de lectura total rwxr-xr-x a un archivo
usuario@computadora:~$ chmod 755 archivo

Para dar todos los derechos a un archivo a cualquiera (rwxrwxrwx)
usuario@computadora:~$ chmod 777 archivo

Ejercicio 8

1. El archivo contigs.fasta en el directorio mis_archivos contiene datos importantes.
2. Por lo tanto, es deseable que únicamente su usuario pueda acceder a estos datos, así como al directorio.
3. Además, quieren evitar que se pueda borrar por equivocación el archivo.

Si no están de acuerdo con los derechos de su HOME, ¿Cómo los modificarían para definir todos estos cambios?

Solución 8

Para evitar el acceso al directorio FASTA a cualquier otro usuario, lo ponemos sólo con los derechos rwx----. También el archivo secuencias.fasta debe tener entonces los derechos r-------

Módulo 3: Manipulación de archivos

> Nota: Git clone

- GitHub es una plataforma que sirve para trabajar en proyectos utilizando el sistema de control de versiones Git.
- Se utiliza principalmente para la creación de código fuente de programas de computadora.
- El software que opera GitHub fue escrito en Ruby on Rails.

Ejercicio 9

Supongamos que hay un script que nos interesa mucho, porque nos va a ayudar a analizar nuestros datos.

1. Entramos a la página del que nos interesa: https://github.com/solouli/IntroductionBioinfo
2. Vamos al proyecto específico.
3. Copiamos la liga que apunta al repositorio.
4. Abrimos la terminal y clonamos el repositorio:

usuario@comp:~$ git clone [https://github.com/solouli/IntroductionBioinfo.git]

5. Exploramos y ejecutamos el script

Módulo 3: Manipulación de archivos

> Nota: flechas ↑ y ↓

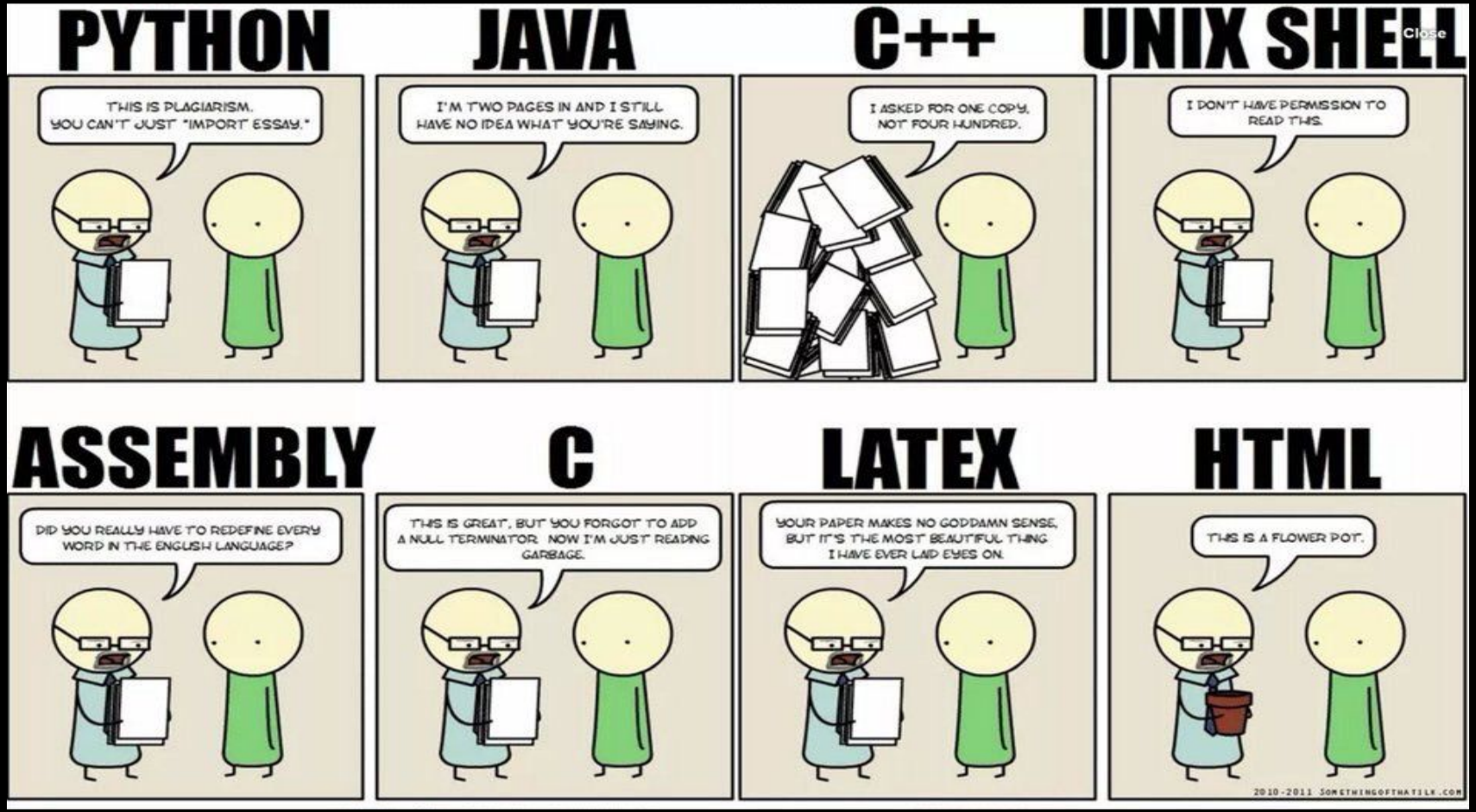
El historial de comandos que se utilizan en una sesión. Si utilizamos la tecla ↑ permite ver los comandos más antiguos mientras que la tecla ↓ permite ver los comandos más recientes.

> Nota: Ctr + r

Espera que tu le des un patrón de búsqueda. Te regresa las líneas que has ejecutado que contengan ese patrón.

> Nota: history

Almacena el historial de comandos ejecutados solamente. Si se hace logout, se pierde el historial. No es un comando.


PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.
C++
I ASKED FOR ONE COPY, NOT FOUR HUNDRED.
UNIX SHELL
I DON'T HAVE PERMISSION TO READ THIS.
ASSEMBLY
DID YOU REALLY HAVE TO REDEFINE EVERY WORD IN THE ENGLISH LANGUAGE?
C
THIS IS GREAT, BUT YOU FORGOT TO ADD A NULL TERMINATOR. NOW I'M JUST READING GARBAGE.
LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.
HTML
THIS IS A FLOWER POT.
2010-2011 SOMETHINGOFTHATILK.COM