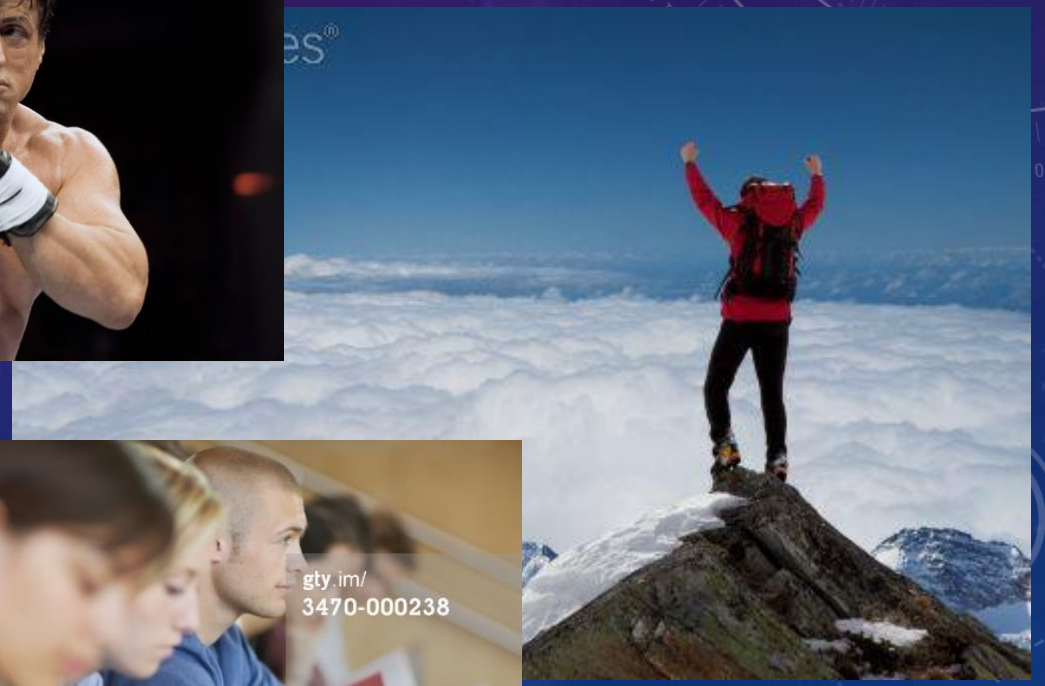
# SHELL SCRIPTING

CÓMO GENERAR SCRIPTS DE SHELL

# EL SECRETO DE SHELL SCRIPTING

- Para comprenderlo, es necesario **experimentar**, practicar e investigar.
- La **experiencia** será importante.



# ¿QUÉ HACE UN SCRIPT?

- Ejecuta los comandos que se le indican.
  - Por ejemplo, yo le puedo indicar que genere una carpeta y mueva ahí el archivo “x”.

- También se pueden incluir
  - Estructuras de control como **`for`** e **`if`**.
  - Variables
  - Cálculos (aunque esta no es la especialidad de un script de shell)

# ¿PARA QUÉ SIRVE UN SCRIPT DE SHELL?

- Facilita grandemente la **automatización de tareas**.
- Ahorra código hasta de lenguajes concisos.
- Puede integrar diversas herramientas en un solo script.
  - Por ejemplo, puedo invocar un script de Python y generar una gráfica de Gnuplot con los resultados que arroje este script.
  - Lo anterior facilita, entre otras cosas, experimentos de índole científico que deben reproducirse muchas veces.

# ¿CÓMO HACER UN SCRIPT DE SHELL?

- Se colocan los comandos a ejecutar en un archivo de texto plano (por convención, utilizamos la extensión **.sh**).
  - Nota: si el script es de una sola línea, no es necesario guardarlo en un archivo para ejecutarlo (simplemente se teclea en la terminal), pero si lo vamos a estar usando con frecuencia, podría ser conveniente guardarlo en un archivo.
- Se otorga un permiso de ejecución al script mediante **chmod u+x** (esto solo se hace una vez)
  - Ejemplo: Users/sara/> chmod u+x michelle.sh
- Se ejecuta el script cuando así se requiera
  - Ejemplo: `Users/sara$ ./michelle.sh`

# MI PRIMER SCRIPT DE SHELL

## hola.sh

```
#!/bin/bash

echo “Hola, mundo”
```

## Terminal

```
$ chmod u+x hola.sh
$ ./hola.sh
$ Hola, mundo
$ ./hola.sh
$ Hola, mundo
```

# echo

- Escribir en la salida estándar (terminal)
- Uso: echo *mensaje*

- Ejemplos:
  - echo “Hola”
  - echo “Hola, ”$nombre [aquí se concatena el contenido de la variable *nombre*]

# touch

- Si un archivo no existe, lo genera.
- Uso: touch *nombre_archivo*

# COMENTARIOS

- Se escriben anteponiendo #
- Ejemplo: #Este es un comentario

# VARIABLES

- Se manejan anteponiéndoles un **$**, excepto cuando se nombran por primera vez.
- Se pueden manejar dentro de estructuras de control (como los ciclos for), para los argumentos o, en general, para almacenar valores que surgen de la aplicación de algún comando.

# ARGUMENTOS

- Los argumentos son parámetros cuyos valores se reciben a la hora de ejecución.
- Dentro del script se especifican mediante **variables**.

# EJEMPLO ARGUMENTOS

## hola.sh

```
#!/bin/bash

echo “Hola, ”$1
```

## Terminal

```
$
$ ./hola.sh Sara
$ Hola, Sara
$ ./hola.sh Pepe
$ Hola, Pepe
```

# CICLO FOR

## Estructura

for variable in rango o colección a iterar

do

instrucciones

done

## Ejemplo

```
#!/bin/bash

for i in {1..10}
  do
     echo $i
 done
```

# OTROS EJEMPLOS FOR

## Imprimir números salteados

```
#!/bin/bash

for i in 2 3 5
  do
     echo $i
  done
```

## Listar contenidos de una carpeta

```
#!/bin/bash

for file in `ls folder`
 do
     echo $file
 done
```

Nota: para acceder a estos archivos, tendríamos que poner folder/$file.

# CONCATENACIÓN

- Mientras que en otros lenguajes la concatenación se hace con “+”, en shell se hace con yuxtaposición, es decir, poniendo los elementos que se van a concatenar, uno junto al otro.
  - Ejemplos:
    - “Hola, ”$nombre
    - “Hola, ”$nombre“, mucho gusto”

# GREP, SED Y AWK

TRES HERRAMIENTAS MUY PODEROSAS DE LINUX

# grep

- Significa “**g**lobally search a **r**egular **e**xpression and **p**rint”
- Sirve para buscar patrones en un archivo de texto
- Uso grep *patrón archivo*
- Las opciones que veremos son:
  - -A *número*: Imprimir un número de líneas posteriores al patrón encontrado.
  - -B *número*: Imprimir un número de líneas anteriores al patrón encontrado.
  - -v: Invertir la búsqueda (dar resultados que *no* contengan ese patrón).

# sed

- Significa “**s**tream **ed**itor”.
- Se usa para modificar archivos (o la entrada que se le proporcione en la terminal).
- Veremos dos usos:
  - Reemplazo de texto
  - Eliminación de líneas en un archivo

# REEMPLAZO CON sed

## Uso

- sed 's/texto a reemplazar/texto por el que se reemplaza/g' *archivo*

## Ejemplos

```
$ sed 's/antiguo/nuevo/g' archivo
```

[Reemplaza todas las ocurrencias de “antiguo” por “nuevo”]

```
$ sed 's/antiguo//g' archivo
```

[Reemplaza todas las ocurrencias de “antiguo” por nada]

# BORRAR LÍNEAS CON sed

## Uso

- Una línea
  - sed ‘**n**d’ *archivo*
  - n indica el número de línea a borrar

- Un rango de líneas
  - Sed ‘**m,n**d’ *archivo*
  - m,n indican el rango [m,n] de líneas

## Ejemplos

```
$ sed ‘1d’ archivo
```

[Borra la primera línea del archivo]

# awk


- Lleva las iniciales de sus creadores: **A**ho, **W**einberger y **K**ernighan.
- Sirve para procesar texto
- Uso: awk ‘{acción}’ *archivo*, donde *acción* se llevará a cabo en cada línea del texto
  - Adicionalmente puede haber pre y postprocesamiento con BEGIN y END.
- Se cuenta con variables especiales, tales como **NR** y **NF**, las cuales contienen, respectivamente la cantidad de líneas y la cantidad de campos para cada línea (se puede especificar un separador con la opción **-F**).
- También se pueden crear scripts de awk, que se invocan mediante **awk -f script.awk**