Tiempo para aprender juntos

Dirección de Servicios de
Infraestructura y Operaciones

Diciembre 2020

Bash Scripting

CopaAirlines

Agenda

- ¿Qué es scripting?
- ¿Qué es shell de Unix?
- ¿Cómo usar Bash?
- ¿Cómo crear un crontab?
- Guía de comandos más utilizados
- Guías de aprendizaje
- Laboratorios

CopaAirlines

¿Qué es scripting?

- Los **scripts** son programas **NO compilados**, es decir, necesitan de un programa lector o **interprete** lo lleve a lenguaje de máquina, para que la información sea procesada y ejecutada por la computadora.
- A nivel de sistema operativo
  - Unix
    - bash
    - ksh
    - csh
  - Windows
    - .bat
    - .cmd
- En diseño web
  - Cliente
    - JavaScript
  - Servidor
    - PHP, ASP

CopaAirlines

¿Qué es shell de unix?

- Es un **intérprete de comandos** para sistemas operativos basados en **Unix**.

- Mediante las instrucciones se puede comunicar con el **kernel** y ejecutar las instrucciones dadas.

- La ejecución puede darse a través:
  - Interfaz tradicional
    - sh, **Bash**, ash, csh, Zsh, ksk, tcsh
  - Interfaz gráfica
    - GNOME, KDE, Xfce

CopaAirlines

¿Cómo usar bash?

- En la primer línea del script hay que agregar el intérprete:
  - #!/bin/bash
- Para comentar una línea, ésta inicia con el carácter "#"
  - # Documentar el código nos ahorra tiempo
- Hay que cambiar el permiso del archivo a ejecución:
  - chmod +x archivo.sh
- Para ejecutarlo
  - ./archivo.sh

¿Cómo crear un crontab?

- Son scripts que se ejecutan de manera periódica

```
* * * * * *
| | | | | |
| | | | | +---comando a ejecutar
| | | | |
| | | | +---- día de la semana (0 - 6) (domingo=0 o 7)
| | | +------- mes (1 - 12)
| | +---------- día del mes (1 - 31)
| +------------ hora (0 - 23)
+-------------- minuto (0 - 59)
```

- Para listarlo
  - crontab -l

- Para activarlo:
  - crontab -e
  - Selecciona la opción 1. /bin/nano

CopaAirlines

#!/bin/bash

Guía de comandos más utilizados en bash shell

EJECUCIÓN

./comando	# ejecuta desde directorio actual
$RUTA/comando	# ejecuta desde cualquier sitio
comando	# ejecuta si está en el $PATH
. script	# ejecuta exportando variables
$(comando param1 ... paramN**)**	# ejecuta de forma literal
`comando param1 ... paramN`	# ejecuta sustituyendo variables
comando **&**	# ejecuta en segundo plano
c1 **|** c2	# redirige salida c1 a entrada c2
c1 ; c2	# ejecuta c1 y luego c2
c1 **&&** c2	# ejecuta c2 si c1 termina sin errores
c1 **||** c2	# ejecuta c2 si c1 termina con errores

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

REDIRECCIONES

output (salida estándar)

`tee fichero`	`# output a fichero y a pantalla`
`> fichero`	`# output a fichero`
`>> fichero`	`# output al final del fichero`
`> /dev/null`	`# descarta output`

error

`2>&1`	`# error a output`
`2> fichero`	`# error a fichero`
`2>> fichero`	`# error al final del fichero`
`2> /dev/null`	`# descarta error`

output y error

`2>&1 | tee fichero`	`# ambos a fichero y a pantalla`
`&> fichero`	`# ambos a fichero`
`&>> fichero`	`# ambos al final del fichero`

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

VARIABLES

variables de entorno

$PWD	# directorio de trabajo actual
$OLDPWD	# directorio de trabajo anterior
$PPID	# identificador del proceso padre
$HOSTNAME	# nombre del ordenador
$USER	# nombre del usuario
$HOME	# directorio del usuario
$PATH	# rutas búsqueda de comandos
$LANG	# idioma para los mensajes
$FUNCNAME	# nombre función en ejecución
$LINENO	# número de línea actual (del script)
$RANDOM	# número aleatorio

variables especiales

$0	# nombre del script
${N}	# parámetro N
$$	# identificador del proceso actual
$!	# identificador del último proceso
$@ (como array) ó $* (como string)	# todos los parámetros recibidos
$#	# número de parámetros recibidos
$? # (0=normal, >0=error)	# código de retorno del último comando
shift	# $1=$2, $2=$3, ... ${N-1}=${N}

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

OPERADORES

operadores aritméticos

+	# suma
-	# resta
*	# multiplicación
/	# división
%	# resto
++	# incremento
--	# decremento

operadores comparaciones numéricas

numero1 **-eq** numero2	# numero1 igual que numero2
numero1 **-ne** numero2	# numero1 distinto que numero2
numero1 **-lt** numero2	# numero1 menor que numero2
numero1 **-le** numero2	# numero1 menor o igual que numero2
numero1 **-gt** numero2	# numero1 mayor que numero2
numero1 **-ge** numero2	# numero1 mayor o igual que numero2

operadores lógicos

!	# NOT
&& , -a	# AND
|| , -o	# OR

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

operadores de ficheros

-e fichero	# existe
-s fichero	# no está vacío
-f fichero	# normal
-d fichero	# directorio
-h fichero	# enlace simbólico
-r fichero	# permiso de lectura
-w fichero	# permiso de escritura
-x fichero	# permiso de ejecución
-O fichero	# propietario
-G fichero	# pertenece al grupo
f1 **-ef** f2	# f1 y f2 enlaces mismo archivo
f1 **-nt** f2	# f1 más nuevo que f2
f1 **-ot** f2	# f1 más antiguo que f2

operadores de cadenas

-n cadena	# no vacía
-z cadena	# vacía
cadena1 = cadena2	# cadena1 igual a cadena2
cadena1 == cadena2	# cadena1 igual a cadena2
cadena1 != cadena2	# cadena1 distinta a cadena2

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

ENTRECOMILLADO

#! RUTA	# ruta al interprete (/bin/bash)
\carácter	# valor literal del carácter
linea1 \ linea2	# para escribir en varias lineas
'cadena'	# valor literal cadena
“cadena”	# valor literal cadena, excepto $ ' \

EXPANSIÓN

[prefijo]{cad1,[,...],cadN}[sufijo]	# = precad1suf ... precadNsuf
${VARIABLE:-valor}	# si VARIABLE nula, retorna valor
${VARIABLE:=valor}	# si VARIABLE nula, asigna valor
${VARIABLE:?mensaje}	# si VARIABLE nula, mensaje error y fin
${VARIABLE:inicio}	# recorta desde inicio hasta el final
${VARIABLE:inicio:longitud}	# recorta desde inicio hasta longitud
${!prefijo*}	# nombres de variables con prefijo
${#VARIABLE}	# número de caracteres de VARIABLE
${#ARRAY[*]}	# elementos de ARRAY
${VARIABLE#patrón}	# elimina mínimo patrón desde inicio
${VARIABLE##patrón}	# elimina máximo patrón desde inicio
${VARIABLE%patrón}	# elimina mínimo patrón desde fin
${VARIABLE%%patrón}	# elimina máximo patrón desde fin
${VARIABLE/patrón/reemplazo}	# reemplaza primera coincidencia
${VARIABLE//patrón/reemplazo}	# reemplaza todas las coincidencias
$((expresión))	# sustituye expresión por su valor
$[expresión]	# sustituye expresión por su valor

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

CopaAirlines

#!/bin/bash

Guía de comandos más utilizados en bash shell

ESTRUCTURAS DE CONTROL

if expresión1; **then**
  bloque1
elif expresión2; **then**
  bloque2
else
  bloque3
fi
	# **condicional**
	# si expresión1 entonces
	# bloque1
	# sino y expresión2 entonces
	# bloque2
	# si ninguna entonces
	# bloque2

case VARIABLE **in**
  patrón11|...|patrón1N)
  bloque1 ;;
  patrón21|...|patrón2N)
  bloque2 ;;
  *)
  bloqueDefecto ;;
esac
	# **selectiva**
	# si VARIABLE coincide con patrones1
	# entonces bloque1
	# si VARIABLE coincide con patrones2
	# entonces bloque2
	# si ninguna
	# entonces bloqueDefecto

for VARIABLE **in** LISTA; **do** bloque **done**	# **iterativa con lista** # ejecuta bloque sustituyendo # VARIABLE por cada elemento de LISTA
for ((expr1; expr2; expr3;)); **do** bloque **done**	# **iterativa con contador** # primero se evalúa exp1 # luego mientras exp2 sea cierta # se ejecutan el bloque y expr3
while expresión; **do** bloque **done**	# **bucle "mientras"** # se ejecuta bloque # mientras expresión sea cierta
until expresion; **do** expresion **done**	# **bucle "hasta"** # se ejecuta bloque # hasta que expresión sea cierta
[function] expresion **() {** **...** **[return** [valor] **]** ... **}**	# **función** # se invoca con # nombreFunción [param1 ... paramN]

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

ARRAYS

`declare -a ARRAY`	`# declaración array`
`ARRAY=(valor1 ... valorN)`	`# asignación compuesta`
`ARRAY[N]=valorN`	`# asignación simple`
`ARRAY=([N]=valorN valorM [P]=valorP)`	`# asigna celdas N, M y P`
`${ARRAY[N]}`	`# valor celda N`
`${ARRAY[*]}`	`# todos los valores`

ARGUMENTOS DE LÍNEA DE COMANDOS

```
while getopts "hs:" option ; do
  case "$option" in
    h) DO_HELP=1 ;;
    s) argument=$OPTARG ; DO_SEARCH=1 ;;
    *) echo "Invalid" ; return ;;
  esac
done
```

```
# getops + "opciones disponibles"
#   mientras haya argumentos
#     seleccionamos
#       -h sin opciones
#       -s con opciones en $OPTARG
#        * error
```

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guía de comandos más utilizados en bash shell

INTERACTIVIDAD

read [-p cadena] [variable1 ...]	# **input** # lee teclado y asigna a variables # puede mostrarse un mensaje antes # si ninguna variable, REPLY = todo
echo cadena **-n** no hace salto de linea **-e** interpreta caracteres con \	# **output** # manda el valor de la cadena # a la salida estándar
printf	# **output formateado** (igual que C)

CONTROL DE PROCESOS

comando **&**	# ejecuta en segundo plano
bg númeroProceso	# continúa ejecución en segundo plano
fg númeroProceso	# continúa ejecución en primer plano
jobs	# muestra procesos en ejecución
kill señal PID1|númeroProceso1	# mata proceso(s) indicado(s)
exit código	# salir con código de retorno # (0=normal, >0=error)
trap [comando] [código1 ...]	# ejecuta comando cuando señal(es)
wait [PID1|númeroProceso1]	# espera hasta fin proceso(s) hijo(s)
nice -n prioridad comando **renice** -n prioridad comando	# ejecuta comando con prioridad [-20/19] # modifica prioridad comando [-20/19] # -20 máxima prioridad y 19 mínima

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

#!/bin/bash

Guías de aprendizaje

- **Programar** es una **habilidad** que se desarrolla **poco a poco** practicando.
- **Llénate** de **paciencia** porque hay que saber como hablarle al sistema operativo a través del lenguaje de programación que estés usando para que haga lo que necesites.
- **Busca** en **internet** y **foros** (Google, https://stackoverflow.com/), hay gente que ya pasó por algo parecido y comparte sus conocimientos.
- **Permítete aprender algo nuevo, se vale equivocarse.**
- **Dato curioso**: Margaret Hamilton fue la primer Ingeniera de Software (misión Apolo de la NASA).

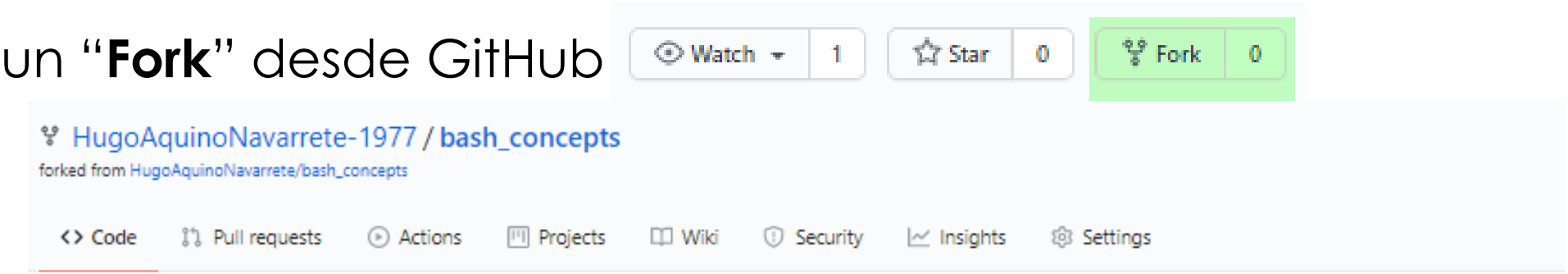
CopaAirlines

#!/bin/bash

Laboratorios – Comienza la diversión

- Ve al siguiente repositorio:
  - https://github.com/HugoAquinoNavarrete/bash_concepts

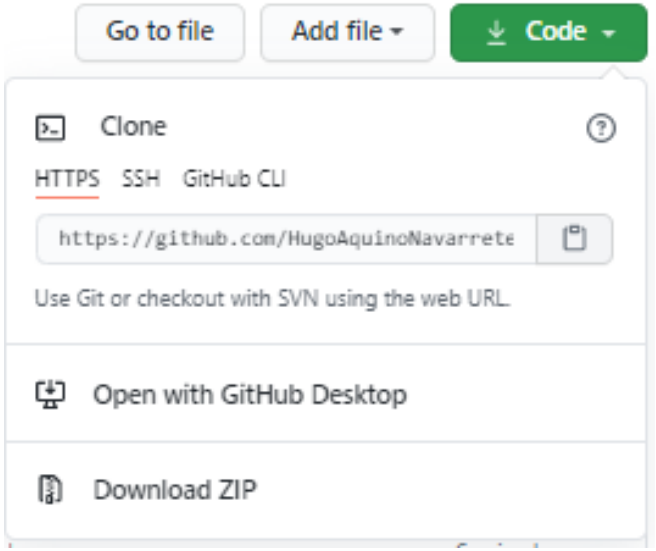
- Haz un "**Fork**" desde GitHub



CopaAirlines

#!/bin/bash

- **Clónalo** en algún directorio de tu VM
  - git clone https://github.com/<tu_usuario>/bash_concepts.git



- Ve al directorio "**bin**" y revisa el contenido del script "**00-primer_script.sh**"
  - cd bin
  - cat 00-primer_script.sh
- Intenta ejecutarlo
  - ./00-primer_script.sh
- ¿Te dejó?

CopaAirlines

#!/bin/bash

- En caso de no poderse ejecutar, cambia el permiso a ejecución e intenta ejecutarlo nuevamente
  - chmod +x 00-primer_script.sh
  - ./00-primer_script.sh

- **Reto # 0. Cambia el permiso a ejecución de todos los archivos de la carpeta "bin" utilizando 1 sola línea de comando.**

#!/bin/bash

Laboratorios

- Revisa el contenido del script "**01-imprime_fecha.sh**"
  - cat 01-imprime_fecha.sh

- Ejecuta el script "**01-imprime_fecha.sh**"
  - ./01-imprime_fecha.sh

- Haz **dos crons** que se ejecuten cada **2 minutos**, el primer archivo se llamará "**fecha.txt**" y se creará cada vez que se ejecute el cron, el otro archivo se llamará "**fecha.log**" y en cada ejecución se irá agregando la fecha. Deposita cada archivo en el directorio "**output**".
  - crontab -e

  - */2 * * * * /home/ubuntu/bash_scripting/bin/01-imprime_fecha.sh > /home/ubuntu/bash_scripting/output/fecha.txt
  - */2 * * * * /home/ubuntu/bash_scripting/bin/01-imprime_fecha.sh >> /home/ubuntu/bash_scripting/output/fecha.log

CopaAirlines

#!/bin/bash

- Un script genera mucho más valor cuando interactúa con su entorno a través del uso de variables y parámetros de la línea de entrada.

- Revisa el contenido del script "**02-parametros.sh**" y ejecútalo
  - ./02-parametros.sh

- Ahora revisa el contenido del script "**03-parametros_uso_if.sh**" y ejecútalo.
  - ./03-parametros_uso_if.sh

- **Reto # 1. Completa el script "reto_1.sh" para que haga lo siguiente:**
  - Si no hay parámetros de entrada, imprime el mensaje "Te faltaron indicar los parámetros"
  - Si solo se ingresa un parámetro de entrada imprime el mensaje "Solo se ingreso un parámetro, falta otro"
  - Si ingresas dos parámetros de entrada imprime este mensaje: "Los valores ingresados fueron: <val_param1> y <val_param2>"

#!/bin/bash

- Revisa el contenido de "**04-operaciones_aritmeticas_enteros.sh**" y "**05-operaciones_aritmeticas_flotantes.sh**", ejecútalos:
  - ./04-operaciones_aritmeticas_enteros.sh
  - ./05-operaciones_aritmeticas_flotantes.sh
- **Reto # 2. Completa el script "reto_2.sh" para que haga lo siguiente:**
  - Solicita 2 variables numérica a través de la línea de comando
  - Imprime alguno de los 3 siguientes mensajes:
    - Si el primer número es mayor que el segundo:
      - "El primer número <numero_1> es mayor que el segundo número <numero_2>"
    - Si el segundo número es mayor que el primero:
      - "El segundo número <numero_2> es mayor que el primer número <numero_1>"
    - Si ambos números son iguales:
      - "Tanto el número <numero_1> como el número <numero_2> son iguales"

#!/bin/bash

Laboratorios

- Revisa el contenido del script "**06-mi_entorno.sh**" y ejecútalo
  - ./06-mi_entorno.sh
- Ahora revisa el contenido del script "**07-imprime_palabras_n_veces.sh**" y ejecútalo
  - ./07-imprime_palabras_n_veces.sh
- **Reto # 3. Completa el script "reto_3.sh" para que haga lo siguiente:**
- Si no ingresaste los 2 parámetros imprime el mensaje de como se debería correr el script: "./reto_3.sh <palabra> <veces>"
- Si ingresaste los 2 parámetros haz lo siguiente:
- 1. Almacena el primer parámetro en una variable que se llame "palabra"
- 2. Obten el tamaño de la variable "palabra" y almacénalo en una variable que se llame "size"
- 3. Imprime la variable "palabra" tantas veces de acuerdo al valor de la variable "size"
- 4. Suma el valor de la variable "size" con el segundo parámetro y almacénalo en una variable que se llame "total"
- 5. Imprime la variable de ambiente $USER tantas veces como el valor de la variable "total"

#!/bin/bash

Laboratorios

- Revisa el contenido del script "**08-lee_archivo.sh**" y ejecútalo una primera vez
  - ./08-lee_archivo.sh
- En el directorio "**input**" edita el archivo "**directorios.txt**" y agrega 5 líneas donde cada línea contenga una sola palabra.
- Nuevamente ejecuta el script "**08-lee_archivo.sh**"
  - ./08-lee_archivo.sh
- Vuelve a ejecutar el script "**08-lee_archivo.sh**" una vez más y mira la salida en pantalla
  - ./08-lee_archivo.sh
- **Reto # 4. Mejora el script "08-lee_archivo.sh", basándote en el "reto_4.sh" para que haga lo siguiente:**
- Ingresa un parámetro de entrada que puede tener el valor "crea" o "borra".
- Si no ingresaste parámetro de entrada, que indique que falta
- Si ingresaste "crea", el script creará los directorios contenidos en el archivo "directorios.txt"
- Si ingresaste "borra", el script eliminará los directorios contenidos en el archivo "directorios.txt"

#!/bin/bash

- Revisa el contenido del script "**09-convierte_fahrenheit_centigrados.sh**" y ejecútalo
  - ./09-convierte_fahrenheit_centigrados.sh

- **Reto # 5. Toma como base el script "09-convierte_fahrenheit_centigrados.sh" para crear el script "reto_5.sh" para que haga lo siguiente:**
- Lee el archivo "libras.txt" que se encuentra en el directorio "input".
- Crea una función que se llame "libras_a_kilogramos" que convierta el valor de libras a kilogramos, redondeando a 3 decimales.
- Crea un archivo en el directorio "output" que se llame "peso.txt", en la primer línea aparecerá "libras,kilogramos" y a partir de la segunda línea, el valor de libras seguido de una "," y posteriormente el valor de la conversión a kilogramos (no debe hacer espacios entre los valores y la "coma").

#!/bin/bash

Laboratorios

- Ve al directorio "**input**" y compartamos sobre lo que hace cada una de las siguientes líneas:
  - cat mlb_parks.csv
  - cat mlb_parks.csv | grep -E X
  - cat mlb_parks.csv | grep -E "^W"
  - cat mlb_parks.csv | grep -E "US$"
  - cat mlb_parks.csv | grep -Ev "US$"
  - cat mlb_parks.csv | grep -E "Washington"
  - cat mlb_parks.csv | grep -E "Washington\,[A-Z]{2}\,[A-Z]{2}$"
  - cat mlb_parks.csv | grep -Ev "^park\." | grep -E "US$" | wc -l
  - cat mlb_parks.csv | grep -Ev "^park\." | grep -Ev "US$" | wc -l

#!/bin/bash

- Usando expresiones regulares (**1 sola línea con el uso de "grep" y "pipes"**) como las que acabamos de ver, contesta las siguientes preguntas al procesar el archivo "**mlb_parks.csv**":

  - ¿Cuántos estadios hay en **total**?
  - ¿Cuántos estadios hay en **Estados Unidos (US)**?
  - ¿Cuántos estadios hay **fuera** de **Estados Unidos**?
  - ¿En que **países** se encuentran los estadios **fuera** de **Estados Unidos**?
  - ¿Cuántos estadios hay en el país **Canadá (CA)**?
  - ¿Cuántos estadios hay en el **Estado de California (CA)**, **Estados Unidos**?
  - ¿Cuántos estadios **tienen** la palabra "**Stadium**"?
  - ¿Cuántos estadios **no tienen** la palabra "**Park**"?

#!/bin/bash

- Usando expresiones regulares (**1 sola línea con el uso de "grep" y "pipes"**), contesta las siguientes preguntas al procesar el archivo "**mlb_teams.csv**":

  - ¿Cuántos equipos había en **1850**?
  - ¿Cuántos equipos había en **1885**?
  - ¿Cuántos equipos había en **1920**?
  - ¿Cuántos equipos había en **1989**?
  - ¿Cuántos equipos había en **2019**?

#!/bin/bash

- Revisa el contenido del script "**10-mlb_equipos.sh**" y ejecútalo.
  - ./10-mlb_equipos.sh <year>

- **Reto # 6. El script "reto_6.sh" es copia del script "10-mlb_equipos.sh", al cual hay que agregarle un par de líneas para que haga lo siguiente:**

- Imprime en pantalla la cantidad de equipos y audiencia total que hubieron en el año que seleccionaste: "En el año <year> hubieron <cantidad_equipos> equipos y la audiencia total a los estadios fue de <total_audiencia> millones de espectadores"
  - Nota 1: el valor de <total_audiencia> divídelo por 1000000 (un millón) y redondea el número a 2 decimales.
  - Nota 2: a partir de 1892 es que aparece el registro de espectadores por equipo.

Quiz

- Tiempo para repasar jugando

Gracias

CopaAirlines