

Computación en Física

Harvey Rodriguez Gil

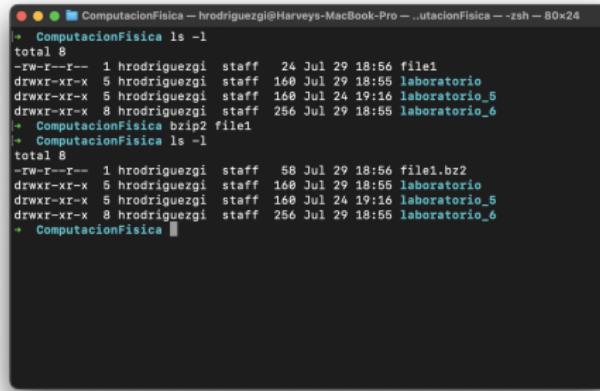
Universidad EIA

30 de Julio de 2024

Comprimiendo Datos - bzip2

En el mundo de linux tenemos más allá de solamente **zip**. Un primer ejemplo es: **bzip2**, el cual tiene los siguientes comandos:

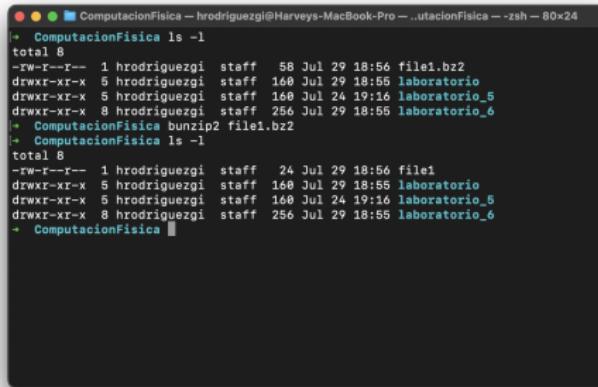
- ▶ **bzip2**: para comprimir
- ▶ **bzcat**: para visualizar el contenido de un fichero
- ▶ **bunzip2**: para descomprimir



```
ComputacionFisica ~ hrodriguezgi@Harveys-MacBook-Pro ~..utacionFisica ~ zsh ~ 80x24
ls -l
total 8
-rw-r--r-- 1 hrodriguezgi staff 24 Jul 29 18:56 file1
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 29 18:55 laboratorio
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezgi staff 256 Jul 29 18:55 laboratorio_6
bzip2 file1
ls -l
total 8
-rw-r--r-- 1 hrodriguezgi staff 58 Jul 29 18:56 file1.bz2
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 29 18:55 laboratorio
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezgi staff 256 Jul 29 18:55 laboratorio_6
```

Comprimiendo Datos - bzip2

Se debe tener en cuenta que este comando solo comprime ficheros individuales, no carpetas.

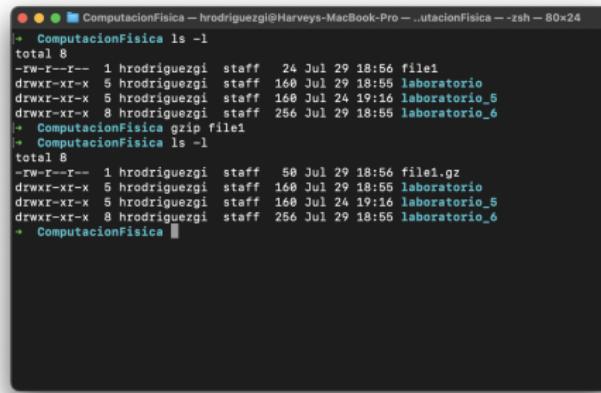


```
ComputacionFisica ~ hrodriguezg@Harveys-MacBook-Pro ~ utacionFisica ~ zsh ~ 80x24
+ ComputacionFisica ls -l
total 8
-rw-r--r-- 1 hrodriguezg staff 58 Jul 29 18:56 file1.bz2
drwxr-xr-x 5 hrodriguezg staff 160 Jul 29 18:56 laboratorio
drwxr-xr-x 5 hrodriguezg staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezg staff 256 Jul 29 18:56 laboratorio_6
+ ComputacionFisica bzip2 file1.bz2
+ ComputacionFisica ls -l
total 8
-rw-r--r-- 1 hrodriguezg staff 24 Jul 29 18:56 file1
drwxr-xr-x 5 hrodriguezg staff 160 Jul 29 18:56 laboratorio
drwxr-xr-x 5 hrodriguezg staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezg staff 256 Jul 29 18:55 laboratorio_6
+ ComputacionFisica
```

Comprimiendo Datos - gzip

Otro popular compresor es **gzip**, el cual es una creación del proyecto GNU.

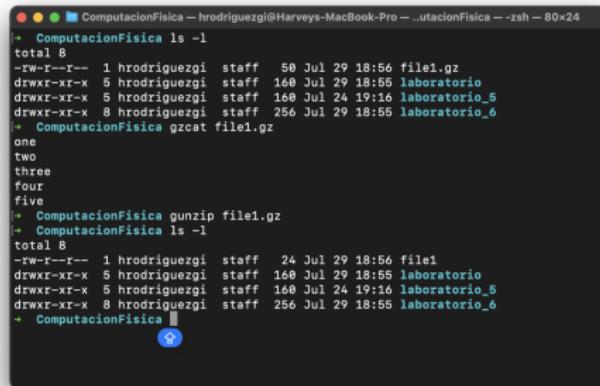
- ▶ **gzip**: para comprimir
- ▶ **gzcat**: para visualizar el contenido de un fichero
- ▶ **gunzip**: para descomprimir



```
ComputacionFisica ~ hrodriguezgi@Harveys-MacBook-Pro ~..utacionFisica ~ zsh ~ 80x24
+- ComputacionFisica ls -l
total 8
-rw-r--r-- 1 hrodriguezgi staff 24 Jul 29 18:56 file1
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 29 18:55 laboratorio
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezgi staff 256 Jul 29 18:55 laboratorio_6
+ ComputacionFisica gzip file1
+ ComputacionFisica ls -l
total 8
-rw-r--r-- 1 hrodriguezgi staff 50 Jul 29 18:56 file1.gz
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 29 18:55 laboratorio
drwxr-xr-x 5 hrodriguezgi staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezgi staff 256 Jul 29 18:55 laboratorio_6
+ ComputacionFisica
```

Comprimiendo Datos - gzip

Se debe tener en cuenta que este comando solo comprime ficheros individuales, no carpetas.



```
ComputacionFisica ~ hrodriguezg@Harveys-MacBook-Pro ~ .utacionFisica ~ zsh ~ 80x24
total 8
-rw-r--r-- 1 hrodriguezg staff 50 Jul 29 18:56 file1.gz
drwxr-xr-x 5 hrodriguezg staff 160 Jul 29 18:56 laboratorio
drwxr-xr-x 5 hrodriguezg staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezg staff 256 Jul 29 18:56 laboratorio_6
ComputacionFisica gzip file1.gz
one
two
three
four
five
ComputacionFisica gunzip file1.gz
ComputacionFisica ls -l
total 8
-rw-r--r-- 1 hrodriguezg staff 24 Jul 29 18:56 file1
drwxr-xr-x 5 hrodriguezg staff 160 Jul 29 18:56 laboratorio
drwxr-xr-x 5 hrodriguezg staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezg staff 256 Jul 29 18:56 laboratorio_6
ComputacionFisica
```

Archivando Datos - tar

Para complementar los dos comandos anteriores, en linux contamos con **tar**, el cual es un comando que nos permite archivar (empaquetar). La sintaxis de este comando es la siguiente:

```
tar function [options] object1 object2...
```

Las funciones determinarán que debe hacer el comando **tar**, como comprimir o crear.

Archivando Datos - tar

Las funciones de tar son las siguientes:

Función	Descripción
-A	Adiciona un archivo tar existente a otro archivo tar.
-c	Crea un nuevo archivo tar.
-r	Adiciona archivos al final de un archivo tar existente.
-t	Lista el contenido de un archivo tar existente.
-x	Extrae los archivos de un archivo tar existente.
-p	Preserva los permisos de los archivos.
-v	Lista los archivos que están siendo procesados.
-f	Nombre del archivo tar (comprimir/descomprimir).

Archivando Datos - tar

Este comando permite trabajar con bzip2 o con gzip para lograr la compresión de carpetas:

Función	Descripción
-j	Redirecciona la salida al comando bzip2 para compresión.
-z	Redirecciona la salida al comando gzip para compresión.

```
ComputacionFisica ~ hrodriguezg@Harveys-MacBook-Pro ~ ..utacionFisica ~ zsh ~ 80x24
+- ComputacionFisica ls -lR laboratorio
total 24
-rw-r--r--@ 1 hrodriguezg staff 39 Jul 24 20:06 data.txt
-rw-r--r--@ 1 hrodriguezg staff 2149 Jul 24 19:16 laboratorio.zip
-rw-r--r--@ 1 hrodriguezg staff 273 Jul 24 20:13 log.txt
+- ComputacionFisica tar -cvzf laboratorio.tar.gz laboratorio
a laboratorio
a laboratorio/log.txt
a laboratorio/data.txt
a laboratorio/laboratorio.zip
+- ComputacionFisica ls -l
total 16
-rw-r--r--@ 1 hrodriguezg staff 24 Jul 29 18:56 file1
drwxr-xr-x 5 hrodriguezg staff 160 Jul 29 19:29 laboratorio
-rw-r--r--@ 1 hrodriguezg staff 2833 Jul 29 19:30 laboratorio.tar.gz
drwxr-xr-x 5 hrodriguezg staff 160 Jul 24 19:16 laboratorio_5
drwxr-xr-x 8 hrodriguezg staff 256 Jul 29 18:55 laboratorio_6
+- ComputacionFisica
```

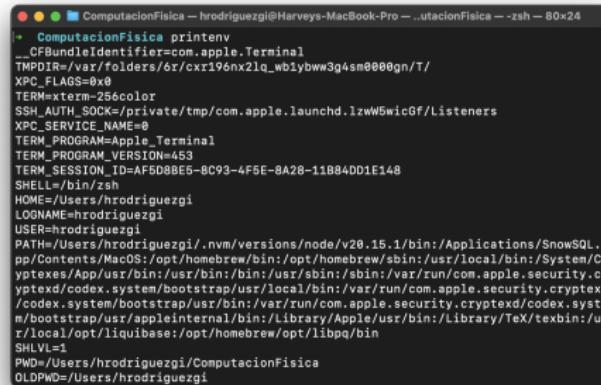
Variab es de entorno

El intérprete de bash utiliza una característica llamada variables de ambiente o de entorno para almacenar información ya sea del intérprete en el que estamos, del ambiente de trabajo o del sistema en si. Estas variables pueden ser accedidas tanto desde el mismo intérprete como desde un script que se esté ejecutando. En Linux existen dos tipos de variables:

- ▶ Variables Globales
- ▶ Variables Locales

Variab es Globales

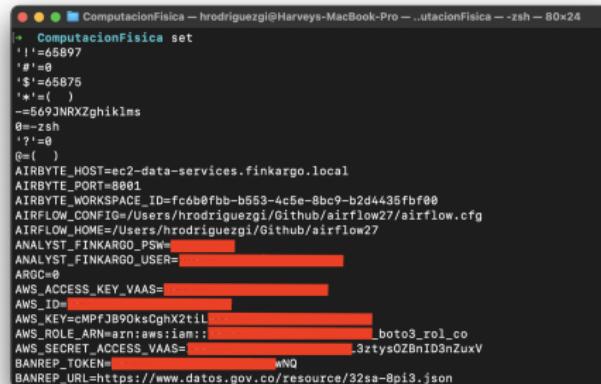
Estas variables son visibles desde la sesión de shell (intérprete) y cualquier proceso que se ejecute en ella. Para ver las variables que tenemos en el sistema podemos ejecutar el comando `printenv`



```
ComputacionFisica ~ hrodriguezgi@Harveys-MacBook-Pro ~ utacionFisica ~ zsh - 80x24
+ ComputacionFisica printenv
__CFBundleIdentifier=com.apple.Terminal
TMPDIR=/var/folders/6r/cxr196nx2lq_wbiybw3g4sm000gn/T/
XPC_FLAGS=0x0
TERM=xterm-256color
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.lzwW5wicGF/Listeners
XPC_SERVICE_NAME=@
TERM_PROGRAM=Apple_Terminal
TERM_PROGRAM_VERSION=453
TERM_SESSION_ID=AF5D8BE5-8C93-4F5E-8A28-11B84DD1E148
SHELL=/bin/zsh
HOME=/Users/hrodriguezgi
LOGNAME=hrodriguezgi
USER=hrodriguezgi
PATH=/Users/hrodriguezgi/.nvm/versions/node/v20.15.1/bin:/Applications/SnowSQL.app/Contents/MacOS:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/var/run/com.apple.security.cryptexd/codex/system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd /codex/system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex/system/bootstrap/usr/appleinternal/bin:/Library/Apple/usr/bin:/Library/TeX/texbin:/usr/local/opt/liquibase:/opt/homebrew/opt/libpq/bin
SHLVL=1
PWD=/users/hrodriguezgi/ComputacionFisica
OLDPWD=/Users/hrodriguezgi
```

Variab es Locales

Estas variables sólo son visibles en el proceso local en el cual ellos están definidos. A diferencia de las variables globales, no tenemos un comando que nos permita ver solo las variables locales, sin embargo podemos utilizar set con cuidado.

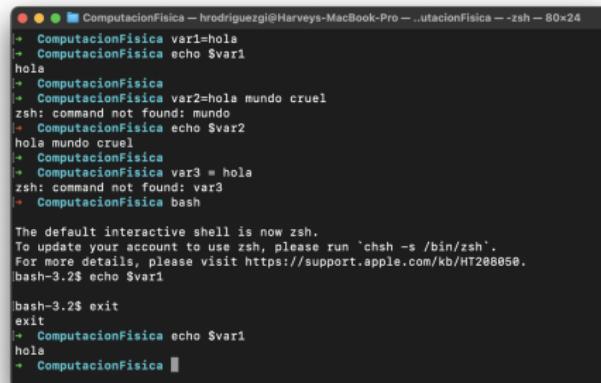


```
+ ComputacionFisica set
':!=65897
':#=0
'$'=65875
'*='( )
-=569JNRXZghiklms
@=zsh
@?=0
@=( )
AIRBYTE_HOST=ec2-data-services.finkargo.local
AIRBYTE_PORT=8001
AIRBYTE_WORKSPACE_ID=fcb6b0fbb-b553-4c5e-8bc9-b2d4435fbf00
AIRFLOW_CONFIG=/Users/hrodiguezgi/Github/airflow27/airflow.cfg
AIRFLOW_HOME=/Users/hrodiguezgi/Github/airflow27
ANALYST_FINKARGO_PSW=[REDACTED]
ANALYST_FINKARGO_USER=[REDACTED]
ARGCH@#
AWS_ACCESS_KEY_VAAS=[REDACTED]
AWS_ID_VAAS=[REDACTED]
AWS_ROLE_ARN=arn:aws:iam::[REDACTED]_boto3_rol_co
AWS_SECRET_ACCESS_VAAS=[REDACTED]_3tysO2BnId3nZuxV
BANREP_TOKEN=[REDACTED]wNQ
BANREP_URL=https://www.datos.gov.co/resource/32sa-8pi3.json
```

Variabes Locales

Podemos crear nuestras variables directamente en el shell (o en los scripts). Para esto podemos ejecutar algo como lo sigue:

```
$ var1=hola
```



The screenshot shows a terminal window titled "ComputacionFisica" running on a MacBook Pro. The session starts with the user defining a variable \$var1 with the value "hola". Then, they attempt to run a command "var2=hola mundo cruel" which fails because "var2" is not defined. Next, they run "echo \$var2" which prints "hola mundo cruel". Finally, they define another variable \$var3 with the value "hola" and try to run "echo \$var3" which fails again. The terminal then prompts for an interactive shell, offering zsh as an option. The user exits the terminal.

```
ComputacionFisica ~ hrodriguezg@Harveys-MacBook-Pro ~ .utacionFisica ~ -zsh ~ 80x24
$ ComputacionFisica var1=hola
$+> ComputacionFisica echo $var1
hola
$+> ComputacionFisica
$+> ComputacionFisica var2=hola mundo cruel
zsh: command not found: mundo
$+> ComputacionFisica echo $var2
hola mundo cruel
$+> ComputacionFisica
$+> ComputacionFisica var3 = hola
zsh: command not found: var3
$+> ComputacionFisica bash

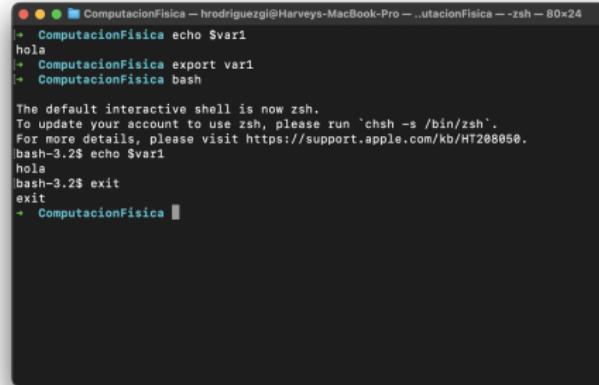
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
bash-3.2$ echo $var1

bash-3.2$ exit
exit
$+> ComputacionFisica echo $var1
hola
$+> ComputacionFisica
```

Variabes Locales

Si queremos convertir una variable en Global, utilizaremos:

```
$ export var1=hola
```



A screenshot of a macOS terminal window titled "ComputacionFisica". The window shows the following command being run:

```
ComputacionFisica echo $var1
hola
ComputacionFisica export var1
ComputacionFisica bash
```

Below this, a message from the system is displayed:

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
```

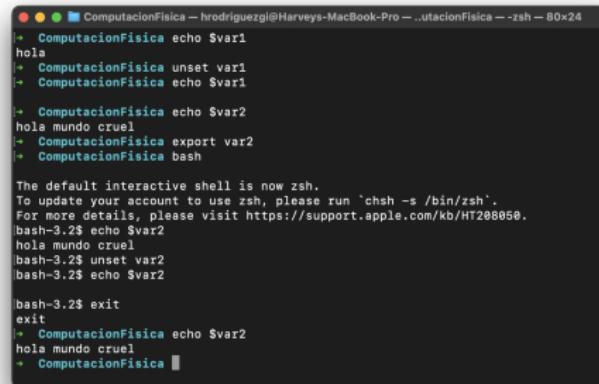
Finally, the user runs:

```
echo $var1
hola
bash-3.2$ exit
exit
ComputacionFisica
```

Variables Locales

Para remover basta con ejecutar:

```
$ unset var1
```



```
ComputacionFisica ~ hrodriguezgi@Harveys-MacBook-Pro ~ .utacionFisica -- zsh - 80x24
+ ComputacionFisica echo $var1
hola
+ ComputacionFisica unset var1
+ ComputacionFisica echo $var1
hola

+ ComputacionFisica echo $var2
hola mundo cruel
+ ComputacionFisica export var2
+ ComputacionFisica bash

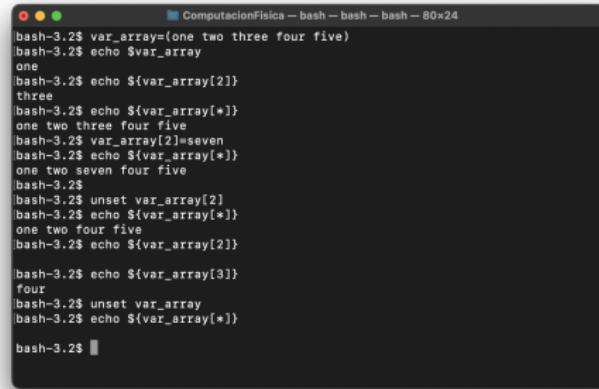
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208056.
bash-3.2$ echo $var2
hola mundo cruel
bash-3.2$ unset var2
bash-3.2$ echo $var2

bash-3.2$ exit
exit
+ ComputacionFisica echo $var2
hola mundo cruel
+ ComputacionFisica
```

Arreglos de Variables

Las variables de entorno también nos permiten almacenar valores que pueden ser utilizados como arreglos, o sea multiples valores que podrán ser referenciados uno a uno o como el arreglo entero.

```
$ var_array=(one two three four five)
```



The screenshot shows a terminal window titled "ComputacionFisica – bash – bash – bash – 80x24". The user has defined an array named "var_array" with elements "one", "two", "three", "four", and "five". They then demonstrate various ways to access and manipulate this array, including printing its contents, referencing individual elements by index, and changing the array's contents. The terminal window has a dark background and light-colored text.

```
ComputacionFisica – bash – bash – bash – 80x24
bash-3.2$ var_array=(one two three four five)
bash-3.2$ echo $var_array
one
two
three
four
five
bash-3.2$ echo ${var_array[2]}
three
bash-3.2$ echo ${var_array[*]}
one two three four five
bash-3.2$ var_array[2]=seven
bash-3.2$ echo ${var_array[*]}
one two seven four five
bash-3.2$ unset var_array[2]
bash-3.2$ echo ${var_array[*]}
one two four five
bash-3.2$ echo ${var_array[2]}
bash-3.2$ echo ${var_array[3]}
four
bash-3.2$ unset var_array
bash-3.2$ echo ${var_array[*]}

bash-3.2$
```

Vamos a practicar

Descargar el archivo `mi_script.sh` del repositorio de GitHub y seguir las instrucciones

