



Linux I

Electiva Profesional

Parte 8: Comandos

<code>*variables</code>	<code>export</code>	<code>unset</code>
<code>if-then-else-fi</code>	<code>seq</code>	

Julían Esteban Gutiérrez Posada

Universidad del Quindío
Facultad de Ingeniería
Ingeniería de Sistemas y Computación

Abril 2020

Un *scripts* de *Shell*

- Es un archivo de texto.
- Contiene un conjunto de comando (propios y de Linux)
- Puede tener variables,
- y en muchos casos, puede tener instrucciones de control
 - Decisiones
 - Ciclos

Un *scripts* de *Shell*

- Es interpretado instrucción a instrucción por el sistema.
- Se suele usar (.sh) como extensión, aunque no es necesario.

Ej: miScript.sh

- Requiere de permisos de ejecución.

```
$ chmod u+x miScript.sh
```

Un *scripts* de *Shell*

- Requiere de un *Shell* o intérprete de comandos, el shell por defecto en Linux se llama bash y todo esta presentación está hecha con él.
- Existen otro shell (algunos no están preinstalados)
sh, tcsh, csh, ash, zsh, ksh, dash, rbash, ...

Usos de *scripts*

- Es empleado principalmente para **AUTOMATIZAR** tareas.
- Linux emplea un gran número de scripts para administrar el sistema, incluyendo la misma administración de los demonios / servicios.

Estructura de un *Script*

plantilla.sh

#!/bin/bash

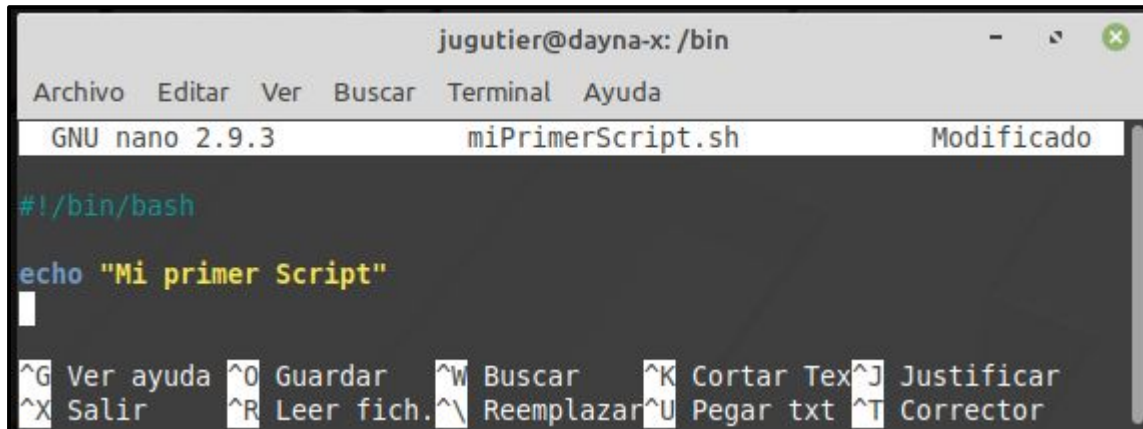
lista de instrucciones

Se le llama **shebang** (*hash-bang* o *sharpbang*) se emplea para indicar el shell que el sistema debe emplear para ejecutar el script.

Para nosotros será siempre
#!/bin/bash

Mi primer Script

```
$ nano miPrimerScript.sh
```



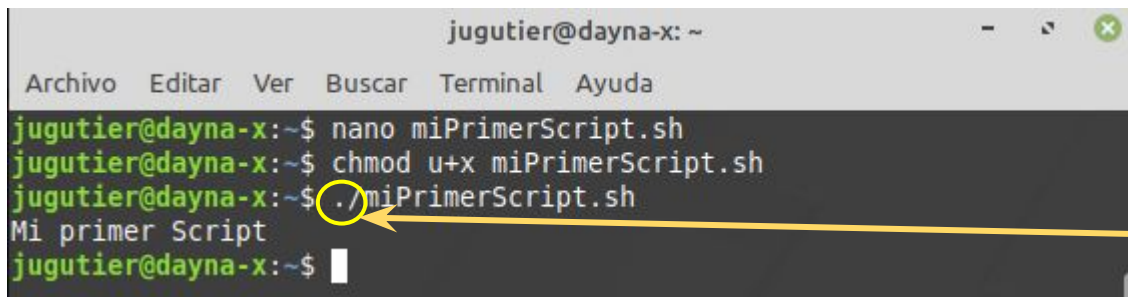
The screenshot shows the nano text editor interface. The title bar indicates the user is 'jugutier@dayna-x' in the directory '/bin'. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The status bar shows 'GNU nano 2.9.3', the filename 'miPrimerScript.sh', and the word 'Modificado'. The editor content shows the shebang line '#!/bin/bash' and the command 'echo "Mi primer Script"'. A help footer is visible at the bottom with various keyboard shortcuts like '^G' for help, '^O' for save, etc.

```
jugutier@dayna-x: /bin
Archivo  Editar  Ver   Buscar  Terminal  Ayuda
GNU nano 2.9.3      miPrimerScript.sh      Modificado

#!/bin/bash

echo "Mi primer Script"

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Tex ^J Justificar
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar txt ^T Corrector
```



The screenshot shows a terminal window with the same user and host. It displays the sequence of commands to create and run the script: 'nano miPrimerScript.sh', 'chmod u+x miPrimerScript.sh', and './miPrimerScript.sh'. The output of the script, 'Mi primer Script', is shown. A yellow circle highlights the './' in the command, with a yellow arrow pointing to it from the right. The prompt returns to '\$' after execution.

```
jugutier@dayna-x: ~
Archivo  Editar  Ver   Buscar  Terminal  Ayuda
jugutier@dayna-x:~$ nano miPrimerScript.sh
jugutier@dayna-x:~$ chmod u+x miPrimerScript.sh
jugutier@dayna-x:~$ ./miPrimerScript.sh
Mi primer Script
jugutier@dayna-x:~$
```

Indispensable para indicar que el script está en el directorio actual.

Variables del sistema

Por defecto, bash ya tiene un conjunto amplio de variables que se pueden consultar y modificar.

Ejecute:

```
$ echo $<tab><tab>
```

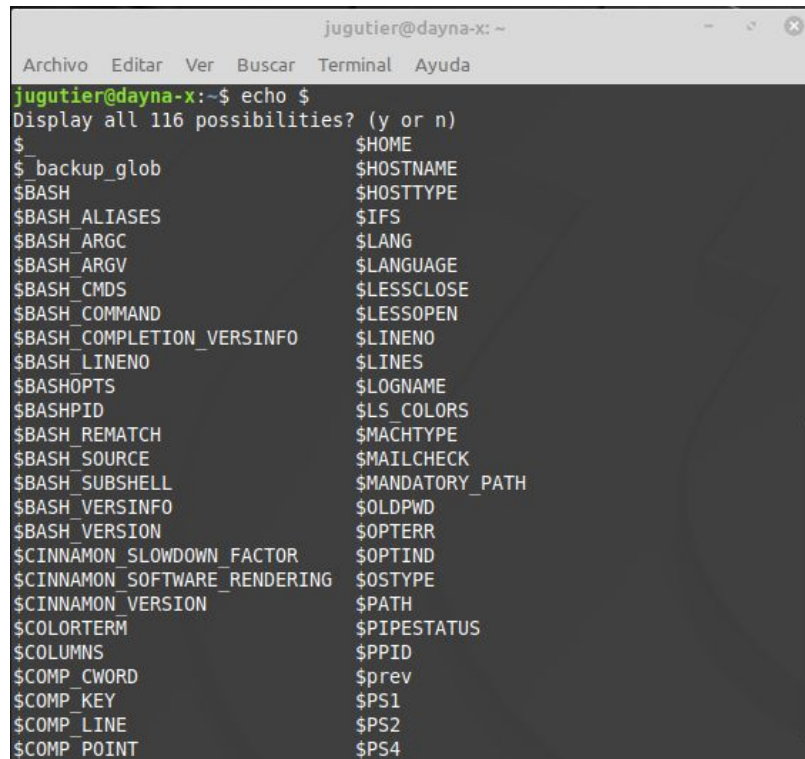
Para esta versión hay 116 variables:

```
$ echo $SHELL  
/bin/bash
```

```
$ echo $LANG  
es_CO.UTF-8
```

```
$ echo $PATH
```

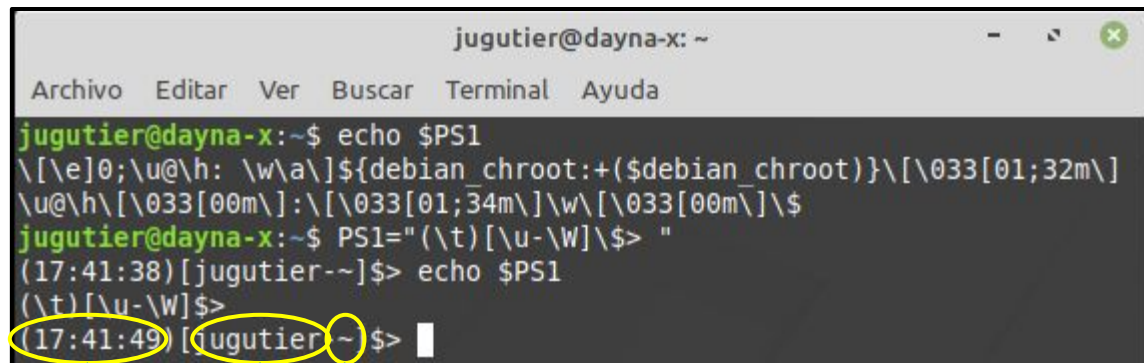
experimente con otras variables



```
jugutier@dayna-x: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
jugutier@dayna-x:~$ echo $  
Display all 116 possibilities? (y or n)  
$_ $HOME  
$_backup_glob $HOSTNAME  
$BASH $HOSTTYPE  
$BASH_ALIASES $IFS  
$BASH_ARGC $LANG  
$BASH_ARGV $LANGUAGE  
$BASH_CMDS $LESSCLOSE  
$BASH_COMMAND $LESSOPEN  
$BASH_COMPLETION_VERSION $LINENO  
$BASH_LINENO $LINES  
$BASHOPTS $LOGNAME  
$BASHPID $LS_COLORS  
$BASH_REMATCH $MACHTYPE  
$BASH_SOURCE $MAILCHECK  
$BASH_SUBSHELL $MANDATORY_PATH  
$BASH_VERSION $OLDPWD  
$BASH_VERSION $OPTERR  
$CINNAMON_SLOWDOWN_FACTOR $OPTIND  
$CINNAMON_SOFTWARE_RENDERING $OSTYPE  
$CINNAMON_VERSION $PATH  
$COLORTERM $PIPESTATUS  
$COLUMNS $PPID  
$COMP_CWORD $prev  
$COMP_KEY $PS1  
$COMP_LINE $PS2  
$COMP_POINT $PS4
```


Variables del sistema

Una de las variables es PS1, la cual da forma al prompt del shell.



A terminal window titled 'jugutier@dayna-x: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The terminal shows the following commands and output:

```
jugutier@dayna-x:~$ echo $PS1
\[ \e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\[ \033[01;32m\
\u@\h\[ \033[00m\]:\[ \033[01;34m\]\w\[ \033[00m\]\$
jugutier@dayna-x:~$ PS1="(\\t) [\\u-\\W]\\$> "
(17:41:38)[jugutier-~]$ echo $PS1
(\\t)[\\u-\\W]$>
(17:41:49) [jugutier-~]$> |
```

Yellow circles highlight the time '17:41:49' and the prompt '[jugutier-~]\$>' in the final line.

Para modificar la variable use el operador igual (=), así:

PS1="(\\t) [\\u-\\W]\\\$> "

Consultar para más información:

https://www.linuxtotal.com.mx/index.php?cont=info_tips_017

Variables propias

```
jugutier@dayna-x: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
jugutier@dayna-x:~$ echo $NOMBRE  
  
jugutier@dayna-x:~$ NOMBRE="Julian E. Gutiérrez P."  
jugutier@dayna-x:~$ echo $NOMBRE  
Julian E. Gutiérrez P.  
jugutier@dayna-x:~$ sh  
$ echo $NOMBRE  
  
$ exit  
jugutier@dayna-x:~$ export NOMBRE  
jugutier@dayna-x:~$ sh  
$ echo $NOMBRE  
Julian E. Gutiérrez P.  
$ exit  
jugutier@dayna-x:~$ unset NOMBRE  
jugutier@dayna-x:~$ echo $NOMBRE  
  
jugutier@dayna-x:~$
```

Es indispensable exportar todas las variables que se requieran acceder desde subprocessos.

```
$ NOMBRE="Julian E."
```

```
$ export NOMBRE
```

o también puede ser:

```
$ export NOMBRE="Julian E."
```

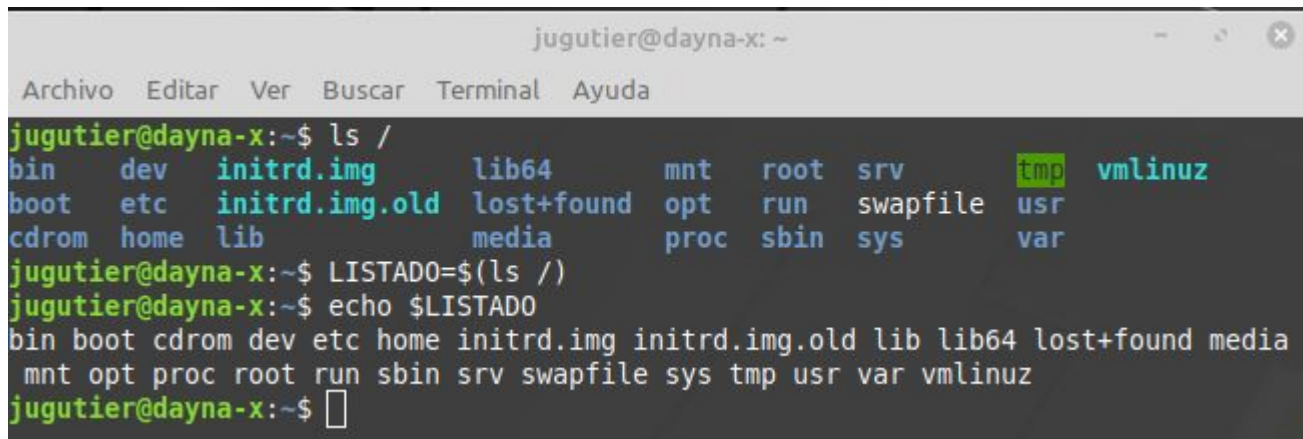
Puede eliminar una variable con: **unset**

¿Qué diferencia hay en usar comillas a apostrofes?

```
$ A="El valor de $SHELL"
```

```
$ B='El valor de $SHELL'
```

Variables propias

A terminal window titled 'jugutier@dayna-x: ~' with a menu bar containing 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows the following commands and output:

```
jugutier@dayna-x:~$ ls /  
bin    dev    initrd.img    lib64    mnt    root    srv    tmp    vmlinuz  
boot   etc    initrd.img.old  lost+found  opt    run    swapfile  usr  
cdrom  home  lib            media    proc   sbin    sys      var  
jugutier@dayna-x:~$ LISTADO=$(ls /)  
jugutier@dayna-x:~$ echo $LISTADO  
bin boot cdrom dev etc home initrd.img initrd.img.old lib lib64 lost+found media  
mnt opt proc root run sbin srv swapfile sys tmp usr var vmlinuz  
jugutier@dayna-x:~$
```

Es posible almacenar en una variable la salida de un comando usando **\$ (Comando)**

```
$ FECHA=$(date)
```

```
$ echo $FECHA
```

...

Script con parámetros

```
$ nano miSegundoScript.sh
```

```
#!/bin/bash
echo "Nombre      : " $0
echo "Argumento 1: " $1
echo "Argumento 2: " $2
echo "No. Argumen: " $#
echo "Argumentos  : " $*
echo "PID:         " $$
```

```
$ ./miSegundoScript.sh
Nombre      : ./miSegundoScript.sh
Argumento 1:
Argumento 2:
No. Argumen: 0
Argumentos  :
PID         : 1233
```

```
$ ./miSegundoScript.sh Hola Mundo
Nombre      : ./miSegundoScript.sh
Argumento 1: Hola
Argumento 2: Mundo
No. Argumen: 2
Argumentos  : Hola Mundo
PID         : 1234
```

```
$ ./miSegundoScript.sh Hola Col Mundo
Nombre      : ./miSegundoScript.sh
Argumento 1: Hola
Argumento 2: Col
No. Argumen: 3
Argumentos  : Hola Col Mundo
PID         : 1235
```

Operaciones aritméticas

```
$ nano sumar.sh
```

```
#!/bin/bash
```

```
echo $1 "+" $2 "=" $(($1+$2))
```

```
echo $(seq 1 10)
```

```
1 2 3 4 5 6 7 8 9 10
```

```
jugutier@dayna-x: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
jugutier@dayna-x:~$ clear  
jugutier@dayna-x:~$ ./sumar.sh  
./sumar.sh: línea 3: + : error sintáctico: se esperaba un operando (el elemento de error es "+ ")  
jugutier@dayna-x:~$ ./sumar.sh 3  
./sumar.sh: línea 3: 3 + : error sintáctico: se esperaba un operando (el elemento de error es "+ ")  
jugutier@dayna-x:~$ ./sumar.sh 3 6  
3 + 6 = 9  
jugutier@dayna-x:~$
```

Comparación de números (cantidad de parámetros)

\$ nano sumar.sh

```
#!/bin/bash

if [ $# -ne 2 ]
then
    echo "Recuerde que: " $0 "n1 n2"
else
    echo $1 "+" $2 "=" $(( $1+$2 ))
fi
```

-eq	==
-ne	!=
-lt	<
-le	<=
-gt	>
-ge	>=

Archivo Editar Ver Buscar Terminal Ayuda

```
jugutier@dayna-x:~$ ./sumar2.sh
Recuerde que: ./sumar2.sh n1 n2
jugutier@dayna-x:~$ ./sumar2.sh 2
Recuerde que: ./sumar2.sh n1 n2
jugutier@dayna-x:~$ ./sumar2.sh 2 6
2 + 6 = 8
jugutier@dayna-x:~$
```

Comparar cadenas

```
$ nano comparar.sh
```

```
#!/bin/bash

if [ -z $1 ]
then
    echo $0 "clave"
else
    if [ $1 == "Uniquindio" ]
    then
        echo "Acceso otorgado"
    else
        echo "Acceso denegado"
    fi
fi
```

==

!=

-n

Verdadero si la cadena no está vacía

-z

Verdadero si la cadena está vacía

```
jugutier@dayna-x:~$ ./comparar.sh
./comparar.sh clave
jugutier@dayna-x:~$ ./comparar.sh 123
Acceso denegado
jugutier@dayna-x:~$ ./comparar.sh Uniquindio
Acceso otorgado
jugutier@dayna-x:~$
```

Comparaciones con archivos

```
$ nano archivo.sh
```

```
#!/bin/bash

if [ -e $1 ]
then
    echo "El archivo SÍ existe"
else
    echo "El archivo NO existe"
fi
```

-e	Si existe el archivo
-s	Si existe y no está vacío
-f	Si existe y es un archivo
-d	Si existe y es un directorio
-w	Si existe y se puede escribir
-r	Si existe y se puede leer
-x	Si existe y se puede ejecutar

Operadores lógicos

!	Negación
-a	Operador Y (AND)
-o	Operador O (OR)