

Iniciamos 16:10

Registra tu asistencia en
tecodi.ng/taller245



bash

Taller de **bash scripting**
nivel intermedio





Han Rodríguez

VP TECoding
8° ITC

- AI Research @ Tec
- 2x Fellow @ MLH
- SWE Intern @ Intel
- SWE Intern @ MSFT

Requisitos

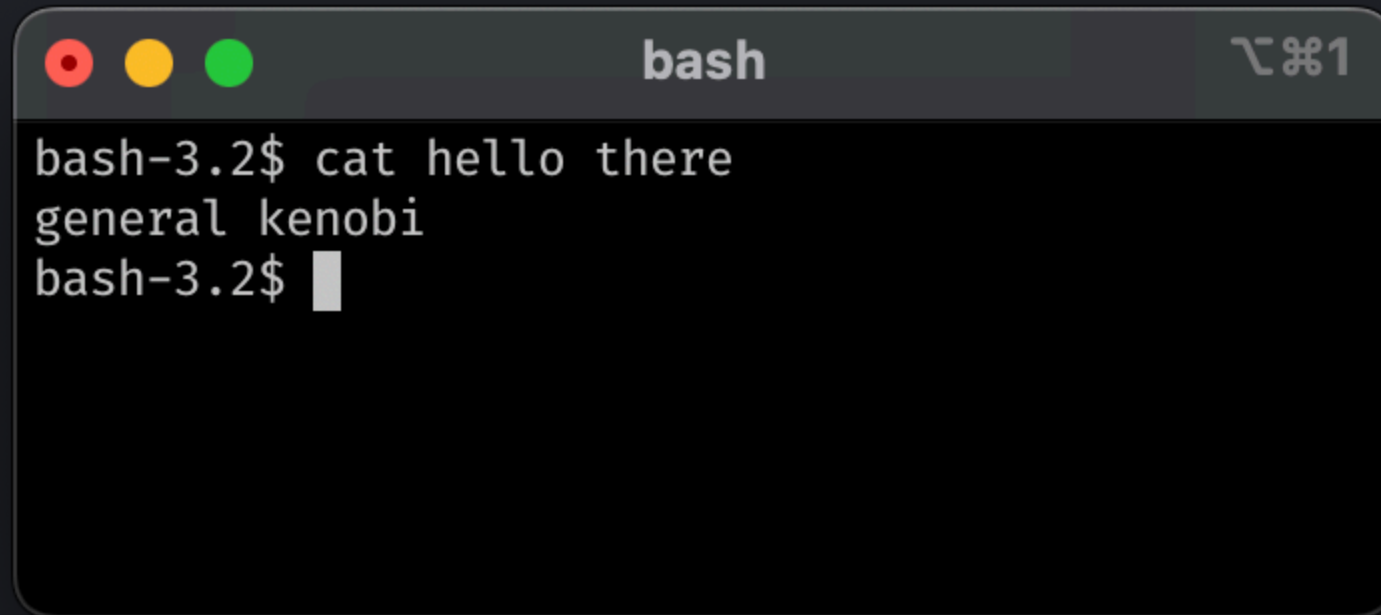
Presentaciones:

tecodi.ng/bash101 - tecodi.ng/bash102

Recursos:

- Tener instalado WSL (Windows solamente) (pág. 5)
- Conocimientos básicos de programación
- Conocimientos de comandos básicos de bash (pág. 7)

Interfaz de línea de comandos

A terminal window with a dark background and a light gray title bar. The title bar contains three colored window control buttons (red, yellow, green) on the left, the text 'bash' in the center, and a window icon followed by '1' on the right. The terminal content shows a command prompt 'bash-3.2\$' followed by the command 'cat hello there'. The output 'general kenobi' is displayed on the next line. A second command prompt 'bash-3.2\$' is shown on the third line with a white cursor block.

```
bash-3.2$ cat hello there
general kenobi
bash-3.2$
```

Repaso de comandos

<code>pwd</code>	# parent working dir.	- dirección de carpeta actual
<code>ls</code>	# list	- archivos en carpeta
<code>cd</code>	# change directory	- cambiar de carpeta
<code>mkdir</code>	# make directory	- crear carpeta
<code>rmdir</code>	# remove directory	- borrar carpeta vacía
<code>touch</code>	# create	- crear archivo vacío
<code>rm</code>	# remove	- borrar archivo o carpeta (-r)
<code>cp</code>	# copy	- copiar archivo o carpeta (-r)
<code>mv</code>	# move	- mover archivo o carpeta (-r)
<code>echo</code>	# print	- imprimir en pantalla
<code>chmod</code>	# change mode	- cambiar permisos de archivo/script
<code>cat</code>	# concatenate	- mostrar contenido de archivo
<code>grep</code>	# global regex print	- buscar patrón de texto
<code>></code>	# output redirection	- pasar salida de comando a archivo
<code> </code>	# pipe	- pasar salida de comando a otro comando
<code>man [cmd]</code>	# q: salir	
<code>[cmd] -h</code>	<code>[cmd] --help</code>	

Repaso de sintaxis

- Declarar y asignar una variable

```
una_var=99  
mi_var1=abc  
var_123="abc"
```

- Acceder a una variable

```
${una_var}  
$mi_var1 # no recomendado
```

Editores de texto

nano

- Salir: `Ctrl + X`

vim

- Entrar al modo edición: `i`
- Salir del modo edición: `Esc`
- Guardar y salir: `:x`

Estructuras de control

Condicionales

```
if [[ `condicion` ]] then
    echo "hello"
elif [[ `condicion` ]] then
    echo "world"
else
    echo "error"
fi
```

- Comparación:

- `-eq` igual
- `-ne` no igual
- `-gt, -lt` mayor que, menor que
- `-ge, -le` mayor o igual, menor o igual

- Archivos:

- `-e` existe
- `-f` es archivo
- `-d` es directorio
- `-r, -w, -x` permite de leer, escribir, ejecutar.

Ciclos

```
for var in a b c d e  
do  
    echo ${var}  
done
```

```
for var in "a b c d e"  
do  
    echo ${var}  
done
```

Comandos avanzados

Redireccionamiento de salida

```
echo "hola" > "archivo.txt"  
echo "mundo" >> "archivo.txt"  
cat "archivo.txt" | grep "la"
```

Sustitución de comandos

```
for line in $(cat "archivo.txt")
do
    echo "linea: ${line}"
done
```

#

Sustitución de comandos

```
for line in $(cat "archivo.txt")
do
    echo "linea: ${line}"
done

cat "archivo.txt" | xargs -I{} echo "linea: {}" #
```


Comandos compuestos

```
cat "archivo.txt" && echo "fin del documento" || echo "archivo inexistente"
```