

corrigeme este codigo por favor: n3o\_usr@endpoint-debianusr:~\$ sha256sum Documents/courses/ISC2-CC/\$(ls)

Claro 😊

El problema no es `sha256sum` en sí, sino el uso de `$(ls)`.

### Qué está mal en tu comando

bash

 Copy code

```
sha256sum Documents/courses/ISC2-CC/$(ls)
```

- `$(ls)` lista todo el directorio actual, no el contenido de `ISC2-CC`.
- Si hay espacios en nombres de archivo, el comando se rompe.
- Puede intentar calcular hashes de directorios, lo cual falla.
- Incluir el prompt (`n3o_usr@endpoint-debianusr:~$`) no forma parte del comando.

### Forma correcta (simple)

Si quieres el hash de todos los archivos dentro de `ISC2-CC`:

bash

 Copy code

```
sha256sum Documents/courses/ISC2-CC/*
```

⚠ Esto falla si hay subdirectorios.

### Forma correcta y robusta (recomendada)

Incluye subdirectorios y maneja espacios correctamente:

bash

 Copy code

```
find Documents/courses/ISC2-CC -type f -print0 | xargs -0 sha256sum
```

### Si solo quieres un archivo específico

bash

 Copy code

```
sha256sum Documents/courses/ISC2-CC/archivo.pdf
```

### Resumen

✗ `$(ls)` → mala práctica

✓ Usa \* o find

Si quieres, dime:

- si hay subdirectorios

We use cookies

- si quieres guardar el resultado en un archivo

X

We use cookies to help this site function, understand service usage, and support marketing efforts. Visit [Manage Cookies](#) to change preferences anytime. View our [Cookie Policy](#) for more info.

 Manage Cookies

 Reject non-essential

 Accept all



Lo use asi: n3o\_usr@endpoint-debianusr:~\$ sha256sum  
Documents/courses/ISC2-CC/\*.png pero que si quiero que  
me los de por SHA265 y luego su nombre osea con un  
header que lo haga mas human readable

Perfecto 👍

Solo una pequeña corrección primero: **es SHA-256**, no SHA-265 (ese algoritmo no existe).

Ahora vamos a hacerlo **más legible para humanos**, con:

- un **header**
- formato claro: Archivo | SHA256
- alineado

### Opción 1: Header + formato bonito (recomendado)

bash

Copy code

```
echo -e "ARCHIVO\t\t\tSHA256\n-----\nsha256sum Documents/courses/ISC2-CC/*.png | awk '{print $2 "\t" $1}'
```

**Salida ejemplo:**

markdown

Copy code

ARCHIVO	SHA256
-----	
image1.png	a3f1c9b7e4c0a9f5...
diagram.png	9b21d4c0e77a1c2a...

### Opción 2: Más limpio aún (alineado automáticamente)

bash

Copy code

```
printf "%-40s %s\n" "ARCHIVO" "SHA256"
printf "%-40s %s\n" "-----" ""
sha256sum Documents/courses/ISC2-CC/*.png | awk '{printf "%-40s %s\n", $2,
```

Esto queda **muy profesional**, ideal para reportes o evidencias.

### Opción 3: Guardarlo en un archivo (por ejemplo para el curso ISC2)

bash

Copy code

```
{
    printf "%-40s %s\n" "ARCHIVO" "SHA256"
    printf "%-40s %s\n" "-----" ""
    sha256sum Documents/courses/ISC2-CC/*.png | awk '{printf "%-40s %s\n", $2,
} > hashes_sha256.txt
```

### Tip de buenas prácticas (importante)

Nunca uses \$(ls) para esto ✗

Tu forma actual con \*.png es **correcta y segura** ✓

Si quieres:

- CSV
- JSON
- verificar hashes luego ( sha256sum -c )
- o hacerlo recursivo

dime y lo ajustamos 🔒📘



```
Exolicame cada parametro de esto: find Documents/
courses/ISC2-CC -type f -print0 | xargs -0 sha256sum y de
esto echo -e
"ARCHIVO\t\tSHA256\n-----"
-----
sha256sum Documents/courses/ISC2-CC/*.png | awk
'{print $2 "\t" $1}'
```

Perfecto, vamos **línea por línea y parámetro por parámetro**, con calma y en español claro



(ideal para entenderlo bien y también para exámenes tipo **ISC2**).

## 1 Comando:

bash

Copy code

```
find Documents/courses/ISC2-CC -type f -print0 | xargs -0 sha256sum
```

Este comando **calcula el SHA-256 de TODOS los archivos**, incluso en subdirectorios, manejando correctamente espacios y caracteres raros.



Busca archivos en un directorio.

bash

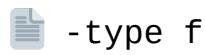
Copy code

```
find Documents/courses/ISC2-CC
```

→ Empieza a buscar **desde ese directorio hacia abajo (recursivo)**.



Ruta inicial donde `find` comienza la búsqueda.

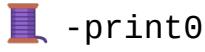


-type f

Filtra **solo archivos normales**.

- f → file (archivo)
- Evita:
  - directorios
  - enlaces simbólicos
  - dispositivos

✓ Necesario porque sha256sum **no trabaja con directorios**.



-print0

Imprime los resultados separados por un **byte NULL ( \0 )** en vez de saltos de línea.

### ¿Por qué es importante?

Porque los nombres de archivo pueden contener:

- espacios
- tabulaciones
- saltos de línea

Ejemplo problemático:

cpp

Copy code

imagen final.png

Con -print0, esto **no rompe el comando**.



|

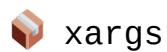
El **pipe**.

Conecta la salida del comando de la izquierda con la entrada del de la derecha.

text

Copy code

find → xargs → sha256sum



xargs

Convierte la entrada estándar en **argumentos para otro comando**.

En este caso:

bash

Copy code

xargs sha256sum

→ ejecuta:

bash

Copy code

sha256sum archivo1 archivo2 archivo3 ...



-0

Le dice a xargs :

"Los datos de entrada están separados por NULL ( \0 ), no por espacios"

🔗 Debe usarse **siempre** junto con `find -print0`.

### 🔑 sha256sum

Calcula el hash **SHA-256** de cada archivo.

Salida por defecto:

php-template

 Copy code

<hash> <nombre\_archivo>

### ✓ Resumen visual

bash

 Copy code

```
find      → busca archivos  
-type f   → solo archivos  
-print0   → seguro con espacios  
|         → conecta  
xargs -0   → recibe seguro  
sha256sum → calcula hash
```

## 2 Comandos:

bash

 Copy code

```
echo -e "ARCHIVO\t\t\tSHA256\n-----  
sha256sum Documents/courses/ISC2-CC/*.png | awk '{print $2 "\t" $1}'
```

Esto **formatea la salida para humanos**, con header y columnas claras.

### 🖨 echo

Imprime texto en la terminal.

### ⚙️ -e

Habilita **secuencias de escape**.

Sin `-e`, `\n` y `\t` se imprimirían literalmente.

👉 "ARCHIVO\t\t\tSHA256"

- `\t` → tabulación (alinear columnas)
- Aquí se usan **3 tabs** para separación visual

Resultado:

nginx

 Copy code

ARCHIVO SHA256

↔ \n

Salto de línea.

```
-----  
-----
```

Línea separadora para mejorar legibilidad.

 sha256sum Documents/courses/ISC2-CC/\* .png

Calcula el SHA-256 **solo de archivos** .png en ese directorio (no recursivo).

Salida:

cpp

 Copy code

```
hash1 imagen1.png  
hash2 imagen final.png
```

 |

Pipe: pasa la salida a awk .

 awk

Lenguaje de procesamiento de texto por columnas.

 {print \$2 "\t" \$1}

Reordena columnas:

Campo	Contenido
\$1	hash SHA256
\$2	nombre del archivo

Este print hace:

bash

 Copy code

```
archivo.png      hash
```

En vez de:

bash

 Copy code

```
hash      archivo.png
```

## Resultado final:

markdown

 Copy code

ARCHIVO	SHA256
image1.png	a3f1c9b7e4c0a9f5...
diagram.png	9b21d4c0e77a1c2a...

## Diferencia clave entre ambos enfoques

Método	Recursivo	Espacios	Formato bonito
find + xargs	✓	✓	✗
sha256sum *.png + awk	✗	⚠	✓

Siquieres, puedo:

- unir **ambos mundos** (recursivo + bonito)
- explicarlo para **responder preguntas de examen**
- pasarlo a **CSV o JSON**

Solo dime ⚡

