Estudiar libro de referencia “El shell bash” antes o durante la resolución de estos ejercicios.

No usar IA, sólo el libro de referencia.

Ejercicios de scripting: paso de parámetros

Ejercicio 1 – Eco de argumentos

Crea un script que muestre en pantalla todos los argumentos que recibe, uno por línea.
👉 Usa $1, $2, … y $@.

Ejercicio 2 – Número de parámetros

Haz un script que imprima cuántos parámetros se han pasado.
👉 Usa la variable $#.

Ejercicio 3 – Suma de dos números

Crea un script que reciba **dos números** como parámetros y muestre su suma.

- Si no se pasan exactamente dos, mostrar un mensaje de error.

Ejercicio 4 – Verificación de fichero

Escribe un script que reciba un **nombre de archivo** como parámetro y:

- Indique si existe.
- Indique si es un fichero normal, un directorio o no existe.

Ejercicio 5 – Script multipropósito

Haz un script que reciba como **primer parámetro** una acción (`fecha`, `usuario`, `ruta`) y ejecute:

- `fecha`: mostrar la fecha actual.
- `usuario`: mostrar el nombre del usuario actual.
- `ruta`: mostrar el directorio actual.
- Si no coincide, mostrar un mensaje de ayuda.
  - Usa `case`.

Ejercicio 6 – Concatenación de parámetros

Crea un script que reciba **N parámetros** y los guarde concatenados en un archivo `parametros.txt`.
- Ejemplo: `./script.sh hola mundo bash` → archivo con `hola mundo bash`.

Ejercicio 7 – Ordenar lista de palabras

Haz un script que reciba varias palabras como parámetros y muestre la lista ordenada alfabéticamente.
- Usa `echo "$@" | tr ' ' '\n' | sort`.

Ejercicio 8 – Validar número de argumentos

Escribe un script que espere **exactamente 3 parámetros**.

- Si recibe menos o más, mostrar: *"Error: se necesitan exactamente 3 parámetros"*.
- Si son 3, mostrarlos en mayúsculas.

Ejercicio 9 – Operaciones matemáticas

Crea un script que reciba:

1. Primer número
2. Operador (+, -, x, /)
3. Segundo número

Ejemplo: `./calc.sh 8 x 3` → salida `24`.
- Usa un `case` para decidir la operación.

Ejercicio 10 – Copia con validación

Haz un script que reciba **dos parámetros**: origen y destino.

- Si el origen existe, copiarlo al destino.
- Si no existe, mostrar error.
- Si el destino ya existe, preguntar antes de sobrescribir.

Ejercicio 11 – Informe multi-OS con `uname`

- Obtén `kernel=$(uname -r)`, `so=$(uname -s)`, `arquitectura=$(uname -m)`.
- Crea `sistema.txt` con esos datos y la fecha (`date`).
- **Si** `so` es `Linux` y `arquitectura` es `x86_64`, añade línea: `Plataforma soportada`.
- **Si no**, añade: `Plataforma no soportada` y termina con `exit 5`.

Ejercicio 12 – Rotación de logs por tamaño

- Dado un fichero `app.log`, **si** supera 10 MB, comprímelo a `app-YYYYMMDD.tar.gz` y vacía `app.log`.
- Si el tar falla, escribe error en `stderr` y `exit 6`.

- Registra la acción en `rotate.log` con fecha.

Ejercicio 13 – Vigilante de puertos

- Recibe puerto como parámetro (`$1`). Comprueba si está en uso (`ss -ltnp | grep ":$1 "` o `netstat -ltnp`).
- **Si** está ocupado, guarda el PID y nombre del proceso en `puerto_$1.txt`.
- **Si no**, imprime `Libre` y crea `escucha_$1.log` con un `nc -l $1` (en segundo plano) durante 10 s.
- Maneja errores a `stderr` y códigos de salida.

Ejercicio 14 – Normalizador de texto con `sed` y `tr`

- Dado `entrada.txt`, genera `salida.txt` con:
  - Todo en minúsculas.
  - Sin tildes (á→a, é→e, …).
  - Una palabra por línea, solo `a-z0-9`.
- Usa `tr`, `sed` y `grep -E`. Quita líneas vacías.
- Muestra top 10 palabras con `sort | uniq -c | sort -nr | head`.

Ejercicio 15 – Scanner de configuración insegura

- Recorre `/etc` buscando archivos `*.conf`.
- Filtra líneas que contengan `permitRootLogin yes`, `PasswordAuthentication yes` o `AllowEmptyPasswords yes` (ignorando may/min).

- Genera `hallazgos.csv` con columnas: ruta, coincidencia, línea.
- **Si** no hay coincidencias, imprime `Sin hallazgos`. Si las hay, `exit 7`.

Ejercicio 16 – Comparador de inventarios con `comm`

- Recibe dos ficheros (listas ordenadas) `A` y `B`.
- Usa `comm` para generar:
  - `solo_en_A.txt`
  - `solo_en_B.txt`
  - `en_ambos.txt`
- **Si** alguno no está ordenado, ordénalo primero a temporales.
- Imprime resumen con conteos por cada categoría.

Ejercicio 17 – Monitor de espacio en disco por punto de montaje

- Crea un bucle que recorra salidas de `df -h` (excluye `tmpfs` y `overlay`).
- Para cada punto de montaje, si uso ≥ 85%, añade línea `ALERTA <mount> <uso>` a `disco_alertas.txt`.
- Si hubo alertas, devuelve `exit 8`; si no, imprime `OK`.

Ejercicio 18 – Verificador de integridad con `sha256sum`

- Dado un directorio, calcula `sha256sum` de todos los ficheros y guarda `manifest.sha256`.
- En una segunda ejecución, verifica cada hash:
    - Diferencias → `integridad_ko.txt`.
    - Coincidencias → `integridad_ok.txt`.
- Si hay al menos una discrepancia, escribe aviso en `stderr` y `exit 9`.

Ejercicio 19 – Paralelización con `xargs -P`

- Dado `imagenes.txt` con URLs, descárgalas en `imgs/` en paralelo con `xargs -P N` (elige N≈ núm. de cores).
- Registra OK/KO en `descargas.log` (usa códigos HTTP con `curl -s -o ... -w '%{http_code}\n'`).
- Al final, muestra un **resumen** (totales OK/KO) y crea `fallidas.txt` con las URLs fallidas.

Ejercicio 20 – Auditor de permisos peligrosos

- Busca en la ruta dada ficheros con permisos `777` o con SUID/SGID activos.
- Genera `permisos_peligrosos.csv` con columnas: tipo (777/SUID/SGID), ruta, propietario, grupo.
- **Si** encuentra algo bajo `/usr/bin` o `/bin` que no sea del propietario `root`, añade `CRÍTICO` al registro.
- Devuelve `exit 10` si hubo hallazgos críticos.

Ejercicio 21 – Agenda de tareas sin `cron`

- Implementa una **agenda simple** que lea `tareas.csv` con formato: `HH:MM,comando`.
- Cada minuto, comprueba si hay tareas para esa hora/minuto y las ejecuta en segundo plano, guardando `stdout`/`stderr` separados por tarea en `logs/`.
- Añade `trap` para terminar limpiamente con `SIGINT`/`SIGTERM`.

Ejercicio 22 – Extractor de métricas de `journalctl` (si systemd)

- Si el SO es Linux (`uname -s == Linux`) y existe `journalctl`, filtra los últimos 5000 registros de `sshd`.
- Cuenta intentos fallidos y exitosos; saca por hora un histograma con `awk`.
- Guarda informe en `sshd_metrics.txt`; si no hay `journalctl`, imprime `No disponible` y sale con 0.

Ejercicio 23 – Linter básico de scripts

- Recorre `scripts/*.sh`. Para cada uno:
  - Comprueba que empieza con `#!/usr/bin/env bash`.
  - Busca `set -euo pipefail`.
  - Asegura que no hay tabs (solo espacios) y que no usa `cat file | grep` (UUOC).
- Crea `linter_reporte.txt` con OK/KO por archivo y reglas incumplidas.
- Si hay incumplimientos, `exit 11`.

Ejercicio 24 – Sincronizador unidireccional “dry-run”

- Recibe `origen` y `destino`.
- Muestra qué ficheros **se copiarían** y **se eliminarían** en destino para sincronizar (sin ejecutar cambios reales).
- Si el usuario confirma (`read -p`), ejecuta la sincronización real.
- Redirige operaciones a `sync_YYYYMMDD.log` y errores a `sync.err`.

Ejercicio 25 – Selector de interfaz de red

- Lista interfaces activas (`ip -o link show | awk '/state UP/ {print $2}' | tr -d ':'`).
- Pide al usuario elegir una.
- Muestra IP, MAC y RX/TX (`ip -s addr show IFACE`).
- **Si** no hay interfaces UP, escribe `No hay interfaces activas` en `stderr` y `exit 12`.