

Ejercicios resueltos de Sistemas Informáticos

Los 49 ejercicios de Sistemas Informáticos de la web, tema a tema: cada uno con su enunciado, los datos que necesitas (código de partida, ejemplos de entrada y salida o la base de datos) y, al final, la solución explicada.

Índice

Componentes Físicos

- [1. ¿Por qué mi disco de 1 TB tiene 931 GB?](#)
- [3. Ancho de banda y cuellos de botella](#)

Sistemas Operativos

- [4. Planificador FCFS y SJF](#)
- [6. Reemplazo de páginas: FIFO, LRU y óptimo](#)

Instalación de Sistemas Operativos

- [7. Un plan de particiones alineado](#)
- [9. Leer la tabla de particiones de un MBR](#)

Sistemas de Ficheros y Copias de Seguridad

- [10. Capacidad y tolerancia de un RAID](#)
- [12. ¿Es buena esta copia de seguridad?](#)

Administración de Sistemas Operativos

- [13. Saludo con validación](#)
- [15. Encuentra el fallo: el saludo que no se guarda](#)
- [17. Usuarios del sistema desde /etc/passwd](#)
- [19. Encuentra el fallo: la alerta que salta siempre](#)
- [21. Encuentra el fallo: el nombre con espacio](#)
- [23. ¿Quién intenta entrar por SSH?](#)
- [25. Encuentra el fallo: las 08:09 que rompen la cuenta](#)

Redes Informáticas

- [27. ¿Llega este cable?](#)
- [29. Puntos únicos de fallo de una red](#)

Configuración de Redes TCP/IP

- [30. ¿Pueden hablar sin router?](#)
- [32. Calculadora VLSM](#)

Servicios de Red, Recursos Compartidos y Seguridad

- [33. Permisos efectivos de una carpeta compartida](#)
- [35. ¿Cuánto aguantaría esta contraseña?](#)

Documentación y Aplicaciones Ofimáticas

- [36. Informe del inventario](#)
- [38. Un conversor de Markdown a HTML](#)

Ejercicios de Sistemas Informáticos

- [39. Contar ficheros por extensión](#)
- [41. Red y broadcast con aritmética de bits](#)

Ejercicios largos

- [2. Dimensionar la fuente de alimentación](#)

- [5. Round Robin con quantum](#)

- [8. ¿Arrancará este esquema de particiones?](#)

- [11. Simulador de estrategias de copia](#)

- [14. Tabla de multiplicar con bucle](#)

- [16. Analizar un fichero de log](#)
- [18. Crear ficheros y asignar permisos](#)
- [20. Encuentra el fallo: el mayor que no lo es](#)
- [22. Alerta de espacio en disco](#)

- [24. Encuentra el fallo: el total que se queda en cero](#)
- [26. Permisos: de letras a octal y al revés](#)

- [28. La tabla MAC de un switch](#)

- [31. Abreviar y expandir direcciones IPv6](#)

- [34. Un cortafuegos de primera coincidencia](#)

- [37. Un índice automático para la documentación](#)

- [40. Validar direcciones IPv4](#)

42. Planificador de procesos: FCFS, SJF, SRTF y Round Robin

44. Detección de ataques de fuerza bruta en el registro de SSH

46. Alta masiva de usuarios a partir de un CSV

48. Monitor de espacio en disco con umbrales

43. Plan de direccionamiento con subredes VLSM

45. Informe de un registro de accesos web

47. Rotación de copias de seguridad abuelo-padre-hijo

49. Analizador de la configuración de Nginx

Componentes Físicos

1. ¿Por qué mi disco de 1 TB tiene 931 GB?

Medio · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Los fabricantes miden en unidades decimales (1 GB = 10^9 bytes) y muchos sistemas operativos en binarias (1 GiB = 2^{30} bytes, aunque Windows lo muestre como «GB»). Cada línea de la entrada es una cantidad y una unidad (B, KB, MB, GB, TB, KiB, MiB, GiB o TiB). Muestra «cantidad unidad = N bytes = X GB = Y GiB», con los bytes como número entero redondeado y X e Y con 2 decimales. Si la unidad no existe, «línea: unidad desconocida».

Código de partida (python)

```
import sys

DECIMAL = {"B": 1, "KB": 10**3, "MB": 10**6, "GB": 10**9, "TB": 10**12}
BINARIO = {"KiB": 2**10, "MiB": 2**20, "GiB": 2**30, "TiB": 2**40}

# TODO: convierte cada tamaño a bytes, a GB y a GiB
```

Ejemplo: Discos y memoria

ENTRADA

```
1 TB
512 GB
16 GiB
4.7 GB
```

SALIDA ESPERADA

```
1 TB = 1000000000000 bytes = 1000.00 GB = 931.32 GiB
512 GB = 512000000000 bytes = 512.00 GB = 476.84 GiB
16 GiB = 17179869184 bytes = 17.18 GB = 16.00 GiB
4.7 GB = 4700000000 bytes = 4.70 GB = 4.38 GiB
```

Ejemplo: Unidades pequeñas

ENTRADA

```
1536 KiB
700 MB
1 B
```

SALIDA ESPERADA

```
1536 KiB = 1572864 bytes = 0.00 GB = 0.00 GiB
700 MB = 700000000 bytes = 0.70 GB = 0.65 GiB
1 B = 1 bytes = 0.00 GB = 0.00 GiB
```

2. Dimensionar la fuente de alimentación

Difícil · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Las primeras líneas son los componentes con su consumo máximo, «nombre vatios» (el nombre puede tener espacios). Después vienen cuatro líneas de configuración «clave=valor»: certificación (bronze, gold o platinum, con eficiencias 0.85, 0.90 y 0.92), uso (porcentaje medio de carga respecto al máximo), horas (de uso al día) y precio (€ por kWh). Muestra: «Consumo de los componentes: N W»; «Con un margen del 30 %: N W» (sin decimales); «Fuente recomendada: N W» (la primera del catálogo que llegue a ese valor, o «ninguna del catálogo es suficiente»); «Consumo real de la pared: X W» (consumo × uso / 100 / eficiencia, 1 decimal); y «Coste anual: X € (Y kWh)» con 365 días al año (2 y 1 decimales).

Código de partida (python)

```
import sys

CATALOGO = [450, 550, 650, 750, 850, 1000, 1200]
EFICIENCIA = {"bronze": 0.85, "gold": 0.90, "platinum": 0.92}

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

# TODO: calcula el consumo, elige la fuente y estima el coste anual
```

Ejemplo: Equipo de desarrollo

ENTRADA

Procesador Ryzen 7 105
Gráfica RTX 4060 115
Placa base 50
RAM 2 módulos 10
SSD NVMe 8
Ventiladores 12
certificacion=gold
uso=40
horas=8
precio=0.18

SALIDA ESPERADA

Consumo de los componentes: 300 W
Con un margen del 30 %: 390 W
Fuente recomendada: 450 W
Consumo real de la pared: 133.3 W
Coste anual: 70.08 € (389.3 kWh)

Ejemplo: Estación de trabajo

ENTRADA

CPU 253
GPU RTX 4090 450
Resto 120
certificacion=platinum
uso=70
horas=10
precio=0.2

SALIDA ESPERADA

Consumo de los componentes: 823 W
Con un margen del 30 %: 1070 W
Fuente recomendada: 1200 W
Consumo real de la pared: 626.2 W
Coste anual: 457.12 € (2285.6 kWh)

3. Ancho de banda y cuellos de botella

Muy difícil · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Cada línea describe un componente. «ram DDR5-6000 2» es memoria del tipo indicado (el número son millones de transferencias por segundo, MT/s) en 2 canales: cada canal transfiere 8 bytes, así que el ancho de banda en GB/s es $MT/s \times 8 \times \text{canales} / 1000$. «ssd 7000 pcie4 x4» es un SSD que puede leer a 7000 MB/s conectado a una ranura PCIe de 4.ª generación con 4 líneas; cada línea transporta 0.985, 1.969 o 3.938 GB/s en las generaciones 3, 4 y 5. Para la memoria muestra «Memoria DDR5-6000 en 2 canales: 96.0 GB/s» («canal» en singular si es 1). Para el SSD, «SSD de V MB/s en PCIe G.0 xN (E GB/s): sin cuello de botella» o, si el enlace es más lento que el SSD, «...: limitado a E MB/s (D MB/s desaprovechados)», con el enlace en MB/s sin decimales ($\times 1000$) y en GB/s con 1 decimal.

Código de partida (python)

```
import sys

PCIE_GBS_POR_LINEA = {"3": 0.985, "4": 1.969, "5": 3.938}

# TODO: calcula el ancho de banda y detecta cuellos de botella
```

Ejemplo: Equipo actual

ENTRADA

ram DDR5-6000 2
ssd 7000 pcie4 x4
ssd 3500 pcie3 x4

SALIDA ESPERADA

Memoria DDR5-6000 en 2 canales: 96.0 GB/s
SSD de 7000 MB/s en PCIe 4.0 x4 (7.9 GB/s): sin
cuello de botella
SSD de 3500 MB/s en PCIe 3.0 x4 (3.9 GB/s): sin
cuello de botella

Ejemplo: SSD en ranura lenta

ENTRADA

```
ssd 7400 pcie3 x4
ssd 12000 pcie5 x4
ram DDR4-3200 1
```

SALIDA ESPERADA

```
SSD de 7400 MB/s en PCIe 3.0 x4 (3.9 GB/s):
limitado a 3940 MB/s (3460 MB/s desaprovechados)
SSD de 12000 MB/s en PCIe 5.0 x4 (15.8 GB/s): sin
cuello de botella
Memoria DDR4-3200 en 1 canal: 25.6 GB/s
```

Sistemas Operativos

4. Planificador FCFS y SJF

Medio · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

La primera línea es el algoritmo: FCFS (el primero que llega, se ejecuta primero) o SJF (de los que ya han llegado, el de ráfaga más corta; a igualdad, el que llegó antes y después por nombre). Ambos son no expropiativos: un proceso se ejecuta hasta terminar. Cada línea siguiente es un proceso «nombre llegada ráfaga». Muestra el diagrama de Gantt en una línea, con los tramos «nombre inicio-fin» separados por « | » (si la CPU queda libre esperando a que llegue alguien, un tramo «ocioso inicio-fin»); después, para cada proceso en el orden de la entrada, «nombre: espera E, retorno R» (espera = inicio – llegada; retorno = fin – llegada); y por último la espera media y el retorno medio con 2 decimales.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
algoritmo = lineas[0][0]
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

# TODO: planifica con FCFS o SJF y muestra el diagrama y los tiempos medios
```

Ejemplo: FCFS del ejemplo

ENTRADA

```
FCFS
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-5 | P2 5-8 | P3 8-9 | P4 9-11
P1: espera 0, retorno 5
P2: espera 4, retorno 7
P3: espera 6, retorno 7
P4: espera 6, retorno 8
Espera media: 4.00
Retorno medio: 6.75
```

Ejemplo: SJF del ejemplo

ENTRADA

```
SJF
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-5 | P3 5-6 | P4 6-8 | P2 8-11
P1: espera 0, retorno 5
P2: espera 7, retorno 10
P3: espera 3, retorno 4
P4: espera 3, retorno 5
Espera media: 3.25
Retorno medio: 6.00
```

Ejemplo: CPU ociosa

ENTRADA

```
FCFS
A 2 3
B 10 2
C 11 1
```

SALIDA ESPERADA

```
ocioso 0-2 | A 2-5 | ocioso 5-10 | B 10-12 | C
12-13
A: espera 0, retorno 3
B: espera 0, retorno 2
C: espera 1, retorno 2
Espera media: 0.33
Retorno medio: 2.33
```

5. Round Robin con quantum

Difícil · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

La primera línea es «quantum Q»; cada línea siguiente, un proceso «nombre llegada ráfaga». Simula Round Robin: el proceso al frente de la cola se ejecuta como mucho Q unidades; si no termina, vuelve al final de la cola. Los procesos que llegan durante ese tramo (o justo cuando termina) entran en la cola antes que el proceso expulsado. Si la cola está vacía, la CPU queda ociosa hasta la siguiente llegada. Muestra el diagrama de Gantt («nombre inicio-fin» separados por « | », con tramos «ocioso inicio-fin» si hace falta), para cada proceso en el orden de la entrada «nombre: espera E, retorno R» (retorno = fin – llegada; espera = retorno – ráfaga), la espera y el retorno medios con 2 decimales y «Cambios de contexto: N» (veces que en el diagrama un tramo es de un proceso distinto al anterior).

Código de partida (python)

```
import sys
from collections import deque

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
quantum = int(lineas[0][1])
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

# TODO: simula Round Robin y muestra el diagrama y los tiempos
```

Ejemplo: Ejemplo con quantum 2

ENTRADA

```
quantum 2
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-2 | P2 2-4 | P3 4-5 | P1 5-7 | P4 7-9 | P2
9-10 | P1 10-11
P1: espera 6, retorno 11
P2: espera 6, retorno 9
P3: espera 2, retorno 3
P4: espera 4, retorno 6
Espera media: 4.50
Retorno medio: 7.25
Cambios de contexto: 6
```

Ejemplo: Quantum grande

ENTRADA

```
quantum 10
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-5 | P2 5-8 | P3 8-9 | P4 9-11
P1: espera 0, retorno 5
P2: espera 4, retorno 7
P3: espera 6, retorno 7
P4: espera 6, retorno 8
Espera media: 4.00
Retorno medio: 6.75
Cambios de contexto: 3
```

Ejemplo: CPU ociosa

ENTRADA

```
quantum 3
A 0 4
B 10 5
C 12 2
```

SALIDA ESPERADA

```
A 0-3 | A 3-4 | ocioso 4-10 | B 10-13 | C 13-15 |
B 15-17
A: espera 0, retorno 4
B: espera 2, retorno 7
C: espera 1, retorno 3
Espera media: 1.00
Retorno medio: 4.67
Cambios de contexto: 4
```

6. Reemplazo de páginas: FIFO, LRU y óptimo

Muy difícil · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

Cuando la memoria está llena y se necesita una página que no está cargada (fallo de página), el sistema operativo elige una víctima. La primera línea es «marcos N» (páginas que caben en memoria, inicialmente vacía) y la segunda, la cadena de referencias separadas por espacios. Cuenta los fallos con tres algoritmos: FIFO (sale la que lleva más tiempo cargada), LRU (sale la que hace más tiempo que no se usa) y óptimo (sale la que tardará más en volver a usarse; si varias no vuelven a usarse, la que lleva más tiempo cargada). Muestra «FIFO: N fallos (P %)», «LRU: ...» y «Óptimo: ...», con el porcentaje sobre el total de referencias con 1 decimal. Prueba la cadena 1 2 3 4 1 2 5 1 2 3 4 5 con 3 y con 4 marcos: verás la anomalía de Bélády.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
marcos = int(lineas[0][1])
referencias = [int(x) for x in lineas[1]]

# TODO: cuenta los fallos de página con FIFO, LRU y el algoritmo óptimo
```

Ejemplo: Cadena clásica con 3 marcos

ENTRADA

```
marcos 3
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```

SALIDA ESPERADA

```
FIFO: 15 fallos (75.0 %)
LRU: 12 fallos (60.0 %)
Óptimo: 9 fallos (45.0 %)
```

Ejemplo: Anomalía de Bélády con 3

ENTRADA

```
marcos 3
1 2 3 4 1 2 5 1 2 3 4 5
```

SALIDA ESPERADA

```
FIFO: 9 fallos (75.0 %)
LRU: 10 fallos (83.3 %)
Óptimo: 7 fallos (58.3 %)
```

Ejemplo: Anomalía de Bélády con 4

ENTRADA

```
marcos 4
1 2 3 4 1 2 5 1 2 3 4 5
```

SALIDA ESPERADA

```
FIFO: 10 fallos (83.3 %)
LRU: 8 fallos (66.7 %)
Óptimo: 6 fallos (50.0 %)
```

Instalación de Sistemas Operativos

7. Un plan de particiones alineado

Medio · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

La primera línea es «disco N GiB» (N entero). Cada línea siguiente es una partición «nombre tamaño», con el tamaño en MiB («512MiB»), en GiB («50GiB», puede tener decimales) o «resto» (todo lo que quede). En un disco GPT, el primer MiB queda reservado (tabla y alineación) y también el último (copia de la tabla). Coloca las particiones una detrás de otra empezando en el MiB 1 y muestra «nombre: inicio A MiB, fin B MiB, tamaño T GiB» (T con 2 decimales). Si una partición no cabe, muestra «ERROR: no cabe nombre (faltan N MiB)» y para. Si todas caben, termina con «Sin asignar: X GiB».

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
disco_mib = int(lineas[0][1].replace("GiB", "")) * 1024

# TODO: coloca las particiones alineadas a 1 MiB y muestra dónde empieza y acaba cada una
```

Ejemplo: Ubuntu en 500 GiB

ENTRADA

```
disco 500GiB
efi 1GiB
raiz 50GiB
swap 8GiB
home resto
```

SALIDA ESPERADA

```
efi: inicio 1 MiB, fin 1025 MiB, tamaño 1.00 GiB
raiz: inicio 1025 MiB, fin 52225 MiB, tamaño
50.00 GiB
swap: inicio 52225 MiB, fin 60417 MiB, tamaño
8.00 GiB
home: inicio 60417 MiB, fin 511999 MiB, tamaño
441.00 GiB
Sin asignar: 0.00 GiB
```

Ejemplo: Windows con hueco libre

ENTRADA

```
disco 256GiB
esp 260MiB
msr 16MiB
windows 180GiB
recuperacion 1GiB
```

SALIDA ESPERADA

```
esp: inicio 1 MiB, fin 261 MiB, tamaño 0.25 GiB
msr: inicio 261 MiB, fin 277 MiB, tamaño 0.02 GiB
windows: inicio 277 MiB, fin 184597 MiB, tamaño
180.00 GiB
recuperacion: inicio 184597 MiB, fin 185621 MiB,
tamaño 1.00 GiB
Sin asignar: 74.73 GiB
```

8. ¿Arrancará este esquema de particiones?

Difícil · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

Las tres primeras líneas son «firmware uefi|bios», «tabla gpt|mbr» y «disco N» (GiB). Cada línea siguiente es una partición «nombre clase sistema tamaño punto» (clase: primaria, extendida o logica; sistema: fat32, ntfs, ext4, swap...; tamaño en GiB; punto de montaje o «-»). Comprueba estas reglas en este orden y muestra un mensaje por cada una que falle, o «Esquema válido» si ninguna falla: (1) «UEFI necesita una tabla GPT (o activar el modo CSM)»; con MBR: (2) «MBR solo puede usar 2048 GiB de los N GiB del disco» si el disco es mayor, (3) «MBR admite como máximo 4 particiones primarias o extendidas», (4) «Solo puede haber una partición extendida», (5) «Las particiones lógicas necesitan una partición extendida»; con GPT: (6) «GPT no tiene particiones extendidas ni lógicas»; (7) con UEFI, «Falta la partición EFI (FAT32 montada en /boot/efi)»; (8) «Falta la partición raíz (/)»; (9) «Las particiones ocupan T GiB y el disco tiene N GiB» (la extendida no cuenta, porque contiene a las lógicas). Los números se muestran sin decimales innecesarios.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
config = {l[0]: l[1] for l in lineas[:3]} # firmware, tabla, disco
particiones = lineas[3:] # nombre clase sistema tamaño_GiB punto_de_montaje

# TODO: comprueba las reglas en orden y muestra los problemas o «Esquema válido»
```

Ejemplo: Esquema correcto

ENTRADA

```
firmware uefi
tabla gpt
disco 500
efi primaria fat32 1 /boot/efi
raiz primaria ext4 50 /
home primaria ext4 400 /home
```

SALIDA ESPERADA

Esquema válido

Ejemplo: UEFI con MBR y sin EFI

ENTRADA

```
firmware uefi
tabla mbr
disco 3000
sistema primaria ntfs 200 C:
datos primaria ntfs 2500 D:
```

SALIDA ESPERADA

UEFI necesita una tabla GPT (o activar el modo CSM)
MBR solo puede usar 2048 GiB de los 3000 GiB del disco
Falta la partición EFI (FAT32 montada en /boot/efi)
Falta la partición raíz (/)

Ejemplo: MBR clásico con lógicas

ENTRADA

```
firmware bios
tabla mbr
disco 1000
win primaria ntfs 300 C:
recup primaria ntfs 1 -
ext extendida - 699 -
raiz logica ext4 100 /
swap logica swap 8 -
home logica ext4 591 /home
```

SALIDA ESPERADA

Esquema válido

9. Leer la tabla de particiones de un MBR

Muy difícil · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

El sector 0 de un disco MBR termina con la tabla de particiones: 4 entradas de 16 bytes (desde el byte 446) y la firma 55 AA. La entrada recibe esos 66 bytes en hexadecimal. En cada entrada: el byte 0 es 0x80 si la partición es la activa (arrancable); el byte 4, el tipo; los bytes 8-11, el sector inicial (LBA); y los bytes 12-15, el número de sectores, ambos enteros de 4 bytes en little endian (el byte menos significativo primero). Las entradas con tipo 00 están vacías. Para cada partición muestra «Partición N: [activa,]tipo XX (nombre), sector inicial S, T sectores (G GiB)» con sectores de 512 bytes y G con 2 decimales (tipo en hexadecimal con 2 cifras en mayúsculas; nombre «desconocido» si no está en la tabla). Si el tipo es EE, añade la línea «El disco usa GPT: esta entrada solo protege el disco de herramientas antiguas». Al final, «Firma: correcta (55 AA)» o «Firma: incorrecta (XX XX)».

Código de partida (python)

```
import sys

datos = bytes.fromhex("".join(sys.stdin.read().split())) # las 4 entradas (64 bytes) y la firma (2 bytes)
TIPOS = {0x07: "NTFS/exFAT", 0x0B: "FAT32", 0x0C: "FAT32 LBA", 0x83: "Linux", 0x82: "Linux swap",
         0x05: "Extendida", 0x0F: "Extendida LBA", 0xEE: "GPT de protección", 0xEF: "Sistema EFI"}

# TODO: interpreta cada entrada de 16 bytes y la firma del sector
```

Ejemplo: Windows y Linux en arranque dual

ENTRADA

```
80 00 00 00 07 00 00 00 00 08 00 00 00 00 40 06
00 00 00 00 83 00 00 00 00 08 40 06 00 00 80 0C
00 00 00 00 82 00 00 00 00 08 C0 12 00 00 00 01
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
55 AA
```

SALIDA ESPERADA

Partición 1: activa, tipo 07 (NTFS/exFAT), sector inicial 2048, 104857600 sectores (50.00 GiB)
Partición 2: tipo 83 (Linux), sector inicial 104859648, 209715200 sectores (100.00 GiB)
Partición 3: tipo 82 (Linux swap), sector inicial 314574848, 16777216 sectores (8.00 GiB)
Firma: correcta (55 AA)

Ejemplo: Disco GPT

ENTRADA

```
00 00 00 00 EE 00 00 00 01 00 00 00 FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
55 AA
```

SALIDA ESPERADA

Partición 1: tipo EE (GPT de protección), sector inicial 1, 4294967295 sectores (2048.00 GiB)
El disco usa GPT: esta entrada solo protege el disco de herramientas antiguas
Firma: correcta (55 AA)

Sistemas de Ficheros y Copias de Seguridad

10. Capacidad y tolerancia de un RAID

Medio · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

Cada línea es «nivel discos tamaño_GB», con el nivel RAID0, RAID1, RAID5, RAID6 o RAID10. Calcula la capacidad útil (RAID0: suma de todos; RAID1: un disco, porque todos son copias; RAID5: $n - 1$ discos; RAID6: $n - 2$; RAID10: la mitad), la eficiencia (útil / total, en % sin decimales) y los fallos de disco que soporta sin perder datos (0, $n - 1$, 1, 2 y en RAID10 «1 fallo por pareja (hasta $n/2$)»). Muestra «RAID5 con 4×2000 GB: 6000 GB útiles (75 %), tolera 1 fallo» (y «fallos» en plural salvo cuando es exactamente 1). Si faltan discos (mínimos: 2, 2, 3, 4 y 4; RAID10 además un número par), muestra «RAID10 con 5 discos: no válido (necesita al menos 4 discos y un número par)», y para un nivel que no existe, «nivel: nivel desconocido».

Código de partida (python)

```
import sys

MINIMO = {"RAID0": 2, "RAID1": 2, "RAID5": 3, "RAID6": 4, "RAID10": 4}

# TODO: calcula la capacidad útil, la eficiencia y los fallos que soporta cada configuración
```

Ejemplo: Niveles habituales

ENTRADA

```
RAID0 2 1000
RAID1 2 4000
RAID5 4 2000
RAID6 6 8000
RAID10 4 1000
```

SALIDA ESPERADA

```
RAID0 con 2 x 1000 GB: 2000 GB útiles (100 %),
tolera 0 fallos
RAID1 con 2 x 4000 GB: 4000 GB útiles (50 %),
tolera 1 fallo
RAID5 con 4 x 2000 GB: 6000 GB útiles (75 %),
tolera 1 fallo
RAID6 con 6 x 8000 GB: 32000 GB útiles (67 %),
tolera 2 fallos
RAID10 con 4 x 1000 GB: 2000 GB útiles (50 %),
tolera 1 fallo por pareja (hasta 2)
```

Ejemplo: Configuraciones no válidas

ENTRADA

```
RAID5 2 1000
RAID10 5 2000
RAID7 4 1000
```

SALIDA ESPERADA

```
RAID5 con 2 discos: no válido (necesita al menos
3 discos)
RAID10 con 5 discos: no válido (necesita al menos
4 discos y un número par)
RAID7: nivel desconocido
```

11. Simulador de estrategias de copia

Difícil · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

La primera línea es «datos N» (GB que se copian), la segunda «cambio P» (porcentaje de los datos que se modifica cada día, siempre ficheros distintos) y la tercera «plan ...» con un tipo de copia por día: C (completa: copia todo), I (incremental: lo que ha cambiado desde la copia anterior, P % de los datos) o D (diferencial: todo lo que ha cambiado desde la última completa, P % por cada día transcurrido desde ella, como máximo el 100 %). Muestra «Día N (T): X GB» (1 decimal) por cada día, «Total ocupado: X GB» y «Para restaurar el último día: día A, día B...» con las copias necesarias: la última completa, la última diferencial posterior (si la hay) y las incrementales posteriores a esa diferencial (o a la completa si no hay diferencial). Una I o una D antes de cualquier completa muestran «Día N: ERROR, una copia T necesita una completa anterior» y no ocupan nada.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
datos = float(lineas[0][1])
cambio = float(lineas[1][1])
plan = lineas[2][1:]          # C = completa, I = incremental, D = diferencial, un día cada una

# TODO: tamaño de cada copia, total ocupado y copias necesarias para restaurar el último día
```

Ejemplo: Semana con incrementales

ENTRADA

```
datos 200
cambio 5
plan C I I I I I I
```

SALIDA ESPERADA

```
Día 1 (C): 200.0 GB
Día 2 (I): 10.0 GB
Día 3 (I): 10.0 GB
Día 4 (I): 10.0 GB
Día 5 (I): 10.0 GB
Día 6 (I): 10.0 GB
Día 7 (I): 10.0 GB
Total ocupado: 260.0 GB
Para restaurar el último día: día 1, día 2, día
3, día 4, día 5, día 6, día 7
```

Ejemplo: Semana con diferenciales

ENTRADA

```
datos 200  
cambio 5  
plan C D D D D D D
```

SALIDA ESPERADA

```
Día 1 (C): 200.0 GB  
Día 2 (D): 10.0 GB  
Día 3 (D): 20.0 GB  
Día 4 (D): 30.0 GB  
Día 5 (D): 40.0 GB  
Día 6 (D): 50.0 GB  
Día 7 (D): 60.0 GB  
Total ocupado: 410.0 GB  
Para restaurar el último día: día 1, día 7
```

Ejemplo: Mixta con dos completas

ENTRADA

```
datos 500  
cambio 10  
plan C I I D I C I D I
```

SALIDA ESPERADA

```
Día 1 (C): 500.0 GB  
Día 2 (I): 50.0 GB  
Día 3 (I): 50.0 GB  
Día 4 (D): 150.0 GB  
Día 5 (I): 50.0 GB  
Día 6 (C): 500.0 GB  
Día 7 (I): 50.0 GB  
Día 8 (D): 100.0 GB  
Día 9 (I): 50.0 GB  
Total ocupado: 1500.0 GB  
Para restaurar el último día: día 6, día 8, día 9
```

12. ¿Es buena esta copia de seguridad?

Muy difícil · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

Para comprobar una copia se compara un manifiesto del original con otro de la copia: cada línea es «ruta tamaño hash» (como lo que genera sha256sum, abreviado). La entrada tiene los dos manifiestos separados por una línea «---». Informa, en este orden: los ficheros MOVIDOS (faltan en una ruta pero aparecen en otra de la copia con el mismo hash) como «MOVIDO ruta -> nueva_ruta»; los que FALTAN; los DISTINTOS (misma ruta, distinto hash) con «(X bytes en el original, Y en la copia)»; y los que SOBРАН; cada grupo ordenado alfabéticamente. Termina con «Correctos: N de M · Datos del original: X MiB» (2 decimales) y «Copia VÁLIDA» si no falta ni difiere nada (los movidos y sobrantes no la invalidan) o «Copia NO VÁLIDA».

Código de partida (python)

```
import sys  
  
texto = sys.stdin.read().split("---")  
original = [l.split() for l in texto[0].splitlines() if l.strip()] # ruta tamaño hash  
copia = [l.split() for l in texto[1].splitlines() if l.strip()]  
  
# TODO: compara los dos manifiestos e informa de las diferencias
```

Ejemplo: Copia perfecta

ENTRADA

```
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 77b1
docs/nominas/febrero.xlsx 22340 01ce
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
---
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 77b1
docs/nominas/febrero.xlsx 22340 01ce
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
```

SALIDA ESPERADA

Correctos: 5 de 5 · Datos del original: 5.65 MiB
Copia VÁLIDA

Ejemplo: Con errores

ENTRADA

```
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 77b1
docs/nominas/febrero.xlsx 22340 01ce
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
---
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 0000
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
tmp/cache.bin 10 ffff
```

SALIDA ESPERADA

FALTA docs/nominas/febrero.xlsx
DISTINTO docs/nominas/enero.xlsx (22100 bytes en
el original, 22100 en la copia)
SOBRA tmp/cache.bin
Correctos: 3 de 5 · Datos del original: 5.65 MiB
Copia NO VÁLIDA

Administración de Sistemas Operativos

13. Saludo con validación

Fácil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Lee un nombre por teclado. Si está vacío, muestra «Error: nombre vacío»; si no, muestra «Hola, NOMBRE» y en otra línea «Tu nombre tiene N letras».

Código de partida (bash)

```
#!/bin/bash
read -r nombre

# TODO: valida y saluda
```

Ejemplo: Nombre normal

ENTRADA

Ana

SALIDA ESPERADA

Hola, Ana
Tu nombre tiene 3 letras

Ejemplo: Otro nombre

ENTRADA

Rodrigo

SALIDA ESPERADA

Hola, Rodrigo
Tu nombre tiene 7 letras

14. Tabla de multiplicar con bucle

Fácil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Lee un número. Si no es un entero positivo, muestra «Error: número no válido». Si lo es, muestra su tabla del 1 al 10 con el formato «N x i = resultado».

Código de partida (bash)

```
#!/bin/bash
read -r n

# TODO
```

Ejemplo: Tabla del 7

ENTRADA

7

SALIDA ESPERADA

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

Ejemplo: Texto

ENTRADA

hola

SALIDA ESPERADA

Error: número no válido

15. Encuentra el fallo: el saludo que no se guarda

Fácil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

El script lee un nombre y debería mostrar «Hola, NOMBRE». En su lugar muestra un error y una línea vacía. Corrígelo.

Código de partida (bash)

```
#!/bin/bash
read -r nombre
saludo = "Hola, $nombre"
echo "$saludo"
```

Ejemplo: Ana

ENTRADA

Ana

SALIDA ESPERADA

Hola, Ana

Ejemplo: Luis

ENTRADA

Luis

SALIDA ESPERADA

Hola, Luis

16. Analizar un fichero de log

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

operativos

Por la entrada llega un log con líneas «FECHA HORA NIVEL IP mensaje». Muestra:

- «Errores: N» (líneas con nivel ERROR)
- «Avisos: N» (nivel WARN)
- «IP con más errores: IP (N)» con la IP que más aparece en las líneas ERROR.

Código de partida (bash)

```
#!/bin/bash
log=$(cat)

# TODO
```

Ejemplo: Log de ejemplo

ENTRADA

```
2025-09-30 10:00:01 INFO 10.0.0.5 Inicio
2025-09-30 10:00:02 ERROR 10.0.0.7 Login fallido
2025-09-30 10:00:05 WARN 10.0.0.5 Disco al 85%
2025-09-30 10:01:00 ERROR 10.0.0.7 Login fallido
2025-09-30 10:02:00 ERROR 10.0.0.9 Timeout
```

SALIDA ESPERADA

```
Errores: 3
Avisos: 1
IP con más errores: 10.0.0.7 (2)
```

17. Usuarios del sistema desde /etc/passwd

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Por la entrada llega el contenido de un /etc/passwd (campos separados por «:» — usuario:x:UID:GID:comentario:home:shell). Muestra, ordenados por UID, los usuarios «reales» (UID ≥ 1000 y distinto de 65534) con el formato «usuario (UID) -> home [shell]», y al final «Total: N».

Código de partida (bash)

```
#!/bin/bash
# La entrada estándar es el contenido de /etc/passwd

# TODO
```

Ejemplo: passwd de ejemplo

ENTRADA

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
lucia:x:1001:1001:Lucía:/home/lucia:/bin/zsh
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
ana:x:1000:1000:Ana:/home/ana:/bin/bash
```

SALIDA ESPERADA

```
ana (1000) -> /home/ana [/bin/bash]
lucia (1001) -> /home/lucia [/bin/zsh]
Total: 2
```

18. Crear ficheros y asignar permisos

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Por la entrada llega una lista de nombres de fichero (uno por línea). En un directorio temporal (mktemp -d), crea cada fichero y asígnele permisos: 750 si termina en .sh, 600 si termina en .key y 644 en cualquier otro caso. Después muestra, ordenados por nombre, sus permisos en octal y su nombre con stat -c '%a %n'.

Código de partida (bash)

```
#!/bin/bash
dir=$(mktemp -d)
cd "$dir" || exit 1

# TODO: crea los ficheros de la entrada y asigna los permisos
```

Ejemplo: Tres tipos

ENTRADA

```
copia.sh
notas.txt
servidor.key
```

SALIDA ESPERADA

```
750 copia.sh
644 notas.txt
600 servidor.key
```

19. Encuentra el fallo: la alerta que salta siempre

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

El script lee el porcentaje de uso del disco y debería mostrar «ALERTA: disco al N %» si pasa de 80, u «OK» si no. Con un uso del 9 % también da la alerta. Corrígelo.

Código de partida (bash)

```
#!/bin/bash
read -r uso
if [ $uso > 80 ]; then
    echo "ALERTA: disco al $uso %"
else
    echo "OK"
fi
```

Ejemplo: Disco casi lleno

ENTRADA

```
95
```

SALIDA ESPERADA

```
ALERTA: disco al 95 %
```

Ejemplo: Disco con espacio

ENTRADA

```
9
```

SALIDA ESPERADA

```
OK
```

Ejemplo: Justo en el límite

ENTRADA

```
80
```

SALIDA ESPERADA

```
OK
```

20. Encuentra el fallo: el mayor que no lo es

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

El script lee dos números en la misma línea y muestra el mayor. Con «7 3» funciona, pero con «10 9» dice que el mayor es 9. Corrígelo.

Código de partida (bash)

```
#!/bin/bash
read -r a b
if [[ $a > $b ]]; then
    echo "El mayor es $a"
else
    echo "El mayor es $b"
fi
```

Ejemplo: Un dígito

ENTRADA

7 3

SALIDA ESPERADA

El mayor es 7

Ejemplo: Dos cifras contra una

ENTRADA

10 9

SALIDA ESPERADA

El mayor es 10

Ejemplo: El segundo es mayor

ENTRADA

4 25

SALIDA ESPERADA

El mayor es 25

21. Encuentra el fallo: el nombre con espacio

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

El script da la bienvenida a Ana López y a cualquier otra persona le dice «No te conozco». Cuando escribe ella su nombre completo, también le dice que no la conoce y aparece un error. Corrígelo.

Código de partida (bash)

```
#!/bin/bash
read -r nombre
if [ $nombre = "Ana López" ]; then
    echo "Bienvenida, Ana"
else
    echo "No te conozco"
fi
```

Ejemplo: Nombre con espacio

ENTRADA

Ana López

SALIDA ESPERADA

Bienvenida, Ana

Ejemplo: Otra persona

ENTRADA

Luis

SALIDA ESPERADA

No te conozco

22. Alerta de espacio en disco

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Por la entrada llega la salida de «df -h» (con su cabecera). Muestra «ALERTA: PUNTO_DE_MONTAJE al N%» para cada sistema de ficheros con un uso igual o superior al umbral que se pasa en la primera línea de la entrada, y «Todo correcto» si no hay ninguno.

Código de partida (bash)

```
#!/bin/bash
read -r umbral

# TODO
```

Ejemplo: Umbral 80

ENTRADA

```
80
S.ficheros Tamaño Usados Disp Uso% Montado en
/dev/sda1 50G 45G 5G 90% /
/dev/sdb1 200G 100G 100G 50% /datos
/dev/sdc1 20G 16G 4G 80% /var
```

SALIDA ESPERADA

```
ALERTA: / al 90%
ALERTA: /var al 80%
```

23. ¿Quién intenta entrar por SSH?

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

La entrada es un fragmento de /var/log/auth.log. Las líneas de contraseñas fallidas tienen la forma «... sshd[PID]: Failed password for USUARIO from IP port N ssh2» o «... Failed password for invalid user USUARIO from IP port N ssh2». Cuenta los intentos fallidos por IP y muéstralos de más a menos intentos (a igualdad, por IP en orden alfabético) con el formato «IP: N intentos», añadiendo « → BLOQUEAR» si son 5 o más. Termina con «Usuarios inexistentes probados: N» (usuarios distintos que aparecen tras «invalid user»). Las demás líneas se ignoran.

Código de partida (bash)

```
#!/bin/bash
# Lee el log por la entrada estándar
log=$(cat)

# TODO: intentos fallidos por IP y usuarios inexistentes probados
```

Ejemplo: Ataque desde una IP

ENTRADA

```
Oct 1 10:00:01 srv sshd[901]: Failed password
for root from 203.0.113.9 port 50122 ssh2
Oct 1 10:00:03 srv sshd[901]: Failed password
for root from 203.0.113.9 port 50122 ssh2
Oct 1 10:00:05 srv sshd[905]: Failed password
for invalid user admin from 203.0.113.9 port
50200 ssh2
Oct 1 10:00:07 srv sshd[905]: Failed password
for invalid user test from 203.0.113.9 port 50200
ssh2
Oct 1 10:00:09 srv sshd[905]: Failed password
for invalid user admin from 203.0.113.9 port
50201 ssh2
Oct 1 10:02:00 srv sshd[950]: Accepted publickey
for ana from 192.168.1.20 port 41000 ssh2
Oct 1 10:05:00 srv sshd[960]: Failed password
for ana from 192.168.1.20 port 41100 ssh2
```

SALIDA ESPERADA

```
203.0.113.9: 5 intentos → BLOQUEAR
192.168.1.20: 1 intentos
Usuarios inexistentes probados: 2
```

Ejemplo: Varias IP empatadas

ENTRADA

```
a sshd[1]: Failed password for root from 10.0.0.9
port 1 ssh2
b sshd[1]: Failed password for root from
10.0.0.10 port 1 ssh2
c sshd[1]: Failed password for invalid user pi
from 10.0.0.9 port 2 ssh2
d sshd[1]: Failed password for invalid user pi
from 10.0.0.10 port 2 ssh2
e sshd[1]: Connection closed by 10.0.0.11 port 3
```

SALIDA ESPERADA

```
10.0.0.10: 2 intentos
10.0.0.9: 2 intentos
Usuarios inexistentes probados: 1
```

24. Encuentra el fallo: el total que se queda en cero

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

El script suma los números que recibe por la entrada, uno por línea, y muestra «Total: N». Siempre muestra «Total: 0». Corrígelo.

Código de partida (bash)

```
#!/bin/bash
total=0
cat | while read -r n; do
    total=$((total + n))
done
echo "Total: $total"
```

Ejemplo: Tres números

ENTRADA

```
3
5
2
```

SALIDA ESPERADA

```
Total: 10
```

Ejemplo: Con un negativo

ENTRADA

```
100
-40
```

SALIDA ESPERADA

```
Total: 60
```

25. Encuentra el fallo: las 08:09 que rompen la cuenta

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

El script lee una hora con el formato HH:MM y muestra los minutos que han pasado desde medianoche. Con 10:30 funciona (630), pero con 08:09 da un error. Corrígelo.

Código de partida (bash)

```
#!/bin/bash
read -r hora
h=${hora%*:}
m=${hora##*:}
echo $(( h * 60 + m ))
```

Ejemplo: Media mañana

ENTRADA

10:30

SALIDA ESPERADA

630

Ejemplo: Con ceros delante

ENTRADA

08:09

SALIDA ESPERADA

489

Ejemplo: Última hora

ENTRADA

23:59

SALIDA ESPERADA

1439

26. Permisos: de letras a octal y al revés

Muy difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Cada línea es un permiso en notación simbólica de 9 caracteres (rwxr-x---) o en octal de 3 o 4 cifras (750, 4755).

Convierte cada uno a la otra notación: «rwxr-x--- = 750» y «0644 = rw-r--r--». Ten en cuenta los bits especiales: setuid (4), setgid (2) y sticky (1), que en la notación simbólica cambian la x del propietario, del grupo o de otros por s o t (o por S o T si ese permiso de ejecución no está). Una entrada simbólica con bits especiales da 4 cifras (rwsr-xr-x = 4755); sin ellos, 3. Una entrada octal se muestra siempre con 4 cifras a la izquierda del «=». Si la línea no es válida, «línea: no válido».

Código de partida (bash)

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

```
# TODO: convierte cada línea de simbólico a octal o de octal a simbólico
```

Ejemplo: Permisos habituales

ENTRADA

```
rwxr-x---  
rw-r--r--  
644  
750
```

SALIDA ESPERADA

```
rwxr-x--- = 750  
rw-r--r-- = 644  
0644 = rw-r--r--  
0750 = rwxr-x---
```

Ejemplo: Bits especiales

ENTRADA

```
rwsr-xr-x  
rwxrwxrwt  
rwxrws---  
4755  
1777  
2770
```

SALIDA ESPERADA

```
rwsr-xr-x = 4755  
rwxrwxrwt = 1777  
rwxrws--- = 2770  
4755 = rwsr-xr-x  
1777 = rwxrwxrwt  
2770 = rwxrws---
```

27. ¿Llega este cable?

Medio · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Cada línea describe un tramo de cable: «medio metros velocidad», con el medio cat5e, cat6, cat6a, cat8 (par trenzado), om3, om4 (fibra multimodo) u os2 (fibra monomodo) y la velocidad en Gbit/s con una G (1G, 10G...). El diccionario MAXIMO indica la distancia máxima de cada medio a cada velocidad que admite. Muestra «medio, M m a VG: SÍ (máximo X m)» si el tramo es válido; «...: NO (máximo X m; sirve a, b...)» si es demasiado largo, con los medios que sí llegarían en orden alfabético (o «ninguno»); «...: NO (no admite VG; velocidades inferiores válidas: ...)» si el medio no admite esa velocidad (o «ninguna»); y «medio: medio desconocido» si no existe.

Código de partida (python)

```
import sys

# Distancia máxima en metros de cada medio para cada velocidad (Gbit/s)
MAXIMO = {
    "cat5e": {1: 100},
    "cat6": {1: 100, 10: 55},
    "cat6a": {1: 100, 10: 100},
    "cat8": {10: 30, 25: 30, 40: 30},
    "om3": {1: 550, 10: 300, 40: 100},
    "om4": {1: 550, 10: 400, 40: 150},
    "os2": {1: 10000, 10: 10000, 40: 10000},
}

# TODO: decide si cada tramo de cable es válido
```

Ejemplo: Oficina

ENTRADA

```
cat6 70 1G
cat6 70 10G
cat6a 90 10G
om4 350 10G
```

SALIDA ESPERADA

```
cat6, 70 m a 1G: SÍ (máximo 100 m)
cat6, 70 m a 10G: NO (máximo 55 m; sirve cat6a,
om3, om4, os2)
cat6a, 90 m a 10G: SÍ (máximo 100 m)
om4, 350 m a 10G: SÍ (máximo 400 m)
```

Ejemplo: Velocidades no admitidas

ENTRADA

```
cat5e 20 10G
cat8 10 1G
coaxial 50 1G
```

SALIDA ESPERADA

```
cat5e, 20 m a 10G: NO (no admite 10G; velocidades
inferiores válidas: 1G)
cat8, 10 m a 1G: NO (no admite 1G; velocidades
inferiores válidas: ninguna)
coaxial: medio desconocido
```

28. La tabla MAC de un switch

Difícil · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Un switch aprende en qué puerto está cada equipo mirando la MAC de origen de las tramas que recibe. La primera línea es «puertos N»; cada línea siguiente, una trama recibida: «puerto MAC_origen MAC_destino». Para cada trama: si la MAC de origen no estaba en la tabla (o estaba en otro puerto, porque el equipo se ha movido), muestra «Aprende MAC en el puerto P». Después, con dos espacios delante: si el destino es FF:FF:FF:FF:FF:FF, «Difusión: inunda por a, b...» (todos los puertos menos el de entrada); si el destino no está en la tabla, «MAC desconocido: inunda por ...»; si está en el mismo puerto de entrada, «MAC está en el mismo puerto: descarta la trama»; y si no, «Reenvía a MAC por el puerto P». Al final muestra «Tabla MAC:» y cada entrada « MAC → puerto P» ordenada por MAC.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
puertos = int(lineas[0][1])
BROADCAST = "FF:FF:FF:FF:FF:FF"

# TODO: simula cómo aprende y reenvía las tramas un switch
```

Ejemplo: Arranque de una red

ENTRADA

```
puertos 4
1 AA:AA:AA:00:00:01 FF:FF:FF:FF:FF:FF
2 BB:BB:BB:00:00:02 AA:AA:AA:00:00:01
1 AA:AA:AA:00:00:01 BB:BB:BB:00:00:02
3 CC:CC:CC:00:00:03 DD:DD:DD:00:00:04
```

SALIDA ESPERADA

```
Aprende AA:AA:AA:00:00:01 en el puerto 1
  Difusión: inunda por 2, 3, 4
Aprende BB:BB:BB:00:00:02 en el puerto 2
  Reenvía a AA:AA:AA:00:00:01 por el puerto 1
  Reenvía a BB:BB:BB:00:00:02 por el puerto 2
Aprende CC:CC:CC:00:00:03 en el puerto 3
  DD:DD:DD:00:00:04 desconocido: inunda por 1, 2, 4
Tabla MAC:
  AA:AA:AA:00:00:01 → puerto 1
  BB:BB:BB:00:00:02 → puerto 2
  CC:CC:CC:00:00:03 → puerto 3
```

29. Puntos únicos de fallo de una red

Muy difícil · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

La primera línea es «raíz NOMBRE»: el equipo por el que se sale a Internet (normalmente el router). Cada línea siguiente es un enlace «enlace A B» entre dos equipos (los enlaces funcionan en los dos sentidos). Para cada equipo con dos o más enlaces que no sea la raíz, en orden alfabético, calcula qué equipos dejarían de poder llegar a la raíz si ese equipo fallara: muestra «Si falla X: sin red a, b...» (en orden alfabético) o «Si falla X: nadie pierde la red (hay caminos alternativos)». Al final, «Puntos únicos de fallo: N» con los equipos cuyo fallo deja a alguien sin red.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
raiz = lineas[0][1] # equipo desde el que se sale a Internet
enlaces = [(a, b) for _, a, b in lineas[1:]]

# TODO: para cada equipo de interconexión, ¿quién se queda sin red si falla?
```

Ejemplo: Estrella extendida

ENTRADA

```
raiz router
enlace router sw-core
enlace sw-core sw-planta1
enlace sw-core sw-planta2
enlace sw-planta1 pc1
enlace sw-planta1 pc2
enlace sw-planta2 pc3
enlace sw-planta2 impresora
```

SALIDA ESPERADA

```
Si falla sw-core: sin red impresora, pc1, pc2, pc3, sw-planta1, sw-planta2
Si falla sw-planta1: sin red pc1, pc2
Si falla sw-planta2: sin red impresora, pc3
Puntos únicos de fallo: 3
```

Ejemplo: Con enlace redundante

ENTRADA

```
raiz router
enlace router sw-a
enlace router sw-b
enlace sw-a sw-b
enlace sw-a pc1
enlace sw-b pc2
enlace sw-b ap
enlace ap portatil
```

SALIDA ESPERADA

```
Si falla ap: sin red portatil
Si falla sw-a: sin red pc1
Si falla sw-b: sin red ap, pc2, portatil
Puntos únicos de fallo: 3
```

Configuración de Redes TCP/IP

30. ¿Pueden hablar sin router?

Medio · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Dos equipos solo se comunican directamente si están en la misma red, es decir, si su IP AND la máscara da el mismo resultado. Cada línea tiene «IP/prefijo IP2»: el primer equipo con su prefijo y la IP del segundo (que usa la misma máscara). Muestra «IP y IP2: misma red RED/prefijo, comunicación directa» o «IP y IP2: redes distintas (RED1 y RED2), hace falta un router». Las funciones `a_entero` y `a_texto` convierten entre la notación con puntos y un entero de 32 bits.

Código de partida (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
    return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

# TODO: ¿están los dos equipos en la misma red?
```

Ejemplo: Redes /24 y /26

ENTRADA

```
192.168.1.10/24 192.168.1.200
192.168.1.10/26 192.168.1.70
192.168.1.65/26 192.168.1.126
```

SALIDA ESPERADA

```
192.168.1.10 y 192.168.1.200: misma red
192.168.1.0/24, comunicación directa
192.168.1.10 y 192.168.1.70: redes distintas
(192.168.1.0 y 192.168.1.64), hace falta un
router
192.168.1.65 y 192.168.1.126: misma red
192.168.1.64/26, comunicación directa
```

Ejemplo: Prefijos que cortan otros octetos

ENTRADA

```
172.16.37.10/20 172.16.40.1
172.16.37.10/20 172.16.48.1
10.1.2.3/8 10.200.0.1
```

SALIDA ESPERADA

```
172.16.37.10 y 172.16.40.1: misma red
172.16.32.0/20, comunicación directa
172.16.37.10 y 172.16.48.1: redes distintas
(172.16.32.0 y 172.16.48.0), hace falta un router
10.1.2.3 y 10.200.0.1: misma red 10.0.0.0/8,
comunicación directa
```

31. Abreviar y expandir direcciones IPv6

Difícil · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Cada línea es una dirección IPv6. Si está completa (8 grupos sin «:»), abrévala: quita los ceros a la izquierda de cada grupo (un grupo todo a ceros queda como 0) y sustituye por «::» la serie más larga de dos o más grupos seguidos a cero (si hay dos igual de largas, la primera; solo se usa «::» una vez). Si está abreviada, expándela a 8 grupos de 4 cifras. Usa siempre minúsculas. Muestra «entrada → resultado».

Código de partida (python)

```
import sys

# TODO: abrevia las direcciones IPv6 completas y expande las abreviadas
```

Ejemplo: Abreviar

ENTRADA

```
2001:0db8:0000:0000:0000:ff00:0042:8329
fe80:0000:0000:0000:0202:b3ff:fe1e:8329
2001:0DB8:0000:0001:0000:0000:0000:0001
```

SALIDA ESPERADA

```
2001:0db8:0000:0000:0000:ff00:0042:8329 →
2001:db8::ff00:42:8329
fe80:0000:0000:0000:0202:b3ff:fe1e:8329 →
fe80::202:b3ff:fe1e:8329
2001:0DB8:0000:0001:0000:0000:0000:0001 →
2001:db8:0:1::1
```

Ejemplo: Expandir

ENTRADA

```
2001:db8::ff00:42:8329
::1
fe80::
```

SALIDA ESPERADA

```
2001:db8::ff00:42:8329 →
2001:0db8:0000:0000:0000:ff00:0042:8329
::1 → 0000:0000:0000:0000:0000:0000:0000:0001
fe80:: → fe80:0000:0000:0000:0000:0000:0000:0000
```

32. Calculadora VLSM

Muy difícil · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

La primera línea es «red A.B.C.D/p», la red que hay que repartir. Cada línea siguiente es «nombre hosts»: un departamento y los equipos que necesita. Con VLSM, asigna las subredes de la que más hosts necesita a la que menos (a igualdad, por nombre), cada una del tamaño mínimo que alcance ($2^{\text{bits}} - 2 \geq \text{hosts}$, con al menos 2 bits) y una detrás de otra desde el principio de la red. Para cada una muestra «nombre (hosts): RED/prefijo, hosts PRIMERO - ÚLTIMO, difusión DIF, sobran N»; si ya no cabe, «nombre: no hay espacio para N hosts» (y se sigue con las demás). Termina con «Libre: N direcciones desde IP» o «Libre: 0 direcciones».

Código de partida (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
    return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
base_ip, base_prefijo = lineas[0][1].split("/")
departamentos = [(nombre, int(hosts)) for nombre, hosts in lineas[1:]]

# TODO: reparte la red con VLSM, de la subred que más hosts necesita a la que menos
```

Ejemplo: Ejemplo del tema

ENTRADA

```
red 192.168.50.0/24
Desarrollo 100
Administracion 50
Direccion 10
Enlace 2
```

SALIDA ESPERADA

```
Desarrollo (100): 192.168.50.0/25, hosts
192.168.50.1 - 192.168.50.126, difusión
192.168.50.127, sobran 26
Administracion (50): 192.168.50.128/26, hosts
192.168.50.129 - 192.168.50.190, difusión
192.168.50.191, sobran 12
Direccion (10): 192.168.50.192/28, hosts
192.168.50.193 - 192.168.50.206, difusión
192.168.50.207, sobran 4
Enlace (2): 192.168.50.208/30, hosts
192.168.50.209 - 192.168.50.210, difusión
192.168.50.211, sobran 0
Libre: 44 direcciones desde 192.168.50.212
```

Ejemplo: Instituto en una /22

ENTRADA

```
red 10.20.0.0/22
Aulas 400
Profesorado 120
Wifi 200
Servidores 20
Impresoras 12
```

SALIDA ESPERADA

```
Aulas (400): 10.20.0.0/23, hosts 10.20.0.1 -
10.20.1.254, difusión 10.20.1.255, sobran 110
Wifi (200): 10.20.2.0/24, hosts 10.20.2.1 -
10.20.2.254, difusión 10.20.2.255, sobran 54
Profesorado (120): 10.20.3.0/25, hosts 10.20.3.1
- 10.20.3.126, difusión 10.20.3.127, sobran 6
Servidores (20): 10.20.3.128/27, hosts
10.20.3.129 - 10.20.3.158, difusión 10.20.3.159,
sobran 10
Impresoras (12): 10.20.3.160/28, hosts
10.20.3.161 - 10.20.3.174, difusión 10.20.3.175,
sobran 2
Libre: 80 direcciones desde 10.20.3.176
```

Servicios de Red, Recursos Compartidos y Seguridad

33. Permisos efectivos de una carpeta compartida

Medio · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-seguridad

En Windows, a una carpeta compartida por la red se le aplican dos capas de permisos: los del recurso compartido y los NTFS. Dentro de cada capa, un usuario suma los permisos de todos sus grupos (se queda con el mayor), una denegación explícita gana a cualquier permiso, y el permiso efectivo es el más restrictivo de las dos capas. Las líneas pueden ser «compartido GRUPO nivel», «ntfs GRUPO nivel», «denegar GRUPO» (denegación NTFS total) o «usuario NOMBRE grupo1,grupo2...». Los niveles, de menor a mayor, son ninguno, lectura, modificar y control. Para cada usuario, en el orden de la entrada, muestra «nombre: compartido X, NTFS Y → efectivo Z».

Código de partida (python)

```
import sys

NIVELES = ["ninguno", "lectura", "modificar", "control"]

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]

# TODO: calcula el permiso efectivo de cada usuario sobre la carpeta compartida
```

Ejemplo: Departamentos

ENTRADA

```
compartido Todos modificar
ntfs Ventas modificar
ntfs Direccion control
ntfs Todos lectura
usuario ana Ventas,Todos
usuario luis Todos
usuario marta Direccion,Todos
```

SALIDA ESPERADA

```
ana: compartido modificar, NTFS modificar →
efectivo modificar
luis: compartido modificar, NTFS lectura →
efectivo lectura
marta: compartido modificar, NTFS control →
efectivo modificar
```

Ejemplo: Denegación y compartido restrictivo

ENTRADA

```
compartido Todos lectura
compartido Admins control
ntfs Admins control
ntfs Todos modificar
denegar Becarios
usuario pedro Todos,Becarios
usuario eva Todos
usuario root Admins
```

SALIDA ESPERADA

```
pedro: compartido lectura, NTFS ninguno →
efectivo ninguno
eva: compartido lectura, NTFS modificar →
efectivo lectura
root: compartido control, NTFS control → efectivo
control
```

34. Un cortafuegos de primera coincidencia

Difícil · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-seguridad

Un cortafuegos revisa sus reglas en orden y aplica la primera que coincide; si ninguna coincide, aplica la política por defecto. La entrada tiene dos partes separadas por «---». La primera empieza con «política allow|deny» y sigue con reglas «allow|deny protocolo puerto from red» (protocolo tcp, udp o any; puerto un número o any; red en CIDR, una IP sola o any). La segunda son paquetes entrantes «protocolo ip_origen puerto_destino». Para cada paquete muestra «proto ip → puerto: ALLOW (regla N)», «...: DENY (regla N)» o «...: ACCIÓN (política por defecto)».

Código de partida (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

texto = sys.stdin.read().split("---")
reglas = [l.split() for l in texto[0].splitlines() if l.strip()] # la primera línea es «política
allow|deny»
paquetes = [l.split() for l in texto[1].splitlines() if l.strip()]

# TODO: aplica las reglas en orden a cada paquete (gana la primera que coincide)
```

Ejemplo: Servidor web

ENTRADA

```
politica deny
allow tcp 22 from 192.168.1.0/24
allow tcp 443 from any
allow tcp 80 from any
deny tcp 3306 from any
allow udp 53 from 10.0.0.0/8
---
tcp 192.168.1.50 22
tcp 203.0.113.7 22
tcp 203.0.113.7 443
tcp 192.168.1.50 3306
udp 10.20.30.40 53
udp 8.8.8.8 53
```

SALIDA ESPERADA

```
tcp 192.168.1.50 → 22: ALLOW (regla 1)
tcp 203.0.113.7 → 22: DENY (política por defecto)
tcp 203.0.113.7 → 443: ALLOW (regla 2)
tcp 192.168.1.50 → 3306: DENY (regla 4)
udp 10.20.30.40 → 53: ALLOW (regla 5)
udp 8.8.8.8 → 53: DENY (política por defecto)
```

35. ¿Cuánto aguantaría esta contraseña?

Muy difícil · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-seguridad

Cada línea es una contraseña. Si en minúsculas está en el DICCIONARIO, o lo está tras quitar los dígitos, puntos y signos de exclamación del final (Barcelona2024!), muestra «clave: muy débil (está en los diccionarios de contraseñas)». Si no, calcula su entropía: longitud × log2(juego), donde el juego suma 26 si hay minúsculas, 26 si hay mayúsculas, 10 si hay dígitos y 33 si hay otros símbolos; si tiene muy pocos caracteres distintos (no más de un tercio de su longitud), divide la entropía entre 2. Clasifícala como débil (< 40 bits), media (< 60), fuerte (< 80) o muy fuerte, y estima el tiempo medio de un ataque de fuerza bruta a 10¹⁰ intentos por segundo: 2^{entropía} / 2 / 10¹⁰ segundos, expresado en años, días, horas o minutos (la mayor unidad con valor ≥ 1; con 1 decimal si es menor que 100, y si no, redondeado y con punto de miles, como «857.012 años»), «más de un millón de ...» si llega al millón, «menos de 1 minuto» o «instantáneo» (< 1 s). Muestra «clave: nivel (X bits, TIEMPO por fuerza bruta)», con los bits con 1 decimal.

Código de partida (python)

```
import sys
import math

DICCIONARIO = {"password", "123456", "qwerty", "admin", "iloveyou", "dragon", "monkey", "letmein",
               "futbol", "barcelona", "madrid", "contraseña", "hola", "princesa", "superman"}

# TODO: estima la fortaleza de cada contraseña
```

Ejemplo: Varias contraseñas

ENTRADA

```
password
Barcelona2024!
Gato
T3rue1!
aaaaaaaaaaaa
caballo-bateria-grapa-correcto
```

SALIDA ESPERADA

```
password: muy débil (está en los diccionarios de contraseñas)
Barcelona2024!: muy débil (está en los diccionarios de contraseñas)
Gato: débil (22.8 bits, instantáneo por fuerza bruta)
T3rue1!: media (46.0 bits, 58.2 minutos por fuerza bruta)
aaaaaaaaaaaa: débil (28.2 bits, instantáneo por fuerza bruta)
caballo-bateria-grapa-correcto: muy fuerte (176.5 bits, más de un millón de años por fuerza bruta)
```

Ejemplo: Largas y aleatorias

ENTRADA

```
Xk9#mP2$VL8q  
z7Q!r4T@w1E%y6U^
```

SALIDA ESPERADA

```
Xk9#mP2$VL8q: fuerte (78.8 bits, 856.735 años por  
fuerza bruta)  
z7Q!r4T@w1E%y6U^: muy fuerte (105.1 bits, más de  
un millón de años por fuerza bruta)
```

Documentación y Aplicaciones Ofimáticas

36. Informe del inventario

Medio · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

La primera línea es «hoy=AAAA-MM-DD» y el resto un CSV separado por punto y coma con cabecera «codigo;tipo;aula;precio;fin_garantia». Agrupa los equipos por aula (en orden alfabético) y muestra «AULA: N equipos, X €» (2 decimales; «1 equipo» en singular), añadiendo «, sin garantía: código1, código2» con los códigos ordenados si alguno tiene la garantía vencida (fecha de fin anterior a hoy). Termina con «TOTAL: N equipos, X €, M sin garantía (P %)» con el porcentaje sin decimales. Es lo que harías con una tabla dinámica en la hoja de cálculo.

Código de partida (python)

```
import sys  
import csv  
from datetime import date  
  
lineas = sys.stdin.read().splitlines()  
hoy = date.fromisoformat(lineas[0].split("=")[1])  
filas = list(csv.DictReader(lineas[1:], delimiter=";")) # codigo;tipo;aula;precio;fin_garantia  
  
# TODO: informe por aula con número de equipos, valor y garantías caducadas
```

Ejemplo: Instituto

ENTRADA

```
hoy=2026-10-01  
codigo;tipo;aula;precio;fin_garantia  
PC-201;sobremesa;2.01;780;2027-05-01  
PC-202;sobremesa;2.01;780;2026-03-15  
IMP-01;impresora;2.01;320.5;2025-11-30  
PC-110;sobremesa;1.10;650;2028-01-10  
PRY-01;proyector;1.10;899.99;2026-12-31  
SRV-01;servidor;CPD;3400;2026-09-30
```

SALIDA ESPERADA

```
1.10: 2 equipos, 1549.99 €  
2.01: 3 equipos, 1880.50 €, sin garantía: IMP-01,  
PC-202  
CPD: 1 equipo, 3400.00 €, sin garantía: SRV-01  
TOTAL: 6 equipos, 6830.49 €, 3 sin garantía (50  
%)
```

37. Un índice automático para la documentación

Difícil · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

La entrada es un documento en Markdown. Genera su índice con los títulos de nivel 2 (##) y 3 (###), numerados (1., 1.1., 1.2., 2....; la numeración de nivel 3 se reinicia en cada nivel 2) y con un enlace a su ancla: «1. [Requisitos previos] (#requisitos-previos)» para el nivel 2 y, con tres espacios delante, « 1.1. [Hardware] (#hardware)» para el nivel 3. El ancla se obtiene del título en minúsculas, sin tildes, eliminando lo que no sea letra, número, espacio o guion, y cambiando los espacios por guiones; si un ancla se repite, la segunda vez se le añade «-1», la tercera «-2»... Ignora el título de nivel 1 y cualquier línea dentro de un bloque de código (entre líneas que empiezan por ```).

Código de partida (python)

```
import sys
import re
import unicodedata

lineas = sys.stdin.read().splitlines()

# TODO: genera el índice numerado de los títulos (## y ###) con su ancla
```

Ejemplo: Manual

ENTRADA

```
# Manual de instalación del servidor

## Requisitos previos
Texto.
### Hardware
### Software
## Instalación
```bash
esto es un comentario dentro de un bloque de
código, no un título
```

### Configuración de la red
### Configuración de la red
## Copias de seguridad: ¿cómo y cuándo?
```

SALIDA ESPERADA

```
1. [Requisitos previos](#requisitos-previos)
  1.1. [Hardware](#hardware)
  1.2. [Software](#software)
2. [Instalación](#instalacion)
  2.1. [Configuración de la red](#configuracion-
de-la-red)
  2.2. [Configuración de la red](#configuracion-
de-la-red-1)
3. [Copias de seguridad: ¿cómo y cuándo?](
#copias-de-seguridad-como-y-cuando)
```

38. Un conversor de Markdown a HTML

Muy difícil · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

Convierte un documento Markdown a HTML con estas reglas. Bloques: los títulos (# a #####) pasan a <h1>...<h6>; las líneas que empiezan por «- » o «* » forman una lista y las que empiezan por «1. », «2. »... una , con cada elemento en un ; las líneas seguidas de texto normal forman un párrafo <p> (unidas con un espacio); una línea en blanco cierra el párrafo o la lista; y lo que hay entre dos líneas que empiezan por `` es un bloque <pre><code>...</code></pre> en el que no se interpreta nada. En línea, dentro de títulos, párrafos y elementos de lista: código → <code>, **negrita** → , *cursiva* → y [texto](url) → texto. Antes de nada, escapa los caracteres <, > y & (html.escape con quote=False), también dentro de los bloques de código. Muestra cada bloque en su propia línea, y las etiquetas de apertura y cierre de las listas en líneas aparte.

Código de partida (python)

```
import sys
import re
import html

lineas = sys.stdin.read().splitlines()

# TODO: convierte el Markdown en HTML
```

Ejemplo: Procedimiento

ENTRADA

```
# Alta de un usuario

Este procedimiento lo sigue el **técnico de guardia** cuando llega una petición *aprobada*.

## Pasos

1. Comprueba la petición en el [gestor de incidencias](https://tickets.empresa.es).
2. Crea la cuenta con `New-ADUser`.
3. Añádela a sus grupos.

Notas:
- Nunca uses contraseñas <123456>.
- Avisa a **RR. HH.** al terminar.

```powershell
New-ADUser -Name "Ana Ruiz" -Enabled $true
if ($x -lt 5) { "<ok>" }
```
```

SALIDA ESPERADA

```
<h1>Alta de un usuario</h1>
<p>Este procedimiento lo sigue el <strong>técnico de guardia</strong> cuando llega una petición <em>aprobada</em>.</p>
<h2>Pasos</h2>
<ol>
<li>Comprueba la petición en el <a href="https://tickets.empresa.es">gestor de incidencias</a>.</li>
<li>Crea la cuenta con <code>New-ADUser</code>.</li>
<li>Añádela a sus grupos.</li>
</ol>
<p>Notas:</p>
<ul>
<li>Nunca uses contraseñas &lt;123456&gt;.</li>
<li>Avisa a <strong>RR. HH.</strong> al terminar.</li>
</ul>
<pre><code>New-ADUser -Name "Ana Ruiz" -Enabled $true
if ($x -lt 5) { "&lt;ok&gt;" }</code></pre>
```

Ejercicios de Sistemas Informáticos

39. Contar ficheros por extensión

Medio · Ejercicios de Sistemas Informáticos · apuntesdam.com/subject/sistemas/topic/ejercicios-sistemas

Por la entrada llega una lista de nombres de fichero (uno por línea). Muestra cuántos hay de cada extensión, en minúsculas y ordenadas alfabéticamente, con el formato «ext: N». Los ficheros sin extensión cuentan como «(sin extensión)» y se muestran al final.

Código de partida (bash)

```
#!/bin/bash
# Entrada: un nombre de fichero por línea

# TODO
```

Ejemplo: Varias extensiones

ENTRADA

```
informe.PDF
foto.jpg
notas.txt
FOTO2.JPG
Makefile
script.sh
factura.pdf
```

SALIDA ESPERADA

```
jpg: 2
pdf: 2
sh: 1
txt: 1
(sin extensión): 1
```

40. Validar direcciones IPv4

Difícil · Ejercicios de Sistemas Informáticos · apuntesdam.com/subject/sistemas/topic/ejercicios-sistemas

Por la entrada llega una dirección por línea. Para cada una muestra «IP válida» o «IP no válida». Es válida si tiene cuatro números separados por puntos, cada uno entre 0 y 255 y sin ceros a la izquierda (salvo el propio 0).

Código de partida (bash)

```
#!/bin/bash
while read -r ip || [ -n "$ip" ]; do
    # TODO: comprueba la IP
    echo "IP válida"
done
```

Ejemplo: Casos variados

ENTRADA

```
192.168.1.10
256.1.1.1
10.0.0
0.0.0.0
192.168.01.1
hola
```

SALIDA ESPERADA

```
IP válida
IP no válida
IP no válida
IP válida
IP no válida
IP no válida
```

41. Red y broadcast con aritmética de bits

Difícil · Ejercicios de Sistemas Informáticos · apuntesdam.com/subject/sistemas/topic/ejercicios-sistemas

Por la entrada llega una dirección con prefijo, por ejemplo 192.168.1.130/26. Calcula con bash (sin herramientas externas) y muestra:

Red: 192.168.1.128

Broadcast: 192.168.1.191

Hosts: 62

Código de partida (bash)

```
#!/bin/bash
read -r entrada
ip=${entrada%/*}
prefijo=${entrada#*/}

# TODO: calcula red, broadcast y hosts
```

Ejemplo: /26

ENTRADA

```
192.168.1.130/26
```

SALIDA ESPERADA

```
Red: 192.168.1.128
Broadcast: 192.168.1.191
Hosts: 62
```

Ejemplo: /20

ENTRADA

```
172.16.37.10/20
```

SALIDA ESPERADA

```
Red: 172.16.32.0
Broadcast: 172.16.47.255
Hosts: 4094
```

Ejercicios largos

42. Planificador de procesos: FCFS, SJF, SRTF y Round Robin

Muy difícil · Python · 100 minutos · apuntesdam.com/ejercicios/python/planificador-de-procesos

Un ordenador ejecuta muchos más procesos que núcleos tiene, así que el planificador del sistema operativo decide en cada momento cuál usa la CPU. La decisión cambia mucho el tiempo que espera cada programa: es uno de los temas que más cae en los exámenes de Sistemas, siempre con el mismo tipo de ejercicio: dibujar el diagrama de Gantt y calcular tiempos.

En este ejercicio vas a escribir ese planificador. Los procesos llegan en distintos instantes con una ráfaga de CPU (lo que necesitan ejecutar), y el programa simula cuatro algoritmos clásicos: FCFS (por orden de llegada), SJF (el de ráfaga más corta entre los que esperan, sin interrumpirlo), SRTF (el de menor tiempo restante, interrumpiendo si llega uno más corto) y Round Robin (turnos de un quantum fijo).

El resultado de cada proceso se mide con dos tiempos: el de retorno (desde que llega hasta que termina) y el de espera (el retorno menos lo que de verdad ha usado la CPU). En la web tienes también el simulador visual de planificación para comprobar tus resultados.

Requisitos

- La primera línea es el algoritmo: `FCFS`, `SJF`, `SRTF` o `RR q` (con `q`, el quantum, un entero mayor que 0). Si no es válida, escribe `Algoritmo no válido` y nada más. Cada línea siguiente es un proceso: `nombre llegada ráfaga`, enteros con la ráfaga mayor que 0. Una línea con otro formato o un nombre repetido escribe `Aviso: línea N ignorada`. Si no queda ningún proceso, `No hay procesos`.
- Desempates: FCFS y Round Robin atienden por orden de llegada y, a igual llegada, por el orden de la entrada. SJF elige la menor ráfaga y SRTF el menor tiempo restante; si empatan, el que llegó antes y luego el de la entrada. En Round Robin, cuando a un proceso se le acaba el quantum, los procesos que han llegado hasta ese instante entran en la cola antes que él.
- SRTF reevalúa cada vez que llega un proceso: si el recién llegado necesita menos que lo que le queda al que está en la CPU, lo expulsa. Si nadie está listo, la CPU queda ociosa hasta la siguiente llegada.
- Escribe `Algoritmo: ...` con su nombre completo (el de la tabla; en Round Robin, `Round Robin con quantum q`) y `Diagrama: | P1 0-3 | P2 3-7 | ... |`: cada tramo con el proceso, el inicio y el fin, `--` para la CPU ociosa, y los tramos seguidos del mismo proceso unidos en uno solo.
- Una tabla con la cabecera `Proceso`, `Llegada`, `Ráfaga`, `Fin`, `Retorno`, `Espera` y una fila por proceso en el orden de la entrada: el nombre en 8 caracteres a la izquierda y los números alineados a la derecha en 8, 8, 6, 9 y 8 caracteres.
- Para terminar, `Media de retorno: X`, `Media de espera: Y` y `Uso de la CPU: Z %` (tiempo ocupado entre el instante final), con dos decimales y coma decimal.

Formato de la entrada

- Línea 1: `FCFS`, `SJF`, `SRTF` o `RR q`. Resto: `nombre llegada ráfaga`.

Nombre de cada algoritmo en la salida

Entrada	Algoritmo
FCFS	FCFS (primero en llegar, primero en ser servido)
SJF	SJF (el más corto primero, sin expulsión)
SRTF	SRTF (menor tiempo restante, con expulsión)
RR q	Round Robin con quantum q

Ejemplo: Round Robin con quantum 2

ENTRADA

```
RR 2
P1 0 5
P2 1 3
P3 2 8
P4 3 6
```

SALIDA ESPERADA

Algoritmo: Round Robin con quantum 2
Diagrama: | P1 0-2 | P2 2-4 | P3 4-6 | P1 6-8 | P4 8-10 | P2 10-11 | P3 11-13 | P1 13-14 | P4 14-16 | P3 16-18 | P4 18-20 | P3 20-22 |
Proceso Llegada Ráfaga Fin Retorno Espera
P1 0 5 14 14 9
P2 1 3 11 10 7
P3 2 8 22 20 12
P4 3 6 20 17 11
Media de retorno: 15,25 · Media de espera: 9,75
Uso de la CPU: 100,00 %

Ejemplo: SRTF con CPU ociosa

ENTRADA

```
SRTF
A 0 7
B 2 4
C 4 1
D 5 4
E 30 2
F x 3
A 1 1
```

SALIDA ESPERADA

Aviso: línea 7 ignorada
Aviso: línea 8 ignorada
Algoritmo: SRTF (menor tiempo restante, con expulsión)
Diagrama: | A 0-2 | B 2-4 | C 4-5 | B 5-7 | D 7-11 | A 11-16 | -- 16-30 | E 30-32 |
Proceso Llegada Ráfaga Fin Retorno Espera
A 0 7 16 16 9
B 2 4 7 5 1
C 4 1 5 1 0
D 5 4 11 6 2
E 30 2 32 2 0
Media de retorno: 6,00 · Media de espera: 2,40
Uso de la CPU: 56,25 %

Código de partida (python)

```
import sys

NOMBRES = {"FCFS": "FCFS (primero en llegar, primero en ser servido)", "SJF": "SJF (el más corto primero, sin expulsión)",
           "SRTF": "SRTF (menor tiempo restante, con expulsión)", "RR": "Round Robin"}

def coma(x):
    return f"{x:.2f}".replace(".", ",")

def simular(algoritmo, quantum, procesos):
    """Devuelve el diagrama de Gantt como lista de [nombre, inicio, fin] y el instante en que termina cada proceso."""
    # TODO
    return [], {}

def main():
    lineas = sys.stdin.read().split("\n")
    # TODO: lee el algoritmo y los procesos, simula y escribe el diagrama, la tabla y las medias

main()
```

43. Plan de direccionamiento con subredes VLSM

Difícil · Python · 70 minutos · apuntesdam.com/ejercicios/python/subredes-vlsm

Una empresa tiene asignada la red 192.168.10.0/24 y quiere dividirla entre sus departamentos. Con subredes del mismo tamaño se desperdician direcciones (el enlace entre dos routers no necesita 62 direcciones), así que se usa VLSM: cada subred recibe la máscara más ajustada a los equipos que va a tener.

El método que se enseña (y el que sigue este programa) es asignar primero las subredes más grandes: así cada bloque empieza en una dirección múltiplo de su tamaño y las subredes nunca se solapan. De cada subred hay que dar la dirección de red, la máscara, la primera y la última dirección que se pueden asignar a equipos y la de difusión.

Hacerlo a mano es lento y es fácil equivocarse con la última dirección o la de difusión. Una dirección IP no es más que un número de 32 bits escrito en cuatro bytes: con ese número, las cuentas son sumas y operaciones de bits.

Requisitos

- La primera línea es la red, `IP/prefijo`, con el prefijo entre 8 y 30. Si no tiene ese formato o la IP no es válida, escribe `Red no válida: texto`. Si la IP no es la dirección de red de ese prefijo (tiene bits de host a 1), escribe `IP no es una dirección de red (sería RED/prefijo)`. En los dos casos no se escribe nada más.
- Cada línea siguiente es una subred: un nombre (puede tener espacios) y el número de hosts que necesita, un entero mayor que 0. Una línea con otro formato escribe `Aviso: línea N ignorada`.
- Las subredes se asignan de mayor a menor número de hosts (si empatan, por orden de la entrada), una detrás de otra desde el principio de la red. Cada una recibe el prefijo más largo que tenga sitio para sus hosts más la dirección de red y la de difusión, y como mucho /30 (un enlace de 2 hosts es un /30).
- Para cada subred: `Nombre (H hosts) → RED/p`, y debajo, con dos espacios delante, `máscara M · hosts útiles U` y `primera A · última B · difusión D`. Si no cabe en lo que queda de la red, `Nombre (H hosts) → no cabe: necesita un /p y quedan N direcciones` y se sigue con la siguiente.
- Antes de las subredes, `Red: RED/prefijo (T direcciones)`. Al final, `Sin asignar: N de T direcciones, desde IP` con la primera dirección libre, o `Sin asignar: 0 de T direcciones` si no queda ninguna.

Formato de la entrada

- Línea 1: la red, `IP/prefijo`. Resto: una subred por línea, `nombre hosts`.

Tamaño de cada prefijo

Prefijo	Máscara	Direcciones	Hosts útiles
/30	255.255.255.252	4	2
/29	255.255.255.248	8	6
/28	255.255.255.240	16	14
/27	255.255.255.224	32	30
/26	255.255.255.192	64	62
/25	255.255.255.128	128	126
/24	255.255.255.0	256	254

Ejemplo: Una oficina con cinco subredes

ENTRADA

192.168.10.0/24
Ventas 50
Administración 20
Almacén 10
Enlace router 2
Dirección 20

SALIDA ESPERADA

Red: 192.168.10.0/24 (256 direcciones)
Ventas (50 hosts) → 192.168.10.0/26
máscara 255.255.255.192 · hosts útiles 62
primera 192.168.10.1 · última 192.168.10.62 ·
difusión 192.168.10.63
Administración (20 hosts) → 192.168.10.64/27
máscara 255.255.255.224 · hosts útiles 30
primera 192.168.10.65 · última 192.168.10.94 ·
difusión 192.168.10.95
Dirección (20 hosts) → 192.168.10.96/27
máscara 255.255.255.224 · hosts útiles 30
primera 192.168.10.97 · última 192.168.10.126 ·
difusión 192.168.10.127
Almacén (10 hosts) → 192.168.10.128/28
máscara 255.255.255.240 · hosts útiles 14
primera 192.168.10.129 · última 192.168.10.142 ·
difusión 192.168.10.143
Enlace router (2 hosts) → 192.168.10.144/30
máscara 255.255.255.252 · hosts útiles 2
primera 192.168.10.145 · última 192.168.10.146 ·
difusión 192.168.10.147
Sin asignar: 108 de 256 direcciones, desde
192.168.10.148

Ejemplo: Una IP que no es de red

ENTRADA

10.0.0.5/24
A 10

SALIDA ESPERADA

10.0.0.5 no es una dirección de red (sería
10.0.0.0/24)

```
import sys

def a_numero(ip):
    """«192.168.10.0» → entero de 32 bits; None si no es una IPv4."""
    # TODO
    return None

def a_ip(n):
    # TODO
    return ""

def main():
    lineas = sys.stdin.read().split("\n")
    # TODO: valida la red, lee las subredes y asígnalas con VLSM

main()
```

44. Detección de ataques de fuerza bruta en el registro de SSH

Difícil · Python · 70 minutos · apuntesdam.com/ejercicios/python/deteccion-de-ataques-ssh

Cualquier servidor con SSH abierto a Internet recibe cientos de intentos de entrada al día: programas que prueban contraseñas para root, admin o test desde máquinas comprometidas. Cada intento queda en el registro de autenticación (/var/log/auth.log en Debian y Ubuntu) con una línea como `Oct 3 10:15:01 web1 sshd[2231]: Failed password for root from 203.0.113.5 port 52314 ssh2`.

Herramientas como fail2ban leen ese registro y bloquean en el cortafuegos las IP que fallan demasiadas veces en poco tiempo. Vas a escribir su lógica: con la configuración típica, 5 fallos dentro de una ventana de 10 minutos bloquean la IP.

Lo más peligroso no son los fallos, sino un acceso correcto desde una IP que antes ha fallado: puede ser alguien que ha acertado la contraseña. Esos casos se marcan como alertas para revisarlos.

Requisitos

- Cada línea empieza con la fecha de syslog en sus primeros 15 caracteres (`Oct 3 10:15:01`, mes en inglés y sin año: se toma 2026). Las líneas vacías o con una fecha que no se puede leer se ignoran, igual que las que no son ni un fallo ni un acceso correcto.
- Un fallo contiene `Failed password for USUARIO from IP port` (o `for invalid user USUARIO`, si el usuario no existe en el sistema). Un acceso correcto contiene `Accepted password for USUARIO from IP port` o `Accepted publickey for ...`.
- Una IP se bloquea en el instante de un fallo si, contando ese fallo, tiene al menos 5 fallos en los últimos 10 minutos (con una diferencia de 10 minutos exactos todavía cuenta). Cada IP se bloquea una sola vez, en el primer instante en que se cumple.
- Un acceso correcto desde una IP que ya ha tenido algún fallo (en cualquier momento anterior) genera la alerta `USUARIO entró desde IP a las HH:MM:SS tras N fallos desde esa IP` («fallo» si es 1).
- Informe: `Intentos fallidos: F · Accesos correctos: A ; Bloquear: con una línea IP a las HH:MM:SS (5 fallos desde las HH:MM:SS)` por IP, en el orden en que se bloquearon, indicando el primer fallo de la ventana (o `Bloquear: ninguna IP`); `Usuarios inexistentes probados: a, b` en orden alfabético (o `ninguno`); si ha habido fallos, `Usuarios más atacados: root (6), admin (3), ...` con los tres con más fallos (a igualdad, alfabético); y `Alertas: con las alertas en orden (o Alertas: ninguna)`.

Formato de la entrada

- Las líneas del registro de autenticación de SSH.

Ejemplo: Una mañana en el servidor web

ENTRADA

```
Oct  3 10:15:01 web1 sshd[2231]: Failed password
for root from 203.0.113.5 port 52314 ssh2
Oct  3 10:15:04 web1 sshd[2231]: Failed password
for invalid user admin from 203.0.113.5 port
52320 ssh2
Oct  3 10:15:09 web1 sshd[2232]: Failed password
for root from 203.0.113.5 port 52331 ssh2
Oct  3 10:16:10 web1 sshd[2240]: Accepted
password for ana from 198.51.100.7 port 50022
ssh2
Oct  3 10:16:12 web1 sshd[2241]: Failed password
for invalid user test from 203.0.113.5 port 52340
ssh2
Oct  3 10:16:30 web1 sshd[2242]: Accepted
publickey for luis from 192.0.2.10 port 40122
ssh2
Oct  3 10:17:02 web1 sshd[2243]: Failed password
for root from 203.0.113.5 port 52351 ssh2
Oct  3 10:20:45 web1 sshd[2250]: Failed password
for ana from 198.51.100.99 port 50100 ssh2
Oct  3 10:21:00 web1 sshd[2251]: Accepted
password for ana from 198.51.100.99 port 50101
ssh2
Oct  3 10:22:10 web1 CRON[2260]:
pam_unix(cron:session): session opened for user
root
```

SALIDA ESPERADA

Intentos fallidos: 6 · Accesos correctos: 3
Bloquear:
203.0.113.5 a las 10:17:02 (5 fallos desde las 10:15:01)
Usuarios inexistentes probados: admin, test
Usuarios más atacados: root (3), admin (1), ana (1)
Alertas:
ana entró desde 198.51.100.99 a las 10:21:00 tras 1 fallo desde esa IP

Ejemplo: La ventana de 10 minutos

ENTRADA

```
Oct  3 22:00:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct  3 22:03:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct  3 22:06:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct  3 22:09:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct  3 22:12:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct  3 22:13:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
línea basura
```

SALIDA ESPERADA

Intentos fallidos: 6 · Accesos correctos: 0
Bloquear:
10.0.0.9 a las 22:13:00 (5 fallos desde las 22:03:00)
Usuarios inexistentes probados: ninguno
Usuarios más atacados: root (6)
Alertas: ninguna

Código de partida (python)

```
import re
import sys
from datetime import datetime, timedelta

FALLO = re.compile(r"Failed password for (invalid user )?(\\S+) from (\\S+) port")
EXITO = re.compile(r"Accepted (password|publickey) for (\\S+) from (\\S+) port")
VENTANA = timedelta(minutes=10)
MAX_FALLOS = 5

def main():
    # TODO: recorre el registro, detecta los ataques y escribe el informe
    pass

main()
```

45. Informe de un registro de accesos web

Difícil · Bash · 75 minutos · apuntesdam.com/ejercicios/bash/informe-registro-accesos-web

Los servidores web como Apache y Nginx anotan cada petición en un registro de accesos (`access.log`). Cada línea dice quién pidió qué, cuándo y cómo respondió el servidor, en un formato estándar llamado Common Log Format.

El administrador de una web pequeña quiere un resumen diario sin instalar herramientas de análisis: un script de Bash que lea el registro por la entrada estándar (`./informe.sh < access.log`) y escriba las cifras importantes.

Un registro real puede tener líneas cortadas (por ejemplo, si el disco se llenó a mitad de una escritura). El script no debe contarlas como peticiones, pero sí decir cuántas había.

Requisitos

- Cada línea tiene la forma `IP - - [fecha zona] "MÉTODO ruta PROTOCOLO" código bytes` . Separada por espacios, la fecha ocupa dos campos y la petición tres; el código es el 9.º campo y los bytes el 10.º. Los bytes valen `-` cuando la respuesta no tiene cuerpo (por ejemplo, una redirección): cuentan como 0.
- Las líneas vacías se ignoran. Una línea no es válida si no llega al campo de los bytes o si el código no es un número de tres cifras.
- Si no hay ninguna petición válida, escribe solo `No hay peticiones válidas.`
- Escribe `Peticiones: N` , `Visitantes distintos: N` (el número de IP distintas), `Respuestas: 2xx A · 3xx B · 4xx C · 5xx D` (cuántas respuestas hay de cada familia, según la primera cifra del código) y `Datos enviados: N KB` (la suma de los bytes dividida entre 1024, sin decimales).
- Bajo `IPs con más peticiones:` , las 3 IP con más peticiones (o menos, si no hay tantas), de más a menos: el número en 3 caracteres alineado a la derecha, un espacio y la IP. A igual número, por orden de texto de la IP, carácter a carácter (por eso `10.0.0.10` va antes que `10.0.0.2`).
- `Página más pedida: ruta (N veces)` , con «vez» si es 1. Si hay empate, la primera en orden de texto.
- `Errores 404:` y debajo, con dos espacios delante, cada ruta distinta que haya respondido 404, en orden de texto. Si no hay ninguna, `Errores 404: ninguno` en una sola línea.
- Si ha habido líneas no válidas, termina con `Líneas no válidas: N` .

Formato de la entrada

- El registro de accesos por la entrada estándar, una petición por línea en Common Log Format.

Campos de una línea del registro

Campo	Ejemplo	Significado
1	192.168.1.10	IP del cliente
2 y 3	--	identidad y usuario (casi siempre vacíos)
4 y 5	[03/Oct/2026:10:15:32 +0200]	fecha, hora y zona horaria
6, 7 y 8	"GET /index.html HTTP/1.1"	método, ruta y protocolo
9	200	código de respuesta
10	5120	bytes enviados (- si no hay cuerpo)

Ejemplo: Diez peticiones

ENTRADA

```
192.168.1.10 - - [03/Oct/2026:10:15:32 +0200]
"GET /index.html HTTP/1.1" 200 5120
192.168.1.10 - - [03/Oct/2026:10:15:33 +0200]
"GET /css/estilos.css HTTP/1.1" 200 2048
10.0.0.7 - - [03/Oct/2026:10:16:01 +0200] "GET
/index.html HTTP/1.1" 200 5120
10.0.0.7 - - [03/Oct/2026:10:16:02 +0200] "GET
/favicon.ico HTTP/1.1" 404 512
172.16.0.3 - - [03/Oct/2026:10:17:45 +0200] "POST
/login HTTP/1.1" 302 -
172.16.0.3 - - [03/Oct/2026:10:17:46 +0200] "GET
/panel HTTP/1.1" 200 8192
192.168.1.10 - - [03/Oct/2026:10:18:10 +0200]
"GET /admin HTTP/1.1" 404 512
10.0.0.7 - - [03/Oct/2026:10:19:00 +0200] "GET
/index.html HTTP/1.1" 200 5120
192.168.1.10 - - [03/Oct/2026:10:20:15 +0200]
"GET /api/datos HTTP/1.1" 500 128
8.8.4.4 - - [03/Oct/2026:10:21:30 +0200] "GET
/index.html HTTP/1.1" 200 5120
```

SALIDA ESPERADA

```
Peticiones: 10
Visitantes distintos: 4
Respuestas: 2xx 6 · 3xx 1 · 4xx 2 · 5xx 1
Datos enviados: 31 KB
IPs con más peticiones:
  4 192.168.1.10
  3 10.0.0.7
  2 172.16.0.3
Página más pedida: /index.html (4 veces)
Errores 404:
  /admin
  /favicon.ico
```

Ejemplo: Empates y líneas rotas

ENTRADA

```
10.0.0.2 - - [04/Oct/2026:09:00:00 +0200] "GET /b
HTTP/1.1" 200 1024
10.0.0.10 - - [04/Oct/2026:09:00:01 +0200] "GET
/a HTTP/1.1" 200 1024
línea rota
10.0.0.2 - - [04/Oct/2026:09:00:02 +0200] "GET /a
HTTP/1.1" 301 -
10.0.0.10 - - [04/Oct/2026:09:00:03 +0200] "GET
/b HTTP/1.1" 200 2048

10.0.0.3 - - [04/Oct/2026:09:00:04 +0200] "GET /c
HTTP/1.1" 200 1024
```

SALIDA ESPERADA

```
Peticiones: 5
Visitantes distintos: 3
Respuestas: 2xx 4 · 3xx 1 · 4xx 0 · 5xx 0
Datos enviados: 5 KB
IPs con más peticiones:
  2 10.0.0.10
  2 10.0.0.2
  1 10.0.0.3
Página más pedida: /a (2 veces)
Errores 404: ninguno
Líneas no válidas: 1
```

Código de partida (bash)

```
#!/bin/bash
# Orden de texto byte a byte: así sort y [[ < ]] ordenan igual en cualquier máquina
export LC_ALL=C

declare -A por_ip por_ruta rutas_404
total=0
bytes_total=0
invalidas=0
c2=0 c3=0 c4=0 c5=0

while read -r ip _ _ _ metodo ruta _ codigo bytes _; do
    [ -z "$ip" ] && continue
    # TODO: valida la línea y actualiza los contadores
done

# TODO: escribe el informe
```

46. Alta masiva de usuarios a partir de un CSV

Difícil · Bash · 70 minutos · apuntesdam.com/ejercicios/bash/alta-masiva-de-usuarios

Al empezar el curso, el administrador del instituto tiene que crear las cuentas de los alumnos nuevos en el servidor Linux. Secretaría le pasa un CSV con nombre, apellidos y grupo, y hacerlo a mano con `useradd` uno por uno no es una opción.

El login sigue la norma del centro: la inicial del nombre, el primer apellido y la inicial del segundo, en minúsculas y sin tildes ni eñes (Lucía Martín Ruiz → lmartinr). Si ya existe, se le añade un número (lmartinr2, lmartinr3...). Hay que tener en cuenta tanto las cuentas que ya hay en el sistema como las que se van creando en el propio fichero.

Un script que crea usuarios es peligroso de probar, así que primero se ejecuta en modo simulación: en lugar de crear nada, escribe las órdenes que ejecutaría. Así se pueden revisar antes de lanzarlas de verdad.

Requisitos

- La primera línea es `GRUPOS grupo1,grupo2,...` (si no, escribe `La primera línea debe ser GRUPOS grupo1,grupo2...` y termina). La segunda es `EXISTEN cuenta1,cuenta2,...` con las cuentas que ya hay en el sistema, que puede estar vacía tras la palabra (si no es esa línea, `Falta la línea EXISTEN` y termina).
- Cada línea siguiente es `nombre;apellido1;apellido2;grupo` (el segundo apellido puede estar vacío). Las líneas vacías se ignoran. Si falta el nombre, el primer apellido o el grupo, o hay más campos, escribe `# Línea N: faltan o sobran campos`; si el grupo no está en la lista, `# Línea N: grupo desconocido G` (N es el número de línea de la entrada).
- Login: inicial del nombre + primer apellido + inicial del segundo apellido, sin tildes (á é í ó ú ü), con la ñ como n, en minúsculas y quitando todo lo que no sea una letra de la a a la z. Si coincide con una cuenta existente o ya creada, se prueba añadiendo 2, 3... hasta encontrar uno libre.
- Por cada alta escribe `useradd -m -s /bin/bash -g grupo -c "Nombre Apellido1 Apellido2" login` (el comentario con el nombre completo tal como viene, sin segundo apellido si está vacío) y `chage -d 0 login` (para obligar a cambiar la contraseña en el primer inicio de sesión).
- Al final, `# Altas: N (grupo1 n1, grupo2 n2, ...)` con todos los grupos en el orden de la línea `GRUPOS`.

Formato de la entrada

- Línea 1: `GRUPOS lista`. Línea 2: `EXISTEN lista`. Resto: `nombre;apellido1;apellido2;grupo`.

Ejemplo: Alumnos y profesores nuevos

ENTRADA

```
GRUPOS alumnos,profesores,administracion
EXISTEN root,ana,lmartinr
Lucía;Martín;Ruiz;alumnos
Luis;Martín;Rodríguez;alumnos
Íñigo;Núñez;Ibáñez;profesores
María José;O'Donnell;;alumnos
Pedro;Gil;López;conserjes
Sofía;Ruiz
```

SALIDA ESPERADA

```
useradd -m -s /bin/bash -g alumnos -c "Lucía
Martín Ruiz" lmartinr2
chage -d 0 lmartinr2
useradd -m -s /bin/bash -g alumnos -c "Luis
Martín Rodríguez" lmartinr3
chage -d 0 lmartinr3
useradd -m -s /bin/bash -g profesores -c "Íñigo
Núñez Ibáñez" inunezi
chage -d 0 inunezi
useradd -m -s /bin/bash -g alumnos -c "María José
O'Donnell" modonnell
chage -d 0 modonnell
# Línea 7: grupo desconocido conserjes
# Línea 8: faltan o sobran campos
# Altas: 4 (alumnos 3, profesores 1,
administracion 0)
```

Ejemplo: Logins repetidos dentro del fichero

ENTRADA

```
GRUPOS aula1
EXISTEN
Ana;García;Pérez;aula1
Antonio;García;Pons;aula1
Alba;García;Puig;aula1

Ana;García;Pérez;aula1
```

SALIDA ESPERADA

```
useradd -m -s /bin/bash -g aula1 -c "Ana García
Pérez" agarciap
chage -d 0 agarciap
useradd -m -s /bin/bash -g aula1 -c "Antonio
García Pons" agarciap2
chage -d 0 agarciap2
useradd -m -s /bin/bash -g aula1 -c "Alba García
Puig" agarciap3
chage -d 0 agarciap3
useradd -m -s /bin/bash -g aula1 -c "Ana García
Pérez" agarciap4
chage -d 0 agarciap4
# Altas: 4 (aula1 4)
```

Código de partida (bash)

```
#!/bin/bash
# Genera las órdenes para dar de alta usuarios a partir de un CSV, sin ejecutarlas (modo simulación)
export LC_ALL=C

# «Lucía Martín Ruiz» → «lmartinr»: sin tildes ni eñes, en minúsculas y solo letras
normalizar() {
    local s=$1
    # TODO
    echo "$s"
}

read -r palabra lista_grupos
# TODO: comprueba la línea GRUPOS, lee la línea EXISTEN y procesa cada usuario
```

47. Rotación de copias de seguridad abuelo-padre-hijo

Difícil · Bash · 70 minutos · apuntesdam.com/ejercicios/bash/rotacion-de-copias-abuelo-padre-hijo

Un servidor hace cada noche una copia completa en un fichero `backup-AAAA-MM-DD.tar.gz`. Si no se borra ninguna, el disco se llena; si se guardan solo las últimas, no hay forma de volver a cómo estaba todo hace dos meses. La solución clásica es la rotación abuelo-padre-hijo (GFS): copias diarias (hijos) para la última semana, semanales (padres) para el último mes y mensuales (abuelos) para el último medio año.

En este servidor la política es: conservar las 7 copias más recientes; además, las hechas en domingo de los últimos 28 días; y además, las hechas el día 1 de mes de los últimos 6 meses (contando el actual). Una copia puede cumplir varias reglas a la vez.

El script recibe la fecha de hoy y el listado de ficheros (lo que daría `ls`) y escribe qué se conserva y por qué, y qué se borra. Así se puede revisar antes de borrar nada.

Requisitos

- La primera línea es `HOY AAAA-MM-DD` con una fecha que exista (si no, La primera línea debe ser `HOY AAAA-MM-DD` y termina). Cada línea siguiente es el nombre de un fichero; las vacías se ignoran.
- Un nombre que no sea exactamente `backup-AAAA-MM-DD.tar.gz` con una fecha que exista escribe `Ignorado (nombre no válido): nombre`; una copia con fecha posterior a hoy, `Ignorado (fecha futura): nombre`. Los avisos salen en el orden de la entrada. Si no queda ninguna copia válida, `No hay copias`.
- Las copias válidas se tratan de la más reciente a la más antigua. Motivos para conservar una copia: `diaria` si está entre las 7 más recientes; `semanal` si es de un domingo y han pasado menos de 28 días desde ella hasta hoy; `mensual` si es del día 1 y es del mes de hoy o de uno de los 5 anteriores.
- Para cada copia escribe `CONSERVAR nombre (motivos)` con los motivos separados por coma y espacio en el orden `diaria, semanal, mensual`, o `BORRAR nombre` (`BORRAR` seguido de cuatro espacios, para que los nombres queden alineados).
- Al final, `Se conservan C y se borran B`.

Formato de la entrada

- Línea 1: `HOY AAAA-MM-DD`. Resto: un nombre de fichero por línea.

Ejemplo: Seis meses de copias

ENTRADA

```
HOY 2026-10-04
backup-2026-10-03.tar.gz
backup-2026-10-02.tar.gz
backup-2026-10-01.tar.gz
backup-2026-09-30.tar.gz
backup-2026-09-29.tar.gz
backup-2026-09-28.tar.gz
backup-2026-09-27.tar.gz
backup-2026-09-26.tar.gz
backup-2026-09-25.tar.gz
backup-2026-09-20.tar.gz
backup-2026-09-13.tar.gz
backup-2026-09-06.tar.gz
backup-2026-09-01.tar.gz
backup-2026-08-30.tar.gz
backup-2026-08-01.tar.gz
backup-2026-07-01.tar.gz
backup-2026-06-01.tar.gz
backup-2026-05-01.tar.gz
backup-2026-04-01.tar.gz
```

SALIDA ESPERADA

```
CONSERVAR backup-2026-10-03.tar.gz (diaria)
CONSERVAR backup-2026-10-02.tar.gz (diaria)
CONSERVAR backup-2026-10-01.tar.gz (diaria,
mensual)
CONSERVAR backup-2026-09-30.tar.gz (diaria)
CONSERVAR backup-2026-09-29.tar.gz (diaria)
CONSERVAR backup-2026-09-28.tar.gz (diaria)
CONSERVAR backup-2026-09-27.tar.gz (diaria,
semanal)
BORRAR    backup-2026-09-26.tar.gz
BORRAR    backup-2026-09-25.tar.gz
CONSERVAR backup-2026-09-20.tar.gz (semanal)
CONSERVAR backup-2026-09-13.tar.gz (semanal)
BORRAR    backup-2026-09-06.tar.gz
CONSERVAR backup-2026-09-01.tar.gz (mensual)
BORRAR    backup-2026-08-30.tar.gz
CONSERVAR backup-2026-08-01.tar.gz (mensual)
CONSERVAR backup-2026-07-01.tar.gz (mensual)
CONSERVAR backup-2026-06-01.tar.gz (mensual)
CONSERVAR backup-2026-05-01.tar.gz (mensual)
BORRAR    backup-2026-04-01.tar.gz
Se conservan 14 y se borran 5
```

Ejemplo: Nombres no válidos y fechas futuras

ENTRADA

```
HOY 2026-03-01
backup-2026-02-29.tar.gz
backup-2026-02-28.tar.gz
copia-2026-02-27.tar.gz
backup-2026-03-02.tar.gz
backup-2026-03-01.tar.gz
backup-2025-11-01.tar.gz
backup-2025-09-01.tar.gz
```

SALIDA ESPERADA

```
Ignorado (nombre no válido): backup-2026-02-
29.tar.gz
Ignorado (nombre no válido): copia-2026-02-
27.tar.gz
Ignorado (fecha futura): backup-2026-03-02.tar.gz
CONSERVAR backup-2026-03-01.tar.gz (diaria,
semanal, mensual)
CONSERVAR backup-2026-02-28.tar.gz (diaria)
CONSERVAR backup-2025-11-01.tar.gz (diaria,
mensual)
CONSERVAR backup-2025-09-01.tar.gz (diaria)
Se conservan 4 y se borran 0
```

```
#!/bin/bash
# Política abuelo-padre-hijo: 7 diarias, domingos de las últimas 4 semanas y el día 1 de los últimos 6 meses
export LC_ALL=C

read -r palabra hoy
# TODO: valida la fecha de hoy, lee los nombres de las copias y decide cuáles se conservan
```

48. Monitor de espacio en disco con umbrales

Medio · Bash · 50 minutos · apuntesdam.com/ejercicios/bash/monitor-de-espacio-en-disco

Un disco lleno tumba bases de datos, impide escribir registros y deja servicios caídos. Por eso los administradores programan con cron un script que revisa el espacio cada pocos minutos y avisa antes de que sea tarde.

La información la da `df -P` (el formato POSIX, pensado para que lo lean los scripts): una línea por sistema de ficheros con el dispositivo, los bloques de 1 KiB, los usados, los disponibles, el porcentaje y el punto de montaje. Muchas líneas son sistemas de ficheros virtuales (tmpfs, devtmpfs, overlay de Docker, squashfs y /dev/loop de los snaps) que no ocupan disco y no hay que vigilar.

Bash solo sabe hacer cuentas con enteros, así que los GiB con un decimal y los redondeos hay que construirlos a mano: es un buen ejercicio de aritmética entera.

Requisitos

- La primera línea es `UMBRALES aviso crítico` con dos enteros, aviso menor que crítico y crítico como mucho 100 (si no, escribe `La primera línea debe ser UMBRALES aviso crítico (aviso < crítico <= 100)` y termina). La segunda es la cabecera de `df`, que se ignora; después, las líneas de `df -P`.
- Se descartan los dispositivos `tmpfs`, `devtmpfs`, `overlay`, `squashfs`, `udev`, `none` y los que empiezan por `/dev/loop`. Si no queda ninguno, `No hay sistemas de ficheros que vigilar`.
- Estado según el porcentaje de la columna Capacity: `URGENTE` si es mayor o igual que el umbral crítico, `AVISO` si es mayor o igual que el de aviso, y `OK` si no.
- Una línea por sistema de ficheros, del porcentaje mayor al menor (a igualdad, por punto de montaje): el estado en 8 caracteres a la izquierda, un espacio, el porcentaje en 3 caracteres a la derecha seguido de `%`, dos espacios, el punto de montaje en 12 caracteres a la izquierda, un espacio y `usado U de T` (usado y usado + disponible). Si el estado no es OK, se añade `· libera N GiB para bajar del A %`: los GiB enteros (redondeando hacia arriba) que hay que borrar para que el uso baje hasta el umbral de aviso.
- Los tamaños se escriben en GiB (1 GiB = 1.048.576 KiB) con un decimal redondeado y coma: `49,7 GiB`. Al final, `Libre en total: X GiB` con la suma de lo disponible en los sistemas vigilados.

Formato de la entrada

- Línea 1: `UMBRALES aviso crítico`. Después, la salida completa de `df -P` (con su cabecera).

Ejemplo: Un servidor con dos discos llenándose

ENTRADA

```
UMBRALES 80 90
Filesystem      1024-blocks      Used Available Capacity Mounted on
udev            4006228              0    4006228      0% /dev
tmpfs           804792             1764    803028      1% /run
/dev/sda2       61255492  52113408    5998944     90% /
tmpfs          4023956              0    4023956      0% /dev/shm
/dev/sda1       523248             6220    517028      2% /boot/efi
/dev/sdb1       976284628 801228314 175056314    83% /datos
/dev/loop0      65536             65536          0    100% /snap/core20/2318
overlay        61255492  52113408    5998944     90% /var/lib/docker/overlay2/x/merged
```

SALIDA ESPERADA

```
URGENTE 90% / usado 49,7 GiB de
55,4 GiB · libera 6 GiB para bajar del 80 %
AVISO 83% /datos usado 764,1 GiB de
931,1 GiB · libera 20 GiB para bajar del 80 %
OK 2% /boot/efi usado 0,0 GiB de 0,5
GiB
Libre en total: 173,2 GiB
```

Ejemplo: Umbrales no válidos

ENTRADA

```
UMBRALES 95 90
Filesystem 1024-blocks Used Available Capacity
Mounted on
```

SALIDA ESPERADA

```
La primera línea debe ser UMBRALES aviso crítico
(aviso < crítico <= 100)
```

Código de partida (bash)

```
#!/bin/bash
# Lee la salida de «df -P» y avisa de los sistemas de ficheros que se llenan
export LC_ALL=C

read -r palabra aviso critico
# TODO: valida los umbrales, salta la cabecera y analiza cada línea de df
```

49. Analizador de la configuración de Nginx

Muy difícil · Bash · 90 minutos · apuntesdam.com/ejercicios/bash/analizador-de-configuracion-nginx

En un servidor con varias webs, la configuración de Nginx crece con el tiempo: un `server` para el dominio, otro para la API, otro que nadie recuerda... y los errores aparecen: un sitio que escucha en el 443 sin certificado (Nginx no arranca), un `autoindex on` olvidado que enseña los ficheros a cualquiera, o un dominio con HTTPS cuyo puerto 80 sigue sirviendo la web sin cifrar.

`nginx -t` comprueba la sintaxis, pero no esas buenas prácticas. Vas a escribir un script que lea un fichero de sitios (el que estaría en `/etc/nginx/sites-available/`), resuma cada bloque `server` y avise de los problemas.

El formato de Nginx es sencillo de leer línea a línea: directivas terminadas en `;` y bloques entre llaves (`server { ... }`), y dentro bloques `location ... { ... }`. Para saber dónde termina cada `server` hay que llevar la cuenta de las llaves abiertas.

Requisitos

- Se ignoran los comentarios (desde `#` hasta el final de la línea), los espacios de los extremos y las líneas vacías. Cada línea es `server {`, otra apertura de bloque (termina en `{`), un cierre `}` o una directiva terminada en `;`. Las líneas se numeran desde 1 contando todas.
- Errores, que detienen el script: `Error: llave de cierre sin abrir en la línea N`, `Error: bloque fuera de un server en la línea N`, `Error: directiva fuera de un server en la línea N` y, al final, `Error: llaves sin cerrar`. Si no hay ningún `server`, `No hay ningún bloque server`.

- De cada `server` se guardan (también desde dentro de sus `location`): los puertos de `listen` (el número tras los dos puntos, si los hay, y `ssl` si la línea lo lleva; sin repetir; 80 si no hay ningún `listen`), los nombres de `server_name`, el primer `root`, el primer `proxy_pass`, la URL del último `return` (lo que va tras el código), si tiene `ssl_certificate`, si tiene `ssl_certificate_key` y si tiene `autoindex on`.
- Por cada servidor: Servidor N (líneas A-B): puerto P · nombres · acción, con los puertos separados por `y` (443 `ssl` y 80), los nombres tal cual o (sin nombre), y como acción `redirige a URL`, `proxy a URL`, `sirve RUTA` o `no sirve nada`, por ese orden de preferencia.
- Avisos, primero los de cada servidor en orden (Servidor N: usa `ssl` pero le falta `ssl_certificate` y `ssl_certificate_key` —o solo el que falte—, Servidor N: `autoindex` activado (muestra la lista de ficheros), Servidor N: no tiene `root`, `return` ni `proxy_pass`) y después, por orden alfabético, dominio: aparece en K servidores del puerto P (para cada dominio y puerto con más de un servidor) y dominio: el puerto 80 no `redirige a HTTPS` (para cada dominio que se sirve en el 443 sin un servidor en el 80 cuyo `return` vaya a `https://`).
- Escribe Avisos: y cada aviso con `-` delante, o Sin avisos.

Formato de la entrada

- Un fichero de configuración de sitios de Nginx.

Ejemplo: Un dominio con API

ENTRADA

```
# Sitio principal
server {
    listen 80;
    listen [::]:80;
    server_name ejemplo.com www.ejemplo.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name ejemplo.com www.ejemplo.com;
    ssl_certificate
/etc/letsencrypt/live/ejemplo.com/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/ejemplo.com/privkey.pem;
    root /var/www/ejemplo;

    location /descargas/ {
        autoindex on; # se dejó activado tras
una prueba
    }
}

server {
    listen 443 ssl;
    server_name api.ejemplo.com;
    ssl_certificate /etc/ssl/api.crt;
    location / {
        proxy_pass http://127.0.0.1:3000;
    }
}
```

SALIDA ESPERADA

```
Servidor 1 (líneas 2-7): puerto 80 · ejemplo.com
www.ejemplo.com · redirige a
https://$host$request_uri
Servidor 2 (líneas 9-20): puerto 443 ssl ·
ejemplo.com www.ejemplo.com · sirve
/var/www/ejemplo
Servidor 3 (líneas 22-29): puerto 443 ssl ·
api.ejemplo.com · proxy a http://127.0.0.1:3000
Avisos:
- Servidor 2: autoindex activado (muestra la
lista de ficheros)
- Servidor 3: usa ssl pero le falta
ssl_certificate_key
- api.ejemplo.com: el puerto 80 no redirige a
HTTPS
```

Ejemplo: Servidores repetidos y uno vacío

ENTRADA

```
server {
    server_name intranet.local;
}
server {
    listen 8080;
    server_name intranet.local;
    root /srv/intranet;
}
server {
    listen 8080;
    server_name intranet.local;
    return 302 /mantenimiento.html;
}
```

SALIDA ESPERADA

Servidor 1 (líneas 1-3): puerto 80 · intranet.local · no sirve nada
Servidor 2 (líneas 4-8): puerto 8080 · intranet.local · sirve /srv/intranet
Servidor 3 (líneas 9-13): puerto 8080 · intranet.local · redirige a /mantenimiento.html
Avisos:
- Servidor 1: no tiene root, return ni proxy_pass
- intranet.local: aparece en 2 servidores del puerto 8080

Código de partida (bash)

```
#!/bin/bash
# Revisa un fichero de sitios de Nginx (como /etc/nginx/sites-available/web) y avisa de los fallos típicos
export LC_ALL=C

n=0 profundidad=0 numero=0
declare -a inicio fin puertos nombres raiz retorno proxy cert clave autoindex

while IFS= read -r linea || [ -n "$linea" ]; do
    numero=$((numero + 1))
    # TODO: quita comentarios y espacios, sigue las llaves y guarda las directivas de cada server
done

# TODO: resumen de cada servidor y avisos
```

Soluciones

1. ¿Por qué mi disco de 1 TB tiene 931 GB?

Medio · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Solución (python)

```
import sys

DECIMAL = {"B": 1, "KB": 10**3, "MB": 10**6, "GB": 10**9, "TB": 10**12}
BINARIO = {"KiB": 2**10, "MiB": 2**20, "GiB": 2**30, "TiB": 2**40}
UNIDADES = {**DECIMAL, **BINARIO}

for linea in sys.stdin.read().splitlines():
    if not linea.strip():
        continue
    cantidad, unidad = linea.split()
    if unidad not in UNIDADES:
        print(f"{linea.strip()}: unidad desconocida")
        continue
    bytes_ = round(float(cantidad) * UNIDADES[unidad])
    print(f"{cantidad} {unidad} = {bytes_} bytes = {bytes_ / 10**9:.2f} GB = {bytes_ / 2**30:.2f} GiB")
```

Todo pasa por los bytes: primero se convierte la cantidad a bytes con el factor de su unidad y después se divide entre 10^9 (GB) y entre 2^{30} (GiB). Unir los dos diccionarios en uno ({**DECIMAL, **BINARIO}) permite buscar cualquier unidad de una vez, y que distinga mayúsculas hace que «Gb» (gigabits) se rechace.

Así se ve de dónde salen los «GB que faltan»: 10^{12} bytes son 1000 GB para el fabricante, pero 931,32 GiB para el sistema operativo, porque un GiB es casi un 7,4 % mayor que un GB.

2. Dimensionar la fuente de alimentación

Difícil · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Solución (python)

```
import sys

CATALOGO = [450, 550, 650, 750, 850, 1000, 1200]
EFICIENCIA = {"bronze": 0.85, "gold": 0.90, "platinum": 0.92}

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

consumo = 0
config = {}
for linea in lineas:
    if "=" in linea:
        clave, valor = linea.split("=")
        config[clave] = valor
    else:
        _, vatios = linea.rsplit(" ", 1)  # el nombre puede tener espacios: los vatios son lo último
        consumo += float(vatios)

margen = consumo * 1.3
fuente = next((f for f in CATALOGO if f >= margen), None)
pared = consumo * float(config["uso"]) / 100 / EFICIENCIA[config["certificacion"]]
kwh = pared * float(config["horas"]) * 365 / 1000

print(f"Consumo de los componentes: {consumo:g} W")
print(f"Con un margen del 30 %: {margen:.0f} W")
print(f"Fuente recomendada: {f'{fuente} W' if fuente else 'ninguna del catálogo es suficiente'}")
print(f"Consumo real de la pared: {pared:.1f} W")
print(f"Coste anual: {kwh * float(config['precio']):.2f} € ({kwh:.1f} kWh)")
```

Las líneas de componentes y las de configuración se distinguen por el «=». En los componentes, `rsplit(" ", 1)` parte solo por el último espacio, porque el nombre puede tener espacios y los vatios siempre van al final. `next(..., None)` devuelve la primera fuente del catálogo que llega al consumo con margen, o `None` si ninguna llega.

El margen se deja porque los componentes tienen picos de consumo y las fuentes rinden peor cerca de su máximo. El consumo real es menor que el máximo (el equipo no siempre va a tope), pero de la pared sale más de lo que llega a los componentes: se divide entre la eficiencia, y la diferencia se pierde en calor.

3. Ancho de banda y cuellos de botella

Muy difícil · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Solución (python)

```
import sys

PCIE_GBS_POR_LINEA = {"3": 0.985, "4": 1.969, "5": 3.938}

for linea in sys.stdin.read().splitlines():
    p = linea.split()
    if not p:
        continue
    if p[0] == "ram":
        mts = int(p[1].split("-")[1])
        canales = int(p[2])
        gbs = mts * 8 * canales / 1000
        print(f"Memoria {p[1]} en {canales} {'canal' if canales == 1 else 'canales'}: {gbs:.1f} GB/s")
    elif p[0] == "ssd":
        ssd = int(p[1])
        gen = p[2].replace("pcie", "")
        lineas = int(p[3].replace("x", ""))
        enlace_gbs = PCIE_GBS_POR_LINEA[gen] * lineas
        enlace_mbs = round(enlace_gbs * 1000)
        texto = f"SSD de {ssd} MB/s en PCIe {gen}.0 x{lineas} ({enlace_gbs:.1f} GB/s)"
        if enlace_mbs < ssd:
            print(f"{texto}: limitado a {enlace_mbs} MB/s ({ssd - enlace_mbs} MB/s desaprovechados)")
        else:
            print(f"{texto}: sin cuello de botella")
```

Cada línea dice primero qué componente es, y el resto se interpreta según el tipo. En la memoria, los MT/s son transferencias por segundo y cada una mueve 8 bytes por canal, así que doble canal duplica el ancho de banda. En el SSD, el enlace es lo que transporta una línea PCIe por el número de líneas.

El cuello de botella aparece cuando el enlace es más lento que el SSD: un SSD de 7400 MB/s en una ranura PCIe 3.0 x4 nunca pasará de unos 3940 MB/s. Se compara todo en MB/s, redondeando el enlace a un número entero.

4. Planificador FCFS y SJF

Medio · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
algoritmo = lineas[0][0]
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

t = 0
gantt = []
inicio, fin = {}, {}
pendientes = list(procesos)
while pendientes:
    llegados = [p for p in pendientes if p[1] <= t]
    if not llegados:
        siguiente = min(p[1] for p in pendientes)
        gantt.append(f"ocioso {t}-{siguiente}")
        t = siguiente
        continue
    if algoritmo == "FCFS":
        p = min(llegados, key=lambda p: p[1])
    else:
        p = min(llegados, key=lambda p: (p[2], p[1], p[0]))
    pendientes.remove(p)
    inicio[p[0]] = t
    t += p[2]
    fin[p[0]] = t
    gantt.append(f"{p[0]} {inicio[p[0]]}-{t}")

print(" | ".join(gantt))
esperas, retornos = [], []
for nombre, llegada, _ in procesos:
    esperas.append(inicio[nombre] - llegada)
    retornos.append(fin[nombre] - llegada)
    print(f"{nombre}: espera {esperas[-1]}, retorno {retornos[-1]}")
print(f"Espera media: {sum(esperas) / len(esperas):.2f}")
print(f"Retorno medio: {sum(retornos) / len(retornos):.2f}")
```

La simulación avanza un reloj `t`. Cada vez que la CPU queda libre se miran los procesos que ya han llegado y se elige uno con `min` y una clave: en FCFS, la llegada; en SJF, la tupla (ráfaga, llegada, nombre), que resuelve los empates en ese orden porque las tuplas se comparan elemento a elemento. Si no ha llegado nadie, se añade un tramo ocioso hasta la siguiente llegada.

Como son no expropiativos, el elegido se ejecuta entero y basta con apuntar cuándo empieza y termina; de ahí salen la espera (inicio – llegada) y el retorno (fin – llegada). SJF da una espera media menor que FCFS porque los procesos cortos no se quedan atascados detrás de uno largo.

5. Round Robin con quantum

Difícil · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

Solución (python)

```
import sys
from collections import deque

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
quantum = int(lineas[0][1])
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

llegadas = sorted(procesos, key=lambda p: p[1]) # sorted es estable: empates en orden de entrada
resto = {nombre: rafaga for nombre, _, rafaga in procesos}
fin = {}
cola = deque()
tramos = []
t = i = 0

def admitir():
    """Mete en la cola a los procesos que ya han llegado en el instante t."""
    global i
    while i < len(llegadas) and llegadas[i][1] <= t:
        cola.append(llegadas[i][0])
        i += 1

while i < len(llegadas) or cola:
    admitir()
    if not cola:
        # CPU ociosa hasta la siguiente llegada
        tramos.append(("ocioso", t, llegadas[i][1]))
        t = llegadas[i][1]
        continue
    nombre = cola.popleft()
    duracion = min(quantum, resto[nombre])
    tramos.append((nombre, t, t + duracion))
    t += duracion
    resto[nombre] -= duracion
    admitir() # los que llegan durante el tramo van antes...
    if resto[nombre] > 0:
        cola.append(nombre) # ...que el proceso expulsado
    else:
        fin[nombre] = t

print(" | ".join(f"{n} {a}-{b}" for n, a, b in tramos))
esperas, retornos = [], []
for nombre, llegada, rafaga in procesos:
    retornos.append(fin[nombre] - llegada)
    esperas.append(retornos[-1] - rafaga)
    print(f"{nombre}: espera {esperas[-1]}, retorno {retornos[-1]}")
print(f"Espera media: {sum(esperas) / len(esperas):.2f}")
print(f"Retorno medio: {sum(retornos) / len(retornos):.2f}")
print(f"Cambios de contexto: {sum(1 for a, b in zip(tramos, tramos[1:]) if a[0] != b[0])}")
```

Round Robin es una cola: el proceso del frente se ejecuta como mucho un quantum y, si le queda ráfaga, vuelve al final. El orden de las dos operaciones tras cada tramo es lo que decide el resultado: primero se admiten los procesos que han llegado durante el tramo (función `admitir`) y después se reencola el expulsado.

En Round Robin un proceso se ejecuta a trozos, así que la espera no es «inicio – llegada»: se calcula como el tiempo total en el sistema menos lo que de verdad usó la CPU. Los cambios de contexto se cuentan comparando cada tramo con el siguiente con `zip(tramos, tramos[1:])`; con un quantum pequeño hay muchos, y cada uno tiene un coste real para el sistema operativo.

6. Reemplazo de páginas: FIFO, LRU y óptimo

Muy difícil · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

Solución (python)

```
import sys
from collections import deque

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
marcos = int(lineas[0][1])
referencias = [int(x) for x in lineas[1]]

def fifo(refs):
    memoria, fallos = deque(), 0
    for p in refs:
        if p not in memoria:
            fallos += 1
            if len(memoria) == marcos:
                memoria.popleft()          # sale la que lleva más tiempo cargada
            memoria.append(p)
    return fallos

def lru(refs):
    memoria, fallos = [], 0                # de la menos a la más recientemente usada
    for p in refs:
        if p in memoria:
            memoria.remove(p)
        else:
            fallos += 1
            if len(memoria) == marcos:
                memoria.pop(0)             # sale la que hace más tiempo que no se usa
            memoria.append(p)
    return fallos

def optimo(refs):
    memoria, fallos = [], 0                # en orden de carga
    for i, p in enumerate(refs):
        if p in memoria:
            continue
        fallos += 1
        if len(memoria) == marcos:
            def proximo_uso(q):
                futuro = refs[i + 1:]
                return futuro.index(q) if q in futuro else float("inf")
            # la que tardará más en usarse; en empate (no vuelven), la cargada antes: max devuelve la primera
            memoria.remove(max(memoria, key=proximo_uso))
        memoria.append(p)
    return fallos

total = len(referencias)
for nombre, algoritmo in (("FIFO", fifo), ("LRU", lru), ("Óptimo", optimo)):
    f = algoritmo(referencias)
    print(f"{nombre}: {f} fallos ({f / total * 100:.1f} %)")
```

Los tres algoritmos solo se diferencian en cómo eligen la víctima cuando la memoria está llena. FIFO usa una cola (sale la primera que entró); LRU mantiene la lista ordenada por uso, moviendo al final cada página que se usa; el óptimo mira el futuro y expulsa la página cuyo próximo uso está más lejos (infinito si no vuelve a usarse).

El óptimo no se puede implementar en un sistema real, porque nadie conoce las referencias futuras, pero sirve de referencia para saber cuánto se aleja un algoritmo del mejor posible. La cadena 1 2 3 4 1 2 5... muestra la anomalía de Bélády: con FIFO, pasar de 3 a 4 marcos aumenta los fallos (de 9 a 10), algo que nunca le pasa a LRU.

7. Un plan de particiones alineado

Medio · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
disco_mib = int(lineas[0][1].replace("GiB", "")) * 1024
ultimo = disco_mib - 1 # el último MiB es para la copia de la tabla GPT

def mib(tamano):
    if tamano.endswith("GiB"):
        return round(float(tamano[:-3]) * 1024)
    return int(tamano[:-3]) # «...MiB»

pos = 1 # el primer MiB es para la tabla y la alineación
for nombre, tamano in lineas[1:]:
    tam = ultimo - pos if tamano == "resto" else mib(tamano)
    if pos + tam > ultimo:
        print(f"ERROR: no cabe {nombre} (faltan {pos + tam - ultimo} MiB)")
        break
    print(f"{nombre}: inicio {pos} MiB, fin {pos + tam} MiB, tamaño {tam / 1024:.2f} GiB")
    pos += tam
else:
    print(f"Sin asignar: {(ultimo - pos) / 1024:.2f} GiB")
```

Trabajar en MiB enteros garantiza la alineación: cada partición empieza justo donde acaba la anterior, siempre en un múltiplo de 1 MiB, que es lo que hacen los instaladores para que las particiones coincidan con los bloques físicos del disco (sobre todo en SSD). El espacio útil va del MiB 1 al penúltimo, porque GPT reserva el principio y el final del disco.

«resto» ocupa lo que quede hasta el final útil. El `else` del `for` solo se ejecuta si el bucle termina sin `break`: así el resumen de espacio sin asignar no aparece cuando una partición no cabe.

8. ¿Arrancará este esquema de particiones?

Difícil · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
config = {l[0]: l[1] for l in lineas[:3]}          # firmware, tabla, disco
particiones = lineas[3:]                          # nombre clase sistema tamaño_GiB punto_de_montaje

def num(x):
    return f"{x:g}"                               # 109.0 → «109», 120.5 → «120.5»

uefi = config["firmware"] == "uefi"
mbr = config["tabla"] == "mbr"
disco = float(config["disco"])
clases = [p[1] for p in particiones]
problemas = []

if uefi and config["tabla"] != "gpt":
    problemas.append("UEFI necesita una tabla GPT (o activar el modo CSM)")
if mbr:
    if disco > 2048:
        problemas.append(f"MBR solo puede usar 2048 GiB de los {num(disco)} GiB del disco")
    if clases.count("primaria") + clases.count("extendida") > 4:
        problemas.append("MBR admite como máximo 4 particiones primarias o extendidas")
    if clases.count("extendida") > 1:
        problemas.append("Solo puede haber una partición extendida")
    if "logica" in clases and "extendida" not in clases:
        problemas.append("Las particiones lógicas necesitan una partición extendida")
elif "extendida" in clases or "logica" in clases:
    problemas.append("GPT no tiene particiones extendidas ni lógicas")
if uefi and not any(p[2] == "fat32" and p[4] == "/boot/efi" for p in particiones):
    problemas.append("Falta la partición EFI (FAT32 montada en /boot/efi)")
if not any(p[4] == "/" for p in particiones):
    problemas.append("Falta la partición raíz (/)")
ocupado = sum(float(p[3]) for p in particiones if p[1] != "extendida") # la extendida contiene a las lógicas
if ocupado > disco:
    problemas.append(f"Las particiones ocupan {num(ocupado)} GiB y el disco tiene {num(disco)} GiB")

print("\n".join(problemas) if problemas else "Esquema válido")
```

Las reglas se comprueban en orden y cada una que falla añade su mensaje a una lista; al final se muestran todos o «Esquema válido». Contar las clases con `clases.count(...)` resuelve casi todas las reglas de MBR: el límite de 4 entradas primarias o extendidas, una sola extendida y que las lógicas necesiten una extendida que las contenga.

En el espacio ocupado no se suma la extendida, porque es solo un contenedor: su tamaño ya es el de las lógicas que lleva dentro. El formato `:g` muestra los números sin decimales innecesarios (109 en vez de 109.0).

9. Leer la tabla de particiones de un MBR

Muy difícil · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-

Solución (python)

```
import sys

datos = bytes.fromhex("".join(sys.stdin.read().split())) # las 4 entradas (64 bytes) y la firma (2 bytes)
TIPOS = {0x07: "NTFS/exFAT", 0x0B: "FAT32", 0x0C: "FAT32 LBA", 0x83: "Linux", 0x82: "Linux swap",
         0x05: "Extendida", 0x0F: "Extendida LBA", 0xEE: "GPT de protección", 0xEF: "Sistema EFI"}

for n in range(4):
    entrada = datos[n * 16:(n + 1) * 16]
    tipo = entrada[4]
    if tipo == 0:
        continue # entrada vacía
    activa = "activa, " if entrada[0] == 0x80 else ""
    inicio = int.from_bytes(entrada[8:12], "little")
    sectores = int.from_bytes(entrada[12:16], "little")
    gib = sectores * 512 / 2**30
    print(f"Partición {n + 1}: {activa}tipo {tipo:02X} ({TIPOS.get(tipo, 'desconocido')}), "
          f"sector inicial {inicio}, {sectores} sectores ({gib:.2f} GiB)")
    if tipo == 0xEE:
        print("El disco usa GPT: esta entrada solo protege el disco de herramientas antiguas")

firma = datos[64:66]
print("Firma: correcta (55 AA)" if firma == b"\x55\xaa" else f"Firma: incorrecta ({firma[0]:02X} {firma[1]:02X})")
```

Cada entrada ocupa 16 bytes, así que la entrada n está en `datos[n*16:(n+1)*16]`, y dentro de ella cada campo tiene una posición fija. Indexar un objeto `bytes` da directamente el número (0x80, el tipo...), y `int.from_bytes(..., "little")` monta los enteros de 4 bytes leyendo el byte menos significativo primero, como lo guarda el PC.

Con 4 bytes se pueden contar como mucho $2^{32} - 1$ sectores de 512 bytes, unos 2 TiB: esa es la razón del límite de MBR, y por eso los discos GPT ponen una única entrada de tipo EE que ocupa todo lo posible, para que las herramientas antiguas vean el disco lleno y no lo toquen.

10. Capacidad y tolerancia de un RAID

Medio · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

Solución (python)

```
import sys

MINIMO = {"RAID0": 2, "RAID1": 2, "RAID5": 3, "RAID6": 4, "RAID10": 4}

for linea in sys.stdin.read().splitlines():
    if not linea.strip():
        continue
    nivel, n, tam = linea.split()
    n, tam = int(n), int(tam)
    if nivel not in MINIMO:
        print(f"{nivel}: nivel desconocido")
        continue
    if n < MINIMO[nivel] or (nivel == "RAID10" and n % 2):
        par = " y un número par" if nivel == "RAID10" else ""
        print(f"{nivel} con {n} discos: no válido (necesita al menos {MINIMO[nivel]} discos{par})")
        continue
    util = {"RAID0": n, "RAID1": 1, "RAID5": n - 1, "RAID6": n - 2, "RAID10": n // 2}[nivel] * tam
    if nivel == "RAID10":
        tolera = f"1 fallo por pareja (hasta {n // 2})"
    else:
        f = {"RAID0": 0, "RAID1": n - 1, "RAID5": 1, "RAID6": 2}[nivel]
        tolera = f"{f} {'fallo' if f == 1 else 'fallos'}"
    print(f"{nivel} con {n} x {tam} GB: {util} GB útiles ({util / (n * tam) * 100:.0f} %), tolera {tolera}")
```

Cada nivel se resume en dos números: cuántos discos de datos útiles quedan y cuántos fallos aguanta. Escribirlos en diccionarios indexados por el nivel evita una cadena de `if`. RAID5 dedica el equivalente a un disco a la paridad y RAID6 a dos, por eso toleran 1 y 2 fallos.

RAID10 es espejos (RAID1) unidos con bandas (RAID0): pierde la mitad de la capacidad y aguanta un fallo en cada pareja, pero si fallan los dos discos de la misma pareja se pierde todo. Ninguno sustituye a una copia de seguridad: un borrado accidental se replica en todos los discos al instante.

11. Simulador de estrategias de copia

Difícil · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
datos = float(lineas[0][1])
cambio = float(lineas[1][1])
plan = lineas[2][1:]          # C = completa, I = incremental, D = diferencial, un día cada una

total = 0
ultima_completa = None
validas = []                  # (día, tipo) de las copias que se han hecho
for dia, tipo in enumerate(plan, 1):
    if tipo != "C" and ultima_completa is None:
        print(f"Día {dia}: ERROR, una copia {tipo} necesita una completa anterior")
        continue
    if tipo == "C":
        tam = datos
        ultima_completa = dia
    elif tipo == "I":
        tam = datos * cambio / 100          # solo lo de un día
    else:
        tam = min(datos, datos * cambio / 100 * (dia - ultima_completa)) # todo desde la completa
    total += tam
    validas.append((dia, tipo))
    print(f"Día {dia} ({tipo}): {tam:.1f} GB")
print(f"Total ocupado: {total:.1f} GB")

# Restaurar: última completa, última diferencial posterior y las incrementales que vengan detrás
c = max(d for d, t in validas if t == "C")
difs = [d for d, t in validas if t == "D" and d > c]
desde = difs[-1] if difs else c
necesarias = [c] + difs[-1:] + [d for d, t in validas if t == "I" and d > desde]
print("Para restaurar el último día: " + " ".join(f"día {d}" for d in necesarias))
```

Las tres estrategias se diferencian en desde cuándo copian los cambios: la incremental, desde la copia anterior (siempre un día de cambios); la diferencial, desde la última completa (cada día un poco más, hasta el total). Por eso las incrementales ocupan poco pero hay que aplicarlas todas para restaurar, y la diferencial ocupa más pero basta con la última.

Para saber qué copias hacen falta se parte de la última completa, se añade la última diferencial posterior (que ya incluye todo lo cambiado hasta ella) y después las incrementales que vengan detrás. `difs[-1:]` es una lista con la última diferencial, o vacía si no hay ninguna.

12. ¿Es buena esta copia de seguridad?

Muy difícil · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

almacenamiento

Solución (python)

```
import sys

texto = sys.stdin.read().split("---")
original = [l.split() for l in texto[0].splitlines() if l.strip()] # ruta tamaño hash
copia = [l.split() for l in texto[1].splitlines() if l.strip()]

orig = {ruta: (int(tam), h) for ruta, tam, h in original}
cop = {ruta: (int(tam), h) for ruta, tam, h in copia}

movidos, faltan, distintos = [], [], []
sobran = [r for r in cop if r not in orig]
for ruta, (tam, h) in orig.items():
    if ruta in cop:
        if cop[ruta][1] != h:
            distintos.append(f"DISTINTO {ruta} ({tam} bytes en el original, {cop[ruta][0]} en la copia)")
            continue
        nueva = next((r for r in sobran if cop[r][1] == h), None) # ¿está en otra ruta con el mismo hash?
        if nueva:
            movidos.append(f"MOVIDO {ruta} -> {nueva}")
            sobran.remove(nueva)
        else:
            faltan.append(f"FALTA {ruta}")

for grupo in (movidos, faltan, distintos, [f"SOBRA {r}" for r in sobran]):
    for linea in sorted(grupo):
        print(linea)

correctos = sum(1 for r, (_, h) in orig.items() if r in cop and cop[r][1] == h)
mib = sum(t for t, _ in orig.values()) / 2**20
print(f"Correctos: {correctos} de {len(orig)} · Datos del original: {mib:.2f} MiB")
print("Copia VÁLIDA" if not faltan and not distintos else "Copia NO VÁLIDA")
```

Los dos manifiestos se convierten en diccionarios ruta → (tamaño, hash), y así comparar es buscar por clave. Lo que se compara es el hash, no el tamaño: un fichero puede tener el mismo tamaño y un contenido distinto (como enero.xlsx en el ejemplo), y solo el hash lo detecta.

Antes de declarar que un fichero falta se busca su hash entre los que sobran en la copia: si aparece, el fichero se ha movido o renombrado, pero sus datos están, y no invalida la copia. Al emparejarlo se quita de los sobrantes para que no salga dos veces.

13. Saludo con validación

Fácil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r nombre

if [ -z "$nombre" ]; then
    echo "Error: nombre vacío"
    exit 0
fi
echo "Hola, $nombre"
echo "Tu nombre tiene ${#nombre} letras"
```

[-z "\$nombre"] es cierto si la variable está vacía; las comillas son necesarias para que la comprobación funcione también con un nombre vacío o con espacios. \${#nombre} da la longitud de la variable sin necesitar ningún programa externo.

14. Tabla de multiplicar con bucle

Fácil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

operativos

Solución (bash)

```
#!/bin/bash
read -r n

if [[ ! $n =~ ^[0-9]+$ ]] || (( 10#$n == 0 )); then
    echo "Error: número no válido"
    exit 0
fi
for i in {1..10}; do
    echo "$n x $i = $(( 10#$n * i ))"
done
```

La expresión regular `^[0-9]+$` comprueba que la entrada sean solo cifras, lo que descarta texto, negativos y decimales; además se rechaza el 0, que no es positivo. `10#` obliga a leer el número en base 10, porque bash interpreta en octal los números que empiezan por 0.

`{1..10}` genera la secuencia del 1 al 10 y `$(( ))` hace la multiplicación.

15. Encuentra el fallo: el saludo que no se guarda

Fácil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r nombre
saludo="Hola, $nombre"
echo "$saludo"
```

En bash una asignación no puede llevar espacios alrededor del `=`. Con `saludo = "..."`, bash entiende que hay que ejecutar un programa llamado «saludo» con los argumentos «=» y el texto; como no existe, da un error y la variable se queda vacía.

16. Analizar un fichero de log

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
log=$(cat)

awk '
$3 == "ERROR" { errores++; porIp[$4]++ }
$3 == "WARN" { avisos++ }
END {
    for (ip in porIp)
        if (porIp[ip] > max || (porIp[ip] == max && ip < mejor)) { max = porIp[ip]; mejor = ip }
    print "Errores: " errores + 0
    print "Avisos: " avisos + 0
    print "IP con más errores: " mejor " (" max + 0 ")"
}' <<< "$log"
```

awk separa cada línea en campos por los espacios: el nivel es el tercero y la IP el cuarto. Un array asociativo `porIp` cuenta los errores de cada IP, y al final se recorre para quedarse con la que tiene más.

Sumar `+ 0` al imprimir convierte en 0 un contador que no se ha llegado a usar (si no, saldría vacío).

17. Usuarios del sistema desde /etc/passwd

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

operativos

Solución (bash)

```
#!/bin/bash
# La entrada estándar es el contenido de /etc/passwd

usuarios=$(awk -F: ' $3 >= 1000 && $3 != 65534 { printf "%d\t%s (%s) -> %s [%s]\n", $3, $1, $3, $6, $7 }'
| sort -n | cut -f2-)
[ -n "$usuarios" ] && echo "$usuarios"
echo "Total: $(grep -c . <<< "$usuarios")"
```

`awk -F:` separa los campos por los dos puntos, y la condición `$3 >= 1000 && $3 != 65534` deja solo los usuarios reales: los UID bajos son del sistema y 65534 es «nobody». Cada línea se escribe precedida de su UID y un tabulador para que `sort -n` pueda ordenar por él, y `cut -f2-` quita después esa columna auxiliar.

`grep -c .` cuenta las líneas que tienen algún carácter, que es el número de usuarios encontrados (0 si no hay ninguno).

18. Crear ficheros y asignar permisos

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
dir=$(mktemp -d)
cd "$dir" || exit 1

while read -r f || [ -n "$f" ]; do
    [ -z "$f" ] && continue
    touch -- "$f"
    case $f in
        *.sh)  chmod 750 -- "$f" ;;      # rwxr-x---: el dueño lo ejecuta, el grupo lo lee y ejecuta
        *.key) chmod 600 -- "$f" ;;      # rw-----: una clave solo la lee su dueño
        *)     chmod 644 -- "$f" ;;      # rw-r--r--: lo normal para un fichero de datos
    esac
done
LC_ALL=C stat -c '%a %n' -- * | LC_ALL=C sort -k2
```

`case` con patrones (`*.sh`, `*.key`) es la forma más clara de elegir según la extensión. Los permisos en octal se leen por cifras: dueño, grupo y otros, con 4 = lectura, 2 = escritura y 1 = ejecución.

El `--` indica a `touch` y a `chmod` que lo que sigue son nombres de fichero, aunque empiecen por un guion. Al final `stat -c '%a %n'` muestra los permisos en octal y el nombre de cada fichero.

19. Encuentra el fallo: la alerta que salta siempre

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r uso
if [ "$uso" -gt 80 ]; then
    echo "ALERTA: disco al $uso %"
else
    echo "OK"
fi
```

Dentro de `[ ]`, el `>` no es una comparación: es una redirección. `[ $uso > 80 ]` ejecuta `[ $uso ]` (cierto si no está vacío) y envía su salida a un fichero llamado «80», así que la condición es siempre cierta. Para comparar números en `[ ]` se usa `-gt` (mayor que), `-lt`, `-ge`, etc.

Otra opción es la aritmética de bash: `if (( uso > 80 ))`, donde `>` sí significa «mayor que».

20. Encuentra el fallo: el mayor que no lo es

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r a b
if (( a > b )); then
    echo "El mayor es $a"
else
    echo "El mayor es $b"
fi
```

Dentro de `[[ ]]`, el `>` compara cadenas en orden alfabético, carácter a carácter: «10» va antes que «9» porque el «1» va antes que el «9», igual que en un diccionario. Para comparar números hay que usar `(( a > b ))` (aritmética) o `[[ $a -gt $b ]]`.

21. Encuentra el fallo: el nombre con espacio

Medio · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r nombre
if [ "$nombre" = "Ana López" ]; then
    echo "Bienvenida, Ana"
else
    echo "No te conozco"
fi
```

Sin comillas, bash parte el valor de la variable por los espacios: `[ $nombre = "Ana López" ]` se convierte en `[ Ana López = "Ana López" ]`, que tiene un argumento de más, y `[` protesta con «too many arguments». Entre comillas, `"$nombre"` es siempre un único argumento.

Regla práctica: poner siempre las variables entre comillas en bash, salvo que se quiera expresamente que se partan.

22. Alerta de espacio en disco

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r umbral

alertas=$(awk -v u="$umbral" 'NR > 1 {
    uso = $5
    sub(/%/,"", uso)
    if (uso + 0 >= u) print "ALERTA: " $NF " al " uso "%"
}')
if [ -n "$alertas" ]; then echo "$alertas"; else echo "Todo correcto"; fi
```

`awk -v` pasa el umbral desde bash a awk. La primera línea que lee awk es la cabecera de `df` (`NR > 1` la salta), el quinto campo es el porcentaje de uso y el último (`$NF`) el punto de montaje. `sub` quita el «%» para poder comparar el número.

Las alertas se guardan en una variable para poder saber, al final, si hubo alguna. Un script así, ejecutado por cron, avisa antes de que un disco lleno deje un servidor sin funcionar.

23. ¿Quién intenta entrar por SSH?

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
# Lee el log por la entrada estándar
log=$(cat)

fallos=$(grep 'Failed password for' <<< "$log")
if [ -n "$fallos" ]; then
    awk '{ for (i = 1; i <= NF; i++) if ($i == "from") print $(i + 1) }' <<< "$fallos" |
    LC_ALL=C sort | uniq -c | LC_ALL=C sort -b -k1,1nr -k2,2 |
    while read -r n ip; do
        if (( n >= 5 )); then echo "$ip: $n intentos → BLOQUEAR"; else echo "$ip: $n intentos"; fi
    done
fi
echo "Usuarios inexistentes probados: $(grep -o 'invalid user [^]*' <<< "$fallos" | sort -u | grep -c
.)"
```

`grep` se queda con las líneas de contraseñas fallidas y `awk` extrae la palabra que va tras «from», que es la IP, esté donde esté en la línea. Después, el clásico `sort | uniq -c` cuenta cuántas veces aparece cada IP, y un segundo `sort` las ordena por número de intentos (de mayor a menor) y, a igualdad, por IP.

Para los usuarios inexistentes, `grep -o` extrae solo el trozo «invalid user nombre» de cada línea y `sort -u` elimina los repetidos. Es lo que hacen herramientas como fail2ban: contar fallos por IP y bloquear las que pasan de un límite.

24. Encuentra el fallo: el total que se queda en cero

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
total=0
while read -r n || [ -n "$n" ]; do
    total=$((total + n))
done
echo "Total: $total"
```

Cada parte de una tubería (`cat | while ...`) se ejecuta en un subproceso. El bucle suma en su propia copia de `total`, y esa copia desaparece al terminar la tubería: el `echo` del final ve la variable original, que sigue en 0.

No hace falta `cat`: el `while` puede leer directamente de la entrada estándar, y así se ejecuta en el propio script. Si los datos vinieran de otro comando, se usaría `done < <(comando)` por la misma razón.

25. Encuentra el fallo: las 08:09 que rompen la cuenta

Difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
read -r hora
h=${hora%:*}
m=${hora##*:}
echo $(( 10#$h * 60 + 10#$m ))
```

En la aritmética de bash, un número que empieza por 0 se interpreta en octal. «08» y «09» no son octales válidos (en octal solo existen las cifras del 0 al 7), y por eso da el error «value too great for base». El prefijo `10#` obliga a leer el número en base 10.

26. Permisos: de letras a octal y al revés

Muy difícil · Administración de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/administracion-sistemas-operativos

Solución (bash)

```
#!/bin/bash
while read -r p || [ -n "$p" ]; do
    [ -z "$p" ] && continue
    if [[ $p =~ ^[r-][w-][xsS-][r-][w-][xsS-][r-][w-][xtT-]$ ]]; then
        # Simbólico → octal
        especial=0; octal=""
        for t in 0 1 2; do
            trio=${p:t*3:3}; v=0
            [ "${trio:0:1}" = r ] && v=$((v + 4))
            [ "${trio:1:1}" = w ] && v=$((v + 2))
            case ${trio:2:1} in x|s|t) v=$((v + 1)) ;; esac
            case ${trio:2:1} in s|S|t|T) especial=$((especial + (4 >> t))) ;; esac # s y t llevan la x # 4, 2 o 1
            octal+=$v
        done
        (( especial )) && octal="$especial$octal"
        echo "$p = $octal"
    elif [[ $p =~ ^[0-7]{3,4}$ ]]; then
        # Octal → simbólico
        o=$p
        [ ${#p} -eq 3 ] && o="0$p"
        esp=${o:0:1}; texto=""
        for t in 0 1 2; do
            d=${o:t+1:1}; r=-; w=-; x=-
            (( d & 4 )) && r=r
            (( d & 2 )) && w=w
            (( d & 1 )) && x=x
            if (( esp & (4 >> t) )); then
                if (( t == 2 )); then
                    if [ $x = x ]; then x=t; else x=T; fi
                else
                    if [ $x = x ]; then x=s; else x=S; fi
                fi
            fi
            texto+="$r$w$x"
        done
        echo "$o = $texto"
    else
        echo "$p: no válido"
    fi
done
```

Cada cifra octal resume tres permisos sumando 4 (r), 2 (w) y 1 (x), así que convertir es sumar o, al revés, comprobar bits con `d & 4`, `d & 2` y `d & 1`. La cuarta cifra, a la izquierda, son los bits especiales: 4 para setuid (en el dueño), 2 para setgid (en el grupo) y 1 para sticky (en otros); `4 >> t` da el bit que corresponde a cada trío.

En la notación simbólica los bits especiales ocupan el lugar de la x: s o t en minúscula si además hay permiso de ejecución, y S o T en mayúscula si no lo hay. Las dos expresiones regulares validan la entrada y deciden en qué sentido hay que convertir.

27. ¿Llega este cable?

Medio · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Solución (python)

```
import sys

# Distancia máxima en metros de cada medio para cada velocidad (Gbit/s)
MAXIMO = {
    "cat5e": {1: 100},
    "cat6": {1: 100, 10: 55},
    "cat6a": {1: 100, 10: 100},
    "cat8": {10: 30, 25: 30, 40: 30},
    "om3": {1: 550, 10: 300, 40: 100},
    "om4": {1: 550, 10: 400, 40: 150},
    "os2": {1: 10000, 10: 10000, 40: 10000},
}

for linea in sys.stdin.read().splitlines():
    if not linea.strip():
        continue
    medio, metros, velocidad = linea.split()
    if medio not in MAXIMO:
        print(f"{medio}: medio desconocido")
        continue
    metros, v = int(metros), int(velocidad.rstrip("G"))
    tramo = f"{medio}, {metros} m a {v}G"
    if v not in MAXIMO[medio]:
        inferiores = ", ".join(f"{x}G" for x in sorted(MAXIMO[medio]) if x < v) or "ninguna"
        print(f"{tramo}: NO (no admite {v}G; velocidades inferiores válidas: {inferiores})")
    elif metros <= MAXIMO[medio][v]:
        print(f"{tramo}: SÍ (máximo {MAXIMO[medio][v]} m)")
    else:
        sirven = ", ".join(sorted(m for m, d in MAXIMO.items() if d.get(v, 0) >= metros)) or "ninguno"
        print(f"{tramo}: NO (máximo {MAXIMO[medio][v]} m; sirve {sirven})")
```

Un diccionario de diccionarios responde directamente a la pregunta «¿hasta dónde llega este medio a esta velocidad?» con `MAXIMO[medio][v]`. Las alternativas salen de recorrer todos los medios con `d.get(v, 0)`, que da 0 a los que no admiten esa velocidad para que nunca cumplan.

`", ".join(...)` or `"ninguno"` usa que una cadena vacía es falsa: si la lista no tiene nada, se muestra el texto alternativo. El ejercicio refleja una decisión real al cablear: el cat6 sirve para 10 Gbit/s solo en tramos cortos, y a partir de unos 100 m hay que pasar a fibra.

28. La tabla MAC de un switch

Difícil · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
puertos = int(lineas[0][1])
BROADCAST = "FF:FF:FF:FF:FF:FF"

tabla = {} # MAC → puerto
for puerto, origen, destino in lineas[1:]:
    p = int(puerto)
    if tabla.get(origen) != p: # nueva, o el equipo se ha movido de puerto
        tabla[origen] = p
        print(f"Aprende {origen} en el puerto {p}")
    otros = ", ".join(str(x) for x in range(1, puertos + 1) if x != p)
    if destino == BROADCAST:
        print(f"Difusión: inunda por {otros}")
    elif destino not in tabla:
        print(f"{destino} desconocido: inunda por {otros}")
    elif tabla[destino] == p:
        print(f"{destino} está en el mismo puerto: descarta la trama")
    else:
        print(f"Reenvía a {destino} por el puerto {tabla[destino]}")

print("Tabla MAC:")
for mac in sorted(tabla):
    print(f" {mac} → puerto {tabla[mac]}")
```

Un switch hace dos cosas con cada trama. Aprende: apunta en su tabla que la MAC de origen está en el puerto por el que ha entrado (y si ya estaba en otro, lo actualiza, porque el equipo se ha movido). Y reenvía: busca la MAC de destino en la tabla; si no la conoce, o es una difusión, la manda por todos los puertos menos el de entrada.

Si el destino está en el mismo puerto por el que llegó la trama (por ejemplo, detrás de un hub), no hace falta enviarla a ningún sitio: el destinatario ya la ha recibido. Por eso, tras unas pocas tramas, un switch deja de inundar la red y cada trama va solo a quien le corresponde.

29. Puntos únicos de fallo de una red

Muy difícil · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
raiz = lineas[0][1] # equipo desde el que se sale a Internet
enlaces = [(a, b) for _, a, b in lineas[1:]]

vecinos = {}
for a, b in enlaces:
    vecinos.setdefault(a, set()).add(b)
    vecinos.setdefault(b, set()).add(a)

def alcanzables(sin):
    """Equipos a los que se llega desde la raíz sin pasar por el equipo «sin»."""
    vistos, pila = {raiz}, [raiz]
    while pila:
        x = pila.pop()
        for v in vecinos[x]:
            if v != sin and v not in vistos:
                vistos.add(v)
                pila.append(v)
    return vistos

puntos = 0
for equipo in sorted(vecinos):
    if equipo == raiz or len(vecinos[equipo]) < 2:
        continue
    aislados = sorted(set(vecinos) - alcanzables(equipo) - {equipo})
    if aislados:
        puntos += 1
        print(f"Si falla {equipo}: sin red {' ', ' '.join(aislados)}")
    else:
        print(f"Si falla {equipo}: nadie pierde la red (hay caminos alternativos)")
print(f"Puntos únicos de fallo: {puntos}")
```

La red es un grafo y la pregunta es de conectividad: para cada equipo de interconexión se recorre el grafo desde la raíz como si ese equipo no existiera, y los que no se alcanzan son los que se quedarían sin red. El recorrido es una búsqueda en profundidad con una pila y un conjunto de visitados.

Los equipos con un solo enlace (los PC) no se analizan porque su fallo no aísla a nadie más. En una estrella cada switch es un punto único de fallo; en un anillo, en cambio, siempre hay otro camino, y solo los equipos que cuelgan de un único switch dependen de él.

30. ¿Pueden hablar sin router?

Medio · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Solución (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
    return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

for linea in sys.stdin.read().splitlines():
    if not linea.strip():
        continue
    primero, segundo = linea.split()
    ip, prefijo = primero.split("/")
    prefijo = int(prefijo)
    mascara = (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF # prefijo unos seguidos de ceros
    red1 = a_entero(ip) & mascara
    red2 = a_entero(segundo) & mascara
    if red1 == red2:
        print(f"{ip} y {segundo}: misma red {a_texto(red1)}/{prefijo}, comunicación directa")
    else:
        print(f"{ip} y {segundo}: redes distintas ({a_texto(red1)} y {a_texto(red2)}), hace falta un router")
```

La máscara de un prefijo /p son p unos seguidos de 32 – p ceros: desplazando 32 unos hacia la izquierda 32 - p posiciones y quedándose con 32 bits (& 0xFFFFFFFF) se obtiene sin tablas. El AND con la máscara borra la parte de host y deja la dirección de red.

Si las dos IP dan la misma red, el equipo puede entregar el paquete directamente; si no, se lo envía a su puerta de enlace. Por eso dos equipos con IP «parecidas» (192.168.1.10 y 192.168.1.70) pueden necesitar un router con una máscara /26.

31. Abreviar y expandir direcciones IPv6

Difícil · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Solución (python)

```
import sys

for linea in sys.stdin.read().splitlines():
    d = linea.strip()
    if not d:
        continue
    if "::" in d or len(d.split(":")) != 8:
        # Expandir: los grupos que faltan son los que representa «::»
        izq, _, der = d.partition("::")
        gi = [g for g in izq.split(":") if g]
        gd = [g for g in der.split(":") if g]
        grupos = gi + ["0"] * (8 - len(gi) - len(gd)) + gd
        resultado = ":".join(g.lower().zfill(4) for g in grupos)
    else:
        # Abreviar: sin ceros a la izquierda y «::» en la serie de ceros más larga
        grupos = [format(int(g, 16), "x") for g in d.split(":")]
        mejor, largo, i = -1, 0, 0
        while i < 8:
            if grupos[i] == "0":
                j = i
                while j < 8 and grupos[j] == "0":
                    j += 1
                if j - i > largo:
                    # con «>», en empate gana la primera
                    mejor, largo = i, j - i
                i = j
            else:
                i += 1
        if largo >= 2:
            resultado = ":".join(grupos[:mejor]) + "::" + ":".join(grupos[mejor + largo:])
        else:
            resultado = ":".join(grupos)
    print(f"{d} → {resultado}")
```

Para expandir, `partition("::")` separa lo que hay a cada lado; los grupos que faltan hasta 8 son los ceros que representa «::», y `zfill(4)` rellena cada grupo con ceros a la izquierda. Para abreviar, `format(int(g, 16), "x")` quita los ceros a la izquierda (un grupo «0000» queda como «0») y pasa a minúsculas a la vez.

Luego se busca la serie de grupos a cero más larga recorriendo la lista. Las reglas de la RFC 5952 son las que fija el enunciado: «::» solo una vez (si no, no se sabría cuántos ceros hay en cada hueco), solo para dos o más grupos, y en empate la primera serie.

32. Calculadora VLSM

Muy difícil · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Solución (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
    return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
base_ip, base_prefijo = lineas[0][1].split("/")
departamentos = [(nombre, int(hosts)) for nombre, hosts in lineas[1:]]

inicio = a_entero(base_ip)
fin_red = inicio + 2 ** (32 - int(base_prefijo))
pos = inicio
for nombre, hosts in sorted(departamentos, key=lambda d: (-d[1], d[0])):
    bits = 2
    while 2 ** bits - 2 < hosts:          # la subred más pequeña que alcanza
        bits += 1
    tam = 2 ** bits
    if pos + tam > fin_red:
        print(f"{nombre}: no hay espacio para {hosts} hosts")
        continue
    print(f"{nombre} ({hosts}): {a_texto(pos)}/{32 - bits}, hosts {a_texto(pos + 1)} - {a_texto(pos + tam - 2)}, "
          f"difusión {a_texto(pos + tam - 1)}, sobran {tam - 2 - hosts}")
    pos += tam
libre = fin_red - pos
print(f"Libre: {libre} direcciones desde {a_texto(pos)}" if libre else "Libre: 0 direcciones")
```

Con las IP convertidas en enteros, repartir la red es sumar: cada subred empieza donde acaba la anterior. El tamaño de cada una es la menor potencia de 2 cuyos hosts útiles ($2^{\text{bits}} - 2$, por la dirección de red y la de difusión) alcanzan, y su prefijo es $32 - \text{bits}$.

Ordenar de mayor a menor no es un capricho: como todos los tamaños son potencias de 2 y van decreciendo, cada subred empieza siempre en un múltiplo de su tamaño, que es la condición para que sea una subred válida. Si se colocaran primero las pequeñas, una grande podría quedar desalineada.

33. Permisos efectivos de una carpeta compartida

Medio · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-

seguridad

Solución (python)

```
import sys

NIVELES = ["ninguno", "lectura", "modificar", "control"]

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]

compartido, ntfs, denegados, usuarios = {}, {}, set(), []
for l in lineas:
    if l[0] in ("compartido", "ntfs"):
        capa = compartido if l[0] == "compartido" else ntfs
        capa[l[1]] = max(capa.get(l[1], 0), NIVELES.index(l[2]))
    elif l[0] == "denegar":
        denegados.add(l[1])
    elif l[0] == "usuario":
        usuarios.append((l[1], l[2].split(",")))

for nombre, grupos in usuarios:
    c = max((compartido.get(g, 0) for g in grupos), default=0)      # dentro de una capa, se suman
    if any(g in denegados for g in grupos):                        # la denegación gana a todo
        n = 0
    else:
        n = max((ntfs.get(g, 0) for g in grupos), default=0)
    efectivo = min(c, n)                                           # entre capas, la más restrictiva
    print(f"{nombre}: compartido {NIVELES[c]}, NTFS {NIVELES[n]} → efectivo {NIVELES[efectivo]}")
```

Guardar los niveles como posiciones de la lista NIVELES convierte las reglas en operaciones con números: dentro de una capa, el usuario se queda con el mayor de sus grupos (`max`); entre las dos capas, con el menor (`min`). La denegación explícita se comprueba aparte, porque anula cualquier permiso que den los demás grupos.

Es la causa de muchas consultas de soporte: alguien tiene «control total» en NTFS y aun así solo puede leer, porque el recurso compartido solo da lectura. Lo habitual es dar control al compartido y ajustar los permisos con NTFS.

34. Un cortafuegos de primera coincidencia

Difícil · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-

seguridad

Solución (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

texto = sys.stdin.read().split("---")
reglas = [l.split() for l in texto[0].splitlines() if l.strip()]      # la primera línea es «politica
allow|deny»
paquetes = [l.split() for l in texto[1].splitlines() if l.strip()]

politica = reglas[0][1]

def en_red(ip, red):
    if red == "any":
        return True
    base, _, prefijo = red.partition("/")
    prefijo = int(prefijo) if prefijo else 32                        # una IP sola es un /32
    mascara = (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF
    return a_entero(ip) & mascara == a_entero(base) & mascara

def coincide(regla, proto, ip, puerto):
    _, r_proto, r_puerto, _, r_red = regla
    return (r_proto in ("any", proto)
            and (r_puerto == "any" or int(r_puerto) == int(puerto))
            and en_red(ip, r_red))

for proto, ip, puerto in paquetes:
    for n, regla in enumerate(reglas[1:], 1):
        if coincide(regla, proto, ip, puerto):
            print(f"{proto} {ip} → {puerto}: {regla[0].upper()} (regla {n})")
            break
    else:
        print(f"{proto} {ip} → {puerto}: {politica.upper()} (política por defecto)")
```

Cada regla coincide si coinciden sus tres condiciones (protocolo, puerto y red de origen), y «any» funciona como comodín. La pertenencia a una red se comprueba como siempre en IP: aplicando la máscara a las dos direcciones y comparando; una IP sin prefijo se trata como /32 (solo ella).

El `for ... else` expresa justo la lógica del cortafuegos: se recorren las reglas en orden y la primera que coincide decide (`break`); el `else` solo se ejecuta si ninguna coincidió, y entonces se aplica la política por defecto. Por eso el orden importa tanto: una regla general colocada antes tapa a una más específica.

35. ¿Cuánto aguantaría esta contraseña?

Muy difícil · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-

Solución (python)

```

import sys
import math

DICCIONARIO = {"password", "123456", "qwerty", "admin", "iloveyou", "dragon", "monkey", "letmein",
               "futbol", "barcelona", "madrid", "contraseña", "hola", "princesa", "superman"}

UNIDADES = [("años", 365 * 86400), ("días", 86400), ("horas", 3600), ("minutos", 60)]

def tiempo(segundos):
    if segundos < 1:
        return "instantáneo"
    for nombre, s in UNIDADES:
        valor = segundos / s
        if valor >= 1:
            if valor >= 1_000_000:
                return f"más de un millón de {nombre}"
            if valor < 100:
                return f"{valor:.1f} {nombre}"
            return f"{round(valor):,} {nombre}".replace(",", ".") # punto de miles
    return "menos de 1 minuto"

for clave in sys.stdin.read().splitlines():
    if not clave:
        continue
    base = clave.lower()
    if base in DICCIONARIO or base.rstrip("0123456789.!") in DICCIONARIO:
        print(f"{clave}: muy débil (está en los diccionarios de contraseñas)")
        continue
    juego = (26 * any(c.islower() for c in clave) + 26 * any(c.isupper() for c in clave)
            + 10 * any(c.isdigit() for c in clave) + 33 * any(not c.isalnum() for c in clave))
    bits = len(clave) * math.log2(juego)
    if len(set(clave)) <= len(clave) / 3: # muchos caracteres repetidos
        bits /= 2
    nivel = "débil" if bits < 40 else "media" if bits < 60 else "fuerte" if bits < 80 else "muy fuerte"
    segundos = 2 ** bits / 2 / 1e10
    print(f"{clave}: {nivel} ({bits:.1f} bits, {tiempo(segundos)} por fuerza bruta)")

```

La entropía mide cuántas contraseñas posibles hay con esa forma: si se usan caracteres de un juego de N símbolos, cada carácter aporta $\log_2(N)$ bits. `any(...)` devuelve un booleano que, multiplicado por el tamaño de cada grupo, suma solo los grupos que aparecen. Un atacante prueba de media la mitad de las 2^{bits} combinaciones, de ahí el `/ 2`.

Pero la entropía solo vale para contraseñas aleatorias: «Barcelona2024!» parece fuerte por su longitud y variedad, y cae en segundos porque los atacantes prueban primero palabras conocidas con números al final. Por eso se comprueba el diccionario antes que nada. La conclusión práctica sale del último ejemplo: cuatro palabras al azar son más fuertes y más fáciles de recordar que una clave corta llena de símbolos.

La función `tiempo` busca la mayor unidad en la que el valor llega a 1 y aplica las reglas de formato; `f"{n:,}"` pone comas de miles, que se cambian por puntos.

36. Informe del inventario

Medio · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

Solución (python)

```
import sys
import csv
from datetime import date

lineas = sys.stdin.read().splitlines()
hoy = date.fromisoformat(lineas[0].split("=")[1])
filas = list(csv.DictReader(lineas[1:], delimiter=";")) # codigo;tipo;aula;precio;fin_garantia

aulas = {}
for f in filas:
    aulas.setdefault(f["aula"], []).append(f)

total_equipos, total_valor, total_vencidos = 0, 0.0, 0
for aula in sorted(aulas):
    equipos = aulas[aula]
    valor = sum(float(e["precio"]) for e in equipos)
    vencidos = sorted(e["codigo"] for e in equipos if date.fromisoformat(e["fin_garantia"]) < hoy)
    linea = f"{aula}: {len(equipos)} {'equipo' if len(equipos) == 1 else 'equipos'}, {valor:.2f} €"
    if vencidos:
        linea += ", sin garantía: " + ", ".join(vencidos)
    print(linea)
    total_equipos += len(equipos)
    total_valor += valor
    total_vencidos += len(vencidos)

porcentaje = total_vencidos / total_equipos * 100
print(f"TOTAL: {total_equipos} equipos, {total_valor:.2f} €, {total_vencidos} sin garantía
({porcentaje:.0f} %)")
```

`csv.DictReader` lee el CSV y da cada fila como un diccionario con los nombres de la cabecera, así el código habla de `e["precio"]` en lugar de posiciones. Agrupar es crear un diccionario aula → lista de equipos con `setdefault`, y después cada aula se resume con `sum` y una comprensión.

Las fechas se comparan como objetos `date`, no como texto: así «antes de hoy» funciona sin trucos. Es exactamente lo que hace una tabla dinámica (agrupar por aula y sumar), pero repetible: el mismo programa sirve para el inventario del año que viene.

37. Un índice automático para la documentación

Difícil · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

Solución (python)

```
import sys
import re
import unicodedata

lineas = sys.stdin.read().splitlines()

def ancla(titulo):
    s = unicodedata.normalize("NFD", titulo.lower())
    s = "".join(c for c in s if unicodedata.category(c) != "Mn") # quita las tildes
    s = re.sub(r"^[a-z0-9 -]", "", s)
    return s.replace(" ", "-")

usadas = {}
en_codigo = False
n2 = n3 = 0
for linea in lineas:
    if linea.startswith("```"):
        en_codigo = not en_codigo
        continue
    if en_codigo:
        continue
    m = re.match(r"^(#{2,3})\s+(.*)$", linea)
    if not m:
        continue
    titulo = m.group(2).strip()
    a = ancla(titulo)
    if a in usadas:
        usadas[a] += 1
        a = f"{a}-{usadas[a]}" # la segunda vez -1, la tercera -2...
    else:
        usadas[a] = 0
    if len(m.group(1)) == 2:
        n2 += 1
        n3 = 0
        print(f"{n2}. [{titulo}](#{a})")
    else:
        n3 += 1
        print(f"    {n2}.{n3}. [{titulo}](#{a})")
```

El ancla se construye en cuatro pasos: minúsculas, quitar tildes (la normalización NFD separa cada letra de su tilde, y las marcas, de categoría «Mn», se descartan), eliminar lo que no sea letra, número, espacio o guion, y cambiar los espacios por guiones. Es el mismo algoritmo que usan GitHub y la mayoría de generadores de documentación.

Un diccionario cuenta cuántas veces ha salido cada ancla para añadir el sufijo -1, -2... a las repetidas. La numeración usa dos contadores, y el de nivel 3 se reinicia cada vez que aparece un título de nivel 2. Las líneas dentro de bloques de código se saltan con una variable que cambia de estado en cada «```».

38. Un conversor de Markdown a HTML

Muy difícil · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-

Solución (python)

```

import sys
import re
import html

lineas = sys.stdin.read().splitlines()

def en_linea(texto):
    """Formato dentro de una línea; el código se aparta primero para que no se interprete."""
    codigos = []
    def guardar(m):
        codigos.append(m.group(1))
        return f"\x00{len(codigos) - 1}\x00"
    texto = re.sub(r"``([^\`]+)`", guardar, texto)
    texto = re.sub(r"\*(.+?)\*", r"<strong>\1</strong>", texto)
    texto = re.sub(r"\*(.+?)\*", r"<em>\1</em>", texto)
    texto = re.sub(r"\[[^\]]+\]\([^\)]+\)", r"<a href='\2'\>\1</a>", texto)
    return re.sub(r"\x00(\d+)\x00", lambda m: f"<code>{codigos[int(m.group(1))]}</code>", texto)

salida, parrafo, codigo = [], [], []
lista = None # "ul", "ol" o None
en_codigo = False

def cerrar_parrafo():
    if parrafo:
        salida.append("<p>" + en_linea(" ".join(parrafo)) + "</p>")
        parrafo.clear()

def cerrar_lista():
    global lista
    if lista:
        salida.append(f"</{lista}>")
        lista = None

for linea in lineas:
    if en_codigo:
        if linea.startswith("```"):
            salida.append("<pre><code>" + "\n".join(codigo) + "</code></pre>")
            en_codigo = False
        else:
            codigo.append(html.escape(linea, quote=False))
            continue
    if linea.startswith("```"):
        cerrar_parrafo(); cerrar_lista()
        en_codigo, codigo = True, []
        continue
    texto = html.escape(linea, quote=False)
    titulo = re.match(r"^(#{1,6})\s+(.*)$", texto)
    elemento = re.match(r"^(?:([-*])|\d+\.)\s+(.*)$", texto)
    if not texto.strip():
        cerrar_parrafo(); cerrar_lista()
    elif titulo:
        cerrar_parrafo(); cerrar_lista()
        n = len(titulo.group(1))
        salida.append(f"<h{n}>{en_linea(titulo.group(2))}</h{n}>")
    elif elemento:
        cerrar_parrafo()
        tipo = "ul" if elemento.group(1) else "ol"
        if lista != tipo:
            cerrar_lista()
            salida.append(f"<{tipo}>")
            lista = tipo
        salida.append(f"<li>{en_linea(elemento.group(2))}</li>")
    else:
        cerrar_lista()
        parrafo.append(texto)
    cerrar_parrafo(); cerrar_lista()
if en_codigo:
    salida.append("<pre><code>" + "\n".join(codigo) + "</code></pre>")

```

```
print("\n".join(salida))
```

El conversor trabaja por bloques con un pequeño estado: si estamos en un párrafo (las líneas acumuladas), en qué lista (ul, ol o ninguna) y si estamos dentro de un bloque de código. Cada línea decide qué bloque abre o cierra; las funciones `cerrar_parrafo` y `cerrar_lista` se llaman siempre que cambia el tipo de bloque, y una vez más al final.

Lo primero con cada línea es escapar `<`, `>` y `&`, para que el texto del documento no se convierta en HTML de verdad. El formato en línea se aplica con expresiones regulares en orden (`**` antes que `*`, para que la negrita no se lea como dos cursivas), y el código en línea se aparta antes con un marcador para que un `*` dentro de él no se interprete.

39. Contar ficheros por extensión

Medio · Ejercicios de Sistemas Informáticos · apuntesdam.com/subject/sistemas/topic/ejercicios-sistemas

Solución (bash)

```
#!/bin/bash
# Entrada: un nombre de fichero por línea

awk '
NF {
    n = split($0, partes, ".")
    if (n > 1 && partes[n] != "") ext[tolower(partes[n])]++
    else sinext++
}
END {
    for (e in ext) print e ": " ext[e] | "LC_ALL=C sort"
    close("LC_ALL=C sort")      # espera a que sort termine antes de seguir
    if (sinext) print "(sin extensión): " sinext
}'
```

`split($0, partes, ".")` parte el nombre por los puntos y devuelve cuántos trozos hay: la extensión es el último, y si solo hay un trozo, el fichero no tiene extensión. Usar el último trozo hace que «a.tar.gz» cuente como «gz», y `tolower` junta «PDF» y «pdf».

Los resultados se envían a `sort` desde dentro de `awk` (`print ... | "sort"`), y `close` espera a que termine antes de escribir la línea de los ficheros sin extensión, que debe ir al final.

40. Validar direcciones IPv4

Difícil · Ejercicios de Sistemas Informáticos · apuntesdam.com/subject/sistemas/topic/ejercicios-sistemas

Solución (bash)

```
#!/bin/bash
octeto='(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-9])'
ip_valida="^$octeto\.$octeto\.$octeto\.$octeto$"

while read -r ip || [ -n "$ip" ]; do
    [ -z "$ip" ] && continue
    if [[ $ip =~ $ip_valida ]]; then
        echo "IP válida"
    else
        echo "IP no válida"
    fi
done
```

La expresión regular de un octeto enumera los casos que van de 0 a 255 sin ceros a la izquierda: 250-255, 200-249, 100-199, 10-99 y 0-9. Así no hace falta partir la IP y comparar números: la expresión completa son cuatro octetos separados por puntos escapados (`\.`), anclados con `^` y `$` para que no sobre nada.

Guardarla en una variable y usarla sin comillas en `[[ $ip =~ $ip_valida ]]` es la forma recomendada en bash: así no hay que escapar los paréntesis y las barras dentro de `[[ ]]`.

41. Red y broadcast con aritmética de bits

Difícil · Ejercicios de Sistemas Informáticos · apuntesdam.com/subject/sistemas/topic/ejercicios-sistemas

Solución (bash)

```
#!/bin/bash
read -r entrada
ip=${entrada%/*}
prefijo=${entrada#*/}

a_texto() { echo "$(( ($1 >> 24) & 255 )).$(( ($1 >> 16) & 255 )).$(( ($1 >> 8) & 255 )).$(( $1 & 255 ))"; }

IFS=. read -r a b c d <<< "$ip"
n=$(( (a << 24) | (b << 16) | (c << 8) | d ))
mascara=$(( (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF ))
red=$(( n & mascara ))
difusion=$(( red | (~mascara & 0xFFFFFFFF) ))

echo "Red: $(a_texto $red)"
echo "Broadcast: $(a_texto $difusion)"
echo "Hosts: $(( (1 << (32 - prefijo)) - 2 ))"
```

La IP se convierte en un único número de 32 bits desplazando cada octeto a su posición (<<) y uniéndolos con OR. Con eso, la red es la IP AND la máscara (se borran los bits de host) y la difusión es la red con todos los bits de host a 1 (OR con la máscara invertida, ~mascara , limitada a 32 bits).

IFS=. read parte la IP por los puntos en cuatro variables, y la función a_texto hace el camino inverso. El número de hosts es $2^{(\text{bits de host})}$ menos la dirección de red y la de difusión.

42. Planificador de procesos: FCFS, SJF, SRTF y Round Robin

Muy difícil · Python · 100 minutos · apuntesdam.com/ejercicios/python/planificador-de-procesos

Solución (python)

```
import sys

NOMBRES = {"FCFS": "FCFS (primero en llegar, primero en ser servido)", "SJF": "SJF (el más corto primero, sin expulsión)",
           "SRTF": "SRTF (menor tiempo restante, con expulsión)", "RR": "Round Robin"}

def coma(x):
    return f"{x:.2f}".replace(".", ",")

def simular(algoritmo, quantum, procesos):
    """Devuelve el diagrama de Gantt como lista de [nombre, inicio, fin] y el instante en que termina cada proceso."""
    restante = {p["nombre"]: p["rafaga"] for p in procesos}
    orden = {p["nombre"]: i for i, p in enumerate(procesos)}
    pendientes = sorted(procesos, key=lambda p: (p["llegada"], orden[p["nombre"]]))
    listos, gantt, fin = [], [], {}
    t = 0

    def admitir(hasta):
        """Pasa a la cola de listos los procesos que han llegado hasta el instante «hasta»."""
        while pendientes and pendientes[0]["llegada"] <= hasta:
            listos.append(pendientes.pop(0))

    def ejecutar(p, duracion):
        nonlocal t
        if gantt and gantt[-1][0] == p["nombre"] and gantt[-1][2] == t:
            gantt[-1][2] += duracion # el mismo proceso sigue: se alarga el tramo
        else:
            gantt.append([p["nombre"], t, t + duracion])
            t += duracion
            restante[p["nombre"]] -= duracion

    admitir(t)
    while listos or pendientes:
        if not listos: # CPU ociosa hasta la siguiente llegada
            gantt.append(["--", t, pendientes[0]["llegada"]])
            t = pendientes[0]["llegada"]
            admitir(t)
            continue
        if algoritmo == "FCFS":
            p = listos.pop(0)
            ejecutar(p, restante[p["nombre"]])
        elif algoritmo == "SJF":
            p = min(listos, key=lambda q: (q["rafaga"], q["llegada"], orden[q["nombre"]]))
            listos.remove(p)
            ejecutar(p, restante[p["nombre"]])
        elif algoritmo == "SRTF":
            p = min(listos, key=lambda q: (restante[q["nombre"]], q["llegada"], orden[q["nombre"]]))
            # Se ejecuta hasta que acabe o hasta la próxima llegada, que puede expulsarlo
            hasta = t + restante[p["nombre"]]
            if pendientes:
                hasta = min(hasta, pendientes[0]["llegada"])
            ejecutar(p, hasta - t)
            if restante[p["nombre"]] > 0:
                admitir(t)
                continue
            listos.remove(p)
        else: # RR
            p = listos.pop(0)
            ejecutar(p, min(quantum, restante[p["nombre"]]))
            admitir(t) # los que llegan entran en la cola antes que el expulsado
            if restante[p["nombre"]] > 0:
                listos.append(p)
                continue
```

```

        fin[p["nombre"]] = t
        admitir(t)
    return gantt, fin

def main():
    lineas = sys.stdin.read().split("\n")
    cabecera = lineas[0].split()
    algoritmo = cabecera[0].upper() if cabecera else ""
    quantum = 0
    if algoritmo == "RR":
        if len(cabecera) != 2 or not cabecera[1].isdigit() or int(cabecera[1]) == 0:
            print("Algoritmo no válido")
            return
        quantum = int(cabecera[1])
    elif algoritmo not in NOMBRES or len(cabecera) != 1:
        print("Algoritmo no válido")
        return

    procesos = []
    for num, linea in enumerate(lineas[1:], start=2):
        partes = linea.split()
        if not partes:
            continue
        if len(partes) != 3 or not partes[1].isdigit() or not partes[2].isdigit() or int(partes[2]) == 0 \
            or any(p["nombre"] == partes[0] for p in procesos):
            print(f"Aviso: línea {num} ignorada")
            continue
        procesos.append({"nombre": partes[0], "llegada": int(partes[1]), "rafaga": int(partes[2])})
    if not procesos:
        print("No hay procesos")
        return

    gantt, fin = simular(algoritmo, quantum, procesos)
    print("Algoritmo: " + NOMBRES[algoritmo] + (f" con quantum {quantum}" if algoritmo == "RR" else ""))
    print("Diagrama: " + " ".join(f"| {n} {a}-{b}" for n, a, b in gantt) + " |")
    print(f"{'Proceso':<8}{'Llegada':>8}{'Ráfaga':>8}{'Fin':>6}{'Retorno':>9}{'Espera':>8}")
    retornos, esperas = [], []
    for p in procesos:
        retorno = fin[p["nombre"]] - p["llegada"]
        espera = retorno - p["rafaga"]
        retornos.append(retorno)
        esperas.append(espera)
        print(f"{p['nombre']:<8}{p['llegada']:>8}{p['rafaga']:>8}{fin[p['nombre']]:>6}{retorno:>9}{espera:>8}")
    print(f"Media de retorno: {coma(sum(retornos) / len(procesos))} · Media de espera: {coma(sum(esperas) / len(procesos))}")
    ocupada = sum(b - a for n, a, b in gantt if n != "--")
    print(f"Uso de la CPU: {coma(100 * ocupada / gantt[-1][2])} %")

main()

```

La simulación avanza por eventos: en lugar de mirar el reloj de unidad en unidad, salta de una decisión a la siguiente (fin de un proceso, fin de un quantum o llegada de un proceso). Es más rápido y deja claro en qué instantes decide el planificador.

Los cuatro algoritmos comparten el bucle y las funciones `admitir` y `ejecutar`; solo cambia la elección del proceso y el tiempo que se le deja. Es la misma estructura que usan los sistemas operativos reales, donde la política de planificación es un módulo intercambiable.

Las tuplas como clave de `min` codifican la política y los desempates en una sola expresión, y hacen que el resultado sea siempre el mismo para la misma entrada, sin depender del orden en que se guardan los procesos.

Retorno y espera salen solos al final: $\text{retorno} = \text{fin} - \text{llegada}$ y $\text{espera} = \text{retorno} - \text{ráfaga}$. Comparar las medias entre algoritmos con los mismos procesos muestra por qué SJF y SRTF minimizan la espera media y por qué Round Robin, aunque espere más de media, responde antes a los procesos cortos.

43. Plan de direccionamiento con subredes VLSM

Difícil · Python · 70 minutos · apuntesdam.com/ejercicios/python/subredes-vlsm

Solución (python)

```
import sys

def a_numero(ip):
    """«192.168.10.0» → entero de 32 bits; None si no es una IPv4."""
    partes = ip.split(".")
    if len(partes) != 4 or not all(p.isdigit() and int(p) <= 255 for p in partes):
        return None
    n = 0
    for p in partes:
        n = n * 256 + int(p)
    return n

def a_ip(n):
    return ".".join(str((n >> desplazamiento) & 255) for desplazamiento in (24, 16, 8, 0))

def mascara(prefijo):
    return (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF

def prefijo_para(hosts):
    """El prefijo más largo cuyo bloque tiene sitio para «hosts» (sin contar la red y la difusión), como
    mucho /30."""
    prefijo = 30
    while 2 ** (32 - prefijo) - 2 < hosts:
        prefijo -= 1
    return prefijo

def main():
    lineas = sys.stdin.read().split("\n")
    texto = lineas[0].strip()
    if texto.count("/") != 1 or a_numero(texto.split("/")[0]) is None or not texto.split("/")
        [1].isdigit() \
        or not 8 <= int(texto.split("/")[1]) <= 30:
        print(f"Red no válida: {texto}")
        return
    ip, prefijo = texto.split("/")
    prefijo = int(prefijo)
    base = a_numero(ip)
    if base & mascara(prefijo) != base:
        print(f"{ip} no es una dirección de red (sería {a_ip(base & mascara(prefijo))}/{prefijo}")
        return
    total = 2 ** (32 - prefijo)

    subredes = []
    for num, linea in enumerate(lineas[1:], start=2):
        partes = linea.split()
        if not partes:
            continue
        if len(partes) < 2 or not partes[-1].isdigit() or int(partes[-1]) == 0:
            print(f"Aviso: línea {num} ignorada")
            continue
        subredes.append((" ".join(partes[:-1]), int(partes[-1])))

    print(f"Red: {a_ip(base)}/{prefijo} ({total} direcciones)")
    # VLSM: de la más grande a la más pequeña, así cada bloque queda alineado con su tamaño
    siguiente = base
    for nombre, hosts in sorted(subredes, key=lambda s: -s[1]):
        p = prefijo_para(hosts)
        tamaño = 2 ** (32 - p)
        if p < prefijo or siguiente + tamaño > base + total:
            print(f"{nombre} ({hosts} hosts) → no cabe: necesita un /{p} y quedan {base + total -
                siguiente} direcciones")
```

```

        continue
    print(f"{nombre} ({hosts} hosts) → {a_ip(siguiete)}/{p}")
    print(f"  máscara {a_ip(mascara(p))} · hosts útiles {tamano - 2}")
    print(f"  primera {a_ip(siguiete + 1)} · última {a_ip(siguiete + tamano - 2)} · difusión
           {a_ip(siguiete + tamano - 1)}")
    siguiete += tamano

libres = base + total - siguiete
if libres:
    print(f"Sin asignar: {libres} de {total} direcciones, desde {a_ip(siguiete)}")
else:
    print(f"Sin asignar: 0 de {total} direcciones")

main()

```

Tratar las direcciones como enteros de 32 bits convierte todo el subnetting en aritmética: la máscara es un desplazamiento, la red un `&` y las direcciones de cada subred, sumas. La conversión a texto solo ocurre al escribir.

Asignar de mayor a menor garantiza la alineación: después de bloques de 64, 32 y 16, la siguiente dirección libre es siempre múltiplo del tamaño del bloque que toca, que es exactamente lo que exige una dirección de red.

La validación tiene dos niveles: que el texto sea una IP con prefijo, y que la IP sea de verdad una dirección de red. Es un error frecuente escribir 192.168.10.5/24 queriendo decir la red 192.168.10.0/24.

El módulo `ipaddress` de Python hace todo esto (`ip_network`, `subnets`, `broadcast_address`). Escribirlo a mano enseña lo que hay debajo; en un programa real conviene usar la librería.

44. Detección de ataques de fuerza bruta en el registro de SSH

Difícil · Python · 70 minutos · apuntesdam.com/ejercicios/python/deteccion-de-ataques-ssh

Solución (python)

```
import re
import sys
from datetime import datetime, timedelta

FALLO = re.compile(r"Failed password for (invalid user )?(\\S+) from (\\S+) port")
EXITO = re.compile(r"Accepted (password|publickey) for (\\S+) from (\\S+) port")
VENTANA = timedelta(minutes=10)
MAX_FALLOS = 5

def main():
    fallos_ip = {}           # ip → lista de instantes de fallo
    bloqueos = {}           # ip → (instante del bloqueo, primer fallo de la ventana)
    inexistentes = set()
    fallos_usuario = {}
    alertas = []
    n_fallos = n_exitos = 0

    for linea in sys.stdin.read().split("\n"):
        if not linea.strip():
            continue
        # Las líneas de syslog empiezan con «Oct  3 10:15:01» (sin año)
        try:
            instante = datetime.strptime(linea[:15], "%b %d %H:%M:%S").replace(year=2026)
        except ValueError:
            continue
        if m := FALLO.search(linea):
            n_fallos += 1
            invalido, usuario, ip = m.groups()
            if invalido:
                inexistentes.add(usuario)
                fallos_usuario[usuario] = fallos_usuario.get(usuario, 0) + 1
                lista = fallos_ip.setdefault(ip, [])
                lista.append(instante)
                # Ventana deslizante: solo cuentan los fallos de los últimos 10 minutos
                recientes = [t for t in lista if instante - t <= VENTANA]
                if len(recientes) >= MAX_FALLOS and ip not in bloqueos:
                    bloqueos[ip] = (instante, recientes[0])
            elif m := EXITO.search(linea):
                n_exitos += 1
                _, usuario, ip = m.groups()
                previos = len(fallos_ip.get(ip, []))
                if previos:
                    alertas.append(f" {usuario} entró desde {ip} a las {instante:%H:%M:%S} tras {previos} {'fallo' if previos == 1 else 'fallos'} desde esa IP")

    print(f"Intentos fallidos: {n_fallos} · Accesos correctos: {n_exitos}")
    if bloqueos:
        print("Bloquear:")
        for ip, (cuando, desde) in sorted(bloqueos.items(), key=lambda b: b[1][0]):
            print(f" {ip} a las {cuando:%H:%M:%S} ({MAX_FALLOS} fallos desde las {desde:%H:%M:%S})")
    else:
        print("Bloquear: ninguna IP")
    print("Usuarios inexistentes probados: " + ("", ".join(sorted(inexistentes)) if inexistentes else "ninguno"))
    if fallos_usuario:
        ranking = sorted(fallos_usuario.items(), key=lambda u: (-u[1], u[0]))[:3]
        print("Usuarios más atacados: " + ("", ".join(f'{u} ({n})' for u, n in ranking))
    print("Alertas:" if alertas else "Alertas: ninguna")
    for a in alertas:
        print(a)

main()
```

Las expresiones regulares con grupos convierten una línea de texto libre en datos estructurados (usuario, IP, si el usuario existe). Es la base de cualquier análisis de registros, desde un script como este hasta herramientas como Logstash.

La ventana deslizante es la clave de fail2ban: no importa cuántos fallos haya en total, sino cuántos caben en 10 minutos. Recalcularla en cada fallo con una lista por IP es sencillo y suficiente para un registro diario; con millones de líneas se usaría una cola que descarta los fallos antiguos.

Separar la recogida de datos (un único recorrido del registro) de la presentación (el informe al final) permite añadir estadísticas nuevas sin volver a leer nada.

Las alertas cruzan dos tipos de eventos: un acceso correcto no es sospechoso por sí solo, pero sí después de fallos desde la misma IP. Ese tipo de correlación es lo que hacen los sistemas de detección de intrusiones (SIEM).

45. Informe de un registro de accesos web

Difícil · Bash · 75 minutos · apuntesdam.com/ejercicios/bash/informe-registro-accesos-web

Solución (bash)

```
#!/bin/bash
# Orden de texto byte a byte: así sort y [[ < ]] ordenan igual en cualquier máquina
export LC_ALL=C

declare -A por_ip por_ruta rutas_404
total=0
bytes_total=0
invalidas=0
c2=0 c3=0 c4=0 c5=0

while read -r ip _ _ _ metodo ruta _ codigo bytes _; do
    [ -z "$ip" ] && continue
    if [ -z "$bytes" ] || ! [[ $codigo =~ ^[0-9]{3}$ ]]; then
        invalidas=$((invalidas + 1))
        continue
    fi
    total=$((total + 1))
    por_ip[$ip]=$(( ${por_ip[$ip]:-0} + 1 ))
    por_ruta[$ruta]=$(( ${por_ruta[$ruta]:-0} + 1 ))
    # Una respuesta sin cuerpo trae «-» en lugar del tamaño
    [ "$bytes" != "-" ] && bytes_total=$((bytes_total + bytes))
    case ${codigo:0:1} in
        2) c2=$((c2 + 1)) ;;
        3) c3=$((c3 + 1)) ;;
        4) c4=$((c4 + 1)) ;;
        5) c5=$((c5 + 1)) ;;
    esac
    [ "$codigo" = "404" ] && rutas_404[$ruta]=1
done

if [ "$total" -eq 0 ]; then
    echo "No hay peticiones válidas."
    exit 0
fi

echo "Peticiones: $total"
echo "Visitantes distintos: ${#por_ip[@]}"
echo "Respuestas: 2xx $c2 · 3xx $c3 · 4xx $c4 · 5xx $c5"
echo "Datos enviados: $((bytes_total / 1024)) KB"

echo "IPs con más peticiones:"
for ip in "${!por_ip[@]}"; do
    echo "${por_ip[$ip]} $ip"
done | sort -k1,1nr -k2,2 | head -n 3 | while read -r n ip; do
    printf "%3d %s\n" "$n" "$ip"
done

mejor=""
for ruta in "${!por_ruta[@]}"; do
    n=${por_ruta[$ruta]}
    if [ -z "$mejor" ] || [ "$n" -gt "${por_ruta[$mejor]}" ] || { [ "$n" -eq "${por_ruta[$mejor]}" ] && [ $ruta < $mejor ] }; then
        mejor=$ruta
    fi
done
veces=${por_ruta[$mejor]}
[ "$veces" -eq 1 ] && palabra="vez" || palabra="veces"
echo "Página más pedida: $mejor ($veces $palabra)"

if [ ${#rutas_404[@]} -eq 0 ]; then
    echo "Errores 404: ninguno"
else
    echo "Errores 404:"
    printf "%s\n" "${!rutas_404[@]}" | sort | while read -r ruta; do
        echo "  $ruta"
    done
done
```

```
fi  
  
[ "$invalidas" -gt 0 ] && echo "Líneas no válidas: $invalidas"  
exit 0
```

Todo el recuento se hace en una sola pasada: el bucle `while read` valida la línea y actualiza los contadores, y el informe se escribe al final con lo acumulado.

El bucle lee directamente de la entrada del script (`done` sin tubería delante). Si se escribiera `cat fichero | while read ...`, el bucle correría en una subshell y los contadores se perderían al terminar: es uno de los errores más típicos en Bash.

Los arrays asociativos hacen de diccionario: por IP, por ruta y, como conjunto, las rutas con 404. Ordenar sí se delega en `sort`, que lo hace mejor que cualquier bucle; la tubería del ranking sí corre en subshell, pero ahí solo se escribe.

`LC_ALL=C` fija el orden byte a byte. Sin él, `sort` usaría las reglas del idioma del sistema y el mismo script podría ordenar distinto en dos máquinas.

46. Alta masiva de usuarios a partir de un CSV

Difícil · Bash · 70 minutos · apuntesdam.com/ejercicios/bash/alta-masiva-de-usuarios

Solución (bash)

```
#!/bin/bash
# Genera las órdenes para dar de alta usuarios a partir de un CSV, sin ejecutarlas (modo simulación)
export LC_ALL=C

# «Lucía Martín Ruiz» → «lmartinr»: sin tildes ni eñes, en minúsculas y solo letras
normalizar() {
    local s=$1 i
    local con=(á é í ó ú ü ñ Á É Í Ó Ú Ü Ñ) sin=(a e i o u n a e i o u n)
    for i in "${!con[@]}"; do
        s=${s//${con[$i]}/${sin[$i]}}
    done
    s=${s,,}
    echo "${s//[a-z]/}"
}

read -r palabra lista_grupos
if [ "$palabra" != "GRUPOS" ] || [ -z "$lista_grupos" ]; then
    echo "La primera línea debe ser GRUPOS grupo1,grupo2..."
    exit 0
fi
IFS=',' read -r -a grupos <<< "$lista_grupos"

declare -A existe por_grupo
read -r palabra lista_existen
if [ "$palabra" = "EXISTEN" ]; then
    IFS=',' read -r -a cuentas <<< "$lista_existen"
    for c in "${!cuentas[@]}"; do existe[$c]=1; done
    numero=2
else
    echo "Falta la línea EXISTEN"
    exit 0
fi

altas=0
while IFS=';' read -r nombre ape1 ape2 grupo resto; do
    numero=$((numero + 1))
    [ -z "$nombre$ape1$ape2$grupo" ] && continue
    if [ -z "$nombre" ] || [ -z "$ape1" ] || [ -z "$grupo" ] || [ -n "$resto" ]; then
        echo "# Línea $numero: faltan o sobran campos"
        continue
    fi
    valido=0
    for g in "${!grupos[@]}"; do [ "$g" = "$grupo" ] && valido=1; done
    if [ $valido -eq 0 ]; then
        echo "# Línea $numero: grupo desconocido $grupo"
        continue
    fi

    # Primero se normaliza y después se toma la inicial: con LC_ALL=C, ${s:0:1} es un byte, y la í ocupa
    # dos
    n=$(normalizar "$nombre")
    a1=$(normalizar "$ape1")
    a2=$(normalizar "$ape2")
    base="${n:0:1}${a1:0:1}${a2:0:1}"
    login=$base
    n=2
    # Si el login ya existe (en el sistema o en este mismo fichero), se le añade un número
    while [ -n "${!existe[$login]}" ]; do
        login="$base$n"
        n=$((n + 1))
    done
    existe[$login]=1
    completo="$nombre $ape1${ape2:+ $ape2}"
    echo "useradd -m -s /bin/bash -g $grupo -c \"$completo\" $login"
    echo "chage -d 0 $login"
    por_grupo[$grupo]=$(( ${por_grupo[$grupo]:-0} + 1 ))
done
```

```

    altas=$((altas + 1))
done

resumen=""
for g in "${grupos[@]}"; do
    resumen+="${resumen:+, }$g ${por_grupo[$g]:-0}"
done
echo "# Altas: $altas ($resumen)"

```

El script lee la configuración (grupos y cuentas existentes) y después recorre los usuarios con un único bucle `while read`, validando cada línea antes de generar nada. Los errores se escriben como comentarios (`#`), así la salida completa se puede guardar en un fichero y ejecutarse tal cual cuando esté revisada.

La normalización se hace con expansiones de parámetros de Bash, que trabajan con las cadenas de bytes exactas de cada letra: funciona igual con cualquier configuración de idioma, cosa que no garantizan `tr` ni `iconv` con `TRANSLIT`.

El array asociativo `existe` es el conjunto de logins ocupados: incluye los del sistema y los que el propio script va asignando. Sin él, dos alumnos con el mismo login en el fichero tendrían la misma cuenta.

Modo simulación primero, ejecución después: es una práctica de administración responsable para cualquier script que cree, borre o modifique cuentas. `chage -d 0` obliga a cambiar la contraseña inicial, que se asignaría aparte (por ejemplo, con `chpasswd`).

47. Rotación de copias de seguridad abuelo-padre-hijo

Difícil · Bash · 70 minutos · apuntesdam.com/ejercicios/bash/rotacion-de-copias-abuelo-padre-hijo

Solución (bash)

```
#!/bin/bash
# Política abuelo-padre-hijo: 7 diarias, domingos de las últimas 4 semanas y el día 1 de los últimos 6
# meses
export LC_ALL=C

read -r palabra hoy
if [ "$palabra" != "HOY" ] || ! date -d "$hoy" +%F >/dev/null 2>&1 || [ "$(date -d "$hoy" +%F)" != "$hoy" ]; then
    echo "La primera línea debe ser HOY AAAA-MM-DD"
    exit 0
fi
dia_hoy=$(( (date -u -d "$hoy" +%s) / 86400 ))
mes_hoy=$(( 10#$(date -d "$hoy" +%Y) * 12 + 10#$(date -d "$hoy" +%m) ))

declare -A fichero_de
fechas=()
while read -r nombre; do
    [ -z "$nombre" ] && continue
    if [[ ! $nombre =~ ^backup-([0-9]{4}-[0-9]{2}-[0-9]{2})\.tar\.gz$ ]] || [ "$(date -d "${BASH_REMATCH[1]}" +%F 2>/dev/null)" != "${BASH_REMATCH[1]}" ]; then
        echo "Ignorado (nombre no válido): $nombre"
        continue
    fi
    fecha=${BASH_REMATCH[1]}
    if [ $fecha > $hoy ]; then
        echo "Ignorado (fecha futura): $nombre"
        continue
    fi
    fichero_de[$fecha]=$nombre
    fechas+=("$fecha")
done

if [ ${#fechas[@]} -eq 0 ]; then
    echo "No hay copias"
    exit 0
fi

# Las fechas en formato AAAA-MM-DD se ordenan bien como texto: de la más reciente a la más antigua
mapfile -t ordenadas <<(printf "%s\n" "${fechas[@]}" | sort -ru)
conservar=0
borrar=0
liberado=()
for i in "${!ordenadas[@]}; do
    f=${ordenadas[$i]}
    motivos=""
    [ "$i" -lt 7 ] && motivos="diaria"
    dias=$(( (dia_hoy - (date -u -d "$f" +%s) / 86400 ))
    if [ "$(date -d "$f" +%u)" -eq 7 ] && [ "$dias" -lt 28 ]; then
        motivos+="{motivos:+, }semanal"
    fi
    meses=$(( mes_hoy - (10#$(date -d "$f" +%Y) * 12 + 10#$(date -d "$f" +%m)) ))
    if [ "$(date -d "$f" +%d)" = "01" ] && [ "$meses" -lt 6 ]; then
        motivos+="{motivos:+, }mensual"
    fi
    if [ -n "$motivos" ]; then
        echo "CONSERVAR ${fichero_de[$f]} ($motivos)"
        conservar=$((conservar + 1))
    else
        echo "BORRAR ${fichero_de[$f]}"
        borrar=$((borrar + 1))
    fi
done
echo "Se conservan $conservar y se borran $borrar"
```

La política se reduce a tres preguntas independientes sobre cada copia (¿está entre las 7 últimas?, ¿es un domingo reciente?, ¿es un día 1 reciente?). Escribirlas por separado y acumular los motivos hace que el informe explique cada decisión, que es lo que permite fiarse de un script que borra.

Toda la aritmética de calendario la resuelve `date`: el día de la semana con `+%u`, el número de día con `+%s`, y la validación de fechas pidiéndole que reescriba la fecha. Contar meses como $\text{año} \times 12 + \text{mes}$ evita casos especiales al cambiar de año.

`[[ =~ ]]` con grupos extrae la fecha del nombre sin `cut` ni `sed`, y `mapfile` con sustitución de procesos convierte la salida de `sort` en un array en una sola línea.

El script solo decide: no borra. En la versión real, la salida se filtraría con `grep ^BORRAR` y se pasaría a `rm`, o mejor, se movería a una carpeta de cuarentena unos días antes de borrar definitivamente.

48. Monitor de espacio en disco con umbrales

Medio · Bash · 50 minutos · apuntesdam.com/ejercicios/bash/monitor-de-espacio-en-disco

Solución (bash)

```
#!/bin/bash
# Lee la salida de «df -P» y avisa de los sistemas de ficheros que se llenan
export LC_ALL=C

read -r palabra aviso critico
if [ "$palabra" != "UMBRALES" ] || ! [[ $aviso =~ ^[0-9]+$ && $critico =~ ^[0-9]+$ ]] || [ "$aviso" -ge
"$critico" ] || [ "$critico" -gt 100 ]; then
    echo "La primera línea debe ser UMBRALES aviso crítico (aviso < crítico <= 100)"
    exit 0
fi

# KiB → «12,3 GiB» con un decimal, redondeado
gib() {
    local decimas=$(( ($1 * 10 + 524288) / 1048576 ))
    echo "$((decimas / 10)),${decimas % 10} GiB"
}

filas=()
libre_total=0
read -r _ # cabecera de df
while read -r dispositivo bloques usado libre capacidad montaje; do
    [ -z "$dispositivo" ] && continue
    # Sistemas de ficheros virtuales: no ocupan disco
    case $dispositivo in
        tmpfs|devtmpfs|overlay|squashfs|udev|none) continue ;;
    esac
    [[ $dispositivo == /dev/loop* ]] && continue
    porcentaje=${capacidad%\%}
    libre_total=$((libre_total + libre))
    if [ "$porcentaje" -ge "$critico" ]; then estado="URGENTE"
    elif [ "$porcentaje" -ge "$aviso" ]; then estado="AVISO"
    else estado="OK"; fi
    linea=$(printf "%-8s %3d% %s -12s usado %s de %s" "$estado" "$porcentaje" "$montaje" "$(gib "$usado")"
        "$(gib $((usado + libre)))")
    if [ "$estado" != "OK" ]; then
        # Lo que hay que liberar para bajar del umbral de aviso: usado - aviso% del total, redondeado hacia
        arriba
        sobra=$(( usado * 100 - aviso * (usado + libre) ))
        liberar=$(( (sobra + 100 * 1048576 - 1) / (100 * 1048576) ))
        linea+=" · libera $liberar GiB para bajar del $aviso %"
    fi
    filas+=("$porcentaje|$montaje|$linea")
done

if [ ${#filas[@]} -eq 0 ]; then
    echo "No hay sistemas de ficheros que vigilar"
    exit 0
fi
# Del más lleno al más vacío; a igual porcentaje, por punto de montaje
printf "%s\n" "${filas[@]}" | sort -t'|' -k1,1nr -k2,2 | cut -d'|' -f3-
echo "Libre en total: $(gib "$libre_total")"
```

`df -P` existe precisamente para los scripts: garantiza una línea por sistema de ficheros y columnas fijas, aunque el nombre del dispositivo sea largo. Leerlo con `read` y nombres de variable deja el script legible.

Los umbrales separan la decisión (qué estado tiene cada disco) de la presentación. En la versión real, el estado URGENTE dispararía un correo o un mensaje, y el script se programaría en cron cada 5 minutos.

Toda la aritmética es entera: el redondeo al más cercano se consigue sumando la mitad del divisor y el redondeo hacia arriba sumando el divisor menos uno. Es la misma técnica que en cualquier lenguaje sin decimales (o con dinero en céntimos).

Guardar las líneas con una clave de ordenación delante y quitarla después (el patrón «decorar, ordenar, quitar») permite ordenar por un dato numérico sin perder el formato de la salida.

49. Analizador de la configuración de Nginx

Muy difícil · Bash · 90 minutos · apuntesdam.com/ejercicios/bash/analizador-de-configuracion-nginx

Solución (bash)

```
#!/bin/bash
# Revisa un fichero de sitios de Nginx (como /etc/nginx/sites-available/web) y avisa de los fallos
típicos
export LC_ALL=C

n=0 profundidad=0 numero=0
declare -a inicio fin puertos nombres raiz retorno proxy cert clave autoindex

while IFS= read -r linea || [ -n "$linea" ]; do
    numero=$((numero + 1))
    linea=${linea%%#*}                                # quita los comentarios
    linea="${linea%%[![:space:]]*}"                  # y los espacios de los extremos
    linea="${linea%%[![:space:]]*}"
    [ -z "$linea" ] && continue

    if [[ $linea =~ ^server[![:space:]]*\{$ ] && [ "$profundidad" -eq 0 ]; then
        n=$((n + 1)); inicio[$n]=$numero; profundidad=1
        continue
    fi
    if [[ $linea == "{" ]]; then
        # un bloque location o similar
        if [ "$profundidad" -eq 0 ]; then echo "Error: bloque fuera de un server en la línea $numero"; exit
        0; fi
        profundidad=$((profundidad + 1))
        continue
    fi
    if [ "$linea" = "}" ]]; then
        if [ "$profundidad" -eq 0 ]; then echo "Error: llave de cierre sin abrir en la línea $numero"; exit
        0; fi
        profundidad=$((profundidad - 1))
        [ "$profundidad" -eq 0 ] && fin[$n]=$numero
        continue
    fi
    if [ "$profundidad" -eq 0 ]; then echo "Error: directiva fuera de un server en la línea $numero"; exit
    0; fi

    linea=${linea%;}
    read -r directiva valor <<< "$linea"
    case $directiva in
        listen)
            puerto=${valor% *}; puerto=${puerto##*:}      # «[:]:443 ssl» → 443
            [[ $valor == *" ssl"* ] && puerto+=" ssl"
            [[ " ${puertos[$n]} " == " $puerto " ] || puertos[$n]="${puertos[$n]:+${puertos[$n]} }$puerto"
            ;;
        server_name) nombres[$n]=$valor ;;
        root) [ -z "${raiz[$n]}" ] && raiz[$n]=$valor ;;
        return) retorno[$n]="${valor##* }" ;;
        proxy_pass) [ -z "${proxy[$n]}" ] && proxy[$n]=$valor ;;
        ssl_certificate) cert[$n]=1 ;;
        ssl_certificate_key) clave[$n]=1 ;;
        autoindex) [ "$valor" = "on" ] && autoindex[$n]=1 ;;
    esac
done

if [ "$profundidad" -ne 0 ]; then echo "Error: llaves sin cerrar"; exit 0; fi
if [ "$n" -eq 0 ]; then echo "No hay ningún bloque server"; exit 0; fi

aviso=()
declare -A servidores_de      # «nombre puerto» → número de servidores
declare -A redirige           # nombre → 1 si su servidor del 80 redirige a https
declare -A en443
for i in $(seq 1 "$n"); do
    [ -z "${puertos[$i]}" ] && puertos[$i]="80"
    if [ -n "${retorno[$i]}" ]; then hace="redirige a ${retorno[$i]}"
    elif [ -n "${proxy[$i]}" ]; then hace="proxy a ${proxy[$i]}"
    elif [ -n "${raiz[$i]}" ]; then hace="sirve ${raiz[$i]}"
    else hace="no sirve nada"; fi
    aviso+=("$hace")
done

for i in $(seq 1 "$n"); do
    echo "Línea $i: ${aviso[$i]}"
done
```

```

echo "Servidor $i (líneas ${inicio[$i]}-${fin[$i]}): puerto ${puertos[$i]// / y } · ${nombres[$i]}:-(sin
nombre)} · $hace" | sed 's/ y ssl/ ssl/g'

if [[ " ${puertos[$i]}" == *" ssl"* ]]; then
    faltan=""
    [ -z "${cert[$i]}" ] && faltan="ssl_certificate"
    [ -z "${clave[$i]}" ] && faltan="${faltan:+$faltan y }ssl_certificate_key"
    [ -n "$faltan" ] && avisos+=("Servidor $i: usa ssl pero le falta $faltan")
fi
[ -n "${autoindex[$i]}" ] && avisos+=("Servidor $i: autoindex activado (muestra la lista de ficheros)")
[ "$hace" = "no sirve nada" ] && avisos+=("Servidor $i: no tiene root, return ni proxy_pass")

for nombre in ${nombres[$i]}; do
    for p in ${puertos[$i]// ssl/}; do
        servidores_de["$nombre $p"]=$(( ${servidores_de["$nombre $p"]:-0} + 1 ))
        if [ "$p" = "80" ] && [[ ${retorno[$i]} == https://* ]]; then redirige[$nombre]=1; fi
        [ "$p" = "443" ] && en443[$nombre]=1
    done
done
done

# Avisos por nombre de dominio, en orden alfabético
mapfile -t claves < <(printf "%s\n" "${!servidores_de[@]}" | sort)
for k in "${claves[@]"; do
    [ -z "$k" ] && continue
    [ "${servidores_de[$k]}" -gt 1 ] && avisos+=("${k%*}: aparece en ${servidores_de[$k]} servidores del
puerto ${k#* }")
done
mapfile -t seguros < <(printf "%s\n" "${!en443[@]}" | sort)
for nombre in "${seguros[@]"; do
    [ -z "$nombre" ] && continue
    [ -z "${redirige[$nombre]}" ] && avisos+=("$nombre: el puerto 80 no redirige a HTTPS")
done

if [ ${#avisos[@]} -eq 0 ]; then
    echo "Sin avisos"
else
    echo "Avisos:"
    for a in "${avisos[@]"; do echo "  - $a"; done
fi

```

El script es un analizador con estado: la profundidad de llaves dice si una línea pertenece a un servidor, a uno de sus `location` o a nada. Es la misma técnica que el conversor CSV con comillas, aplicada a un formato con bloques anidados.

Guardar los datos en arrays paralelos indexados por el número de servidor convierte el fichero en una pequeña tabla. Con ella, el resumen de cada servidor es un recorrido y los avisos son preguntas sobre la tabla.

Los avisos de dominios necesitan una segunda pasada que cruce información de varios servidores (el 80 de un dominio y su 443 suelen estar en bloques distintos). Por eso se recogen primero todos los datos y se analizan después.

En un servidor real, el script se ejecutaría antes de `nginx -t && systemctl reload nginx`, por ejemplo en un pipeline de despliegue: los fallos de sintaxis los encuentra Nginx; los de criterio, este script.