

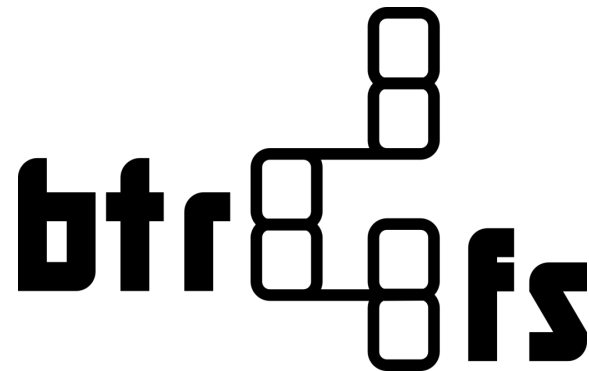


BTRFS: B-Tree File System



Avila Reyes Iker
Romero Pizano Christian Gustavo

¿Qué es?



- Es un sistema de archivos para Linux.
- Fue lanzado en 2007 y declarado estable hasta el 2013.
- Diseñado para ser robusto, eficiente y escalable.
- Incorpora compresión, control de versiones y detección de errores.
- Basado en la técnica Copy-On-Write.
- Usa una estructura de Árbol-B.

¿Qué es un sistema de archivos?

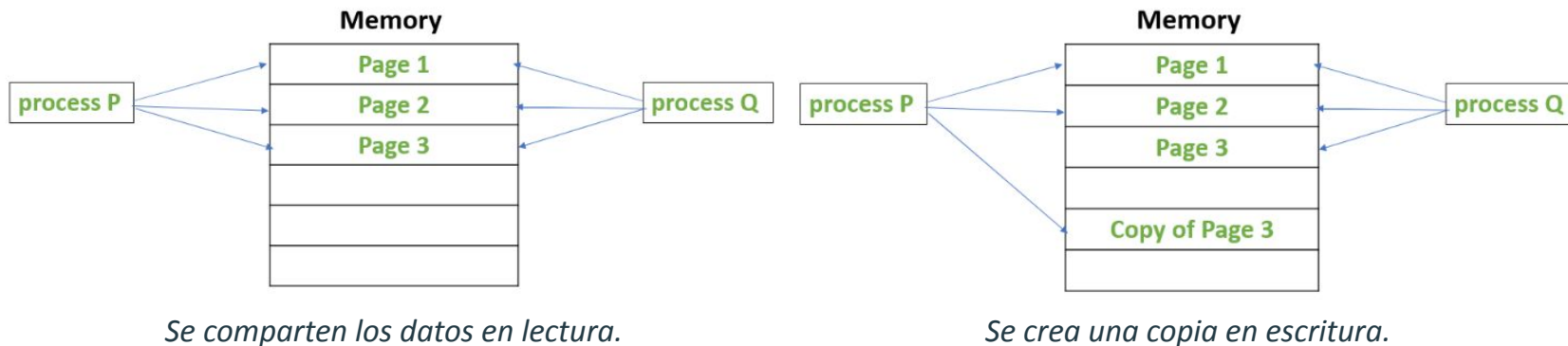
Es una estructura para organizar y administrar archivos en un dispositivo de almacenamiento, como pueden ser un disco duro, una unidad de estado sólido (SSD) o un dispositivo flash USB. Ejemplos:

- FAT y exFAT
- NTFS
- APFS
- HFS, HFS+
- Ext4



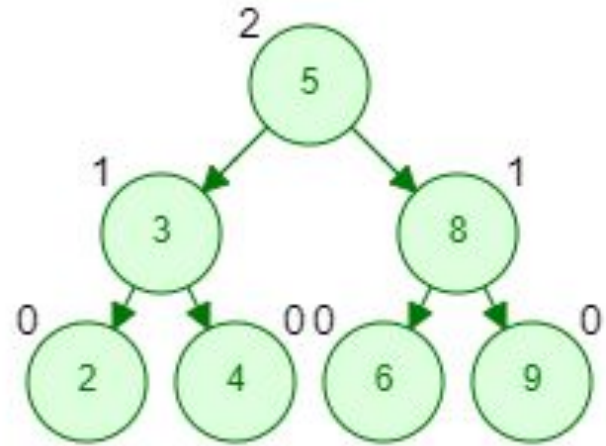
Copy-on-Write

- También conocido como “Shadowing” o “Implicit Sharing”.
- Comparte bloques de almacenamiento entre archivos y programas hasta que uno necesita ser modificado.
- No sobrescribe datos, solo guarda una copia en un nuevo bloque de memoria.
- Mejora la integridad de los datos y se facilita la creación de snapshots.

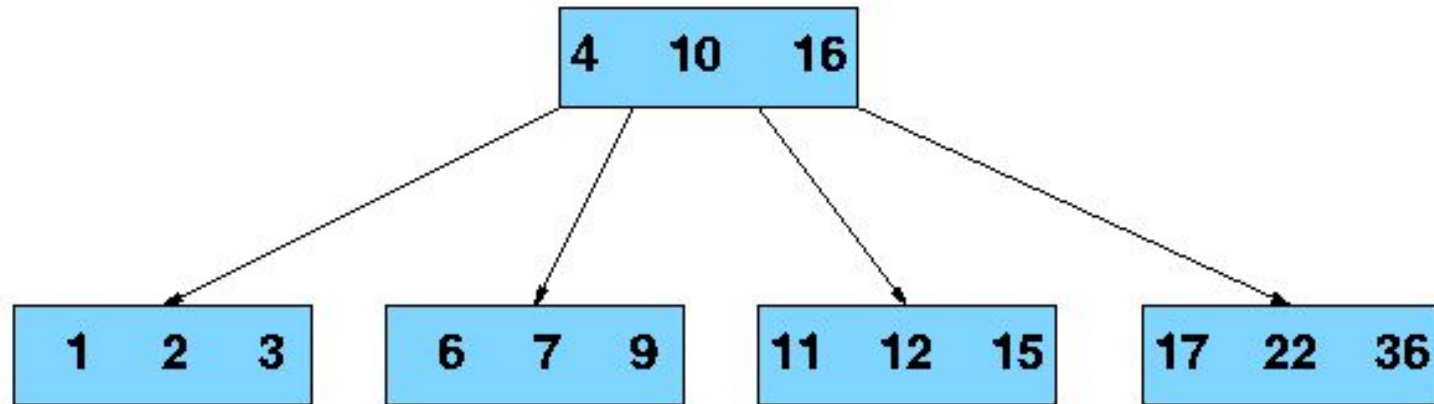


Árboles B

- Estructura de datos comúnmente utilizada dentro de los sistemas de archivos.
- Se usa para administrar grandes cantidades de datos.
- Mantiene a los datos organizados y fácilmente accesibles.
- Minimiza la cantidad de accesos al disco.



- Cada nodo puede almacenar ' $m-1$ ' claves, y ' m ' hijos.
- Altura reducida significativamente.
- Minimiza el número de accesos al disco
- Mejora el rendimiento en operaciones de inserción, eliminación y búsqueda.
- Todos sus nodos “hoja” se mantienen en el mismo nivel.



Ejemplo: Árbol con 3 claves y 4 hijos en el nodo raíz.



¿Como es el proceso de escritura?

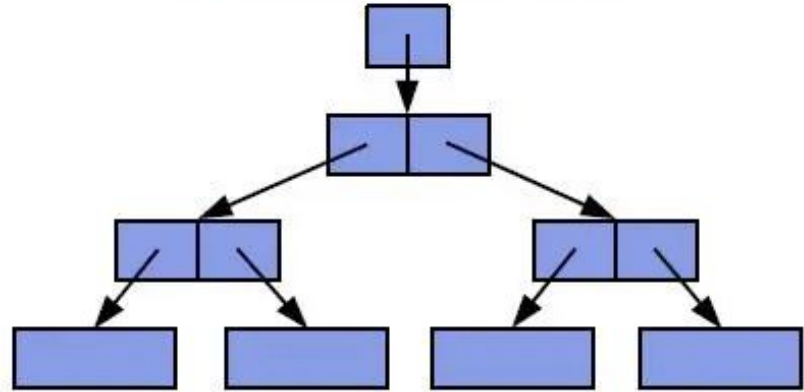
Ejemplo: Queremos modificar “archivo.txt”.



Paso 1 - Encontrar el archivo

Cada nodo contiene referencias a bloques de datos.

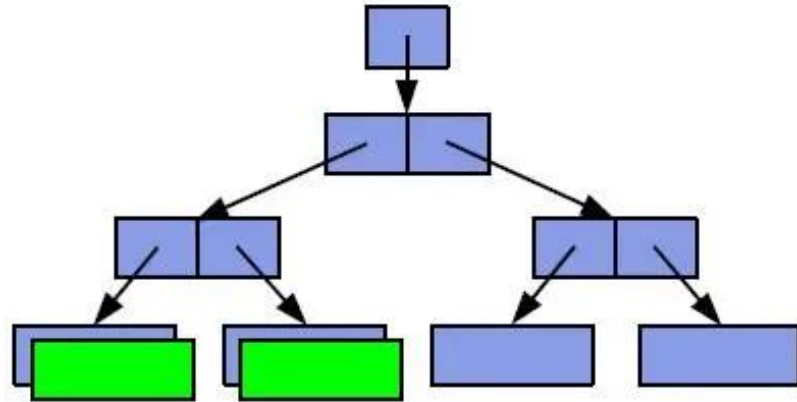
Para encontrar archivo.txt, recorre desde el Árbol Raíz → Árbol de Metadatos → Entrada del archivo.



Paso 2 - Copiar el archivo

Como se quiere modificar el archivo, se realiza una copia.

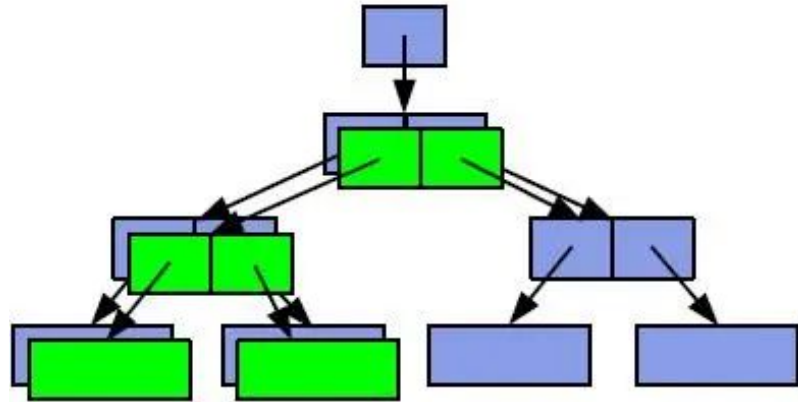
En lugar de sobrescribir el archivo sobre ese bloque, se asigna un nuevo bloque de memoria, y se escribe la nueva versión de los datos en ese bloque.



Paso 3 - Actualizar los nodos

Primero se copia el nodo padre del bloque de datos antiguo a un nuevo nodo y éste se actualiza para que apunte al nuevo bloque.

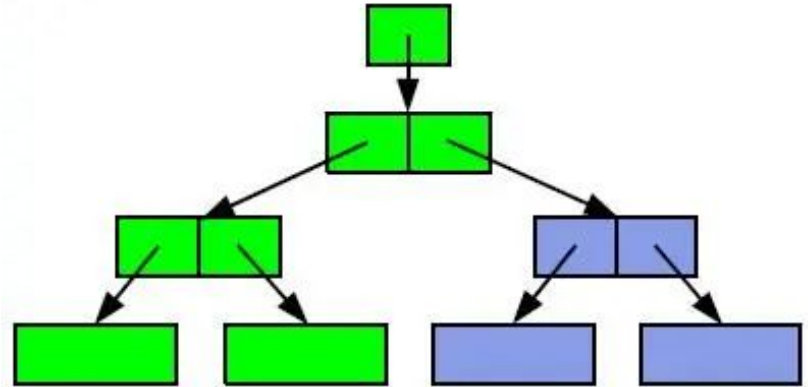
Este proceso se repite recursivamente hacia arriba hasta llegar a la raíz del árbol.



Paso 4 - Actualizar la raíz

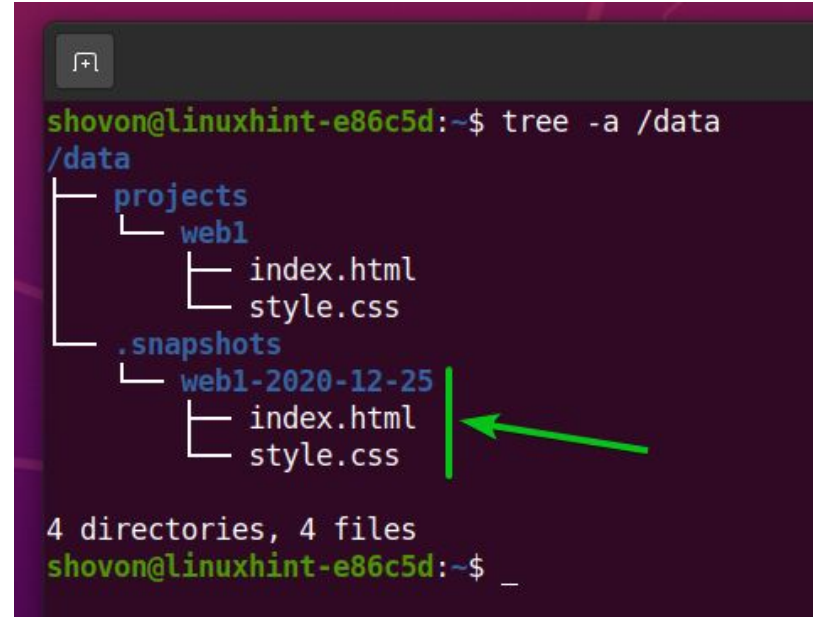
Se copia y modifica la raíz para que apunte a la nueva versión del árbol.

Al mantener la versión anterior del archivo, se puede regresar a ella en caso de corrupción o algún otro fallo.



Snapshots

- Subvolumenes de datos.
- Mantienen referencias a versiones anteriores de su estructura. Básicamente una imagen del sistema de archivos en un momento dado.
- Evitan duplicar información.
- Gracias a CoW, son relativamente ligeros.
- Solo almacenan diferencias respecto a los datos originales.




```
shovon@linuxhint-e86c5d:~$ tree -a /data
/data
├── projects
│   ├── web1
│   │   ├── index.html
│   │   └── style.css
│   └── .snapshots
│       └── web1-2020-12-25
│           ├── index.html
│           └── style.css
└── 4 directories, 4 files
shovon@linuxhint-e86c5d:~$ _
```

Snapshot de un folder.



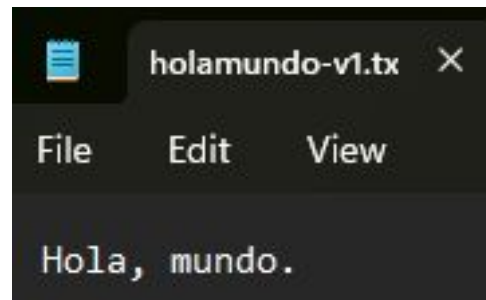
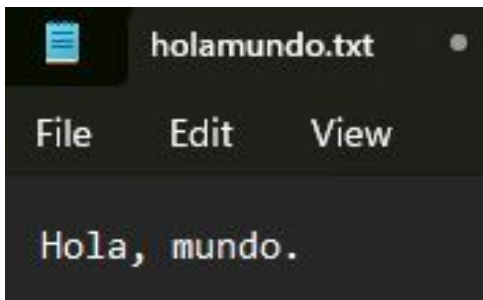
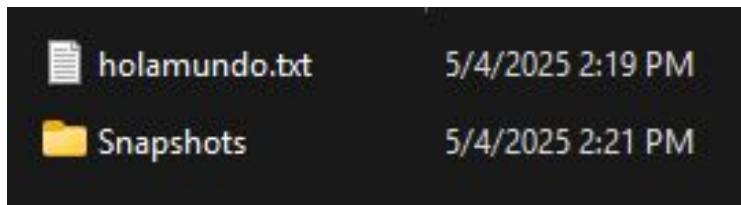
¿Cómo funcionan?

Ejemplo: Queremos modificar un archivo
“holamundo.txt”.



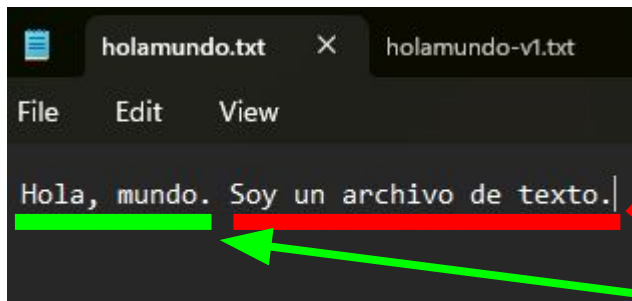
¿Cómo funcionan?

- Decidimos crear un snapshot en la carpeta “snapshots/holamundo-v1.txt”.
- Entramos a esta carpeta, abrimos el archivo de texto, y vemos que contiene el mismo “Hola, mundo.”



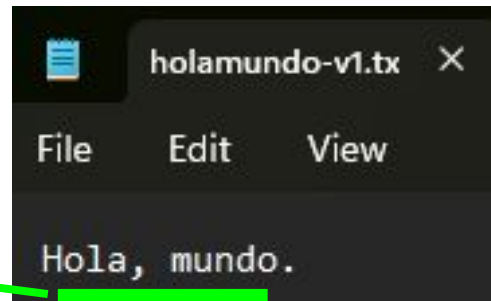
¿Cómo funcionan?

- Modificamos el “holamundo.txt” original para que diga: “Hola, mundo. Soy un archivo de texto.”.
- En vez de copiar completamente la cadena de “Hola, mundo.” se hace referencia al “Hola, mundo.” del snapshot.



holamundo.txt (modificado)

Se guarda en un nuevo bloque de datos



holamundo-v1.txt (snapshot)

Mismo bloque de datos



Otras aplicaciones de los snapshots

Detección de Errores

- Se realiza a través de sumas de verificación, o “checksums”.
- BTRFS realiza un checksum cada que se realiza una operación de escritura, y el resultado se guarda en un árbol de checksums.
- Cada vez que se lee un bloque, se vuelve a calcular su checksum y se compara con el almacenado para verificar que no haya corrupción.
- Ayuda a mantener la integridad de los datos.

Ejemplo de Checksums

Volvemos al ejemplo inicial, cuando modificamos “archivo.txt”:

1. Se realiza un checksum.
2. Los datos modificados se guardan en el bloque de código “B1”.
3. Se calcula su checksum, y esto nos da como resultado “C1”.
4. Se guarda “C1” en el árbol de checksums. Este nodo apunta al bloque “B1” donde se guardaron los datos.

Ahora intentamos leer el archivo:

1. BTRFS realiza la operación de checksum otra vez.
2. El resultado de esta operación es “C2”.
3. Busca en el árbol de checksum la referencia a “B1”.
4. Ve que está en “C1”, pero el resultado de su operación fué “C2”.
5. Concluye que hubo una corrupción en el archivo.

Seguridad y Fiabilidad

- *BTRFS soporta SELinux y AppArmor, permitiendo políticas de acceso granular. Ideal para entornos multi-usuario o contenedores, donde el aislamiento es crítico.*
- Funciones como RAID 5/6 aún son inestables. Se recomienda RAID 1/10 o verificar actualizaciones del kernel para usos avanzados.
- A pesar de las advertencias, sigue siendo una opción robusta y fiable :)

Comparativas

Aspecto	BTRFS	EXT4	ZFS
Snapshots	Nativos	✗	Avanzados
Checksums	✓	✗	✓
Kernel Linux	Nativo	Nativo	✗ Módulo externo
Recomendado	Contenedores	Uso general	Enterprise/NAS

Conclusiones