Linux File Systems

Task 1: Btrfs.

Btrfs (**B-tree file system)** is a GPL -licensed copy-on-write file system for Linux. The file system is still in development and marked as unstable. Btrfs implements the following features:

Cloning
Btrfs provides a *clone* operation which atomically creates a copy-on-write snapshot of a file.

Subvolumes and snapshots
Subvolumes allow a single instance of Btrfs to have multiple root directories, all using the file system as a pooled store. Subvolumes can be nested inside other subvolumes, where they appear as subdirectories. Subvolumes are always created empty.
Snapshots are writeable copy-on-write clones of whole subvolumes. Snapshots of individual subdirectories are not possible.

Send/receive
Given any pair of snapshots, Btrfs can generate a binary diff between them (via the `btrfs send` command) that can be replayed later (via `btrfs receive`) possibly on a different Btrfs file system.

Quotas
A *quota group* imposes an upper limit to the space a subvolume or snapshot may consume. A new snapshot initially consumes no quota because its data is shared with its parent, but thereafter incurs a charge for new files and copy-on-writes on existing files. When quotas are active, a qgroup is automatically created with each new subvolume/snapshot. These initial qgroups are building blocks which can be grouped into hierarchies to implement quota pools.

In-place ext3/4 conversion
The `btrfs-convert` tool exploits this ability to do an in-place conversion of any ext2,3,4 file system by nesting equivalent Btrfs metadata in its unallocated space. This produces a hybrid file system that can be mounted as either ext2,3,4 or Btrfs.

Seed devices
When creating a new Btrfs, an existing Btrfs can be used as a read-only "seed" file system. The new file system will then act as a copy-on-write overlay on the seed. The seed can be later detached from the Btrfs, at which point the rebalancer will simply copy over any seed data still referenced by the new file system before detaching.

Encryption
Encryption is unlikely to be implemented for some time, if ever, due to the complexity of implementation and pre-existing tested and peer-reviewed solutions.

Checking and recovery
Unix systems traditionally rely on “fsck” programs to check and repair filesystems. There is another tool, named btrfs-restore, that can be used to recover files from an unmountable filesystem, without modifying the broken filesystem itself.

Windows File Systems

Task 4: Compression in NTFS.
NTFS can compress files using LZNT1 algorithm. Files are compressed in 16-cluster chunks. With 4 kB clusters, files are compressed in 64 kB chunks. If the compression reduces 64 kB of data to 60 kB or less, NTFS treats the unneeded 4 kB pages like empty sparse fileclusters—they are not written. This allows not unreasonable random-

access times - the OS just has to follow the chain of fragments. However, large compressible files become highly fragmented since every chunk is less than 64KB becomes a fragment. Single-user systems with limited hard disk space can benefit from NTFS compression for small files, from 4 kB to 64 kB or more, depending on compressibility. Files less than 900 bytes or so are stored with the directory entry in the Master File Table.
The best use of compression is for files that are repetitive, written seldom, usually accessed sequentially, and not themselves compressed. Log files are an ideal example. Compressing system files used at bootup like drivers, NTLDR, winload.exe, or BOOTMGR may prevent the system from booting correctly. However, in later editions of Windows, compression of important system files is disallowed.
Files may be compressed or decompressed individually (via changing the advanced attributes) for a drive, directory, or directory tree, becoming a default for the files inside.