NetDrive Server

Introduction

The mTCP NetDrive server is a program written in Go that provides remote disk services for the NetDrive DOS device driver. The server has the following features:

- A text user interface console window so you can monitor the server and interact with it.
- The ability to handle many concurrently connected clients.
- Supports journaling of disk images allowing clients the ability to "undo" recent writes.
- Allows the sharing of a single disk image (read-only mode). A single disk image may also be shared in a simulated "read-write" mode where the writes from a particular client are only visible to that client, and only while that client is connected.
- The ability to create new floppy disk and hard disk images.

The server is cross-platform and runs on Windows 10, Windows 11, and Linux (64 bit x86 or Arm). The server requires no special permissions to run.

Running the server

The server has three major commands:

- create: create disk images (floppy or hard drive)
- image: perform maintenance on disk images
- serve: serve disk images to DOS clients

The server side program is named netdrive.exe (Windows) or netdrive (Linux). NetDrive takes a command, optional flags, and possibly positional parameters depending on the command being used.

The command line format is:

```
netdrive [<common_flags>] command [<command_flags>] <command_args>
```

Flags (common or command specific) are optional. There are two common flags that apply to all commands:

-log_file <filename>	Append log messages to the specified file (default: no logging)
-log_level <level>	Set the logging level to DEBUG, INFO, WARN or ERROR (default: INFO)

Run "netdrive help" to see the list of commands and common flags. Each command also has a help feature.

The create command is discussed in Creating disk images. The image command is discussed in Advanced disk images: Session scoped images and Journaled images.

Flags for serving images

The serve command starts the code that serves read and write requests from DOS clients. To start the server code use the following command:

```
netdrive <common_flags> serve <server_flags>
```

The optional server flags are:

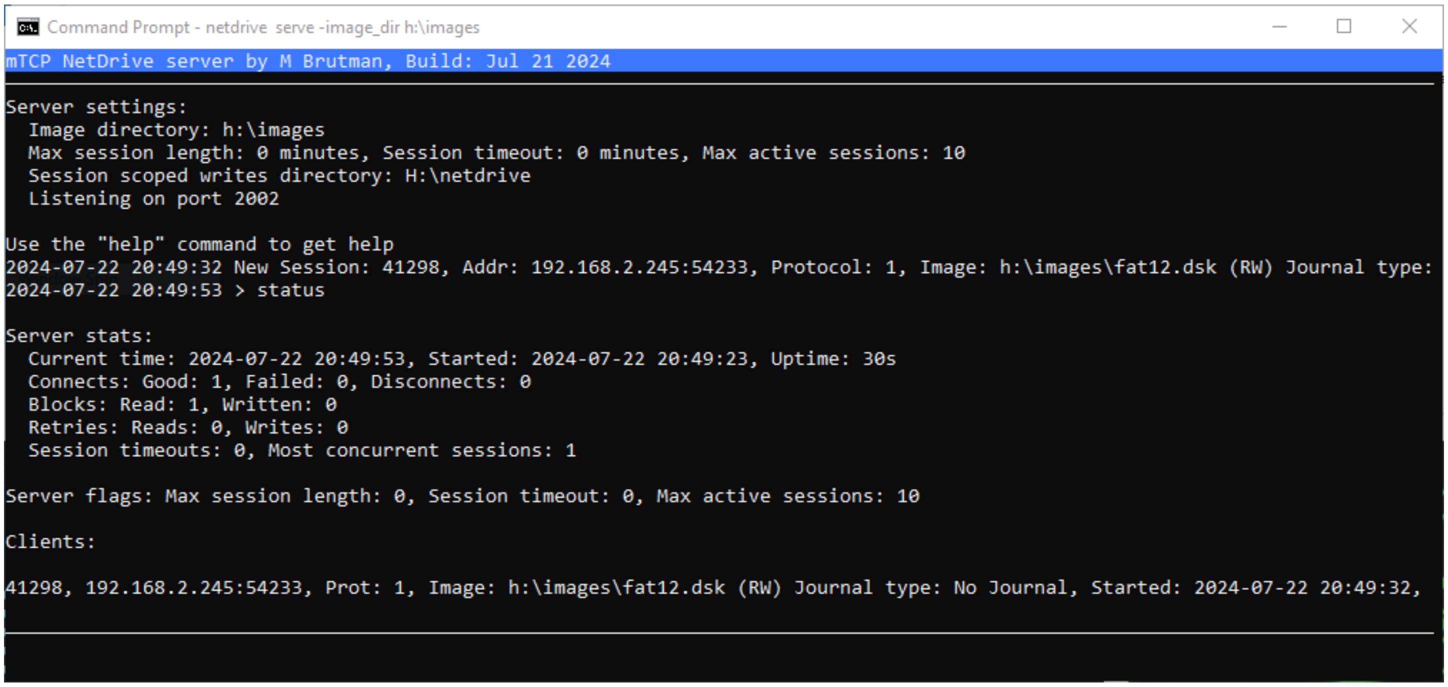
-port <n>	Serve using UDP port n (default is port 2002)
-image_dir <dir>	Use <dir> when looking for disk images (default: current dir)
-headless	Run in “headless” (non-interactive) mode
-timeout <n>	Set the session timeout in minutes (default: unlimited)
-max_time <n>	Set the max session duration in minutes (default: unlimited)
-max_active_sessions	Set the maximum number of concurrent sessions (default: 20)
-session_scoped_writes_dir <d>	Set the directory to use for Session scoped disk image journal files

Example command line:

```
netdrive -log_file serving.log serve -port 8086 -image_dir my_images
```

will start the server using UDP port 8086, log messages to serving.log, and look for images in the my_images subdirectory.

Run the program at the Windows command prompt (cmd.exe) or a terminal window in Linux. When the server starts it will draw a screen that consists of a messages area and a command input line at the bottom.



```
mTCP NetDrive server by M Brutman, Build: Jul 21 2024

Server settings:
  Image directory: h:\images
  Max session length: 0 minutes, Session timeout: 0 minutes, Max active sessions: 10
  Session scoped writes directory: H:\netdrive
  Listening on port 2002

Use the "help" command to get help
2024-07-22 20:49:32 New Session: 41298, Addr: 192.168.2.245:54233, Protocol: 1, Image: h:\images\fat12.dsk (RW) Journal type:
2024-07-22 20:49:53 > status

Server stats:
  Current time: 2024-07-22 20:49:53, Started: 2024-07-22 20:49:23, Uptime: 30s
  Connects: Good: 1, Failed: 0, Disconnects: 0
  Blocks: Read: 1, Written: 0
  Retries: Reads: 0, Writes: 0
  Session timeouts: 0, Most concurrent sessions: 1

Server flags: Max session length: 0, Session timeout: 0, Max active sessions: 10

Clients:

41298, 192.168.2.245:54233, Prot: 1, Image: h:\images\fat12.dsk (RW) Journal type: No Journal, Started: 2024-07-22 20:49:32,
```

Messages are displayed as machines connect, disconnect, or other notable things happen. Output from commands also appears in the messages area. The messages area holds up to 10,000 lines in a backscroll buffer. If the window is not wide enough some messages may look truncated; just make the window wider to see them.

There are four commands available:

help	Show help text
kill <session>	Force session number <session> to end
set	Set options (set run by itself displays help text)
status	Show the status of the server
quit	End the server

The following keys are also recognized:

Esc	Clear the command input area
Ctrl-L	Refresh the screen
Ctrl-C	Quit (Use it twice to confirm it.)
PgUp and PgDown	Scroll the messages are up or down

In headless mode all you will see is a message telling you that headless mode was specified. Ctrl-C is used to terminate the server in headless mode.

The server must be running before you try to connect a DOS machine to a virtual hard drive.

The -timeout flag can be used to protect the server against clients that crash before they can terminate their session cleanly. The server checks for inactive sessions every minute. If a client is inactive for longer than the session timeout period the server assumes that the client has crashed or gone away and the session is terminated. If that should happen and the client is still viable then the next operation that the client sends will result in an error because the session was terminated. If this should happen the client can use the DOS NETDRIVE.EXE command to disconnect and reconnect again.

The -max_time flag is different in that it sets the maximum amount of time that a client may be connected. This is useful when running a public server as it helps to prevent a user from monopolizing all of the available connections to the server.

The -max_active_sessions flag sets the limit for the number of clients that can be connected at the same time. This is also useful on a public server, as it helps the server protect itself from excessive loads. While the default limit is 20 clients, serving client requests is not that expensive so it is possible to serve hundreds of clients using a single server.

The -session_scopes_write_dir flag is used to set the directory where the server will create the per-session private journal files for images using that feature. See “mTCP NetDrive disk images and journals” for information on Session scoped disk images.

Serving standard disk images

If a disk image is read-write then only one DOS client may open and use the disk image at a time. This is because the file systems used by DOS (FAT) and DOS assume they have total control of the emulated disk and there is no way to coordinate writes from other possible clients. (Allowing multiple clients write access to a disk image at the same time will probably corrupt the file allocation table.)

When opening an image the server will check to see if the image is read-only. If it is then the DOS client will be able to read blocks from the image but writes will be blocked by the server's operating system. This is useful for protecting imaged floppy disks or for doing a forensic examination on a hard drive image.

Another advantage of read-only disk image files is that several DOS computers can mount the same read-only disk image file at the same time. Normally a disk image file can only be used by one computer at a time because writes to the disk image file can't be properly coordinated. Read-only disk image files do not have that problem and thus the server will allow several DOS machines to mount the same read-only disk image file at the same time.

Serving Journaled disk images

Note: See "Advanced disk images: Session scoped images and Journaled images" for more information on Journaled disk images and how they are different from standard image files.

Journaled disk images are effectively the same as read-write disk image files, so the same limitation of only one connected client at a time applies to them too. It does not matter if the base disk image is marked read-only or read-write, as writes will go to the journal file and not the base disk image. The journal file must be read-write.

Serving Session scoped disk images

Note: See "Advanced disk images: Session scoped images and Journaled images" for more information on Session scoped disk images and how they are different from standard image files.

Session scoped disk images are a special case of journaled images where the journal file is private to a specific client session and temporary, lasting only as long as the client session. As a result, Session scoped disk images are effectively read-only even though the client has the illusion of being able to do writes during their session. (The writes are lost at the end of the session.) This means that multiple clients can concurrently connect to Session scoped disk images, just like with read-only disk images.

The temporary session journal files will be created in the image directory by default, or in the directory specified by the -session_scoped_writes_dir flag if that flag was used. (These files are allowed to be placed separately from image files because of their transient nature.)

Session scoped disk images are useful for running software on images and allowing that software to make writes if it needs to, but without having to worry about backing up and restoring the original image after each connection. A good example of this would be if you were sharing a games archive you would want people to be able to play the games and modify high score files; on a read-only image they would get an error if a write was attempted but with session_scoped_writes the high score file can be created or updated while the shared image remains untouched.

Disk images

A disk image is a file that looks like a DOS disk; instead of blocks on a mass storage device there are blocks in the file. The disk image can be any format that is valid for DOS as long as 512 byte blocks are used. (DOS can use different block sizes but 512 byte blocks are the standard for disk storage.) Under normal operation reads and writes go directly into the disk image file, simulating what would happen on a real mass storage device. Like with a real mass storage device, writes persist across connections.

mTCP NetDrive uses standard disk images, which are plain files where each 512 byte block of the file can be treated as a sector or block of a mass storage device. There is no metadata in the file and no compression is used. These files are the same format used by Linux 'dd' and can be mounted by Linux systems using the loopback mount option. These files can be converted to and from other formats, such as those used by VirtualBox, QEMU or VMWare.

Disk image formats (FAT variations)

The mTCP NetDrive server basically just does read and write operations into disk image files on behalf of a connected DOS client. While the server really doesn't care about the contents of the file, DOS absolutely does care and DOS requires that the image file contents look like a valid FAT (File Allocation Table) based filesystem.

There are three versions of FAT supported by DOS:

- FAT12:
  - Required for DOS 2.x.
  - Supports hard drives up to 32 MB. (65535 sectors max)
  - Maximum number of files: Just under 4K
  - Max format: Just under 4K clusters at 16 sectors per cluster
  - Used by floppy disks and small hard drives
- FAT16 (also known as original FAT16):
  - Introduced with DOS 3.x.
  - Supports hard drives up to 32 MB, but more efficiently than FAT12. (Also 65535 sectors max)
  - Maximum number of files: Just under 64K
  - Max format: 64K clusters at 1 sector per cluster
- FAT16B (also known as final FAT16):
  - Introduced with Compaq MS-DOS 3.31, widely available in DOS 4.x and up
  - Supports hard drives up to 2GB
  - Max format: 64K clusters at 64 sectors per cluster

While FAT does not specify the sector or block size, it should be 512 bytes, which ensures maximum compatibility. (As of this writing mTCP NetDrive only supports 512 byte blocks.)

The number of sectors used for the File Allocation Table (FAT), the number of Root directory entries, the number of FAT copies, and other parameters can be varied. In general one should choose enough FAT sectors to be able to use the full size of the drive, 1 copy of the FAT, and a reasonable number of Root directory entries:

- FAT12: Each entry in the table is 12 bits. A full table requires 12 512 byte sectors.
- FAT16 and FAT16B: Each entry in the table requires 16 bits. A full table requires 256 512 byte sectors.
- Root directory entries: 32 bytes each, 16 per 512 byte sector.

The size of the table determines the maximum number of files across the entire hard disk. You can use less than a full table if you know that you do not need the maximum number of files.

DOS looks at the number of clusters to determine if the FAT is using 12 bit entries or 16 bit entries. If you use less than 4085 clusters DOS uses 12 bit FAT entries. If you use 4085 or more clusters then DOS uses 16 bit FAT entries. The decision between FAT16 and FAT16B is made by looking at the total number of sectors in the BPB; if it is 65535 or less then FAT16 is in use. If the total number of sectors is 0 then a 32 bit sector count from a field in the extended BPB is used and FAT16B is being used.

If you create a floppy image with NetDrive you will get one of the standard floppy images. If you create a hard disk image with NetDrive you can choose the FAT type, the number of FAT copies and the number of root directory entries. The size of the FAT will always be the maximum allowed and the cluster size will be the minimum necessary for the given disk size.

Creating disk images

You can use any tool that creates a standard disk image to create a disk image usable with mTCP NetDrive. (In Linux I've used dd and mkfs.vfat.)

The mTCP NetDrive server also has a command line function that creates new floppy or hard disk images. Floppy disk images are restricted to 8 well known formats, while hard drive images have more options.

To create a floppy image use the following command:

```
netdrive create floppy [create flags] <size_in_KB> <image_name>
```

where <size_in_KB> is a diskette capacity size in KB and <image_name> is the name of the image file you want to create. The following sizes are supported:

160 or 320	DOS 1.x 160 and 320K images (40 tracks, 8 sectors per track, 1 or two sides)
180 or 360	DOS 2.x and up 180 and 360K images (40 tracks, 9 sectors per track, 1 or two sides)
1200	DOS 3.0 and up 1200K images (80 tracks, 15 sectors per track, two sides)
720	DOS 3.2 and up 720K images (80 tracks, 8 sectors per track, 2 sides)
1440	DOS 3.3 and up 1440K images (80 tracks, 18 sectors per track, 2 sides)
2880	DOS 5.0 and up 2880K images (80 tracks 36 sectors per track, 2 sides)

The following optional flag can be used when creating floppy images:

-image_dir <dir>	Use <dir> when looking for disk images (default: current dir)

Example:

```
netdrive create floppy -image_dir floppy_images 1440 empty.dsk
```

will create a standard, 1.44 MB floppy image in the floppy_images directory.

To create a hard drive image use the netdrive create hd command:

```
netdrive create hd [create flags] <size_in_MB> <fat_type> <image_name>
```

Size may range from 1 to 2047 MB and fat type is either FAT12, FAT16, or FAT16B. The cluster size will be automatically selected based on the image size.

(Hint: to ensure the most efficient use of space choose hard drive sizes that are just under a power of 2. For example, 255 MB, 511 MB, 1023 MB, etc. The cluster size has to double near the power of 2 boundaries, so being just under a power of 2 makes the best use of the space and cluster size.)

The following optional flags can be used when creating hard drive images:

-image_dir <dir>	Use <dir> when looking for disk images (default: current dir)
-num_fats <n>	Create <n> copie of the File Allocation Table. (n must be 1 or 2)
-root_dir_entries <n>	Create a root directory with <n> entries

The default number of root directory entries is 512. By default two FAT copies are made although only one is actually ever used. (Blame Microsoft.)

Example:

```
netdrive create hd -num_fats 1 511 FAT16B bigdisk.dsk
```

will create a 511 MB FAT16B drive image with just one FAT copy.

Manipulating disk images with Linux

Linux makes it fairly easy to create and work with disk images:

- The “dd” command can be used to create an empty file for a new disk image. Use /dev/zero as the source.
- The mkfs.fat command can write a filesystem on the new disk image.
- The mount command can mount images in read-only mode for inspection or read-write mode for manipulation. (Use the -o loop,check=normal -t msdos options. You might also need to use the uid option to be able to make changes.)
- mtools can be used to manipulate image files.

Remember that two systems can’t write to the same image at the same time. This includes a remotely connected DOS system and a Linux system mounting the image using the loopback option.

Caution: Linux tools are not capable of manipulating mTCP NetDrive Journaled disk images because they are not aware of the journal file that mTCP NetDrive uses. You can use Linux tools to create new image files but if you turn an image file into a Journaled disk image you should not use Linux tools to manipulate it.

Manipulating disk images with Windows

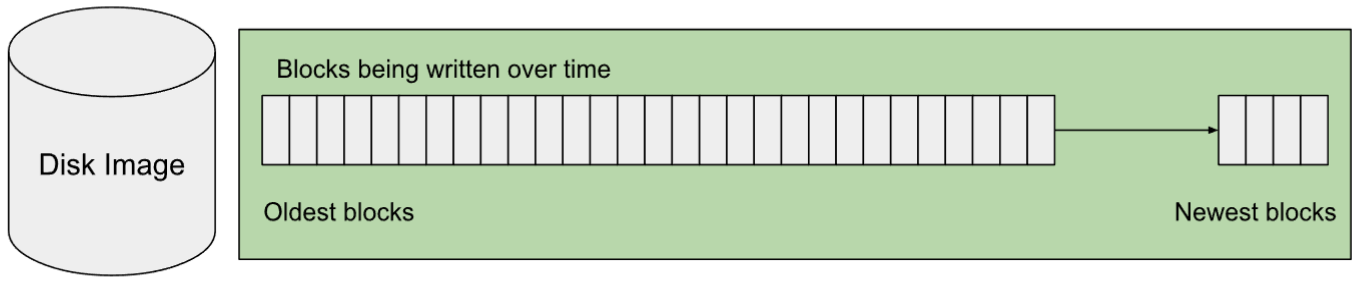
(I’m not as familiar with manipulating disk images with Windows, so if you have a tip please send it to me.)

- WinImage seems to be able to open and manipulate floppy and hard drive disk images. (I tried it a little bit during a trial period; it was awkward but it might be usable.)
- qemu-img can convert hard drive image formats such as VMDK, VDI, VHD, and raw (what NetDrive uses.)
- Virtual machines such as VirtualBox can mount floppy images in their emulated floppy drives.

Caution: Windows tools are not capable of manipulating mTCP NetDrive Journaled disk images because they are not aware of the journal file that mTCP NetDrive uses. You can use Windows tools to create new image files but if you turn an image file into a Journaled disk image you should not use Windows tools to manipulate it.

Advanced disk images: Session scoped images and Journaled images

In computing, a journal is a data structure or a file where pending transactions are recorded. When a journal file is used with a disk image, newly written data goes to the journal file and the disk image is left unmodified. When a block is read the journal file is checked first to get the newest version of the block. If the block is not in the journal file (meaning the block has not been written to) then the block is fetched from the disk image.



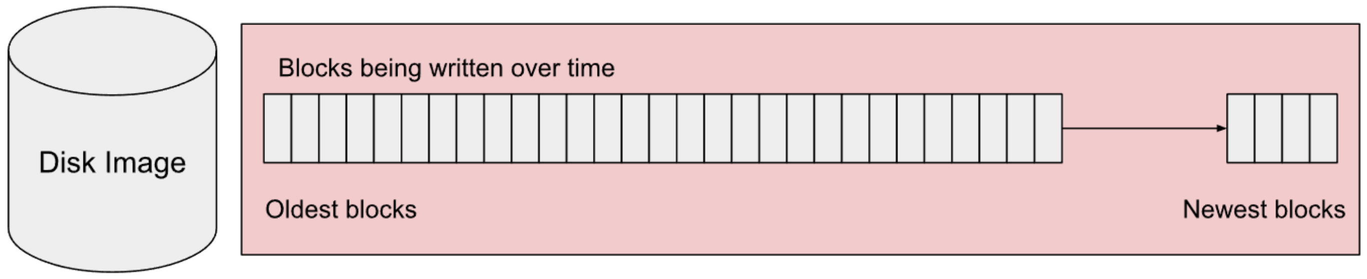
The journal file (green) takes all writes instead of the disk image.

Using a journal file for writes allows one to quickly undo those writes by just simply deleting the journal file. You can also "undo" a subset of writes (the newest writes) by truncating the journal file, effectively wiping out writes past the truncation point.

This property of journal files is used to implement two additional disk image types in mTCP NetDrive, each of which has an advanced feature.

Session scoped disk images

When a DOS client connects to the server it starts a session, which is used to track the state of the client and the disk image while the client is connected. When a Session scoped disk image is used a private journal file for the session is added to a disk image, allowing the client to have the illusion of disk writes that are only visible during that session. When the session ends the journal file is deleted, making it look as though those writes never happened.



Deleting the per-session journal (denoted in red) throws away any writes the connected client made.

Since all writes go to the private journal file the disk image itself is never altered, effectively making the disk image read-only. As the disk image is effectively read-only, multiple clients can connect to and use the same disk image when it is a Session scoped disk image.

This can be used to add a "demo mode" disk to images, allowing people to interact with them (including writes) without worrying about the possibility of permanent changes to the disk image. A good example of this is a public server with a DOS shareware game collection on it; a Session scoped disk image lets you share one copy of the disk while giving everybody the illusion of being able to write to it. (At least temporarily during a session.)

To mark a disk image as a session scoped disk image create a new file in the same directory as the image with the same name, but add ".session_scoped" to it. For example, if your disk image is called "dos_utilities.dsk" then create "dos_utilities.dsk.session_scoped" in the same directory to turn on session scoped writes for that disk image. The file can be empty - it just has to exist.

To disable the session scoping feature on a disk image just delete that file.

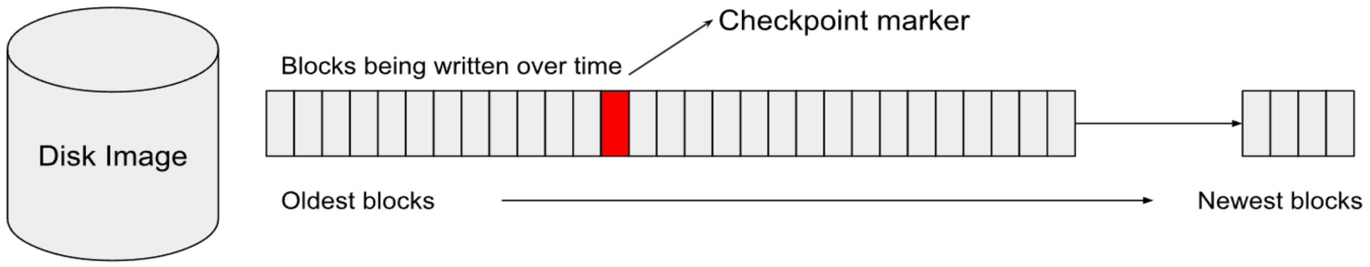
When a client connects to a Session scoped disk image new writes go to a journal that is private to the session. The filename for the journal has the session number as the name and '.log' as the extension, and it is only created if the client attempts to write data. At the end of the session the journal file will be deleted.

By default the journal files for Session scoped disk images will be in the same directory as the disk images. Use the -session_scoped_writes_dir flag when starting the server to put the Session scoped disk image journal files in a different directory.

No maintenance operations are needed for Session scoped disk images.

Journaled disk images

A Journaled disk image is a disk image that has a single, persistent journal file that survives across sessions. This allows a DOS client to undo writes that are in the journal file, effectively "going back in time" on the disk image, including writes that were made during earlier sessions. As the journal file for the image persists across sessions there can only be one journal file, so this resembles a normal read-write image but with the special undo feature. Disk images with this support are known as Journaled disk images.



Deleting new blocks after the checkpoint marker block effectively "goes back in time" to when the checkpoint marker was written.

(Note: While Session scoped disk images use a journal file, the implementation is simpler than the full journalling implementation used by Journaled disk images. As a result GoBack/Undo is not available when using a Session scoped disk image.)

Enabling journaling on a disk image

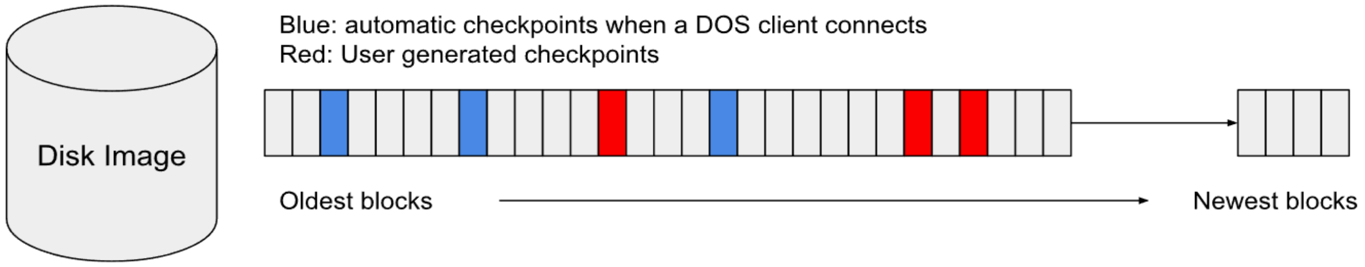
To enable journaling on a disk image create a new file in the same directory as the image with the same name but add ".journal" to it. For example, if your disk image is called "dos_utilities.dsk" then create "dos_utilities.dsk.journal" in the same directory to turn on journaling for that disk image. The file must start as an empty file with 0 length.

If the journal file exists, writes to the image will be appended to the journal file. Do not delete this file - if you do you will lose those writes.

In addition to the ".journal" file a new file with the same name as the disk image and ".journal.map" appended to the end will also be created in the directory. This file (the "map" file) is used to speed up the process of re-connecting to an image. If this file gets deleted by accident it will be recreated and no data will be lost.

Showing checkpoint markers in a journal

Checkpoint markers are metadata written to a journal that are used to mark points in time that are available for GoBack function. Each time a client connects to a journaled disk image a generic "session start" checkpoint marker is written to the journal. The user can also add their own checkpoint markers at any time.


Blue: automatic checkpoints when a DOS client connects
Red: User generated checkpoints
Disk Image
Oldest blocks
Newest blocks

To see the checkpoint markers in a journal use the netdrive image list_cp command.

While the DOS client also has a list_cp command, it can only show a limited number of checkpoint markers because it has to transfer them over the network and buffer space is limited. The version of list_cp in the netdrive server does not have that limitation.

Undo writes (GoBack)

(This function is the same as the goto_cp function in the DOS client.)

A journaled drive supports to GoBack feature, which allows you to "go back" in time with respect to writes, making the drive look like an earlier version of itself. This is effectively an "undo" feature where you select how far back you want to go.

Use the "netdrive image goto_cp" command to delete all of the writes past a specified checkpoint marker. (Use the "netdrive list_cp" command first to see the available checkpoint markers.) This operation can not be un-done - if you use it those writes are lost and can not be recovered. The journal file will be truncated at the point you specify, throwing those writes away.

Compacting journal files

All writes to a journaled disk image get appended to the journal file, causing the journal file to grow in size. Only the newest write to a specific block is valid, as the previous writes represent stale data that was valid at an earlier point in time. While the GoBack feature makes use of this behavior it can result in a significant amount of dead space inside of the journal file. This is especially true when the same set of blocks is repeatedly written.

The compaction process scans a journal file and copies the active journal records to a new journal file. Checkpoint markers are preserved and there should be no difference between a journal file before and after compaction except for the size, which will be smaller.

To run compaction on a journal file use the “netdrive image compact_journal” command.

Committing journaled writes

As discussed above, all writes to a journaled disk image get appended to the journal file, causing the journal file to grow in size. A large journal file should not be a problem for the system but eventually you will want to make the journalled writes permanent in the disk image file.

To do this use the “netdrive image commit_writes” command. The command will let you commit all writes in a journal file, or only writes up to a specified checkpoint marker. After the command is run the journal will either be empty (all writes committed) or will have only writes made after the specified checkpoint marker.

Disabling journaling on a disk image

The journal file on a disk image contains your writes to the disk image, so do not just delete it - if you do you will lose those writes.

To disable journaling while preserving your changes you need to commit those changes to the disk image file:

- Make sure the disk image is not in use by a connected client.
- Use the “netdrive image commit_writes” command to move the writes in the journal file into the disk image. This makes the writes permanent in the disk image.
- After the commit_writes operation the .journal file should be empty (0 length). It is now safe to delete the .journal and .journal.map files.

To disable journaling while throwing away any changes in the journal file just delete the .journal and .journal.map files.

After the .journal and .journal.map files are done the image will be treated like a standard disk image and any writes will be made inside the disk image file.

Other operations on disk images

Copying and renaming disk images

Disk images are just normal files so you can copy them or rename them as needed.

If a disk is journaled be sure to copy or rename the .journal file too, otherwise you will lose the writes to the image. You should also copy the .journal.map file but if you forget it will be recreated.

Comparing disk images

The “netdrive image compare” command can be used to compare two images for equivalence. Journaled and non-journaled images are supported, and the journals can be of different sizes.

This command is basically only useful as a safety check after committing writes to a disk image. If you make a backup copy of a disk image and (and its journal) before committing writes to the disk image you can compare the backup to the new version of the disk image and they should be equivalent.