|  |  |  |
| --- | --- | --- |
| Bochs User Manual | | |
| [Prev](http://docs.google.com/cirrus-notes.html) | Chapter 8. Tips and Techniques | [Next](http://docs.google.com/using-bximage.html) |

8.20. Disk Image Modes

Bochs can handle independent disk image format for each disk present on the ata interfaces. The disk image type is selected in the configuration file by the "mode" option of the ataX-xxx directives. Example:

ata0-master: type=disk, mode=flat, path=10M.sample, cylinders=306, heads=4, spt=17

**Note:** If unspecified, the default "mode" is flat.

**Table 8-5. Supported Disk Modes**

|  |  |  |
| --- | --- | --- |
| Name | Description | Features |
| flat | one file, flat layout | accessible with mtools or winimage-like tools |
| concat | multiple files, concatenated | mappable to contained partitions |
| external | accessed through an external C++ class | developer specific, needs a C++ class at compile time |
| dll | accessed through a DLL | developer specific, windows only |
| sparse | up to 10 layers stackable files | commitable, rollbackable, growing |
| vmware3 | vmware3 disk support | vmware version 3 compatibility |
| vmware4 | vmware4 disk support | vmware version 4 (VMDK) compatibility |
| undoable | read-only base file with a commitable redolog | commitable, rollbackable |
| growing | one growing file | growing |
| volatile | read-only base file with a volatile redolog | always rollbacked |
| vpc | VirtualPC disk support | fixed / dynamic size supported |
| vbox | Oracle(tm) VM VirtualBox disk support | VDI version 1.1 fixed / dynamic size supported |
| vvfat | local directory appears as VFAT disk (with volatile redolog) | optional commit or rollback |

8.20.1. flat

8.20.1.1. description

In flat mode, all sectors of the harddisk are stored in one flat file, in lba order.

8.20.1.2. image creation

Flat disk images can be created with the bximage utility (see [Section 8.21](http://docs.google.com/using-bximage.html) for more information).

8.20.1.3. path

The "path" option of the ataX-xxx directive in the configuration file must point to the flat image file.

8.20.1.4. external tools

Flat images content can be accessed from the host by the following tools :

* mtools (see [Section 8.3](http://docs.google.com/mtools.html))
* mount with a loopback (see [Section 8.8](http://docs.google.com/loop-device-usage.html))
* Winimage / DiskExplorer (see [Section 8.5](http://docs.google.com/winimage.html))
* Bochs Tools (see [Section 8.4](http://docs.google.com/bochs-linux-disktools.html))

8.20.1.5. typical use

Flat mode is Bochs default harddisk layout. This is also the layout of disk images provided on Bochs websites.

8.20.1.6. limitations

On some host OSes, Bochs flat disk images are limited to 2GiB.

8.20.2. concat

8.20.2.1. description

In concat mode, all sectors of the harddisk are stored in several flat files, in lba order.

8.20.2.2. image creation

Disk images for the usage in 'concat' mode can be created as 'flat' mode image files with the bximage utility (see [Section 8.21](http://docs.google.com/using-bximage.html) for more information).

8.20.2.3. path

The "path" option of the ataX-xxx directive in the configuration file must point to the first file (e.g. win95-1). The lower layer files names are found by adding 1 to the last character (e.g. win95-2, win95-3, etc.).

8.20.2.4. external tools

If every single file contains a complete partition, they can be accessed with same tools as the 'flat' mode images.

8.20.2.5. typical use

If the partition sizes and file sizes are set up correctly, this allows you to store each partition in a separate file, which is very convenient if you want to operate on a single partition (e.g. mount with loopback, create file system, fsck, etc.).

8.20.2.6. limitations

On some host OSes, there is a limit of 2GiB per file.

8.20.3. external/dll

8.20.3.1. description

This mode is only useful for developers and needs an additional C++ class compiled in, or an additional DLL linked to Bochs.

8.20.4. sparse

8.20.4.1. description

Sparse disk support has been added by JustinSB. Sparse disk features are:

* Large hard drive can be created, and only used space will be stored in the file. In practice, on Unix, this is not a large gain as it is done anyway.
* Multiple sparse drive images can be mounted on top of each other. Writes go to the top image. This allows several similar configurations to share a master "base" file, and also allows file system rollback or no-write options. Up to 10 disk images can be layered on top of each other.

8.20.4.2. image creation

Sparse disk images must be created with the bximage utility (see [Section 8.21](http://docs.google.com/using-bximage.html) for more information). Be sure to enter "sparse" when selecting the image type.

8.20.4.3. path

The "path" option of the ataX-xxx directive in the configuration file must point to the top layered file. The lower layer files names are found by substracting 1 from the last character (must be a digit)

8.20.4.4. external tools

No external tool support Sparse disk images yet.

8.20.4.5. typical use

8.20.4.5.1. Space Saving

Create a sparse disk image using bximage. Set size to eg 10GB. Only allocated space will be stored, so your drive image should be only about as large as the files stored on it.

8.20.4.5.2. Disk Rollback

* Create a sparse disk image called "c.img.0". Point .bochsrc at "c.img.0". In bochs, install your favorite OS. Switch off bochs.
* Create a sparse disk image (of the same size) and name it "c.img.1". Point .bochsrc at "c.img.1" "c.img.0" is visible, but all writes go to "c.img.1". After using bochs, you can simply delete "c.img.1" to undo changes and go back to a clean OS install.

8.20.4.5.3. Disk Optional Commit

* Create a sparse disk image called "c.img.0". Point .bochsrc at "c.img.0". In bochs, install your favorite OS. Switch off bochs.
* Create a sparse disk image (of the same size) and name it "c.img.1". Point .bochsrc at "c.img.1" "c.img.0" is visible, but all writes go to "c.img.1". After using bochs, if you want to keep the changes, use the (currently non-existent) merge utility to make a single unified drive image.
* Alternatively simply create a new partition on top called "c.img.2".

8.20.4.5.4. Common Base

* Create a sparse disk image called "base.img". Point .bochsrc at "base.img". In bochs, install your favorite OS. Switch off bochs.
* Create a sparse disk image (of the same size) and name it "www.img.1". Make "wwww.img.0" a symlink to "base.img". Point .bochsrc at "www.img.1". Using bochs, install a webserver.
* Create a symlink to "base.img" called "db.img.0". Create a sparse disk image (of the same size) and name it "db.img.1". Point .bochsrc at "db.img.1". Using bochs, install a database server.

Now both a database server and webserver can be run in separate virtual machines, but they share the common OS image, saving drive space.

8.20.4.6. limitations

There is a need for supporting utilities (yet unwritten) :

* to merge two sparse disk images into a single image
* to defragment a sparse disk image and remove unused space

8.20.5. vmware3 / vmware4

8.20.5.1. description

Sharvil Nanavati has added vmware3/4 disk image support into Bochs for Net Integration Technologies, Inc. You should be able to use disk images created by vmware version 3 and 4.

8.20.5.2. image creation

Create such disk image with vmware version 3 or 4.

8.20.5.3. path

The "path" option of the ataX-xxx directive in the configuration file must point to the vmware3/4 disk image.

8.20.5.4. external tools

 In addition to the utilities provided by VMware Inc. some other freeware tools are available.

8.20.5.5. typical use

If you want to use an existing vmware3/4 disk image.

8.20.5.6. limitations

Only vmware versions 3 and 4 disk image files are supported.

8.20.6. undoable

8.20.6.1. description

Undoable disks are commitable/rollbackable disk images. An undoable disk is based on a read-only image, associated with a growing redolog, that contains all changes (writes) made to the base image content. Currently, base images of types 'flat', 'sparse', 'growing', 'vmware3', 'vmware4' and 'vpc' are supported.

This redolog is dynamically created at runtime, if it does not previously exists.

All writes go to the redolog, reads are done from the redolog if previously written, or from the base file otherwise.

If unspecified with the "journal" option of the ataX-xxx directive, the redolog file name is created by adding a ".redolog" suffix to the base image name.

File size of the redolog can grow up to the total disk size plus a small overhead due to internal data management (about 3% for a 32MiB disk, less than 0.5% for a 2GiB disk).

After a run, the redolog will still be present, so the changes are still visible the next time you run Bochs with this disk image.

8.20.6.2. image creation

The flat / sparse / growing disk images must be created with the bximage utility (see [Section 8.21](http://docs.google.com/using-bximage.html) for more information). The growing redolog is created automatically if needed.

8.20.6.3. path

The "path" option of the ataX-xxx directive in the configuration file must be the base image name. The redolog name can be set with the "journal" option of the same directive. If not set, the redolog name is created by adding the ".redolog" suffix to the base image name.

8.20.6.4. external tools

To access the flat disk image content, see [Section 8.20.1.4](http://docs.google.com/harddisk-modes.html#HARDDISK-MODE-FLAT-TOOLS) for available tools.

**Note:**  The up-to-date content can only be seen after you commit the redolog to the flat file with the bximage utility.

8.20.6.5. typical use

* **Commit**  
  After a run, the redolog can be committed (merged) to the base image with the bximage utility.
* **Rollback**  
  After a run, the redolog can be rollbacked (discarded) by simply deleting the redolog file.
* **Common Base**  
  One base disk image with a guest OS installed can be used to install different software as described above for the "sparse" disk image mode.
* **Harddisk Image on a Read-Only Medium**  
  In the 'undoable' mode, the base file is always opened in read-only mode, so it can safely be stored on a read-only medium (for example on a cdrom). In that case it is recommended to specify the redolog file with the "journal" option.

8.20.6.6. limitations

The "undoable" disk depends on the limitations of base disk image used.

8.20.7. growing

8.20.7.1. description

Growing disk images start as a small files, and grow whenever new data is written to them.

Once a sector is written in the growing file, subsequent writes to the same sector will happen in place.

File size of Growing disk images can go up to the total disk size plus a small overhead due to internal data management. (about 3% for a 32MiB disk, less than 0.5% for a 2GiB disk).

8.20.7.2. image creation

Growing disk images must be created with the bximage utility (see [Section 8.21](http://docs.google.com/using-bximage.html) for more information). Be sure to enter "growing" when selecting the image type.

8.20.7.3. path

The "path" option of the ataX-xxx directive in the configuration file must be the growing image name.

8.20.7.4. external tools

No external tool support Growing disk images yet.

8.20.7.5. typical use

Growing disk images can be used whenever you want to maximize disk space. However, please note that Bochs will not check if enough disk space is available before writing new data. If no disk space is available, a panic will occur.

8.20.7.6. limitations

The size of the virtual disk is currently limited to 8 TB, but the maximum size of the image file depends on host OS limitations.

8.20.8. volatile

8.20.8.1. description

Volatile disks are always-rollbacked disk images. An volatile disk is based on a read-only image, associated with a growing redolog, that contains all changes (writes) made to the base image content. Currently, base images of types 'flat', 'sparse', 'growing', 'vmware3', 'vmware4' and 'vpc' are supported.

The redolog is dynamically created at runtime, when Bochs starts, and is deleted when Bochs closes (win32) or just after it has been created (Unix).

All writes go to the redolog, reads are done from the redolog if previously written, or from the base file otherwise.

If unspecified with the "journal" option of the ataX-xxx directive, the redolog file name is created by adding a ".redolog" suffix to the base image name.

File size of the redolog can grow up to the total disk size plus a small overhead due to internal data management (about 3% for a 32MiB disk, less than 0.5% for a 2GiB disk).

After a run, the redolog is not any more present, so the changes are discarded.

8.20.8.2. image creation

The flat / sparse / growing disk images must be created with the bximage utility (see [Section 8.21](http://docs.google.com/using-bximage.html) for more information). The growing redolog is created automatically.

8.20.8.3. path

The "path" option of the ataX-xxx directive in the configuration file must be the base image name. The redolog name can be set with the "journal" option of the same directive. If not set, the redolog name is created by adding the ".redolog" suffix to the base image name. A random suffix is also appended to the redolog name.

8.20.8.4. external tools

See [Section 8.20.1.4](http://docs.google.com/harddisk-modes.html#HARDDISK-MODE-FLAT-TOOLS) for tools to access the flat disk image content.

8.20.8.5. typical use

* **Repeatable simulations**  
   to be completed
* **Multiple Bochs instances**  
   to be completed
* **Harddisk Image on a Read-Only Medium**  
  In the 'volatile' mode, the base file is always opened in read-only mode, so it can safely be stored on a read-only medium (for example on a cdrom). In that case it is recommended to specify the redolog file with the "journal" option.

8.20.8.6. limitations

The "volatile" disk depends on the limitations of base disk image used.

8.20.9. vpc

8.20.9.1. description

The "vpc" disk image mode is mostly a port of Qemu's "vpc" block driver for VirtualPC disk images (written by Alex Beregszaszi and Kevin Wolf).

8.20.9.2. image creation

Create such disk image with Microsoft VirtualPC (tm) or Qemu's disk image utility (qemu-img).

8.20.9.3. path

The "path" option of the ataX-xxx directive in the configuration file must point to the VirtualPC disk image.

8.20.9.4. external tools

Use Microsoft VirtualPC (tm) tools to manipulate these disk images.

8.20.9.5. typical use

Share disk images with VirtualPC.

8.20.9.6. limitations

The based on the Qemu code the disk size is limited to 127 GB.

8.20.10. vvfat

8.20.10.1. description

The "vvfat" disk image mode is mostly a port of Qemu's "virtual VFAT" block driver (written by Johannes E. Schindelin). It supports the read-only part of this implementation plus some additions. The structures of the virtual disk are created from the contents of the specified local directory when Bochs is starting. All writes to this virtual disk go to a volatile redolog and when closing Bochs, the user can decide whether or not to commit the changes. If "No" is seclected, all changes will be lost when Bochs quits and the files of the local directory are not modified. Otherwise the changes of files and directories are committed. WARNING: Don't use important data without backup in the "vvfat" directory when using this "optional commit" feature.

In addition to Qemu's "vvfat" implementation the Bochs one supports configurable disk geometry, FAT32 and reading MBR and/or boot sector from file. When using the autodetection feature, the CHS values 1024/16/63 (Qemu defaults) are used. To use an MBR image file from a real disk it must be named vvfat\_mbr.bin and placed in the specified directory. Bochs uses the geometry and file system type from this file and doesn't show it on the virtual disk. The same applies to a boot sector image file, but it must be named vvfat\_boot.bin. Using both image file is also possible if they match (grabbed from the same disk).

FAT32 is enabled for disk with minimum 2 GB size, or if MBR / boot sector image enable this filesystem. When using "vvfat" for a floppy, FAT12 is used (1.44M media only). In all other cases FAT16 is used. Long filename support is always present.

Special FAT file attributes are stored in a special file named vvfat\_attr.cfg. When using the "optional commit" feature, modified attributes are saved to this file. The "optional commit" also supports setting the file modification date and time.

8.20.10.2. image creation

Not necessary. The directory must exist before running Bochs.

8.20.10.3. path

The "path" option of the ataX-xxx directive in the configuration file must point to an existing local directory. The redolog name can be set with the "journal" option of the same directive. If not set, the redolog name is created by adding the filename "vvfat.dir.redolog" to the path. A random suffix is also appended to the redolog name.

8.20.10.4. external tools

You can use your favourite file manager to copy file to the directory you'd like to use. The directory should not be modified while Bochs is running.

8.20.10.5. typical use

Accessing files of the host hard drive is only possible with vvfat.

8.20.10.6. limitations

There is no runtime write support present yet. All changes are written to the volatile redolog and they can only be committed on Bochs exit.

When using the MBR from image file only the first partition is used and it must be of type FAT16 or FAT32. FAT12 can be used for the floppy only. An extended partition or other file system types are not supported.

The "vvfat" implementation does not support save/restore, since there is no check if the host directory tree and all files are unmodified after saving the Bochs state.

|  |  |  |
| --- | --- | --- |
| [Prev](http://docs.google.com/cirrus-notes.html) | [Home](http://docs.google.com/index.html) | [Next](http://docs.google.com/using-bximage.html) |
| Notes about Cirrus SVGA usage | [Up](http://docs.google.com/howto.html) | Using the bximage tool |