NAME

xorrisofs − Emulation of ISO 9660 program mkisofs by program xorriso

SYNOPSIS

xorrisofs [options] [-o filename] pathspec [pathspecs ...]

DESCRIPTION

xorrisofs produces Rock Ridge enhanced ISO 9660 filesystems and add−on sessions to such filesystems. Optionally it can produce Joliet directory trees too.

xorrisofs understands options of program mkisofs from cdrtools by Joerg Schilling. Its implementation is part of program xorriso which shares no source code with cdrtools.

ISO 9660, Rock Ridge, Joliet, HFS+:

ISO 9660 (aka **ECMA−119**) is a read−only filesystem that is mainly used for optical media CD, DVD, BD, but may also reside on other storage devices like disk files, USB sticks or disk partitions. It is widely readable by many operating systems and by boot facilities of personal computers.
ISO 9660 describes directories and data files by very restricted filenames with no distinction of upper case and lower case. Its metadata do not comply to fundamental POSIX specifications.
Rock Ridge is the name of a set of additional information which enhance an ISO 9660 filesystem so that it can represent a POSIX compliant filesystem with ownership, access permissions, symbolic links, and other attributes. Rock Ridge allows filenames of up to 255 bytes and paths of up to 1024 bytes.
xorrisofs produces Rock Ridge information by default. It is strongly discouraged to disable this feature.
Joliet is the name of an additional directory tree which provides filenames up to 64 characters encoded as UTF−16. A Joliet tree is mainly interesting for reading the ISO image by operating systems of Microsoft Corporation. Production of this directory tree may be enabled by option −J.
ISO 9660:1999 is the name of an additional directory tree which provides longer filenames. It allows single file names to have up to 207 characters. It might be of use with some older computer system boot facilities which read neither Rock Ridge nor Joliet but need longer filenames nevertheless. Production of this directory tree may be enabled by option −iso−level 4.
HFS+ is the name of a filesystem which is normally used for writing and reading on hard disks and similar devices. It is possible to embed a HFS+ partition into the emerging ISO 9660 image and to mark it by Apple Partition Map entries. This interferes with options which copy data into the first 32 KiB of the ISO image, like −G or −isohybrid−mbr. See option −hfsplus.
The main purpose for having an embedded HFS+ partition is booting of certain models of Apple computers.

Inserting files into the ISO image:

xorrisofs deals with two kinds of file addresses:
disk_path is a path to an object in the local filesystem tree.
iso_rr_path is the Rock Ridge address of a file object in the ISO image. If no Rock Ridge information shall be stored in an emerging ISO, then the names will get mapped to ISO 9660 names of limited length and character set.

A program argument is handled as a **pathspec**, if it is not recognized as original mkisofs option or additional **xorrisofs** option. A pathspec depicts an input file object by a disk_path. If option −graft−points is not present, then the behavior depends on the file type of disk_path. Directories get merged with the /−directory of the ISO image. Files of other types get copied into the /−directory.
If −graft−points is present then each pathspec gets split at the first occurrence of the =−character. The part before the = is taken as **target**, i.e. the iso_rr_path for the file object in the ISO image. The part after the first = is taken as **source**, i.e. the disk_path of the input object.
It is possible to make =−characters part of the iso_rr_path by preceding them with a \−character. The same must be done for \−characters which shall be part of the iso_rr_path.

If the source part of the pathspec leads to a directory, then all files underneath this directory get inserted into the image, too. It is possible to exclude particular files from being inserted by help of option −m.
In case that target already exists, the following rules apply: Directories and other files may overwrite existing non−directories. Directories get merged with existing directories. Non−directories may not overwrite existing directories.

Relation to program xorriso:

xorrisofs is actually a command mode of program **xorriso**, which gets entered either by xorriso command "−as mkisofs" or by starting the program by one of the names "xorrisofs", "mkisofs", "genisoimage", or "genisofs".
This command mode can be left by argument "−−" which leads to generic xorriso command mode. See **man xorriso** for its description.

xorriso performs image reading and writing by help of libburn, which is mainly intended for optical drives, but also operates on all POSIX file types except directories.
The program messages call any image file a "drive". File types which are not supported for reading are reported as "blank". The reported free media space may be quite fictional.
Nevertheless **xorrisofs** does not operate directly on optical drives, but rather forces libburn to regard them as general device files. So for writing of sequential optical media (CD, DVD−R, DVD+R, BD−R) one will have to use a burn program. E.g the cdrecord emulation of xorriso. See EXAMPLES.

OPTIONS

Image loading:

The following options control loading of an existing ISO image for the purpose of preparing a suitable add−on session. If they are missing then a new image is composed from scratch.

−M disk_path

Set the path from which to load the existing ISO image directory tree on which to base the upcoming directory tree as add−on session. The path must lead to a random−access readable file object. On GNU/Linux: regular data files or block device files.
A special kind of pseudo disk_path has the form "/dev/fd/"number. It depicts the open file descriptor with the given number, regardless whether the operating system supports this feature by file nodes in /dev/fd or not. E.g. /dev/fd/3 is file descriptor 3 which was opened by the program that later started xorriso.

−prev-session disk_path

Alias of −M.

−dev disk_path

Alias of −M.

−C last_session_start,next_writeable_address

Set the 2 KiB block address last_session_start from where to read the ISO image out of the file given by option −M.
Separated by a comma, set the next_writeable_address to which the add−on session will finally be written. Decisive is actually the block address which the intended readers will have to use as superblock address on the intended medium.
Both values can be inquired from optical media by help of burn programs and cdrecord option −msinfo. xorriso itself can obtain it in its cdrecord emulation.

```
  values=$(xorriso −as cdrecord dev=/dev/... −msinfo)
  echo $values
```

Option −C may be used without option −M to create an ISO image from scratch and prepare it for being finally written to a block address other than 0. Parameter last_session_start must then be set to 0.

−cdrecord-params last_session_start,next_writeable_address

Alias of −C.

Settings for file insertion:

−path-list disk_path

Read pathspecs line−by−line from disk_file and insert the depicted file objects into the ISO image. If disk_path is "−" then read the pathspecs from standard input.

--quoted_path_list disk_path
Like option −path−list but reading quoted words rather than plain lines. Whitespace outside of quotes will be discarded. On the other hand it is possible to represent pathspecs which contain newline characters.
The double quotation mark " and the single quotation mark ’ can be used to enclose whitespace and make it part of pathspecs. Each mark type can enclose the marks of the other type. A trailing backslash \ outside quotations or an open quotation cause the next input line to be appended.

−f
Resolve symbolic links on disk rather than storing them as symbolic links in the ISO image.

−follow-links
Alias of −f.

−graft-points
Enable interpretation of input file pathspecs as combination of iso_rr_path and disk_path, separated by a =−character.

−m disk_pattern
Exclude files from being inserted into the image. Silently ignored are those files of which the disk_path matches the given shell parser pattern. If no /−character is part of the pattern, then it gets matched against the leaf name of the disk file.
It is possible to give more than one −m option.

−exclude
Alias of −m.

−x
Alias of −m.

−old-exclude
Alias of −m.

−exclude-list disk_path
Perform −m using each line out of file disk_path as argument disk_pattern.

−z
Enable recognition and proper processing of zisofs compressed files as produced by program mkzftree. These files will get equipped with the necessary meta data so that a Linux kernel will recognize them and deliver their content in uncompressed form.

−transparent-compression
Alias of −z.

−-zisofs-version-2
Enable the recognition and proper processing of experimental zisofs version 2 compressed files. The Linux kernel (as of 5.9) does not yet know this format and will complain like
  isofs: Unknown ZF compression algorithm: PZ
This complaint can be prevented by option −−zisofs2−susp−z2 .
The files will be shown by unaware kernels as they were submitted to xorriso, i.e. with zisofs2 header, block pointer list, and compressed data.
−−zisofs−version−2 also enables −z.

−-zisofs2-susp-z2
Enable the production of SUSP entries "Z2" instead of "ZF" with zisofs2 compressed files. Unaware Linux kernels silently ignore "Z2" entries.

−-zisofs2-susp-zf
Enable the production of SUSP entries "ZF" instead of "Z2" with zisofs2 compressed files. Unaware Linux kernels complain about zisofs2 "ZF" by "Unknown ZF compression algorithm" and thus leave a mark in the system log.

−root iso_rr_path
Insert all files under the given iso_rr_path. If option −graft−points is given, then iso_rr_path is prepended to each target part of a pathspec.
The default for −root is "/".

−old-root iso_rr_path
Enable incremental insertion of files into the loaded image. The effective target and source addresses of given pathspecs get compared whether the target already exists in the ISO image and is still identical to the source on disk. Metadata in the ISO image will get adjusted, if they differ from those on disk. New files and files with changed content will get newly added. Target files which do not exist in any of the according pathspec sources will get removed from the ISO directory tree.
If the effective setting of −root differs from the iso_rr_path given with −old−root, then the files underneath the −old−root directory get cloned underneath the −root directory. Cloning happens before file comparison.

--old-root-no-ino
Disable recording and use of disk inode numbers. If no disk inode numbers are recorded, then option −old−root will have to read disk file content and compare it with the MD5 checksum that is recorded in the ISO image.
With recorded disk inode numbers and with credible ctime and mtime, it is possible to detect potential changes in the content without actually reading it. A loophole remains if multiple different filesystems may get mounted at the same directory, like it is habit with /mnt. In this case one has to use option −−old−root−devno or disable the inode number shortcut by −−old−root−no−ino.

--old-root-devno
Enable comparison of recorded device numbers together with recorded inode numbers. This works only with good old stable device numbers which get out of fashion, regrettably. If the hard disk has a different device number after each reboot, then this comparison will see all files as changed and thus prevent any incremental size saving.

--old-root-no-md5
Disable recording and use of MD5 checksums for data file content. If neither checksums and nor disk inode numbers are recorded, then option −old−root will have to read ISO image file content when comparing it with disk file content.

Settings for image production:

−o disk_path
Set the output file address for the emerging ISO image. If the address exists as regular file, it will be truncated to length 0 when image production begins. It may not already exist as directory. If it does not exist yet then its parent directory must exist and a regular file will get created.
A special kind of pseudo disk_path has the form "/dev/fd/"number. It depicts the open file descriptor with the given number, regardless whether the operating system supports this feature by file nodes in /dev/fd or not. E.g. /dev/fd/4 is file descriptor 4 which was opened by the program that later started xorriso.
Default is standard output (/dev/fd/1) which may also be set by disk_path "−".

−output disk_path
Alias of −o.

--stdio_sync "on"|"off"|"end"|number
Set the number of bytes after which to force output to disk in order to keep the memory from being clogged with lots of pending data for slow devices. "on" is the same as "16m". Forced output can be disabled by "off", or be delayed by "end" until all data are produced. If a number is chosen, then it must be at least 64k.
The default with xorriso mkisofs emulation is −−stdio_sync "off".
xorriso uses an inner fifo buffer with default size 4 MiB. So forcing the operating system i/o cache

to disk does not necessarily block the simultaneous production of more image content.

--emul-toc
Write a second superblock with the first session into random−access files. If further sessions get appended and the first superblock gets updated, then the second superblock will not be overwritten. So it is still possible to mount the first session and to find the start blocks of the further sessions.
The price is 64 KiB extra space consumption. If −partition_offset is non−zero, then it is 128 KiB plus twice the partition setup.

--no-emul-toc
Do not write a second superblock with the first session into random−access files.
This is the default.

--sort-weight weight_number iso_rr_path
Attribute a LBA weight number to regular files. If iso_rr_path leads to a directory then all regular files underneath will get the weight_number.
The weight_number may range from −2147483648 to 2147483647. The higher it is, the lower will be the block address of the file data in the emerging ISO image. Currently the El Torito boot catalog has a hardcoded weight of 1 billion. Normally it should occupy the block with the lowest possible address. Data files get added or loaded with initial weight 0. Boot image files have a default weight of 2.

--sort-weight-list disk_path
Read pairs of weight number and iso_rr_path from a file of the local filesystem. Apply each pair like with −−sort−weight.
Only the last −−sort−weight−list or −−sort−weight−patterns of a xorrisofs run gets into effect.
The weight number is read from the start of the line. The iso_rr_path part of an input line begins immediately after the first blank or tab character of the line.
Notes for the case that this feature is used within a sequence of generic xorriso commands (not an issue with a pure mkisofs emulation run):
The addressed files must already be in the ISO image model when you execute
```
  −as mkisofs −−sort−weight−list disk_path −−
```
Several such commands may be used to apply more than one weight file.
Data files which are loaded by −indev or −dev get a weight between 1 and 2 exp 28 = 268,435,456, depending on their block address. This shall keep them roughly in the same order if the write method of modifying is applied.

--sort-weight-patterns disk_path
Like −−sort−weight−list , but expanding the iso_rr_paths as shell parser patterns and applying −−sort−weight to each matching file.

−uid number|name
Use the given number or locally existing user name as owner id of all files and directories in the emerging filesystem. Empty name or name "−" revoke this feature.

−gid number|name
Use the given number or locally existing group name as group id of all files and directories in the emerging filesystem. Empty name or name "−" revoke this feature.

−dir-mode mode
Set the access permissions for all directories in the image to the given mode which is either an octal number beginning with "0" or a comma separated list of statements of the form [ugoa]*[+−=][rwxst]* . E.g. ug=rx,a−rwx

−file-mode mode
Like −dir−mode but for all regular data files in the image.

−pad
Add 300 KiB to the end of the produced ISO image. This circumvents possible read errors from

ISO images which have been written to CD media in TAO mode. The additional bytes are claimed as part of the ISO image if not −−emul−toc is given.
Option −pad is the default.

−no-pad
Disable padding of 300 KiB to the end of the produced ISO image. This is safe if the image is not meant to be written on CD or if it gets written to CD as only track in write mode SAO.

--old-empty
Use the old way of of giving block addresses in the range of [0,31] to files with no own data content. The new way is to have a dedicated block to which all such files will point.

Settings for standards compliance:

−iso-level number
Specify the ISO 9660 version which defines the limitations of file naming and data file size. The naming restrictions do not apply to the Rock Ridge names but only to the low−level ISO 9660 names. There are three conformance levels:
Level 1 allows ISO names of the form 8.3 and file size up to 4 GiB − 1.
Level 2 allows ISO names with up to 32 characters and file size up to 4 GiB − 1.
Level 3 allows ISO names with up to 32 characters and file size of up to 400 GiB − 200 KiB. (This size limitation is set by the xorriso implementation and not by ISO 9660 which would allow nearly 8 TiB.)
Pseudo−level 4 enables production of an additional ISO 9660:1999 directory tree.

−disallow_dir_id_ext
Do not follow a bad habit of mkisofs which allows dots in the ISO names of directories. On the other hand, some bootable GNU/Linux images depend on this bad habit.

−U
This option allows ISO file names without dot and up to 37 characters, ISO file paths longer than 255 characters, and all ASCII characters in file names. Further it omits the semicolon and the version numbers at the end of ISO names.
This all violates ISO 9660 specs.

−untranslated-filenames
Alias of −U.

−untranslated_name_len number
Allow ISO file names up to the given number of characters without any character conversion. The maximum number is 96. If a file name has more characters, then image production will fail deliberately.
This violates ISO 9660 specs.

−allow-lowercase
Allow lowercase character in ISO file names.
This violates ISO 9660 specs.

−relaxed-filenames
Allow nearly all 7−bit characters in ISO file names. Not allowed are 0x0 and ’/’. If not option −allow−lowercase is given, then lowercase letters get converted to uppercase.
This violates ISO 9660 specs.

−d
Do not add trailing dot to ISO file names without dot.
This violates ISO 9660 specs.

−omit-period
Alias of −d.

−l
Allow up to 31 characters in ISO file names.

−full-iso9660-filenames
Alias of −l.

−max-iso9660-filenames
Allow up to 37 characters in ISO file names.
This violates ISO 9660 specs.

−N
Omit the semicolon and the version numbers at the end of ISO names.
This violates ISO 9660 specs.

−omit-version-number
Alias of −N.

Settings for standards extensions:

−R
With mkisofs this option enables Rock Ridge extensions. **xorrisofs** produces them by default. It is strongly discouraged to disable them by option −−norock.

−rock
Alias of −R.

−r
Enable Rock Ridge and set user and group id of all files in the ISO image to 0. Grant r−permissions to all. Deny all w−permissions. If any x−permission is set, grant x−permission to all. Remove s−bit and t−bit.
These attribute changes stay delayed until mkisofs emulation ends. Within the same −as mkisofs emulation command they can be revoked by a subsequent option −−norock. For compatibility reasons, option −R does not revoke the changes ordered by −r.

−rational-rock
Alias of −r.

--norock
This option disables the production of Rock Ridge extensions for the ISO 9660 file objects. The multi−session capabilities of **xorrisofs** depend much on the naming fidelity of Rock Ridge. So it is strongly discouraged to disable it by this option, except for the special use case to revoke the effect of −r by:
−−norock −R

--set_all_file_dates timestring
Set mtime, atime, and ctime of all files and directories to the given time.
Valid timestring formats are: 'Nov 8 14:51:13 CET 2007', 110814512007.13, 2007110814511300.
See also −−modification−date= and man xorriso, Examples of input timestrings.
If the timestring is "set_to_mtime", then the atime and ctime of each file and directory get set to the value found in their mtime.
These actions stay delayed until actual ISO production begins. Up to then they can be revoked by −−set_all_file_dates with empty timestring or timestring "default".
The timestamps of the El Torito boot catalog file get refreshed when the ISO is produced. They can be influenced by −−modification−date=.

−file_name_limit number
Set the maximum permissible length for file names in the range of 64 to 255. Path components which are longer than the given number will get truncated and have their last 33 bytes overwritten by a colon ':' and the hex representation of the MD5 of the first 4095 bytes of the whole oversized name. Potential incomplete UTF−8 characters will get their leading bytes replaced by '_'.
Linux kernels up to at least 4.1 misrepresent names of length 254 and 255. If you expect such names in or under disk_paths and plan to mount the ISO by such Linux kernels, consider to set −file_name_limit 253.

−D The standard ECMA−119 demands that no path in the image shall have more than 8 name components or 255 characters. Therefore it would be necessary to move deeper directory trees to a higher directory. Rock Ridge offers an opportunity to let these relocated directories appear at their original deep position, but this feature might not be implemented properly by operating systems which mount the image.
Option −D disables this deep directory relocation, and thus violates ISO 9660 specs.
xorrisofs has −D set by default. If given explicitly then it overrides the options −rr_reloc_dir and −hide−rr−moved.

−disable-deep-relocation
Alias of −D.

−rr_reloc_dir name
Enable the relocation of deep directories and thus avoid ECMA−119 file paths of more than 8 name components or 255 characters. Directories which lead to such file paths will get moved to a directory in the root directory of the image. Its name gets set by this option. It is permissible to use the root directory itself.
The overall directory tree will appear originally deep when interpreted as Rock Ridge tree. It will appear as re−arranged if only ECMA−119 information is considered.
If the given relocation target directory does not already exist when image production begins, then it will get created and marked for Rock Ridge as relocation artefact. At least on GNU/Linux it will not be displayed in mounted Rock Ridge images.
The name must not contain a ’/’ character after its first character and it must not be longer than 255 bytes.
This option has no effect if option −D is present.

−hide-rr-moved
Alias of −rr_reloc_dir "/.rr_moved"

--for_backup
Enable all options which improve backup fidelity:
−−acl, −−xattr−any, −−md5, −−hardlinks.
If you later restore a backup with xattr from non−user namespaces, then make sure that the target operating system and filesystem know what these attributes mean. Possibly you will need administrator privileges to record or restore such attributes. At recording time, xorriso will try to tolerate missing privileges and just record what is readable.
Option −xattr after option −for_backup excludes non−user attributes from being recorded.

--acl
Enable recording and loading of ACLs from GNU/Linux or FreeBSD (see man getfacl, man acl). They will not be in effect with mounted ISO images. But xorriso can restore them on the same systems when extracting files from the ISO image.

--xattr
Enable recording and loading of GNU/Linux or FreeBSD extended attributes in user namespace (see man getfattr and man attr, man getextattr and man 9 extattr, respectively). They will not be in effect with mounted ISO images. But xorriso can restore them on the same systems when extracting files from the ISO image.

--xattr-any
Enable recording and loading of GNU/Linux or FreeBSD extended attributes in all namespaces. This might need administrator privileges, even if the owner of the disk file tries to read the attributes.

--md5
Enable recording of MD5 checksums for the overall ISO image and for each single data file in the image. xorriso can check the content of an ISO image with these sums and raise alert on mismatch. See man xorriso, options −check_media, check_md5_r. xorriso can print recorded MD5 checksums. E.g. by:

−find / −exec get_md5

--hardlinks

Enable loading and recording of hardlink relations. Search for families of iso_rr files which stem from the same disk file, have identical content filtering and have identical properties. The members of each family get the same inode number in the ISO image.
Whether these numbers are respected at mount time depends on the operating system. xorriso can create hardlink families when extracting files from the ISO image.

--scdbackup_tag disk_path record_name

Append a scdbackup checksum record to the image. This works only if the parameter next_writeable_address of option −C is 0 and −−md5 is enabled. If disk_path is not an empty string, then append a scdbackup checksum record to the end of this file. record_name is a word that gets part of tag and record.
Program scdbackup_verify will recognize and verify tag and file record.
An empty record_name disables this feature.

−J

Enable the production of an additional Joliet directory tree along with the ISO 9660 Rock Ridge tree.

−joliet Alias of −J.

−joliet-long

Allow 103 characters in Joliet file names rather than 64 as is prescribed by the specification. Allow Joliet paths longer than the prescribed limit of 240 characters.
Oversized names get truncated. Without this option, oversized paths get excluded from the Joliet tree.

−joliet-utf16

Encode Joliet file names in UTF−16BE rather than UCS−2. The difference is with characters which are not present in UCS−2 and get encoded in UTF−16 by 2 words of 16 bit each. Both words then stem from a reserved subset of UCS−2.

−hfsplus

Enable the production of an additional HFS+ filesystem inside the ISO 9660 image and mark it by Apple Partition Map (APM) entries in the System Area, the first 32 KiB of the image.
This may collide with options like −G or −isohybrid−mbr which submit user data for inclusion in the same address range. The first 8 bytes of the System Area get overwritten by { 0x45, 0x52, 0x08 0x00, 0xeb, 0x02, 0xff, 0xff } which can be executed as x86 machine code without negative effects. So if an MBR gets combined with this feature, then its first 8 bytes should contain no essential commands.
The next blocks of 2 KiB in the System Area will be occupied by APM entries. The first one covers the part of the ISO image before the HFS+ filesystem metadata. The second one marks the range from HFS+ metadata to the end of file content data. If more ISO image data follow, then a third partition entry gets produced. Other features of xorriso might cause the need for more APM entries.
Be aware that HFS+ is case−insensitive although it can record file names with upper−case and lower−case letters. Therefore, file names from the iso_rr name tree may collide in the HFS+ name tree. In this case they get changed by adding underscore characters and counting numbers. In case of very long names, it might be necessary to map them to "MANGLED_...".
WARNING:
The HFS+ implementation in libisofs has a limit of 125,829,120 bytes for the size of the overall directory tree. This suffices for about 300,000 files of normal name length. If the limit gets exceeded, a FAILURE event will be issued and the ISO production will not happen.

−hfsplus-serial-no

Set a string of 16 digits "0" to "9" and letters "a" to "f", which will be used as unique serial number of an emerging HFS+ filesystem.

−hfsplus-block-size number

Set the allocation block size to be used when producing HFS+ filesystems. Permissible are 512, 2048, or 0. The latter lets the program decide.

−apm-block-size number

Set the block size to be used when describing partitions by an Apple Partition Map. Permissible are 512, 2048, or 0. The latter lets the program decide.
Note that size 512 is not compatible with production of GPT, and that size 2048 will not be mountable −t hfsplus at least by older Linux kernels.

−hfsplus-file-creator-type creator type iso_rr_path

Set the HFS+ creator and type attributes of a file in the emerging image. These are two codes of 4 characters each.

−hfs-bless-by blessing iso_rr_path

Issue a HFS+ blessing. They are roles which can be attributed to up to four directories and a data file:
"ppc_bootdir", "intel_bootfile", "show_folder", "os9_folder", "osx_folder".
They may be abbreviated as "p", "i", "s", "9", and "x".
Each such role can be attributed to at most one file object. "intel_bootfile" is the one that would apply to a data file. All others apply to directories. No file object can bear more than one blessing.

−hfs-bless disk_path

Issue HFS+ blessing "ppc_bootdir" to the directory which stems from the directory disk_path in the local filesystem tree.
This works only if there is at least one data file underneath the directory. disk_path can become ambiguous if files from different local filesystem sub−trees are put into the same sub−tree of the ISO image. Consider to use −hfs−bless−by "p" for unambiguous addressing via iso_rr_path.

Settings for file hiding:

−hide disk_path_pattern

Make files invisible in the directory tree of ISO 9660 and Rock Ridge, if their disk_path matches the given shell parser pattern. The data content of such hidden files will be included in the resulting image, even if they do not show up in any directory. But you will need own means to find nameless data in the image.
This command does not apply to the boot catalog.

−hide-list disk_path

Perform −hide using each line out of file disk_path as argument disk_path_pattern.

−hide-joliet disk_path_pattern

Like option −hide but making files invisible in the directory tree of Joliet, if their disk_path matches the given shell parser pattern.

−hide-joliet-list disk_path

Perform −hide−joliet using each line out of file disk_path as argument disk_path_pattern.

−hide-hfsplus disk_path_pattern

Like option −hide but making files invisible in the directory tree of HFS+, if their disk_path matches the given shell parser pattern.

−hide-hfsplus-list disk_path

Perform −hide−hfsplus using each line out of file disk_path as argument disk_path_pattern.

ISO image ID strings:

The following strings and file addresses get stored in the Primary Volume Descriptor of the ISO9660 image. The file addresses are ISO 9660 paths. These files should have iso_rr_paths which consist only of the characters [A−Z0−9_] and exactly one dot which separates at most 8 characters from at most 3 characters.

−V text
Set the Volume Id of the ISO image. xorriso accepts any text up to 32 characters, but according to rarely obeyed specs stricter rules apply:
Conformant are ASCII characters out of [A−Z0−9_]. Like: "IMAGE_23"
Joliet allows 16 UCS−2 characters. Like: "Windows name"
Be aware that the volume id might get used automatically as name of the mount point when the medium is inserted into a playful computer system.

−volid text
Alias of −V.

−volset text
Set the Volume Set Id of the ISO image. Permissible are up to 128 characters.

−P text
Set the Publisher Id of the ISO image. This may identify the person or organisation who specified what shall be recorded. Permissible are up to 128 characters.

−publisher text
Alias of −P.

−A text
Set the Application Id of the ISO image. This may identify the specification of how the data are recorded. Permissible are up to 128 characters.
The special text "@xorriso@" gets converted to the id string of xorriso which is normally written as Preparer Id. It is a wrong tradition to write the program id as Application Id.

−appid text
Alias of −A.

−sysid text
Set the System Id of the ISO image. This may identify the system which can recognize and act upon the content of the System Area in image blocks 0 to 15. Permissible are up to 32 characters.

−p text
Set the Preparer Id of the ISO image. This may identify the person or other entity which controls the preparation of the data which shall be recorded. Normally this should be the id of xorriso and not of the person or program which operates xorriso. Please avoid to change it. Permissible are up to 128 characters.
The special text "@xorriso@" gets converted to the id string of xorriso which is default at program startup.

−preparer text
Alias of −p.

−abstract iso_path
Set the address of the Abstract File of the ISO image. This should be the ISO 9660 path of a file in the image which contains an abstract statement about the image content. Permissible are up to 37 characters.

−biblio iso_path
Set the address of the Biblio File of the ISO image. This should be the ISO 9660 path of a file in the image which contains bibliographic records. Permissible are up to 37 characters.

−copyright iso_path
Set the address of the Copyright File of the ISO image. This should be the ISO 9660 path of a file in the image which contains a copyright statement. Permissible are up to 37 characters.

--modification-date=YYYYMMDDhhmmsscc
Set a timestring that overrides ISO image creation and modification timestamps literally. It must consist of 16 decimal digits which form YYYYMMDDhhmmsscc, with YYYY between 1970 and 2999. Time zone is GMT. It is supposed to match this GRUB line:
```
 search −−fs−uuid −−set YYYY−MM−DD−hh−mm−ss−cc
```
E.g. 2010040711405800 is 7 Apr 2010 11:40:58 (+0 centiseconds).
Among the influenced timestamps are: isohybrid MBR id, El Torito boot catalog file, HFS+ superblock.

--application_use character|0xXY|disk_path

Specify the content of the Application Use field which can take at most 512 bytes.
If the parameter of this command is empty, then the field is filled with 512 0−bytes. If it is a single character, then it gets repeated 512 times. If it begins by "0x" followed by two hex digits [0−9a−fA−F], then the digits are read as byte value which gets repeated 512 times.
Any other parameter text is used as disk_path to open a data file and to read up to 512 bytes from it. If the file is smaller than 512 bytes, then the remaining bytes in the field get set to binary 0.

El Torito Bootable ISO images:

The precondition for a bootable ISO image is to have in the ISO image the files of a boot loader. The boot facilities of computers get directed to such files, which usually execute further program files from the ISO image. **xorrisofs** can produce several kinds of boot block or boot record, which become part of the ISO image, and get interpreted by the according boot facility.

An **El Torito** boot record points the bootstrapping facility to a boot catalog with one or more boot images, which are binary program files stored in the ISO image. The content of the boot image files is not in the scope of El Torito.
xorriso composes the boot catalog according to the boot image files given and structured by options −b, −e, −eltorito−alt−boot, and −−efi−boot. Often it contains only one entry.
Normally the boot images are data files inside the ISO filesystem. By special path "−−interval:appended_partition_NNN:all::" it is possible to refer to an appended partition. The number NNN gives the partition number as used with the corresponding option −append_partition. E.g.:
```
 −append_partition 2 0xef /tmp/efi.img
 −e −−interval:appended_partition_2:all::
```
El Torito gets interpreted by boot facilities PC−BIOS and EFI. Most bootable GNU/Linux CDs are equipped with ISOLINUX or GRUB boot images for PC−BIOS.
xorrisofs supports the example options out of the ISOLINUX wiki, the options used in GRUB script grub−mkrescue, and the example in the FreeBSD AvgLiveCD wiki.

For CD booting via boot facilities other than PC−BIOS and EFI, and for booting from USB sticks or hard disks, see the next section about the System Area.

−b iso_rr_path

Specify the boot image file which shall be mentioned in the current entry of the El Torito boot catalog. It will be marked as suitable for PC−BIOS.
With boot images from ISOLINUX and GRUB this option should be accompanied by options −c , −no−emul−boot , −boot−load−size 4 , −boot−info−table.

−eltorito-boot iso_rr_path

Alias of −b.

−eltorito-alt-boot

Finalize the current El Torito boot catalog entry and begin a new one. A boot image file and all its necessary options shall be specified before option −eltorito−alt−boot. All further El Torito boot options apply to the new catalog entry. Up to 32 catalog entries are possible.

−e iso_rr_path

Specify the boot image file which shall be mentioned in the current entry of the El Torito boot catalog. It will be marked as suitable for EFI.
Option −e should be followed by option −no−emul−boot and no other El Torito options before an eventual −eltorito−alt−boot.

--efi-boot iso_rr_path

Perform −eltorito−alt−boot, option −e with the given iso_rr_path, −no−emul−boot, and again −eltorito−alt−boot. This gesture is used for achieving EFI−bootability of the GRUB2 rescue CD.

−eltorito-platform "x86"|"PPC"|"Mac"|"efi"|0xnn|nnn

Set the Platform Id number for the next option −b or −eltorito−boot. The number may be chosen by a platform name or by a number between 0 and 255 (0x00 and 0xFF). "x86" = 0 is for

PC−BIOS, "PPC" = 1 for some PowerPC systems, "Mac" = 2 for some MacIntosh systems, "efi" = 0xEF for EFI on modern PCs with x86 compatible CPUs or others.
If the new platform id differs from the previous one, −eltorito−alt−boot gets performed.

−boot-load-size number|"full"

Set the number of 512−byte blocks to be loaded at boot time from the boot image in the current catalog entry.
Non−emulating BIOS bootimages usually need a load size of 4. Nevertheless the default setting of mkisofs is to use the full size of the boot image rounded up to a multiple of 4 512−byte blocks. This default may be explicitly enforced by the word "full" instead of a number.
EFI boot images usually get set the number of blocks occupied by the boot image file.
El Torito cannot represent load sizes higher than 65535.

−hard-disk-boot

Mark the boot image in the current catalog entry as emulated hard disk. (Not suitable for any known boot loader.)

−no-emul-boot

Mark the boot image in the current catalog entry as not emulating floppy or hard disk. (This is to be used with all known boot loaders.)
If neither −hard−disk−boot nor −no−emul−boot is given, then the boot image will be marked as emulating a floppy. (Not suitable for any known boot loader.)

−eltorito-id text|56_hexdigits

Define the ID string of the boot catalog section where the boot image will be listed. If the value consists of 56 characters [0−9A−Fa−f] then it is converted into 28 bytes, else the first 28 characters become the ID string. The ID string of the first boot image becomes the overall catalog ID. It is limited to 24 characters. Other id_strings become section IDs.

−eltorito-selcrit hexdigits

Define the Selection Criteria of the boot image. Up to 20 bytes get read from the given characters [0−9A−Fa−f]. They get attributed to the boot image entry in the catalog.

−boot-info-table

Overwrite bytes 8 to 63 in the current boot image. The information will be supplied by xorriso in the course of image production: Block address of the Primary Volume Descriptor, block address of the boot image file, size of the boot image file.

--grub2-boot-info

Overwrite bytes 2548 to 2555 in the current boot image by the address of that boot image. The address is written as 64 bit little−endian number. It is the 2KB block address of the boot image content, multiplied by 4, and then incremented by 5.

−c iso_rr_path

Set the address of the El Torito boot catalog file within the image. This file address is not significant for the booting PC−BIOS or EFI, but it may later be read by other programs in order to learn about the available boot images.

−eltorito-catalog iso_rr_path

Alias of −c.

--boot-catalog-hide

Prevent the El Torito boot catalog from appearing as file in the directory trees of the image.

System Area, MBR, GPT, APM, other boot blocks:

The first 16 blocks of an ISO image are the System Area. It is reserved for system dependent boot software. This may be the boot facilities and partition tables of various hardware architectures.
A **MBR** (Master Boot Record) contains boot code and a partition table. It is read by PC−BIOS when booting from USB stick or hard disk, and by PowerPC CHRP or PReP when booting. An MBR partition with type 0xee indicates the presence of GPT.
A **GPT** (GUID Partition Table) marks partitions in a more modern way. It is read by EFI when booting

from USB stick or hard disk, and may be used for finding and mounting a HFS+ partition inside the ISO image.
An **APM** (Apple Partition Map) marks the HFS+ partition. It is read by Macs for booting and for mounting.
MBR, GPT and APM are combinable. APM occupies the first 8 bytes of MBR boot code. All three do not hamper El Torito booting from CDROM.
xorrisofs supports further boot facilities: MIPS Big Endian (SGI), MIPS Little Endian (DEC), SUN SPARC, HP−PA, DEC Alpha. Those are mutually not combinable and also not combinable with MBR, GPT, or APM.

Several of the following options expect disk paths as input but also accept description strings for the libisofs interval reader, which is able to cut out data from disk files or −indev and to zeroize parts of the content: −G, −generic−boot, −−embedded−boot, −−grub2−mbr, −isohybrid−mbr, −efi−boot−part, −prep−boot−part, −B, −sparc−boot, −append_partition.
The description string consists of the following components, separated by colon ':'
  "−−interval:"Flags":"Interval":"Zeroizers":"Source
The component "−−interval" states that this is not a plain disk path but rather a interval reader description string.
The component Flags modifies the further interpretation:
"local_fs" demands to read from a file depicted by the path in Source.
"imported_iso" demands to read from the −indev. This works only if −outdev is not the same as −indev. The Source component is ignored.
"appended_partition_NNN" with a decimal number NNN works only for options which announce El Torito boot image paths: −b, −e, −−efi−boot. The number gives the partition number as used with the corresponding option −append_partition.
The component Interval consists of two byte address numbers separated by a "−" character. E.g. "0−429" means to read bytes 0 to 429.
The component Zeroizers consists of zero or more comma separated strings. They define which part of the read data to zeroize. Byte number 0 means the byte read from the Interval start address. Each string may be one of:
"zero_mbrpt" demands to zeroize the MBR partition table if bytes 510 and 511 bear the MBR signature 0x55 0xaa.
"zero_gpt" demands to check for a GPT header in bytes 512 to 1023, to zeroize it and its partition table blocks.
"zero_apm" demands to check for an APM block 0 and to zeroize its partition table blocks.
Start_byte"−"End_byte demands to zeroize the read−in bytes beginning with number Start_byte and ending after End_byte.
The component Source is the file path with flag "local_fs", and ignored with flag "imported_iso".
Byte numbers may be scaled by a suffix out of {k,m,g,t,s,d} meaning multiplication by {1024, 1024k, 1024m, 1024g, 2048, 512}. A scaled value end number depicts the last byte of the scaled range.
E.g. "0d−0d" is "0−511".
Examples:
  "local_fs:0−32767:zero_mbrpt,zero_gpt,440−443:/tmp/template.iso"
  "imported_iso:45056d−47103d::"

−G disk_path
: Copy at most 32768 bytes from the given disk file to the very start of the ISO image.
Other than a El Torito boot image, the file disk_path needs not to be added to the ISO image. It will not show up as file in the directory trees.
In multi−session situations, the special disk_path "." prevents reading of a disk file but nevertheless causes the adjustments in the existing MBR, which were ordered by other options.

−generic-boot disk_path
: Alias of −G.

--embedded-boot disk_path
Alias of −G.

--grub2-mbr disk_path
Install disk_path in the System Area and treat it as modern GRUB2 MBR. The content start address of the first boot image is converted to a count of 512 byte blocks, and an offset of 4 is added. The result is written as 64 bit little−endian number to byte address 0x1b0.

−isohybrid-mbr disk_path
Install disk_path as ISOLINUX isohybrid MBR which makes the boot image given by option −b bootable from USB sticks and hard disks via PC−BIOS. This preparation is normally done by ISOLINUX program isohybrid on the already produced ISO image.
The disk path should lead to one of the Syslinux files isohdp[fp]x*.bin . The MBR gets patched according to isohybrid needs. The first partition describes the range of the ISO image. Its start is at block 0 by default, but may be set to 64 disk blocks by option −partition_offset 16.
For the meaning of special disk_path "." see option −G.

−isohybrid-gpt-basdat
Mark the current El Torito boot image (see options −b and −e) in an actually invalid GPT as partition of type Basic Data. This works only with −isohybrid−mbr and has the same impact on the system area as −efi−boot−part. It cannot be combined with −efi−boot−part or −hfsplus.
The first three boot images which are marked by GPT will also show up as partition entries in MBR. The MBR partition of type 0xEF is what actually is used by EFI firmware for booting from USB stick. The MBR partition for PC−BIOS gets type 0x00 rather than 0x17 in this case. Often the further MBR entries are the ones which actually get used by EFI.

−isohybrid-gpt-hfsplus
Mark the current El Torito boot image (see options −b and −e) in GPT as partition of type HFS+. Impact and restrictions are like with −isohybrid−gpt−basdat.

−isohybrid-apm-hfsplus
Mark the current El Torito boot image (see options −b and −e) in Apple Partition Map as partition of type HFS+. This works only with −isohybrid−mbr and has a similar impact on the system area as −hfsplus. It cannot be combined with −efi−boot−part or −hfsplus.
The ISOLINUX isohybrid MBR file must begin by a known pattern of 32 bytes of x86 machine code which essentially does nothing. It will get overwritten by 32 bytes of APM header mock−up.

−part_like_isohybrid
Control whether −isohybrid−gpt−basdat, −isohybrid−gpt−hfsplus, and −isohybrid−apm−hfsplus apply even if not −isohybrid−mbr is present. No MBR partition of type 0xee emerges, even if GPT gets produced. Gaps between GPT and APM partitions will not be filled by more partitions. Appended partitions get mentioned in APM if other APM partitions emerge.

−iso_mbr_part_type "default"|number|type_guid
Set the partition type of the MBR or GPT partition which represents the ISO or at least protects it.
Number may be 0x00 to 0xff. The text "default" re−enables the default types of the various occasions to create an ISO MBR partition. This is without effect if no such partition emerges by other settings or if the partition type is prescribed mandatorily like 0xee for GPT protective MBR or 0x96 for CHRP.
If instead a type_guid is given by a 32−digit hex string like a2a0d0ebe5b9334487c068b6b72699c7 or by a structured text like EBD0A0A2−B9E5−4433−87C0−68B6B72699C7, then it will be used as partition type if the ISO filesystem appears as partition in GPT. In MBR, C12A7328−F81F−11D2−BA4B−00A0C93EC93B will be mapped to 0xef. Any other GUID will be mapped to 0x83.

--protective-msdos-label
Patch the System Area by a simple PC−DOS partition table where partition 1 claims the range of the ISO image but leaves the first block unclaimed. This is mutally exclusive to option −isohybrid−mbr.

--mbr-force-bootable

Enforce an MBR partition with "bootable/active" flag if options like −−protective−msdos−label or −−grub2−mbr are given. These options normally cause the flag to be set if there is an MBR partition of type other than 0xee or 0xef. If no such partition exists, then no bootflag is set, unless −−mbr−force−bootable forces creation of a dummy partition of type 0x00 which covers only the first block of the ISO image.

If no bootable MBR is indicated by other options and a partition gets created by −append_partition, then −−mbr−force−bootable causes a bootflag like it would do with e.g. −−protective−msdos−label.

−partition_offset 2kb_block_adr

Cause a partition table with a single partition that begins at the given block address. This is counted in 2048 byte blocks, not in 512 byte blocks. If the block address is non−zero then it must be at least 16. Values larger than 16 are hardly of use. A non−zero partition offset causes two superblocks to be generated and two sets of directory trees. The image is then mountable from its absolute start as well as from the partition start.

The offset value of an ISO image gets preserved when a new session is added to a loaded image. So the value defined here is only in effect if a new ISO image gets written.

−partition_hd_cyl number

Set the number of heads per cylinder for the MBR partition table. 0 chooses a default value. Maximum is 255.

−partition_sec_hd number

Set the number of sectors per head for the MBR partition table. 0 chooses a default value. Maximum is 63.

The product partition_sec_hd * partition_hd_cyl * 512 is the cylinder size. It should be divisible by 2048 in order to make exact alignment possible. With appended partitions and −appended_part_as_gpt there is no limit for the number of cylinders. Else there may be at most 1024 of them. If the cylinder size is too small to stay below the limit, then appropriate values of partition_hd_cyl are chosen with partition_sec_hd 32 or 63. If the image is larger than 8,422,686,720 bytes, then the cylinder size constraints cannot be fulfilled for MBR. They seem not overly important anyway. Flat block addresses in partition tables are good for 1 TiB.

−partition_cyl_align mode

Control image size alignment to an integer number of cylinders. It is prescribed by isohybrid specs and it seems to please program fdisk. Cylinder size must be divisible by 2048. Images larger than 8,323,596,288 bytes cannot be aligned in MBR partition table.

Mode "auto" is default. Alignment by padding happens only if option −isohybrid−mbr is given.

Mode "on" causes alignment by padding with option −−protective−msdos−label too. Mode "all" is like "on" but also pads up partitions from −append_partition to an aligned size.

Mode "off" disables alignment unconditionally.

−append_partition partition_number type_code disk_path

Cause a prepared filesystem image to be appended to the ISO image and to be described by a partition table entry in a boot block at the start of the emerging ISO image. The partition entry will bear the size of the submitted file rounded up to the next multiple of 2048 bytes or to the next multiple of the cylinder size.

Beware of subsequent multi−session runs. The appended partition will get overwritten.

partition_number may be 1 to 4. Number 1 will put the whole ISO image into the unclaimed space before partition 1. So together with most xorriso MBR or GPT features, number 2 would be the most natural choice.

The type_code may be "FAT12", "FAT16", "Linux", or a hexadecimal number between 0x00 and 0xff. Not all those numbers will yield usable results. For a list of codes search the Internet for "Partition Types" or run fdisk command "L". If the partition appears in GPT then type_code 0xef is mapped to the EFI System Partition Type GUID. All others get mapped to Basic Data Type GUID.

type_code may also be a type GUID as plain hex string like a2a0d0ebe5b9334487c068b6b72699c7 or as structured text like EBD0A0A2−B9E5−4433−87C0−68B6B72699C7. It will be used if the partition is mentioned in GPT. In MBR, C12A7328−F81F−11D2−BA4B−00A0C93EC93B will be mapped to 0xef. Any other GUID will be mapped to 0x83. In APM, 48465300−0000−11AA−AA11−00306543ECAC will be mapped to partition type "Apple_HFS", any other to "Data".
If some other command causes the production of GPT, then the appended partitions will be mentioned there too, even if not −appended_part_as_gpt is given.

−appended_part_as_gpt
Marks partitions from −append_partition in GPT rather than in MBR. In this case the MBR shows a single partition of type 0xee which covers the whole output data.
By default, appended partitions get marked in GPT only if GPT is produced because of other options.

−appended_part_as_apm
Marks partitions from −append_partition in Apple Partition Map, too.
By default, appended partitions get marked in APM only if APM is produced because of other options and −part_like_isohybrid is enabled.

−efi-boot-part disk_path
Copy a file from disk into the emerging ISO image and mark it by a GPT entry as EFI System Partition. EFI boot firmware is supposed to use a FAT filesystem image in such a partition for booting from USB stick or hard disk.
Instead of a disk_path, the word −−efi−boot−image may be given. It exposes in GPT the content of the first El Torito EFI boot image as EFI system partition. EFI boot images are introduced by options −e or −−efi−boot. The affected EFI boot image cannot show up in HFS+ because it is stored outside the HFS+ partition.

--gpt_disk_guid value
Control whether an emerging GPT shall get a randomly generated disk GUID or whether the GUID is supplied by the user. Value "random" is default. Value "modification−date" produces a low quality GUID from the value set by option −−modification−date=.
A string of 32 hex digits, or a RFC 4122 compliant GUID string may be used to set the disk GUID directly. UEFI prescribes the first three components of a RFC 4122 GUID string to be byte−swapped in the binary representation:
E.g. −−gpt_disk_guid 2303cd2a−73c7−424a−a298−25632da7f446 equals −−gpt_disk_guid 2acd0323c7734a42a29825632da7f446
The partition GUIDs get generated by minimally varying the disk GUID.

−chrp-boot-part
Mark the block range of the whole emerging ISO image as MBR partition of type 0x96. This is not compatible with any other feature that produces MBR partition entries. It makes GPT unrecognizable.
CHRP is often used in conjunction with HFS. It is not yet tested whether HFS+ filesystems produced with option −hfsplus would boot on any CHRP capable machine which does not boot pure ISO 9660 as well.

−chrp-boot
Alias of −chrp−boot−part.

−prep-boot-part disk_path
Copy a file from disk into the emerging ISO image and mark it by a MBR partition entry of type 0x41. PReP boot firmware is supposed to read the content of the partition as single ELF executable file. This option is compatible with other MBR partitions and with GPT.

−mips-boot iso_rr_path
Declare a data file in the image to be a MIPS Big Endian boot file and cause production of a MIPS Big Endian Volume Header. This is mutually exclusive with production of other boot blocks like

MBR. It will overwrite the first 512 bytes of any data provided by −G. Up to 15 boot files can be declared by multiple −mips−boot options.

−mipsel-boot iso_rr_path
Declare a data file in the image to be the MIPS Little Endian boot file. This is mutually exclusive with other boot blocks. It will overwrite the first 512 bytes of any data provided by −G. Only a single boot file can be declared by −mipsel−boot.

−B disk_path[,disk_path ...]
Cause one or more data files on disk to be written after the end of the ISO image. A SUN Disk Label will be written into the first 512 bytes of the ISO image which lists this image as partition 1 and the given disk_paths as partition 2 up to 8.
The disk files should contain suitable boot images for SUN SPARC systems.
The pseudo disk_path "..." causes that all empty partition entries become copies of the last non−empty entry. If no other disk_path is given before "..." then all partitions describe the ISO image. In this case, the boot loader code has to be imported by option −G.

−sparc-boot disk_path[,disk_path ...]
Alias of −B.

−sparc-label text
Set the ASCII label text of a SUN Disk Label.

--grub2-sparc-core iso_rr_path
Cause the content address and size of the given data file in the image to be written after the SUN Disk Label. Both numbers are counted in bytes. The address is written as 64 bit big−endian number to byte 0x228. The size is written as 32 bit big−endian number to byte 0x230.

−hppa-cmdline text
Set the PALO command line for HP−PA. Up to 1023 characters are permitted by default. With −hppa−hdrversion 4 the limit is 127.
Note that the first five −hppa options are mandatory, if any of the −hppa options is given. Only option −hppa−hdrversion is allowed to be missing.

−hppa-bootloader iso_rr_path
Designate the given path as HP−PA bootloader file.

−hppa-kernel-32 iso_rr_path
Designate the given path as HP−PA 32 bit kernel file.

−hppa-kernel-64 iso_rr_path
Designate the given path as HP−PA 64 bit kernel file.

−hppa-ramdisk iso_rr_path
Designate the given path as HP−PA RAM disk file.

−hppa-hdrversion number
Choose between PALO header version 5 (default) and version 4. For the appropriate value see in PALO source code: PALOHDRVERSION.

−alpha-boot iso_rr_path
Declare a data file in the image to be the DEC Alpha SRM Secondary Bootstrap Loader and cause production of a boot sector which points to it. This is mutually exclusive with production of other boot blocks like MBR.

Character sets:

Character sets should not matter as long as only english alphanumeric characters are used for file names or as long as all writers and readers of the medium use the same character set. Outside these constraints it may be necessary to let xorriso convert byte codes.
A conversion from input character set to the output character set is performed when an ISO image gets written. Vice versa there is a conversion from output character set to the input character set when an ISO image gets loaded. The sets can be defined by options −input−charset and −output−charset, if needed.

−input-charset character_set_name
Set the character set from which to convert disk file names when inserting them into the ISO image.

−output-charset character_set_name
Set the character set from which to convert names of loaded ISO images and to which to convert names when writing ISO images.

Jigdo Template Extraction:

From man genisoimage: "Jigdo is a tool to help in the distribution of large files like CD and DVD images; see http://atterer.net/jigdo/ for more details. Debian CDs and DVD ISO images are published on the web in jigdo format to allow end users to download them more efficiently."
If the use of libjte was enabled at compile time of xorriso, then **xorrisofs** can produce a .jigdo and a .template file together with a single−session ISO image. If not, then Jigdo options will cause a FAILURE event, which normally leads to program abort.
One may determine the ability for Jigdo by:

```
  $ xorrisofs −version 2>&1 | grep ’^libjte’ && echo YES
```

The .jigdo file contains checksums and symbolic file addresses. The .template file contains the compressed ISO image with reference tags instead of the content bytes of the listed files.
Input for this process are the normal arguments for a **xorrisofs** session with no image loaded, and a checksum file which lists those data files which may be listed in the .jigdo file and externally referenced in the .template file. Each designated file is represented in the checksum file by a single text line:
Checksum as hex digits, 2 blanks, size as 12 decimal digits or blanks, 2 blanks, symbolic file address
The kind of checksum is chosen by −jigdo "checksum_algorithm" with values "md5" (32 hex digits) or "sha256" (64 hex digits). It will also be used for the file address lines in the .jigdo file.
The default is "md5".
The file address in a checksum file line has to bear the same basename as the disk_path of the file which it shall match. The directory path of the file address is decisive for To=From mapping, not for file recognition. After To=From mapping, the file address gets written into the .jigdo file. Jigdo restore tools will convert these addresses into really reachable data source addresses from which they can read.
If the list of jigdo parameters is not empty, then padding will be counted as part of the ISO image.

−jigdo-checksum-algorithm "md5"|"sha256"
Set the checksum algorithm which shall be used for the data file entries in the .jigdo file and is expected in the checksum file. Default is "md5".

−jigdo-jigdo disk_path
Set the disk_path for the .jigdo file with the checksums and download addresses for filling the holes in .template.

−jigdo-template disk_path
Set the disk_path for the .template file with the holed and compressed ISO image copy.

−jigdo-min-file-size size
Set the minimum size for a data file to be listed in the .jigdo file and being a hole in the .template file. size may be a plain number counting bytes, or a number with appended letter "k", "m", "g" to count KiB (1024 bytes), MiB (1024 KiB), or GiB (1024 MiB).

−jigdo-force-checksum disk_path_pattern
adds a regular expression pattern which will get compared with the absolute disk_path of any data file that was not found in the checksum file. A match causes a MISHAP event, which normally does not abort the program run but finally causes a non−zero exit value of the program.

−jigdo-force-md5 disk_path_pattern
Outdated alias of −jigdo−force−checksum.

−jigdo-exclude disk_path_pattern
Add a regular expression pattern which will get compared with the absolute disk_path of any data file. A match causes the file to stay in .template in any case.

−jigdo-map To=From
Add a string pair of the form To=From to the parameter list. If a data file gets listed in the .jigdo file, then it is referred by the file address from its line in the checksum file. This file address gets checked whether it begins with the From string. If so, then this string will be replaced by the To string and a ’:’ character, before it goes into the .jigdo file. The From string should end by a ’/’ character.

−checksum-list disk_path
Set the disk_path where to find the checksum file file with symbolic file addresses and checksums according to −jigdo−checksum−algorithm.

−md5-list disk_path
Outdated alias of −checksum−list.

−jigdo-template-compress "gzip"|"bzip2"
Choose one of "bzip2" or "gzip" for the compression of the template file. The jigdo file is put out uncompressed.

−checksum_algorithm_iso list_of_names
Choose one or more of "md5", "sha1", "sha256", "sha512" for the auxiliary "# Image Hex" checksums in the .jigdo file. The list_of_names may e.g. look like "md5,sha1,sha512". Value "all" chooses all available algorithms. Note that MD5 stays always enabled.

−checksum_algorithm_template list_of_names
Choose the algorithms for the "# Template Hex" checksums in the .jigdo file. The rules for list_of_names are the same as with −checksum_algorithm_iso.

Miscellaneous options:

−print-size
Print to stdandard output the foreseeable number of 2048 byte blocks in the emerging ISO image. Do not produce this image.
The result depends on several settings.
If option −−emul−toc is given, then padding (see −pad) is not counted as part of the image size. In this case either use −no−pad or add 150 (= 300 KiB) to the resulting number.
If mkisofs emulation ends after option −print−size, then the properties of the most recently specified boot image file cannot be edited by subsequent xorriso commands.

--no_rc Only if used as first argument this option prevents reading and interpretation of startup files. See section FILES below.

−help
List supported options to stderr. Original mkisofs options bear their original mkisofs description texts.

−quiet
Suppress most messages of the program run, except those which indicate problems or errors.

−gui
Increase the frequency of pacifier messages while writing an ISO image.

−log-file disk_path
Truncate file disk_path to 0 size and redirect to it all messages which would normally appear on stderr. −log−file with empty text as disk_path re−enables output to stderr.

−v
Enable the output of informational program messages.

−verbose
Alias of −v.

−version

Print to standard output a text that begins with
```
 "mkisofs 2.01−Emulation Copyright (C)"
```
and to standard error the version information of xorriso.

EXAMPLES

Overview of examples:

A simple image production run
Set ISO image paths by -graft-points
Perform multi-session runs
Let xorrisofs work underneath growisofs
Incremental backup of a few directory trees
Incremental backup with accumulated trees
Create bootable images for PC-BIOS and EFI

A simple image production run

A prepared file tree in directory ./for_iso gets copied into the root directory of the ISO image. File permissions get set to read−only for everybody. Joliet attributes for Microsoft systems get added. The resulting image gets written as data file ./image.iso on disk.
```
  $ xorrisofs −r −J −o ./image.iso ./for_iso
```

Set ISO image paths by -graft-points

Without option −graft−points each given disk file is copied into the root directory of the ISO image, maintaining its name. If a directory is given, then its files and sub−directories are copied into the root directory, maintaining their names.
```
  $ xorrisofs ... /home/me/datafile /tmp/directory
```
yields in the ISO image root directory:
```
  /datafile
  /file_1_from_directory
  ...
  /file_N_from_directory
```

With option −graft−points it is possible to put files and directories to arbitrary paths in the ISO image.
```
  $ xorrisofs ... −graft−points /home/me/datafile /dir=/tmp/directory
```
yields in the ISO image root directory:
```
  /datafile
  /dir
```
Eventually needed parent directories in the image will be created automatically:
```
  /datafiles/file1=/home/me/datafile
```
yields in the ISO image:
```
  /datafiles/file1
```
The attributes of directory /datafiles get copied from /home/me on disk.

Normally one should avoid = and \ characters in the ISO part of a pathspec. But if it must be, one may escape them:
```
  /with_\=_and_\\/file=/tmp/directory/file
```
yields in the ISO image:
```
  /with_=_and_\/file
```

Perform multi-session runs

This example works for multi−session media only: CD−R[W], DVD−R[W], DVD+R, BD−R. Add cdrskin option −−grow_overwriteable_iso to all −as cdrecord runs in order to enable multi−session emulation on overwritable media.
The first session is written like this:
```
  $ xorrisofs −graft−points \
            /tree1=prepared_for_iso/tree1 \
    | xorriso −as cdrecord −v dev=/dev/sr0 blank=fast −multi −eject −
```

Follow−up sessions are written like this (the run of dd is only to give demons a chance to spoil it):

```
  $ m=$(xorriso −as cdrecord dev=/dev/sr0 −msinfo)
  $ dd if=/dev/sr0 count=1 >/dev/null 2>&1
  $ xorrisofs −M /dev/sr0 −C $m −graft−points \
            /tree2=prepared_for_iso/tree2 \
    | xorriso −as cdrecord −v dev=/dev/sr0 −waiti −multi −eject −
```

Always eject the drive tray between sessions.
The run of xorriso −as mkisofs will read old sessions via the CD−ROM driver of /dev/sr0. This driver might not be aware of the changed content as long as the medium is not loaded again. In this case the previous session would not be properly assessed by xorriso and the new session would contain only the newly added files.
Some systems have not enough patience with automatic tray loading and some demons may interfere with a first CD−ROM driver read attempt from a freshly loaded medium.
When loading the tray manually, wait 10 seconds after the drive has stopped blinking.
A safe automatic way seems to be a separate run of xorriso for loading the tray with proper waiting, and a subsequent run of dd which shall offer itself to any problems caused by demons assessing the changed drive status. If this does not help, insert a run of "sleep 10" between xorriso and dd.

Let xorrisofs work underneath growisofs

growisofs expects an ISO formatter program which understands options −C and −M. A variable is defined to override the hardcoded default name.

```
  $ export MKISOFS="xorrisofs"
  $ growisofs −Z /dev/dvd /some/files
  $ growisofs −M /dev/dvd /more/files
```

If no "xorrisofs" is available on your system, then you will have to create a link pointing to the xorriso binary and tell growisofs to use it. E.g. by:

```
  $ ln −s $(which xorriso) "$HOME/xorrisofs"
  $ export MKISOFS="$HOME/xorrisofs"
```

One may quit mkisofs emulation by argument "−−" and make use of all xorriso commands. growisofs dislikes options which start with "−o" but −outdev must be set to "−". So use "outdev" instead:

```
  $ growisofs −Z /dev/dvd −−for_backup −− \
            outdev − −update_r /my/files /files
  $ growisofs −M /dev/dvd −−for_backup −− \
            outdev − −update_r /my/files /files
```

Note that −−for_backup is given in the mkisofs emulation. To preserve the recorded extra data it must already be in effect, when the emulation loads the image.

Incremental backup of a few directory trees

This changes the directory trees /open_source_project and /personal_mail in the ISO image so that they become exact copies of their disk counterparts. ISO file objects get created, deleted or get their attributes adjusted accordingly.
ACL, xattr, hard links and MD5 checksums will be recorded. It is expected that inode numbers in the disk filesystem are persistent over cycles of mounting and booting. Files with names matching *.o or *.swp get excluded explicitly.

To be used several times on the same medium, whenever an update of the two disk trees to the medium is desired. Begin with a blank medium and update it until he run fails gracefully due to lack of remaining space on the old one.
Always eject the drive tray between sessions. A run of dd shall give demons a chance to spoil the first read on freshly loaded media.

```
  $ msinfo=$(xorriso −as cdrecord dev=/dev/sr0 −msinfo)
  $ dd if=/dev/sr0 count=1 >/dev/null 2>&1
  $ load_opts=
  $ test −n "$msinfo" && load_opts="−M /dev/sr0 −C $msinfo"
  $ xorrisofs $load_opts −o − −−for_backup −m ’*.o’ −m ’*.swp’ \
    −V PROJ_MAIL_"$(date ’+%Y_%m_%d_%H%M%S’)" −graft−points \
    −old−root / \
    /projects=/home/thomas/projects \
    /personal_mail=/home/thomas/personal_mail \
    | xorriso −as cdrecord dev=/dev/sr0 −v −multi −waiti −eject −
```

This makes sense if the full backup leaves substantial remaining capacity on media and if the expected changes are much smaller than the full backup.

Better do not use your youngest backup for −old−root. Have at least two media which you use alternatingly. So only older backups get endangered by the new write operation, while the newest backup is stored safely on a different medium.
Always have a blank medium ready to perform a full backup in case the update attempt fails due to insufficient remaining capacity. This failure will not spoil the old medium, of course.

If inode numbers on disk are not persistent, then use option −−old−root−no−ino . In this case an update run will compare recorded MD5 sums against the current file content on hard disk.

With **mount** option **−o "sbsector="** on GNU/Linux or **−s** on FreeBSD or NetBSD it is possible to access the session trees which represent the older backup versions. With CD media, GNU/Linux mount accepts session numbers directly by its option "session=".
Multi−session media and most overwritable media written by xorriso can tell the sbsectors of their sessions by xorriso option −toc:

```
 $ xorriso −dev /dev/sr0 −toc
```

xorriso can print the matching mount command for a session number:

```
 $ xorriso −mount_cmd /dev/sr0 session 12 /mnt
```

or for a volume id that matches a search expression:

```
 $ xorriso −mount_cmd /dev/sr0 volid '*2008_12_05*' /mnt
```

Both yield on standard output something like:

```
 mount −t iso9660 −o nodev,noexec,nosuid,ro,sbsector=1460256 '/dev/sr0' '/mnt'
```

The superuser may let xorriso execute the mount command directly:

```
 # osirrox −mount /dev/sr0 "volid" '*2008_12_05*' /mnt
```

Incremental backup with accumulated trees

Solaris does not offer the option to mount older sessions. In order to keep them accessible, one may map all files to a file tree under a session directory and accumulate those directories from session to session. The −root tree is cloned from the −old−root tree before it gets compared with the appropriate trees on disk.
This demands to know the previously used session directory name.
With the first session:

```
 $ xorrisofs −root /session1 \
  −o − −−for_backup −m '*.o' −m '*.swp' \
  −V PROJ_MAIL_"$(date '+%Y_%m_%d_%H%M%S')" −graft−points \
  /projects=/home/thomas/projects \
  /personal_mail=/home/thomas/personal_mail \
  | xorriso −as cdrecord dev=/dev/sr0 −v blank=as_needed \
        −multi −waiti −eject −
```

With the second session, option −old−root refers to /session1 and the new −root is /session2.
Always eject the drive tray between sessions. A run of dd shall give demons a chance to spoil the first read on freshly loaded media.

```
 $ msinfo=$(xorriso −as cdrecord dev=/dev/sr0 −msinfo)
 $ dd if=/dev/sr0 count=1 >/dev/null 2>&1
 $ load_opts=
 $ test −n "$msinfo" && load_opts="−M /dev/sr0 −C $msinfo"
 $ xorrisofs $load_opts −root /session2 −old−root /session1 \
  −o − −−for_backup −m '*.o' −m '*.swp' \
  −V PROJ_MAIL_"$(date ’+%Y_%m_%d_%H%M%S’)" −graft−points \
  /projects=/home/thomas/projects \
  /personal_mail=/home/thomas/personal_mail \
  | xorriso −as cdrecord dev=/dev/sr0 −v −multi −waiti −eject −
```

With the third session, option −old−root refers to /session2. The new −root is /session3. And so on.

Create bootable images for PC-BIOS and EFI

The SYSLINUX/ISOLINUX boot loader suite is popular for booting PC−BIOS. The ISOLINUX wiki prescribes to create on disk a directory ./CD_root and to copy all desired files underneath that directory. Especially file isolinux.bin shall be copied to ./CD_root/isolinux/isolinux.bin . This is the boot image file.
The prescribed mkisofs options can be used unchanged with **xorrisofs**:

```
 $ xorrisofs −o output.iso \
    −b isolinux/isolinux.bin −c isolinux/boot.cat \
    −no−emul−boot −boot−load−size 4 −boot−info−table \
    ./CD_root
```

Put it on CD by a burn program. E.g.:

```
 $ xorriso −as cdrecord −v dev=/dev/sr0 blank=as_needed output.iso
```

The image from above example will boot from CD, DVD or BD, but not from USB stick or other hard−disk−like devices. This can be done by help of an isohybrid MBR. Syslinux provides matching template files as isohdp[fp]x*.bin . E.g. /usr/lib/syslinux/isohdpfx.bin .
If a few hundred KB of size do not matter, then option −partition_offset can be used to create a partition table where partition 1 starts not at block 0. This facilitates later manipulations of the USB stick by tools for partitioning and formatting.
The image from the following example will be prepared for booting via MBR and its first partition will start at hard disk block 64.
It will also boot from optical media.

```
 $ xorrisofs −o output.iso \
    −b isolinux/isolinux.bin −c isolinux/boot.cat \
    −no−emul−boot −boot−load−size 4 −boot−info−table \
    −isohybrid−mbr /usr/lib/syslinux/isohdpfx.bin \
    −partition_offset 16 \
    ./CD_root
```

Become superuser and copy the image to the unpartitioned base device file of the USB stick. On GNU/Linux this is e.g. /dev/sdb, not /dev/sdb1.
CAUTION: This will overwrite any partitioning on the USB stick and make remaining data unaccessible.
So first make sure you got the correct address of the intended device. E.g. by reading 100 MiB data from it and watching it blinking:

```
 # dd bs=2K if=/dev/sdb count=50K >/dev/null
```

Now copy the image onto it

```
 # dd bs=2K if=output.iso of=/dev/sdb
```

Now for EFI:
The boot image file has to be the image of an EFI System Partition, i.e. a FAT filesystem with directory /EFI/BOOT and boot files with EFI prescribed names: BOOTIA32.EFI for 32 bit x86, BOOTx64.EFI for 64 bit AMD/x86 (in UEFI−2.4 there is indeed a lower case "x"), BOOTAA64.EFI for 64 bit ARM. The software in the FAT filesystem should be able to find and inspect the ISO filesystem for boot loader configuration and start of operating system. GRUB2 program grub−mkimage can produce such a FAT filesystem with suitable content, which then uses further GRUB2 software from the ISO filesystem.
EFI boot equipment may be combined with above ISOLINUX isohybrid for PC−BIOS in a not really UEFI−2.4 compliant way, which obviously works well. It yields MBR and GPT partition tables, both with nested partitions. Assumed the EFI System Partition image is ready as ./CD_root/boot/grub/efi.img, add the following options before the directory address ./CD_root:

```
    −eltorito−alt−boot −e ’boot/grub/efi.img’ −no−emul−boot \
    −isohybrid−gpt−basdat \
```

More compliant with UEFI−2.4 is to decide for either MBR or GPT and to append a copy of the EFI System Partition in order to avoid overlap of ISO partition and EFI partition. Here for MBR:

```
    −eltorito−alt−boot −e 'boot/grub/efi.img' −no−emul−boot \
    −append_partition 2 0xef ./CD_root/boot/grub/efi.img \
```

The resulting ISOs are supposed to boot from optical media and USB stick. One may omit option −eltorito−alt−boot if no option −b is used to make the ISO bootable via PC−BIOS.

For ISOs with pure GRUB2 boot equipment consider to use GRUB2 tool grub−mkrescue as frontend to xorrisofs.

If you have a bootable ISO filesystem and want to know its equipment plus a proposal how to reproduce it, try:

```
 $ xorriso −hfsplus on −indev IMAGE.iso \
    −report_el_torito plain −report_system_area plain \
    −print "" −print "======= Proposal for xorrisofs options:" \
    −report_el_torito as_mkisofs
```

FILES

Startup files:

If not −−no_rc is given as the first argument then **xorrisofs** attempts on startup to read and execute lines from the following files:

```
  /etc/default/xorriso
  /etc/opt/xorriso/rc
  /etc/xorriso/xorriso.conf
  $HOME/.xorrisorc
```

The files are read in the sequence given here, but none of them is required to exist. The lines are not interpreted as **xorrisofs** options but as generic xorriso commands. See man xorriso.

After the xorriso startup files, the program tries one by one to open for reading:

```
  ./.mkisofsrc
  $MKISOFSRC
  $HOME/.mkisofsrc
  $(dirname $0)/.mkisofsrc
```

On success it interprets the file content and does not try further files. The last address is used only if start argument 0 has a non−trivial dirname.
The reader currently interprets the following NAME=VALUE pairs:

```
 APPI default for −A
 PUBL default for −publisher
 SYSI default for −sysid
 VOLI default for −V
 VOLS default for −volset
```

Any other lines will be silently ignored.

ENVIRONMENT

The following environment variables influence the program behavior:
HOME is used to find xorriso and mkisofs startup files.
MKISOFSRC may be used to point the program to a mkisofs startup file.
SOURCE_DATE_EPOCH belongs to the specs of reproducible−builds.org. It is supposed to be either undefined or to contain a decimal number which tells the seconds since january 1st 1970. If it contains a number, then it is used as time value to set the default of −−modification−date=. −−gpt_disk_guid defaults to "modification−date". The default of −−set_all_file_dates is then "set_to_mtime". Further the "now" time for ISO nodes without disk source is then set to the SOURCE_DATE_EPOCH value.
Startup files and program options can override the effect of SOURCE_DATE_EPOCH.

SEE ALSO

For generic xorriso command mode
xorriso(1)

For the cdrecord emulation of xorriso
xorrecord(1)

For mounting xorriso generated ISO 9660 images (-t iso9660)
mount(8)

Other programs which produce ISO 9660 images
mkisofs(8), genisoimage(8)

Programs which burn sessions to optical media
growisofs(1), cdrecord(1), wodim(1), cdrskin(1), xorriso(1)

ACL and xattr
getfacl(1), setfacl(1), getfattr(1), setfattr(1)

MD5 checksums
md5sum(1)

On FreeBSD the commands for xattr and MD5 differ
getextattr(8), setextattr(8), md5(1)

BUGS

To report bugs, request help, or suggest enhancements for **xorriso**, please send electronic mail to the public list <bug−xorriso@gnu.org>. If more privacy is desired, mail to <scdbackup@gmx.net>.
Please describe what you expect **xorriso** to do, the program arguments or dialog commands by which you tried to achieve it, the messages of **xorriso**, and the undesirable outcome of your program run.
Expect to get asked more questions before solutions can be proposed.

AUTHOR

Thomas Schmitt <scdbackup@gmx.net>
for libburnia−project.org

COPYRIGHT

Copyright (c) 2011 − 2021 Thomas Schmitt
Permission is granted to distribute this text freely. It shall only be modified in sync with the technical properties of xorriso. If you make use of the license to derive modified versions of xorriso then you are entitled to modify this text under that same license.

CREDITS

xorrisofs is in part based on work by Vreixo Formoso who provides libisofs together with Mario Danic who also leads the libburnia team. Vladimir Serbinenko contributed the HFS+ filesystem code and related knowledge.
Compliments towards Joerg Schilling whose cdrtools served me for ten years.