NAME

xorriso − creates, loads, manipulates and writes ISO 9660 filesystem images with Rock Ridge extensions.

SYNOPSIS

xorriso [*settings*|*actions*]

DESCRIPTION

xorriso is a program which copies file objects from POSIX compliant filesystems into Rock Ridge enhanced ISO 9660 filesystems and performs session−wise manipulation of such filesystems. It can load the management information of existing ISO images and it writes the session results to optical media or to filesystem objects.
Vice versa **xorriso** is able to copy file objects out of ISO 9660 filesystems.

A special property of **xorriso** is that it needs neither an external ISO 9660 formatter program nor an external burn program for CD, DVD or BD but rather incorporates the libraries of libburnia−project.org .

Overview of features:

Operates on an existing ISO image or creates a new one.
Copies files from disk filesystem into the ISO image.
Copies files from ISO image to disk filesystem (see osirrox).
Renames or deletes file objects in the ISO image.
Changes file properties in the ISO image.
Updates ISO subtrees incrementally to match given disk subtrees.
Writes result either as completely new image or as add−on session to optical media or filesystem objects.
Can activate ISOLINUX and GRUB boot images via El Torito and MBR.
Can perform multi−session tasks as emulation of mkisofs and cdrecord.
Can record and restore hard links and ACL.
Content may get zisofs compressed or filtered by external processes.
Can issue commands to mount older sessions on GNU/Linux or FreeBSD.
Can check media for damages and copy readable blocks to disk.
Can attach MD5 checksums to each data file and the whole session.
Scans for optical drives, blanks re−usable optical media.
Reads its instructions from command line arguments, dialog, and files.
Provides navigation commands for interactive ISO image manipulation.
Adjustable thresholds for abort, exit value, and problem reporting.

Note that **xorriso** does not write audio CDs and that it does not produce UDF filesystems which are specified for official video DVD or BD.

General information paragraphs:

Session model
Media types and states
Creating, Growing, Modifying, Blind Growing
Libburn drives
Rock Ridge, POSIX, X/Open, El Torito, ACL, xattr
Command processing
Dialog, Readline, Result pager

Maybe you first want to have a look at section EXAMPLES near the end of this text before reading the next few hundred lines of background information.

Session model:

Unlike other filesystems, **ISO 9660** (aka **ECMA−119**) is not intended for read−write operation but rather for being generated in a single sweep and being written to media as a **session**.
The data content of the session is called filesystem **image**.

The written image in its session can then be mounted by the operating system for being used read−only. GNU/Linux is able to mount ISO images from block devices, which may represent optical media, other media or via a loop device even from regular disk files. FreeBSD mounts ISO images from devices that represent arbitrary media or from regular disk files.

This session usage model has been extended on CD media by the concept of **multi−session** , which adds information to the CD and gives the mount programs of the operating systems the addresses of the entry points of each session. The mount programs recognize block devices which represent CD media and will by default mount the image in the last session.
This session usually contains an updated directory tree for the whole medium which governs the data contents in all recorded sessions. So in the view of the mount program all sessions of a particular medium together form a single filesystem image.
Adding a session to an existing ISO image is in this text referred as **growing**.
The multi−session model of the MMC standard does not apply to all media types. But program growisofs by Andy Polyakov showed how to extend this functionality to overwritable media or disk files which carry valid ISO 9660 filesystems.

xorriso provides growing as well as an own method named **modifying** which produces a completely new ISO image from the old one and the modifications. See paragraph Creating, Growing, Modifying, Blind Growing below.

xorriso adopts the concept of multi−session by loading an image directory tree if present, by offering to manipulate it by several actions, and by writing the new image to the target medium.
The first session of a **xorriso** run begins by the definition of the input drive with the ISO image or by the definition of an output drive. The session ends by command −commit which triggers writing. A −commit is done automatically when the program ends regularly.

After −commit a new session begins with the freshly written one as input. A new input drive can only be chosen as long as the loaded ISO image was not altered. Pending alteration can be revoked by command −rollback.

Writing a session to the target is supposed to be very expensive in terms of time and of consumed space on appendable or write−once media. Therefore all intended manipulations of a particular ISO image should be done in a single session. But in principle it is possible to store intermediate states and to continue with image manipulations.

Media types and states:

There are two families of media in the MMC standard:
Multi−session media are CD−R, CD−RW, DVD−R, DVD+R, DVD+R/DL, BD−R, and unformatted DVD−RW. These media provide a table of content which describes their existing sessions. See command **−toc**.
Similar to multi−session media are DVD−R DL and minimally blanked DVD−RW. They record only a single session of which the size must be known in advance. **xorriso** will write onto them only if command −close is set to "on".
Overwritable media are DVD−RAM, DVD+RW, BD−RE, and formatted DVD−RW. They offer random write access but do not provide information about their session history. If they contain one or more ISO 9660 sessions and if the first session was written by **xorriso**, then a table of content can be emulated. Else only a single overall session will be visible.
DVD−RW media can be formatted by −format "full". They can be made unformatted by −blank "deformat".
Regular files and block devices are handled as overwritable media. Pipes and other writeable file types are handled as blank multi−session media.

These media can assume several states in which they offer different capabilities.
Blank media can be written from scratch. They contain no ISO image suitable for **xorriso**.
Blank is the state of newly purchased optical media. With used CD−RW and DVD−RW it can be achieved by action −blank "as_needed". Overwritable media are considered blank if they are new or if they have been marked as blank by **xorriso**. Action −blank "as_needed" can be used to do this marking on overwritable media, or to apply mandatory formatting to new media if necessary.
Appendable media accept further sessions. Either they are MMC multi−session media in appendable state, or they are overwritable media which contain an ISO image suitable for **xorriso**.
Appendable is the state after writing a session with command −close off.

Closed media cannot be written. They may contain an ISO image suitable for **xorriso**.
Closed is the state of DVD−ROM media and of multi−session media which were written with command −close on. If the drive is read−only hardware then it will probably show any media as closed CD−ROM or DVD−ROM.
Overwritable media assume this state in such read−only drives or if they contain unrecognizable data in the first 32 data blocks.
Read−only drives may or may not show session histories of multi−session media. Often only the first and the last session are visible. Sometimes not even that. Command −rom_toc_scan might or might not help in such cases.

Creating, Growing, Modifying, Blind Growing:

A new empty ISO image gets **created** if there is no input drive with a valid ISO 9660 image when the first time an output drive is defined. This is achieved by command −dev on blank media or by command −outdev on media in any state.
The new empty image can be populated with directories and files. Before it can be written, the medium in the output drive must get into blank state if it was not blank already.

If there is a input drive with a valid ISO image, then this image gets loaded as foundation for manipulations and extension. The constellation of input and output drive determines which write method will be used. They have quite different capabilities and constraints.

The method of **growing** adds new data to the existing data on the medium. These data comprise of new file content and they override the existing ISO 9660 + Rock Ridge directory tree. It is possible to hide files from previous sessions but they still exist on the medium and with many types of optical media it is quite easy to recover them by mounting older sessions.
Growing is achieved by command −dev.

The write method of **modifying** produces compact filesystem images with no outdated files or directory trees. Modifying can write its images to target media which are completely unsuitable for multi−session operations. E.g. DVD−RW which were treated with −blank deformat_quickest, DVD−R DL, named pipes, character devices, sockets. On the other hand modified sessions cannot be written to appendable media but to blank media only.
So for this method one needs either two optical drives or has to work with filesystem objects as source and/or target medium.
Modifying takes place if input drive and output drive are not the same and if command −grow_blindly is set to its default "off". This is achieved by commands −indev and −outdev.

If command −grow_blindly is set to a non−negative number and if −indev and −outdev are both set to different drives, then **blind growing** is performed. It produces an add−on session which is ready for being written to the given block address. This is the usage model of

```
 mkisofs −M $indev −C $msc1,$msc2 −o $outdev
```

which gives much room for wrong parameter combinations and should thus only be employed if a strict distinction between ISO formatter **xorriso** and the burn program is desired. −C $msc1,$msc2 is equivalent to:

```
 −load sbsector $msc1 −grow_blindly $msc2
```

Libburn drives:

Input drive, i.e. source of an existing or empty ISO image, can be any random access readable libburn drive: optical media with readable data, blank optical media, regular files, block devices.
Output drive, i.e. target for writing, can be any libburn drive. Some drive types do not support the method of growing but only the methods of modifying and blind growing. They all are suitable for newly created images.

All drive file objects have to offer rw−permission to the user of **xorriso**. Even those which will not be usable for reading an ISO image.
With any type of drive object, the data are considered to be organized in blocks of 2 KiB. Access happens in terms of Logical Block Address (**LBA**) which gives the number of a particular data block.

MMC compliant (i.e. optical) drives on GNU/Linux usually get addressed by the path of their block device

or of their generic character device. E.g.
```
  −dev /dev/sr0
  −dev /dev/hdc
  −dev /dev/sg2
```
By default xorriso will try to map the given address to /dev/hd* and /dev/sr*. The command −scsi_dev_family can redirect the mapping from sr to scd or sg. The latter does not suffer from the concurrency problems which plague /dev/sr of Linux kernels since version 3. But it does not yield the same addresses which are used by mount(8) or by open(2) for read(2).
On FreeBSD the device files have names like
```
  −dev /dev/cd0
```
On NetBSD:
```
  −dev /dev/rcd0d
```
On OpenSolaris:
```
  −dev /dev/rdsk/c4t0d0s2
```
Get a list of accessible drives by command
```
  −device_links
```
It might be necessary to do this as **superuser** in order to see all drives and to then allow rw−access for the intended users. Consider to bundle the authorized users in a group like old "floppy".

Filesystem objects of nearly any type can be addressed by prefix "stdio:" and their path in the filesystem. E.g.:
```
  −dev stdio:/dev/sdc
```
The default setting of −drive_class allows the user to address files outside the /dev tree without that prefix. E.g.:
```
  −dev /tmp/pseudo_drive
```
If path leads to a regular file or to a block device then the emulated drive is random access readable and can be used for the method of growing if it already contains a valid ISO 9660 image. Any other file type is not readable via "stdio:" and can only be used as target for the method of modifying or blind growing. Non−existing paths in existing directories are handled as empty regular files.

A very special kind of pseudo drive are open file descriptors. They are depicted by "stdio:/dev/fd/" and descriptor number (see man 2 open).
Addresses "−" or "stdio:/dev/fd/1" depict standard output, which normally is the output channel for result texts. To prevent a fatal intermingling of ISO image and text messages, all result texts get redirected to stderr if −*dev "−" or "stdio:/dev/fd/1" is among the start arguments of the program.
Standard output is currently suitable for creating one session per program run without dialog. Use in other situations is discouraged and several restrictions apply:
It is not allowed to use standard output as pseudo drive if it was not among the start arguments. Do not try to fool this ban via backdoor addresses to stdout.
If stdout is used as drive, then −use_readline is permanently disabled. Use of backdoors can cause severe memory and/or tty corruption.

Be aware that especially the superuser can write into any accessible file or device by using its path with the "stdio:" prefix. By default any address in the /dev tree without prefix "stdio:" will work only if it leads to a MMC drive.
One may use command **−ban_stdio_write** to surely prevent this risk and to restrict drive usage to MMC drives.
One may prepend "mmc:" to a path to surely disallow any automatic "stdio:".
By command −drive_class one may ban certain paths or allow access without prefix "stdio:" to other paths.

Rock Ridge, POSIX, X/Open, El Torito, ACL, xattr:

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.
This is what **xorriso** uses for a decent representation of the disk files within the ISO image. **xorriso** produces Rock Ridge information by default. It is strongly discouraged to disable this feature.

xorriso is not named "porriso" because POSIX only guarantees 14 characters of filename length. It is the X/Open System Interface standard XSI which demands a file name length of up to 255 characters and paths of up to 1024 characters. Rock Ridge fulfills this demand.

An **El Torito** boot record points the BIOS bootstrapping facility to 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.
Most bootable GNU/Linux CDs are equipped with ISOLINUX or GRUB boot images. **xorriso** is able to create or maintain an El Torito object which makes such an image bootable. For details see command −boot_image.
It is possible to make ISO images bootable from USB stick or other hard−disk−like media. Several options install a **MBR** (Master Boot Record), It may get adjusted according to the needs of the intended boot firmware and the involved boot loaders, e.g. GRUB2 or ISOLINUX. A MBR contains boot code and a partition table. The new MBR of a follow−up session can get in effect only on overwritable media.
MBR 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.
Emulation −as mkisofs 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.
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.
There is support for further facilities: MIPS Big Endian (SGI), MIPS Little Endian (DEC), SUN SPARC, HP−PA. Those are mutually not combinable and also not combinable with MBR, GPT, or APM.

ACL are an advanced way of controlling access permissions to file objects. Neither ISO 9660 nor Rock Ridge specify a way to record ACLs. So libisofs has introduced a standard conformant extension named AAIP for that purpose. It uses this extension if enabled by command **−acl**.
AAIP enhanced images are supposed to be mountable normally, but one cannot expect that the mounted filesystem will show and respect the ACLs. For now, only **xorriso** is able to retrieve those ACLs. It can bring them into effect when files get restored to an ACL enabled file system or it can print them in a format suitable for tool setfacl.
Files with ACL show as group permissions the setting of entry "mask::" if that entry exists. Nevertheless the non−listed group members get handled according to entry "group::". When removing ACL from a file, **xorriso** brings "group::" into effect.
Recording and restoring of ACLs from and to local files works currently only on GNU/Linux and FreeBSD.

xattr (aka EA, or extattr) are pairs of name and value which can be attached to file objects. AAIP is able to represent them and **xorriso** can record and restore them.
But be aware that pairs with names of non−user namespaces are not necessarily portable between operating systems and not even between filesystems. Only those which begin with "user.", like "user.x" or "user.whatever", can unconditionally be expected to be appropriate on other machines and disks. Processing of other xattr may need administrator privileges.
Name has to be a 0 terminated string. Value may be any array of bytes which does not exceed the size of 4095 bytes. xattr processing happens only if it is enabled by command **−xattr**.
As with ACL, currently only **xorriso** is able to retrieve xattr from AAIP enhanced images, to restore them to xattr capable file systems, or to print them.
Recording and restoring of xattr from and to local files works currently only on GNU/Linux and FreeBSD, where they are known as extattr.

Command processing:

Commands are either actions which happen immediately or settings which influence following actions. So their sequence does matter, unless they are given as program arguments and command **−x** is among them.
Commands consist of a command word, followed by zero or more parameter words. If the list of parameter

words is of variable length (indicated by "[...]" or "[***]") then it must be terminated by either the **list delimiter**, occur at the end of the argument list, or occur at the end of an input line.

At program start the list delimiter is the string "−−". This may be changed with the −list_delimiter command in order to allow "−−" as parameter in a variable length list. However, it is advised to reset the delimiter to "−−" immediately afterwards.
For brevity the list delimiter is referred as "−−" throughout this text.
The list delimiter is silently ignored if it appears after the parameters of a command with a fixed list length. It is handled as normal text if it appears among the parameters of such a command.

Pattern expansion converts a list of pattern words into a list of existing file addresses. Unmatched pattern words will appear unaltered in that result list.
Pattern matching supports the usual shell parser wildcards '*' '?' '[xyz]' and respects '/' as the path separator, which may only be matched literally.
Pattern expansion is a property of some particular commands and not a general feature. It is controlled by commands −iso_rr_pattern and −disk_pattern. Commands which use pattern expansion all have variable parameter lists which are specified in this text by "[***]" rather than "[...]".
Some other commands perform pattern matching unconditionally.

Command and parameter words are either read from the program arguments, where one argument is one word, or from quoted input lines where words are recognized similar to the quotation rules of a shell parser.
xorriso is not a shell, although it might appear so at first glimpse. Be aware that the interaction of quotation marks and pattern symbols like "*" differs from the usual shell parsers. In **xorriso**, a quotation mark does not make a pattern symbol literal.

Quoted input converts whitespace−separated text into words. The double quotation mark " and the single quotation mark ' can be used to enclose whitespace and make it part of words (e.g. of file names). 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.
Quoted input accepts any 8−bit character except NUL (0) as the content of the quotes. Nevertheless it can be cumbersome for the user to produce those characters directly. Therefore quoted input and program arguments offer optional **Backslash Interpretation** which can represent all 8−bit characters except NUL (0) via backslash codes as in $'...' of bash.
This is not enabled by default. See command −backslash_codes.

When the program starts then it first looks for argument −no_rc. If this is not present then it looks for its startup files and reads their content as command input lines. Then it interprets the program arguments as commands and parameters. Finally it enters dialog mode if command −dialog "on" has been executed by this point.

The program ends either by command −end, or by the end of program arguments if dialog mode has not been enabled at that point, or by a problem event which triggers the threshold of command −abort_on.

Dialog, Readline, Result pager:

Dialog mode prompts for a quoted input line, parses it into words, and performs them as commands with their parameters. It provides assisting services to make dialog more comfortable.

Readline is an enhancement for the input line. You may already know it from the bash shell. Whether it is available in **xorriso** depends on the availability of package readline−dev at the time when **xorriso** was built from its sourcecode.
Readline lets the user move the cursor over the text in the line by help of the Left and the Right arrow keys. Text may be inserted at the cursor position. The Delete key removes the character under the cursor. Up and Down arrow keys navigate through the history of previous input lines.
See man readline for more info about libreadline.

Command −page activates a built−in result text pager which may be convenient in dialog mode. After an action has output the given number of terminal lines, the pager prompts the user for a line of input.
An empty line lets **xorriso** resume work until the next page is output.
The single character "@" disables paging for the current action.
"@@@", "x", "q", "X", or "Q" request that the current action aborts and suppress further result output.

Any other line input will be interpreted as new dialog line. The current action is requested to abort. Afterwards, the input line is executed.

Some actions apply paging to their info output, too.
The request to abort may or may not be obeyed by the current action. All actions try to abort as soon as possible.

OPTIONS

All command words are shown with a leading dash although this dash is not mandatory for the command to be recognized. Nevertheless within command −as the dashes of the emulated commands are mandatory.
Normally any number of leading dashes is ignored with command words and inner dashes are interpreted as underscores.

Execution order of program arguments:

By default the program arguments of a xorriso run are interpreted as a sequence of commands which get performed exactly in the given order. This requires the user to write commands for desired settings before the commands which shall be influenced by those settings.
Many other programs support program arguments in an arbitrary ordering and perform settings and actions in a sequence at their own discretion. xorriso provides an option to enable such a behavior at the cost of loss of expressivity.

−x Enable automatic sorting of program arguments into a sequence that (most likely) is sensible. This command may be given at any position among the commands which are handed over as program arguments.
Note: It works only if it is given as program argument and with a single dash (i.e. "−x"). It will not work in startup files, nor with −options_from_file, nor in dialog mode, nor as "x" and finally not as "−−x". It affects only the commands given as program arguments.

−list_arg_sorting
List all xorriso commands in the order which applies if command −x is in effect.
This list may also be helpful without −x for a user who ponders over the sequence in which to put commands. Deviations from the listed sorting order may well make sense, though.

Acquiring source and target drive:

The effect of acquiring a drive may depend on several commands in the next paragraph "Influencing the behavior of image loading". If desired, their enabling commands have to be performed before the commands which acquire the drive.

−dev address
Set input and output drive to the same address and load an ISO image if it is present. If there is no ISO image then create a blank one. Set the image expansion method to growing.
This is only allowed as long as no changes are pending in the currently loaded ISO image. If changes are pending, then one has to perform −commit or −rollback first.
Special address string "−" means standard output, to which several restrictions apply. See above paragraph "Libburn drives".
An empty address string "" gives up the current device without acquiring a new one.

−indev address
Set input drive and load an ISO image if present. If the new input drive differs from −outdev then switch from growing to modifying or to blind growing. It depends on the setting of −grow_blindly which of both gets activated. The same rules and restrictions apply as with −dev.

−outdev address
Set output drive and if it differs from the input drive then switch from growing to modifying or to blind growing. Unlike −dev and −indev this action does not load a new ISO image. So it can be performed even if there are pending changes.
−outdev can be performed without previous −dev or −indev. In that case an empty ISO image with no changes pending is created. It can either be populated by help of −map, −add et.al. or it can be discarded silently if −dev or −indev are performed afterwards.

Special address string "−" means standard output, to which several restrictions apply. See above paragraph "Libburn drives".
An empty address string "" gives up the current output drive without acquiring a new one. No writing is possible without an output drive.

−drive_class "harmless"|"banned"|"caution"|"clear_list" disk_pattern
Add a drive path pattern to one of the safety lists or make those lists empty. There are three lists defined which get tested in the following sequence:
If a drive address path matches the "harmless" list then the drive will be accepted. If it is not a MMC device then the prefix "stdio:" will be prepended automatically. This list is empty by default.
Else if the path matches the "banned" list then the drive will not be accepted by **xorriso** but rather lead to a FAILURE event. This list is empty by default.
Else if the path matches the "caution" list and if it is not a MMC device, then its address must have the prefix "stdio:" or it will be rejected. This list has by default one entry: "/dev".
If a drive path matches no list then it is considered "harmless". By default these are all paths which do not begin with directory "/dev".
A path matches a list if one of its parent paths or itself matches a list entry. Address prefix "stdio:" or "mmc:" will be ignored when testing for matches.
By pseudo−class "clear_list" and pseudo−patterns "banned", "caution", "harmless", or "all", the lists may be made empty.
E.g.: −drive_class clear_list banned
One will normally define the −drive_class lists in one of the **xorriso** Startup Files.
Note: This is not a security feature but rather a bumper for the superuser to prevent inadverted mishaps. For reliably blocking access to a device file you have to deny its rw−permissions in the filesystem.

−drive_access "exclusive"|"shared":"unrestricted"|"readonly"
Control whether device file locking mechanisms shall be used when acquiring a drive, and whether status or content of the medium in the drive may be altered. Useful and most harmless are the setting "shared:readonly" and the default setting "exclusive:unrestricted".
"exclusive" enables tests and locks when acquiring the drive. It depends on the operating system which locking mechanisms get applied, if any. On GNU/Linux it is open(O_EXCL). On FreeBSD it is flock(LOCK_EX).
"shared" disables the use of these mechanisms to become able to acquire drives which are mounted, or opened by some process, or guarded by /dev/pktcdvd*.
"unrestricted" enables all technically appropriate operations on an acquired drive. "shared:unrestricted" risks to get own burn runs spoiled by other processes or to vice versa spoil activities of such processes. So use "exclusive:unrestricted" unless you know for sure that "shared" is safe.
"readonly" disables operations which might surprise a co−user of the drive. For −outdev these are formatting, blanking, writing, ejecting. For −indev this is ejecting. Be aware that even reading and drive status inquiries can disturb an ongoing burn run on CD−R[W] and DVD−R[W].

−scsi_dev_family "default"|"sr"|"scd"|"sg"
GNU/Linux specific:
By default, xorriso tries to map Linux drive addresses to /dev/sr* before they get opened for operating the drive. This coordinates well with other use cases of optical drives, like mount(8). But since year 2010 all /dev/sr* share a global lock which allows only one drive to process an SCSI command while all others have to wait for its completion. This yields awful throughput if more than one drive is writing or reading simultaneously. The global lock is not applied to device files /dev/sg* and also not if the xorriso drive address is prepended by "stdio:".
So for simultaneous burn runs on modern GNU/Linux it is advisable to perform −scsi_dev_family "sg" before any −dev, −indev, or −outdev. The drive addresses may then well be given as /dev/sr* but will nevertheless get used as the matching /dev/sg*.
If you decide so, consider to put the command into a global startup file like /etc/opt/xorriso/rc.

−grow_blindly "off"|predicted_nwa

If predicted_nwa is a non−negative number then perform blind growing rather than modifying if −indev and −outdev are set to different drives. "off" or "−1" switch to modifying, which is the default.
predicted_nwa is the block address where the add−on session of blind growing will finally end up. It is the responsibility of the user to ensure this final position and the presence of the older sessions. Else the overall ISO image will not be mountable or will produce read errors when accessing file content. **xorriso** will write the session to the address as obtained from examining −outdev and not necessarily to predicted_nwa.
During a run of blind growing, the input drive is given up before output begins. The output drive is given up when writing is done.

Influencing the behavior of image loading:

The following commands should normally be performed before loading an image by acquiring an input drive. In rare cases it is desirable to activate them only after image loading.

−read_speed code|number[k|m|c|d|b]

Set the speed for reading. Default is "none", which avoids to send a speed setting command to the drive before reading begins.
Further special speed codes are:
"max" (or "0") selects maximum speed as announced by the drive.
"min" (or "−1") selects minimum speed as announced by the drive.
Speed can be given in media dependent numbers or as a desired throughput per second in MMC compliant kB (= 1000) or MB (= 1000 kB). Media x−speed factor can be set explicitly by "c" for CD, "d" for DVD, "b" for BD, "x" is optional.
Example speeds:

```
 706k = 706kB/s = 4c = 4xCD
 5540k = 5540kB/s = 4d = 4xDVD
```

If there is no hint about the speed unit attached, then the medium in the −indev will decide. Default unit is CD = 176.4k.
Depending on the drive, the reported read speeds can be deceivingly low or high. Therefore "min" cannot become higher than 1x speed of the involved medium type. Read speed "max" cannot become lower than 52xCD, 24xDVD, or 20xBD, depending on the medium type.
MMC drives usually activate their own idea of speed and take the speed value given by the burn program only as hint for their own decision. Friendly drives adjust their constant angular velocity so that the desired speed is reached at the outer rim of the medium. But often there is only the choice between very slow and very loud.
Sometimes no speed setting is obeyed at all, but speed is adjusted to the demand frequency of the reading program. So xorriso offers to set an additional software enforced limit by prefix "soft_force:". The program will take care not to read faster than the soft_force speed. This may be combined with setting the drive speed to a higher value. Setting "soft_force:0" disables this feature.
"soft_force:" tries to correct in subsequent waiting periods lost or surplus time of up to 0.25 seconds. This smoothens the overall data stream but also enables short times of higher speed to compensate short times of low speed. Prefix "soft_corr:" sets this hindsight span by giving a number of microseconds. Not more than 1 billion = 1000 seconds. Very short times can cause speed deviations, because systematic inaccuracies of the waiting function cannot be compensated.
Examples (combinable):

```
 −read_speed 6xBD
 −read_speed soft_force:4xBD −read_speed soft_corr:100000
```

−load entity id

Load a particular (possibly outdated) ISO session from −dev or −indev. Usually all available sessions are shown with command −toc.
entity depicts the kind of addressing. id depicts the particular address. The following entities are defined:

"auto" with any id addresses the last session in −toc. This is the default.
"session" with id being a number as of a line "ISO session", column "Idx".
"track" with id being a number as of a line "ISO track", column "Idx".
"lba" or "sbsector" with a number as of a line "ISO ...", column "sbsector".
"volid" with a search pattern for a text as of a line "ISO ...", column "Volume Id".
Addressing a non−existing entity or one which does not represent an ISO image will either abandon −indev or at least lead to a blank image.
If an input drive is set at the moment when −load is executed, then the addressed ISO image is loaded immediately. Else, the setting will be pending until the next −dev or −indev. After the image has been loaded once, the setting is valid for −rollback until next −dev or −indev, where it will be reset to "auto".

−displacement [-]lba

Compensate a displacement of the image versus the start address for which the image was prepared. This affects only loading of ISO images and reading of their files. The multi−session method of growing is not allowed as long as −displacement is non−zero. I.e. −indev and −outdev must be different. The displacement gets reset to 0 before the drive gets re−acquired after writing.
Examples:
If a track of a CD starts at block 123456 and gets copied to a disk file where it begins at block 0, then this copy can be loaded with
```
 −displacement −123456
```
If an ISO image was written onto a partition with offset of 640000 blocks of 512 bytes, then it can be loaded from the base device by
```
 −load sbsector 160000 −displacement 160000
```
(If the partition start address is not divisible by 4, then you will have to employ a loop device instead.)
In both cases, the ISO sessions should be self contained, i.e. not add−on sessions to an ISO image outside their track or partition.

−read_fs "any"|"norock"|"nojoliet"|"ecma119"

Specify which kind of filesystem tree to load if present. If the wish cannot be fulfilled, then ECMA−119 names are loaded and converted according to −ecma119_map.
"any" first tries to read Rock Ridge. If not present, Joliet is tried.
"norock" does not try Rock Ridge.
"nojoliet" does not try Joliet.
"ecma119" tries neither Rock Ridge nor Joliet.

−assert_volid pattern severity

Refuse to load ISO images with volume IDs which do not match the given search pattern. When refusing an image, give up the input drive and issue an event of the given severity (like FAILURE, see −abort_on). An empty search pattern accepts any image.
This command does not hamper the creation of an empty image from blank input media and does not discard an already loaded image.

−in_charset character_set_name

Set the character set from which to convert file names when loading an image. See paragraph "Character sets" for more explanations. When loading the written image after −commit the setting of −out_charset will be copied to −in_charset.

−auto_charset "on"|"off"

Enable or disable recording and interpretation of the output character set name in an xattr attribute of the image root directory. If enabled and if a recorded character set name is found, then this name will be used as name of the input character set when reading an image.
Note that the default output charset is the local character set of the terminal where **xorriso** runs. Before attributing this local character set to the produced ISO image, check whether the terminal properly displays all intended filenames, especially exotic national characters.

−hardlinks mode[:mode...]

Enable or disable loading and recording of hardlink relations.
In default mode "off", iso_rr files lose their inode numbers at image load time. Each iso_rr file object which has no inode number at image generation time will get a new unique inode number if −compliance is set to new_rr.
Mode "on" preserves inode numbers from the loaded image if such numbers were recorded. When committing a session it searches for families of iso_rr files which stem from the same disk file, have identical content filtering and have identical properties. The family members all get the same inode number. Whether these numbers are respected at mount time depends on the operating system.
Command −lsl displays hardlink counts if "lsl_count" is enabled. This can slow down the command substantially after changes to the ISO image have been made. Therefore the default is "no_lsl_count".
Commands −update and −update_r track splits and fusions of hard links in filesystems which have stable device and inode numbers. This can cause automatic last minute changes before the session gets written. Command −hardlinks "perform_update" may be used to do these changes earlier, e.g. if you need to apply filters to all updated files.
Mode "without_update" avoids hardlink processing during update commands. Use this if your filesystem situation does not allow −disk_dev_ino "on".
xorriso commands which extract files from an ISO image try to hardlink files with identical inode number. The normal scope of this operation is from image load to image load. One may give up the accumulated hard link addresses by −hardlinks "discard_extract".
A large number of hardlink families may exhaust −temp_mem_limit if not −osirrox "sort_lba_on" and −hardlinks "cheap_sorted_extract" are both in effect. This restricts hard linking to other files restored by the same single extract command. −hardlinks "normal_extract" re−enables wide and expensive hardlink accumulation.

−acl "on"|"off"

Enable or disable processing of ACLs. If enabled, then**xorriso** will obtain A CLs from disk file objects, store ACLs in the ISO image using the libisofs specific AAIP format, load AAIP data from ISO images, test ACL during file comparison, and restore ACLs to disk files when extracting them from ISO images. See also commands −getfacl, −setfacl.

−xattr "on"|"user"|"any"|"off"

Enable or disable processing of xattr attributes. If enabled, then**xorriso** will handle xattr similar to ACL. See also commands −getfattr, −setfattr and above paragraph about xattr.
Modes "on" and "user" read and write only attributes from namespace "user".
Mode "any" processes attributes of all namespaces. This might need administrator privileges, even if the owner of the disk file tries to read or write the attributes.
Note that xattr from namespace "isofs." are never read from disk or restored to disk. Further it is not possible to set them via xorriso xattr manipulation commands.

−md5 "on"|"all"|"off"|"load_check_off"

Enable or disable processing of MD5 checksums for the overall session and for each single data file. If enabled then images with checksum tags get loaded only if the tags of superblock and directory tree match properly. The MD5 checksums of data files and whole session get loaded from the image if there are any.
With commands −compare and −update the recorded MD5 of a file will be used to avoid content reading from the image. Only the disk file content will be read and compared with that MD5. This can save much time if −disk_dev_ino "on" is not suitable.
Commands which copy whole data files from ISO to hard disk will verify the copied data stream by the recorded MD5, if −osirrox "check_md5_on" is set.
At image generation time they are computed for each file which gets its data written into the new session. The checksums of files which have their data in older sessions get copied into the new session. Superblock, tree and whole session get a checksum tag each.
Mode "all" will additionally check during image generation whether the checksum of a data file

changed between the time when its reading began and the time when it ended. This implies reading every file twice.
Mode "load_check_off" together with "on" or "all" will load recorded MD5 sums but not test the recorded checksum tags of superblock and directory tree. This is necessary if growisofs was used as burn program, because it does not overwrite the superblock checksum tag of the first session. Therefore load_check_off is in effect when **xorriso** −as mkisofs option −M is performed.
The test can be re−enabled by mode "load_check_on".
Checksums can be exploited via commands −check_md5, −check_md5_r, via find actions get_md5, check_md5, and via −check_media.

−for_backup
Enable all extra features which help to produce or to restore backups with highest fidelity of file properties. Currently this is a shortcut for:
−hardlinks on −acl on −xattr any −md5 on
If you 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. But at restore time, missing privileges will cause failure events.
Command −xattr "user" after command −for_backup excludes non−user attributes from being recorded or restored.

−ecma119_map "stripped"|"unmapped"|"lowercase"|"uppercase"
Choose the conversion of file names when a session gets loaded, if they stem neither from a Rock Ridge name nor from a Joliet name.
Mode "stripped" is the default. It shows the names as found in the ISO but removes trailing ";1" or ".;1" if present.
Mode "unmapped" shows names as found without removing characters. Warning: Multi−session converts "xyz;1" to "xyz_1" and maybe adds new ";1".
Mode "lowercase" is like "stripped" but also maps uppercase letters to lowercase letters. This is compatible to default GNU/Linux mount behavior.
Mode "uppercase" is like "stripped" but maps lowercase letters to uppercase, if any occur despite the prescriptions of ECMA−119.

−joliet_map "stripped"|"unmapped"
Choose the conversion of file names when a session gets loaded from a Joliet tree.
Mode "stripped" is the default. It removes trailing ";1" or ".;1" if present.
Mode "unmapped" shows names as found without removing characters. Warning: Multi−session converts "xyz;1" to "xyz_1" and maybe adds new ";1".

−iso_nowtime "dynamic"|timestring
Choose whether to use the current time ("dynamic") or a fixed time point for timestamps of ISO 9660 nodes without a disk source file and as default for superblock timestamps.
If a timestring is given, then it is used for such timestamps. For the formats of timestrings see command **−alter_date**.

−disk_dev_ino "on"|"ino_only"|"off"
Enable or disable processing of recorded file identification numbers (dev_t and ino_t). If enabled they are stored as xattr and can substantially accelerate file comparison. The root node gets a global start timestamp. If during comparison a file with younger timestamps is found in the ISO image, then it is suspected to have inconsistent content.
If device numbers and inode numbers of the disk filesystems are persistent and if no irregular alterations of timestamps or system clock happen, then potential content changes can be detected without reading that content. File content change is assumed if any of mtime, ctime, device number or inode number have changed.
Mode "ino_only" replaces the precondition that device numbers are stable by the precondition that mount points in the compared tree always lead to the same filesystems. Use this if mode "on"

always sees all files changed.
The speed advantage appears only if the loaded session was produced with −disk_dev_ino "on" too.
Note that −disk_dev_ino "off" is totally in effect only if −hardlinks is "off", too.

−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 ’_’.
iso_rr_paths with the long components will still be able to access the file paths with truncated components.
If −file_name_limit is executed while an ISO tree is present, the file names in the ISO tree get checked for existing truncated file names of the current limit and for name collisions between newly truncated files and existing files. In both cases, the setting will be refused with a SORRY event.
One may lift this ban by prepending the character "+" to the argument of −file_name_limit. Truncated filenames may then get truncated again, invalidating their MD5 part. Colliding truncated names are made unique, consuming at least 9 more bytes of the remaining name part.
If writing of xattr is enabled, then the length will be stored in "isofs.nt" of the root directory. If reading of xattr is enabled and "isofs.nt" is found, then the found length will get into effect if it is smaller than the current setting of −file_name_limit.
File name patterns will only work if they match the truncated name. This might change in future.
Files with truncated names get deleted and re−added unconditionally during −update and −update_r. This might change in future.
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. Else just avoid names longer than 253 characters.

−rom_toc_scan "on"|"force"|"off"[:"emul_off"][:"emul_wide"]
Read−only drives do not tell the actual media type but show any media as ROM (e.g. as DVD−ROM). The session history of MMC multi−session media might be truncated to first and last session or even be completely false. (The emulated history of overwritable media is not affected by this.)
To have in case of failure a chance of getting the session history and especially the address of the last session, there is a scan for ISO 9660 filesystem headers which might help but also might yield worse results than the drive’s table of content. At its end it can cause read attempts to invalid addresses and thus ugly drive behavior. Setting "on" enables that scan for alleged read−only media.
Some operating systems are not able to mount the most recent session of multi−session DVD or BD. If on such a system **xorriso** has no own MMC capabilities then it may still find that session from a scanned table of content. Setting "force" handles any media like a ROM medium with setting "on".
On the other hand the emulation of session history on overwritable media can hamper reading of partly damaged media. Setting "off:emul_off" disables the elsewise trustworthy table−of−content scan for those media.
The table−of−content scan on overwritable media normally searches only up to the end of the session that is pointed to by the superblock at block 0. Setting "on:emul_wide" lets the scan continue up to the end of the medium. This may be useful after copying a medium with −check_media patch_lba0=on when not the last session was loaded.

−calm_drive "in"|"out"|"all"|"revoke"|"on"|"off"
Reduce drive noise until it is actually used again. Some drives stay alert for substantial time after they have been used for reading. This reduces the startup time for the next drive operation but can be loud and waste energy if no i/o with the drive is expected to happen soon.
Modes "in", "out", "all" immediately calm down −indev, −outdev, or both, respectively. Mode

"revoke" immediately alerts both. Mode "on" causes −calm_drive to be performed automatically after each −dev, −indev, and −outdev. Mode "off" disables this.

−ban_stdio_write

Allow for writing only the usage of MMC optical drives. Disallow to write the result into files of nearly arbitrary type. Once set, this command cannot be revoked.

−early_stdio_test "on"|"appendable_wo"|"off"

If enabled by "on" then regular files and block devices get tested for effective access permissions. This implies to try opening those files for writing, which otherwise will happen only later and only if actual writing is desired.
The test result is used for classifying the pseudo drives as overwritable, read−only, write−only, or uselessly empty. This may lead to earlier detection of severe problems, and may avoid some less severe error events.
Mode "appendable_wo" is like "on" with the additional property that non−empty write−only files are regarded as appendable rather than blank.

−data_cache_size number_of_tiles blocks_per_tile

Set the size and granularity of the data cache which is used when ISO images are loaded and when file content is read from ISO images. The cache consists of several tiles, which each consists of several blocks. A larger cache reduces the need for tiles being read multiple times. Larger tiles might additionally improve the data throughput from the drive, but can be wasteful if the data are scattered over the medium.
Larger cache sizes help best with image loading from MMC drives. They are an inferior alternative to −osirrox option "sort_lba_on".
blocks_per_tile must be a power of 2. E.g. 16, 32, or 64. The overall cache size must not exceed 1 GiB. The default values can be restored by parameter "default" instead of one or both of the numbers. Currently the default is 32 tiles of 32 blocks = 2 MiB.

Inserting files into ISO image:

The following commands expect file addresses of two kinds:
disk_path is a path to an object in the local filesystem tree.
iso_rr_path is the Rock Ridge name of a file object in the ISO image. If no Rock Ridge information is recorded in the loaded ISO image, then you will see ISO 9660 names which are of limited length and character set. If no Rock Ridge information shall be stored in an emerging ISO image, then their names will get mapped to such restricted ISO 9660 (aka ECMA−119) names.

Note that in the ISO image you are as powerful as the superuser. Access permissions of the existing files in the image do not apply to your write operations. They are intended to be in effect with the read−only mounted image.

If the iso_rr_path of a newly inserted file leads to an existing file object in the ISO image, then the following collision handling happens:
If both objects are directories then they get merged by recursively inserting the subobjects from filesystem into ISO image. If other file types collide then the setting of command **−overwrite** decides.
Renaming of files has similar collision handling, but directories can only be replaced, not merged. Note that if the target directory exists, then −mv inserts the source objects into this directory rather than attempting to replace it. Command −move, on the other hand, would attempt to replace it.

The commands in this section alter the ISO image and not the local filesystem.

−disk_pattern "on"|"ls"|"off"

Set the pattern expansion mode for the disk_path parameters of several commands which support this feature.
Setting "off" disables this feature for all commands which are marked in this man page by "disk_path [***]" or "disk_pattern [***]".
Setting "on" enables it for all those commands.
Setting "ls" enables it only for those which are marked by "disk_pattern [***]".
Default is "ls".

−add pathspec [...] | disk_path [***]
: Insert the given files or directory trees from filesystem into the ISO image.
If −pathspecs is set to "on" or "as_mkisofs" then pattern expansion is always disabled and character ’=’ has a special meaning. It separates the ISO image path from the disk path:
iso_rr_path=disk_path
Character ’=’ in the iso_rr_path must be escaped by ’\’ (i.e. as "\=").
With −pathspecs "on", the character ’\’ must not be escaped. The character ’=’ in the disk_path must not be escaped.
With −pathspecs "as_mkisofs", all characters ’\’ must be escaped in both, iso_rr_path and disk_path. The character ’=’ may or may not be escaped in the disk_path.
If iso_rr_path does not begin with ’/’ then −cd is prepended. If disk_path does not begin with ’/’ then −cdx is prepended.
If no ’=’ is given then the word is used as both, iso_rr_path and disk path. If in this case the word does not begin with ’/’ then −cdx is prepended to the disk_path and −cd is prepended to the iso_rr_path.
If −pathspecs is set to "off" then −disk_pattern expansion applies, if enabled. The resulting words are used as both, iso_rr_path and disk path. Relative path words get prepended the setting of −cdx to disk_path and the setting of −cd to iso_rr_path.

−add_plainly mode
: If set to mode "unknown" then any command word that does not begin with "−" and is not recognized as known command will be subject to a virtual −add command. I.e. it will be used as pathspec or as disk_path and added to the image. If enabled, −disk_pattern expansion applies to disk_paths.
Mode "dashed" is similar to "unknown" but also adds unrecognized command words even if they begin with "−".
Mode "any" announces that all further words are to be added as pathspecs or disk_paths. This does not work in dialog mode.
Mode "none" is the default. It prevents any words from being understood as files to add, if they are not parameters to appropriate commands.

−path_list disk_path
: Like −add but read the parameter words from file disk_path or standard input if disk_path is "−". The list must contain exactly one pathspec or disk_path pattern per line.

−quoted_path_list disk_path
: Like −path_list but with quoted input reading rules. Lines get split into parameter words for −add. Whitespace outside quotes is discarded.

−map disk_path iso_rr_path
: Insert file object disk_path into the ISO image as iso_rr_path. If disk_path is a directory then its whole sub tree is inserted into the ISO image.

−map_single disk_path iso_rr_path
: Like −map, but if disk_path is a directory then its sub tree is not inserted.

−map_l disk_prefix iso_rr_prefix disk_path [***]
: Perform −map with each of the disk_path parameters. iso_rr_path will be composed from disk_path by replacing disk_prefix by iso_rr_prefix.

−update disk_path iso_rr_path
: Compare file object disk_path with file object iso_rr_path. If they do not match, then perform the necessary image manipulations to make iso_rr_path a matching copy of disk_path. By default this comparison will imply lengthy content reading before a decision is made. Commands −disk_dev_ino or −md5 may accelerate comparison if they were already in effect when the loaded session was recorded.
If disk_path is a directory and iso_rr_path does not exist yet, then the whole subtree will be inserted. Else only directory attributes will be updated.

−update_r disk_path iso_rr_path

Like −update but working recursively. I.e. all file objects below both addresses get compared whether they have counterparts below the other address and whether both counterparts match. If there is a mismatch then the necessary update manipulation is done.
Note that the comparison result may depend on command −follow. Its setting should always be the same as with the first adding of disk_path as iso_rr_path.
If iso_rr_path does not exist yet, then it gets added. If disk_path does not exist, then iso_rr_path gets deleted.

−update_l disk_prefix iso_rr_prefix disk_path [***]

Perform −update_r with each of the disk_path parameters. iso_rr_path will be composed from disk_path by replacing disk_prefix by iso_rr_prefix.

−update_li iso_rr_prefix disk_prefix iso_rr_path [***]

Perform −update_r with each of the iso_rr_path parameters. disk_path will be composed from iso_rr_path by replacing iso_rr_prefix by disk_prefix.

−update_lxi disk_prefix iso_rr_prefix disk_path [***]

Perform −update_r with each of the disk_path parameters and with iso_rr_paths in the ISO filesystem which are derived from the disk_path parameters after exchanging disk_prefix by iso_rr_prefix. So, other than −update_l, this detects missing matches of disk_path and deletes the corresponding iso_rr_path.
Note that relative disk_paths and disk_path patterns are interpreted as sub paths of the current disk working directory −cdx. The corresponding iso_rr_paths are derived by exchanging disk_prefix by iso_rr_prefix before pattern expansion happens. The current −cdi directory has no influence.

−cut_out disk_path byte_offset byte_count iso_rr_path

Map a byte interval of a regular disk file into a regular file in the ISO image. This may be necessary if the disk file is larger than a single medium, or if it exceeds the traditional limit of 2 GiB − 1 for old operating systems, or the limit of 4 GiB − 1 for newer ones. Only the newest Linux kernels seem to read properly files >= 4 GiB − 1.
A clumsy remedy for this limit is to backup file pieces and to concatenate them at restore time. A well tested chopping size is 2047m. It is permissible to request a higher byte_count than available. The resulting file will be truncated to the correct size of a final piece. To request a byte_offset higher than available yields no file in the ISO image but a SORRY event. E.g:

```
 −cut_out /my/disk/file 0 2047m \
 /file/part_1_of_3_at_0_with_2047m_of_5753194821 \
 −cut_out /my/disk/file 2047m 2047m \
 /file/part_2_of_3_at_2047m_with_2047m_of_5753194821 \
 −cut_out /my/disk/file 4094m 2047m \
 /file/part_3_of_3_at_4094m_with_2047m_of_5753194821
```

While command −split_size is set larger than 0, and if all pieces of a file reside in the same ISO directory with no other files, and if the names look like above, then their ISO directory will be recognized and handled like a regular file. This affects commands −compare*, −update*, and overwrite situations. See command −split_size for details.

−cpr disk_path [***] iso_rr_path

Insert the given files or directory trees from filesystem into the ISO image.
The rules for generating the ISO addresses are similar as with shell command cp −r. Nevertheless, directories of the iso_rr_path are created if necessary. Especially a not yet existing iso_rr_path will be handled as directory if multiple disk_paths are present. The leafnames of the multiple disk_paths will be grafted under that directory as would be done with an existing directory.
If a single disk_path is present then a non−existing iso_rr_path will get the same type as the disk_path.
If a disk_path does not begin with '/' then −cdx is prepended. If the iso_rr_path does not begin with '/' then −cd is prepended.

−mkdir iso_rr_path [...]

Create empty directories if they do not exist yet. Existence as directory generates a WARNING event, existence as other file causes a FAILURE event.

−lns target_text iso_rr_path

Create a symbolic link with address iso_rr_path which points to target_text. iso_rr_path may not exist yet.
Hint: Command −clone produces the ISO equivalent of a hard link.

−clone iso_rr_path_original iso_rr_path_copy

Create a copy of the ISO file object iso_rr_path_original with the new address iso_rr_path_copy. If the original is a directory then copy all files and directories underneath. If iso_rr_path_original is a boot catalog file, then it gets not copied but is silently ignored.
The copied ISO file objects have the same attributes. Copied data files refer to the same content source as their originals. The copies may then be manipulated independendly of their originals.
This command will refuse execution if the address iso_rr_path_copy already exists in the ISO tree.

−cp_clone iso_rr_path_original [***] iso_rr_path_dest

Create copies of one or more ISO file objects as with command −clone. In case of collision merge directories with existing ones, but do not overwrite existing ISO file objects.
The rules for generating the copy addresses are the same as with command −cpr (see above) or shell command cp −r. Other than with −cpr, relative iso_rr_path_original will get prepended the −cd path and not the −cdx path. Consider to −mkdir iso_rr_path_dest before −cp_clone so the copy address does not depend on the number of iso_rr_path_original parameters.

Settings for file insertion:

−file_size_limit value [value [...]] --

Set the maximum permissible size for a single data file. The values get summed up for the actual limit. If the only value is "off" then the file size is not limited by **xorriso**. Default is a limit of 100 extents, 4g −2k each:
 −file_size_limit 400g −200k −−
When mounting ISO 9660 filesystems, old operating systems can handle only files up to 2g −1 −−. Newer ones are good up to 4g −1 −−. You need quite a new Linux kernel to read correctly the final bytes of a file >= 4g if its size is not aligned to 2048 byte blocks.
xorriso's own data read capabilities are not affected by operating system size limits. Such limits apply to mounting only. Nevertheless, the target filesystem of an −extract must be able to take the file size.

−not_mgt code[:code[...]]

Control the behavior of the exclusion lists.
Exclusion processing happens before disk_paths get mapped to the ISO image and before disk files get compared with image files. The absolute disk path of the source is matched against the −not_paths list. The leafname of the disk path is matched against the patterns in the −not_leaf list. If a match is detected then the disk path will not be regarded as an existing file and not be added to the ISO image.
Several codes are defined. The _on/_off settings persist until they are revoked by their_off/_on counterparts.
"erase" empties the lists which were accumulated by −not_paths and −not_leaf.
"reset" is like "erase" but also re−installs default behavior.
"off" disables exclusion processing temporarily without invalidating the lists and settings.
"on" re−enables exclusion processing.
"param_off" applies exclusion processing only to paths below disk_path parameter of commands. I.e. explicitly given disk_paths are exempted from exclusion processing.
"param_on" applies exclusion processing to command parameters as well as to files below such parameters.
"subtree_off" with "param_on" excludes parameter paths only if they match a −not_paths item exactly.

"subtree_on" additionally excludes parameter paths which lead to a file address below any −not_paths item.
"ignore_off" treats excluded disk files as if they were missing. I.e. they get reported with −compare and deleted from the image with −update.
"ignore_on" keeps excluded files out of −compare or −update activities.

−not_paths disk_path [***]
Add the given paths to the list of excluded absolute disk paths. If a given path is relative, then the current −cdx is prepended to form an absolute path. Pattern matching, if enabled, happens at definition time and not when exclusion checks are made.
(Do not forget to end the list of disk_paths by "−−")

−not_leaf pattern
Add a single shell parser style pattern to the list of exclusions for disk leafnames. These patterns are evaluated when the exclusion checks are made.

−not_list disk_path
Read lines from disk_path and use each of them either as −not_paths parameter, if they contain a / character, or as −not_leaf pattern.

−quoted_not_list disk_path
Like −not_list but with quoted input reading rules. Each word is handled as one parameter for −not_paths or −not_leaf.

−follow occasion[:occasion[...]]
Enable or disable resolution of symbolic links and mountpoints under disk_paths. This applies to actions −add, −du*x, −ls*x, −findx, −concat, and to −disk_pattern expansion.
There are three kinds of follow decisison to be made:
link is the hop from a symbolic link to its target file object for the purpose of reading. I.e. not for command −concat. If enabled then symbolic links are handled as their target file objects, else symbolic links are handled as themselves.
mount is the hop from one filesystem to another subordinate filesystem. If enabled then mountpoint directories are handled as any other directory, else mountpoints are handled as empty directories if they are encountered in directory tree traversals.
concat is the hop from a symbolic link to its target file object for the purpose of writing. I.e. for command −concat. This is a security risk !
Less general than above occasions:
pattern is mount and link hopping, but only during −disk_pattern expansion.
param is link hopping for parameter words (after eventual pattern expansion). If enabled then −ls*x will show the link targets rather than the links themselves. −du*x, −findx, and −add will process the link targets but not follow links in an eventual directory tree below the targets (unless "link" is enabled).
Occasions can be combined in a colon separated list. All occasions mentioned in the list will then lead to a positive follow decision.
off prevents any positive follow decision. Use it if no other occasion applies.
Shortcuts:
default is equivalent to "pattern:mount:limit=100".
on always decides positive. Equivalent to "link:mount:concat".

Not an occasion but an optional setting is:
limit=<number> which sets the maximum number of link hops. A link hop consists of a sequence of symbolic links and a final target of different type. Nevertheless those hops can loop. Example:

```
  $ ln −s .. uploop
```

Link hopping has a built−in loop detection which stops hopping at the first repetition of a link target. Then the repeated link is handled as itself and not as its target. Regrettably one can construct link networks which cause exponential workload before their loops get detected. The number given with "limit=" can curb this workload at the risk of truncating an intentional sequence of link hops.

−pathspecs "on"|"off"|"as_mkisofs"
Control parameter interpretation with **xorriso** actions −add and −path_list.
Mode "as_mkisofs" enables pathspecs of the form
iso_rr_path=disk_path
like with program mkisofs −graft−points.
All characters '\' must be escaped in both, iso_rr_path and disk_path. The character '=' must be escaped in the iso_rr_path and may or may not be escaped in the disk_path. This mode temporarily disables −disk_pattern expansion for command −add.
Mode "on" does nearly the same. But '=' must only be escaped in the iso_rr_path and '\' must not be escaped at all. This has the disadvantage that one cannot express an iso_rr_path which ends by '\'.
Mode "off" disables pathspecs of the form target=source and re−enables −disk_pattern expansion.

−overwrite "on"|"nondir"|"off"
Allow or disallow overwriting of existing files in the ISO image by files with the same name.
With setting "off", name collisions with at least one non−directory file cause FAILURE events. Collisions of two directories lead to merging of their file lists.
With setting "nondir", only directories are protected by such events, other existing file types get treated with −rm before the new file gets added. Setting "on" enables automatic −rm_r. I.e. a non−directory can replace an existing directory and all its subordinates.
If restoring of files is enabled, then the overwrite rule applies to the target file objects on disk as well, but "on" is downgraded to "nondir".

−split_size number["k"|"m"]
Set the threshold for automatic splitting of regular files. Such splitting maps a large disk file onto a ISO directory with several part files in it. This is necessary if the size of the disk file exceeds −file_size_limit. Older operating systems can handle files in mounted ISO 9660 filesystems only if they are smaller than 2 GiB or in other cases 4 GiB.
Default is 0 which will exclude files larger than −file_size_limit by a FAILURE event. A well tested −split_size is 2047m. Sizes above −file_size_limit are not permissible.
While command −split_size is set larger than 0 such a directory with split file pieces will be recognized and handled like a regular file by commands −compare* , −update*, and in overwrite situations. There are −osirrox parameters "concat_split_on" and "concat_split_off" which control the handling when files get restored to disk.
In order to be recognizable, the names of the part files have to describe the splitting by 5 numbers:
 part_number,total_parts,byte_offset,byte_count,disk_file_size
which are embedded in the following text form:
 part_#_of_#_at_#_with_#_of_#
Scaling characters like "m" or "k" are taken into respect. All digits are interpreted as decimal, even if leading zeros are present.
E.g: /file/part_1_of_3_at_0_with_2047m_of_5753194821
No other files are allowed in the directory. All parts have to be present and their numbers have to be plausible. E.g. byte_count must be valid as −cut_out parameter and their contents may not overlap.

File manipulations:

The following commands manipulate files in the ISO image, regardless whether they stem from the loaded image or were newly inserted.

−iso_rr_pattern "on"|"ls"|"off"
Set the pattern expansion mode for the iso_rr_path parameters of several commands which support this feature.
Setting "off" disables pattern expansion for all commands which are marked in this man page by "iso_rr_path [***]" or "iso_rr_pattern [***]".
Setting "on" enables it for all those commands.

Setting "ls" enables it only for those which are marked by "iso_rr_pattern [***]".
Default is "on".

−rm iso_rr_path [***]
Delete the given files from the ISO image.
Note: This does not free any space on the −indev medium, even if the deletion is committed to that same medium.
The image size will shrink if the image is written to a different medium in modification mode.

−rm_r iso_rr_path [***]
Delete the given files or directory trees from the ISO image. See also the note with command −rm.

−rmdir iso_rr_path [***]
Delete empty directories.

−move iso_rr_path iso_rr_path
Rename the file given by the first (origin) iso_rr_path to the second (destination) iso_rr_path. Deviate from rules of shell command mv by not moving the origin file underneath an existing destination directory. The origin file will rather replace such a directory, if this is allowed by command −overwrite.

−mv iso_rr_path [***] iso_rr_path
Rename the given file objects in the ISO tree to the last parameter in the list. Use the same rules as with shell command mv.
If pattern expansion is enabled and if the last parameter contains wildcard characters then it must match exactly one existing file address, or else the command fails with a FAILURE event.

−chown uid iso_rr_path [***]
Set ownership of file objects in the ISO image. uid may either be a decimal number or the name of a user known to the operating system.

−chown_r uid iso_rr_path [***]
Like −chown but affecting all files below eventual directories.

−chgrp gid iso_rr_path [***]
Set group attribute of file objects in the ISO image. gid may either be a decimal number or the name of a group known to the operating system.

−chgrp_r gid iso_rr_path [***]
Like −chgrp but affecting all files below eventual directories.

−chmod mode iso_rr_path [***]
Equivalent to shell command chmod in the ISO image. mode is either an octal number beginning with "0" or a comma separated list of statements of the form [ugoa]*[+−=][rwxst]* .
Like: go−rwx,u+rwx .
Personalities: u=user, g=group, o=others, a=all
Operators: + adds given permissions, − revokes given permissions, = revokes all old permissions and then adds the given ones.
Permissions: r=read, w=write, x=execute|inspect, s=setuid|setgid, t=sticky bit
For octal numbers see man 2 stat.

−chmod_r mode iso_rr_path [***]
Like −chmod but affecting all files below eventual directories.

−setfacl acl_text iso_rr_path [***]
Attach the given ACL to the given iso_rr_paths. If the files already have ACLs, then those get deleted before the new ones get into effect. If acl_text is empty, or contains the text "clear" or the text "−−remove−all", then the existing ACLs will be removed and no new ones will be attached. Any other content of acl_text will be interpreted as a list of ACL entries. It may be in the long multi−line format as put out by −getfacl but may also be abbreviated as follows:
ACL entries are separated by comma or newline. If an entry is empty text or begins with "#" then

it will be ignored. A valid entry has to begin by a letter out of {ugom} for "user", "group", "other", "mask". It has to contain two colons ":". A non−empty text between those ":" gives a user id or group id. After the second ":" there may be letters out of {rwx− #}. The first three give read, write, or execute permission. Letters "−", " " and TAB are ignored. "#" causes the rest of the entry to be ignored. Letter "X" or any other letters are not supported. Examples:

```
 g:toolies:rw,u:lisa:rw,u:1001:rw,u::wr,g::r,o::r,m::rw
 group:toolies:rw−,user::rw−,group::r−−,other::r−−,mask::rw−
```

A valid entry may be prefixed by "d", some following characters and ":". This indicates that the entry goes to the "default" ACL rather than to the "access" ACL. Example:

```
 u::rwx,g::rx,o::,d:u::rwx,d:g::rx,d:o::,d:u:lisa:rwx,d:m::rwx
```

−setfacl_r acl_text iso_rr_path [***]

Like −setfacl but affecting all files below eventual directories.

−setfacl_list disk_path

Read the output of −getfacl_r or shell command getfacl −R and apply it to the iso_rr_paths as given in lines beginning with "# file:". This will change ownership, group and ACL of the given files. If disk_path is "−" then lines are read from standard input. Line "@" ends the list, "@@@" aborts without changing the pending iso_rr_path.

Since −getfacl and getfacl −R strip leading "/" from file paths, the setting of −cd does always matter.

−setfattr [-]name value iso_rr_path [***]

Attach the given xattr pair of name and value to the given iso_rr_paths. If the given name is prefixed by "−", then the pair with that name gets removed from the xattr list. If name is "−−remove−all" then all user namespace xattr of the given iso_rr_paths get deleted. In case of deletion, value must be an empty text.

Which names are permissible depends on the setting of command −xattr. "on" or "user" restricts them to namespace "user". I.e. a name has to look like "user.x" or "user.whatever".

−xattr setting "any" enables names from all namespaces except "isofs".

Values and names undergo the normal input processing of **xorriso**. See also command −backslash_codes. Other than with command −setfattr_list, the byte value 0 cannot be expressed via −setfattr.

−setfattr_r [-]name value iso_rr_path [***]

Like −setfattr but affecting all files below eventual directories.

−setfattr_list disk_path

Read the output format of −getfattr_r or shell command getfattr −Rd and apply it to the iso_rr_paths as given in lines beginning with "# file:". All previously existing xattr of the acceptable namespaces will be deleted before the new xattr get attached. The set of acceptable names depends on the setting of command −xattr.

If disk_path is "−" then lines are read from standard input.

Since −getfattr and getfattr −Rd strip leading "/" from file paths, the setting of −cd does always matter.

Empty input lines and lines which begin by "#" will be ignored (except "# file:"). Line "@" ends the list, "@@@" aborts without changing the pending iso_rr_path. Other input lines must have the form

```
 name="value"
```

The separator "=" is not allowed in names. Value may contain any kind of bytes. It must be in quotes. Trailing whitespace after the end quote will be ignored. Non−printables bytes and quotes must be represented as \XYZ by their octal 8−bit code XYZ. Use code \000 for 0−bytes.

−alter_date type timestring iso_rr_path [***]

Alter the date entries of files in the ISO image. type may be one of the following:

"a" sets access time, updates ctime.

"m" sets modification time, updates ctime.

"b" sets access time and modification time, updates ctime.

"a−c", "m−c", and "b−c" set the times without updating ctime.
"c" sets the ctime.
timestring may be in the following formats (see also section EXAMPLES):
As expected by program date:
 MMDDhhmm[[CC]YY][.ss]]
As produced by program date:
 [Day] MMM DD hh:mm:ss [TZON] YYYY
Relative times counted from current clock time:
 +|−Number["s"|"h"|"d"|"w"|"m"|"y"]
where "s" means seconds, "h" hours, "d" days, "w" weeks, "m"=30d, "y"=365.25d plus 1d added to multiplication result.
Absolute seconds counted from Jan 1 1970:
 =Number
xorriso's own timestamps:
 YYYY.MM.DD[.hh[mm[ss]]]
scdbackup timestamps:
 YYMMDD[.hhmm[ss]]
where "A0" is year 2000, "B0" is 2010, etc.
ECMA−119 volume timestamps:
 YYYYMMDDhhmmsscc
These are normally given as GMT. The suffix "LOC" causes local timezone conversion. E.g. 2013010720574700, 2013010720574700LOC. The last two digits cc (centiseconds) will be ignored, but must be present in order to make the format recognizable.
Example:
  −alter_date m−c 2013.11.27.103951 /file1 /file2 −−
This command does not persistently apply to the boot catalog, which gets fresh timestamps at −commit time. Command −volume_date "uuid" can set this time value.

−alter_date_r type timestring iso_rr_path [***]
Like −alter_date but affecting all files below eventual directories.

−hide hide_state iso_rr_path [***]
Prevent the names of the given files from showing up in the directory trees of ISO 9660 and/or Joliet and/or HFS+ when the image gets written. 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.
Warning: Data which are hidden from the ISO 9660 tree will not be copied by the write method of modifying.
Possible values of hide_state are: "iso_rr" for hiding from ISO 9660 tree, "joliet" for Joliet tree, "hfsplus" for HFS+, "on" for them all. "off" means visibility in all directory trees.
These values may be combined. E.g.: joliet:hfsplus
This command does not apply to the boot catalog. Rather use: −boot_image "any" "cat_hidden=on"

Tree traversal command -find:

−find iso_rr_path [test [op] [test ...]] [-exec action [params]] --
A restricted substitute for shell command find in the ISO image. It performs an action on matching file objects at or below iso_rr_path.
If not used as last command in the line then the parameter list needs to get terminated by "−−".
Tests are optional. If they are omitted then action is applied to all file objects. If tests are given then they form together an expression. The action is applied only if the expression matches the file object. Default expression operator between tests is −and, i.e. the expression matches only if all its tests match.
Available tests are:
−name pattern : Matches if pattern matches the file leaf name. If the pattern does not contain any of the characters "*?[", then it will be truncated according to −file_name_limit and thus match the

truncated name in the ISO filesystem.
−wholename pattern : Matches if pattern matches the file path as it would be printed by action "echo". Character ’/’ can be matched by wildcards. If pattern pieces between ’/’ do not contain any of the characters "*?[", they will be truncated according to −file_name_limit.
−disk_name pattern : Like −name but testing the leaf name of the file source on disk. Can match only data files which do not stem from the loaded image, or for directories above such data files. With directories the result can change between −find runs if their content stems from multiple sources.
−disk_path disk_path : Matches if the given disk_path is equal to the path of the file source on disk. The same restrictions apply as with −disk_name.
−type type_letter : Matches files of the given type: "block", "char", "dir", "pipe", "file", "link", "socket", "eltorito", and "Xotic" which matches what is not matched by the other types.
Only the first letter is interpreted. E.g.: −find / −type d
−maxdepth number : Matches only files which are at most at the given depth level relative to the iso_rr_path where −find starts. That path itself is at depth 0, its directory children are at 1, their directory children at 2, and so on.
−mindepth number : Matches only files which are at least at the given depth level.
−damaged : Matches files which use data blocks marked as damaged by a previous run of −check_media. The damage info vanishes when a new ISO image gets loaded.
Note that a MD5 session mismatch marks all files of the session as damaged. If finer distinction is desired, perform −md5 off before −check_media.
−pending_data : Matches files which get their content from outside the loaded ISO image.
−lba_range start_lba block_count : Matches files which use data blocks within the range of start_lba and start_lba+block_count−1.
−has_acl : Matches files which have a non−trivial ACL.
−has_xattr : Matches files which have xattr name−value pairs from user namespace.
−has_aaip : Matches files which have ACL or any xattr.
−has_any_xattr : Matches files which have any xattr other than ACL.
−has_md5 : Matches data files which have MD5 checksums.
−has_hfs_crtp creator type : Matches files which have the given HFS+ creator and type attached. These are codes of 4 characters which get stored if −hfsplus is enabled. Use a single dash ’−’ as wildcard that matches any such code. E.g:.
```
 −has_hfs_crtp YYDN TEXT
 −has_hfs_crtp − −
```
−has_hfs_bless blessing : Matches files which bear the given HFS+ blessing. It may be one of : "ppc_bootdir", "intel_bootfile", "show_folder", "os9_folder", "osx_folder", "any". See also action set_hfs_bless.
−has_filter : Matches files which are filtered by −set_filter.
−hidden hide_state : Matches files which are hidden in "iso_rr" tree, in "joliet" tree, in "hfsplus" tree, in all trees ("on"), or not hidden in any tree ("off").
Those which are hidden in some tree match −not −hidden "off".
−bad_outname namespace : Matches files with names which change when converted forth and back between the local character set and one of the namespaces "rockridge", "joliet", "ecma119", "hfsplus".
All applicable −compliance rules are taken into respect. Rule "omit_version" is always enabled, because else namespaces "joliet" and "ecma119" would cause changes with every non−directory name. Consider to also enable rules "no_force_dots" and "no_j_force_dots".
The namespaces use different character sets and apply further restrictions to name length, permissible characters, and mandatory name components. "rockridge" uses the character set defined by −out_charset, "joliet" uses UCS−2BE, "ecma119" uses ASCII, "hfsplus" uses UTF−16BE.
−name_limit_blocker length : Matches file names which would prevent command −file_name_limit with the given length. The command itself reports only the first problem file.
−prune : If this test is reached and the tested file is a directory then −find will not dive into that

directory. This test itself does always match.
−use_pattern "on"|"off" : This pseudo test controls the interpretation of wildcards with tests −name, −wholename, and −disk_name. Default is "on". If interpretation is disabled by "off", then the parameters of −name, −wholename, and −disk_name have to match literally rather than as search pattern. This test itself does always match.
−or_use_pattern "on"|"off" : Like −use_pattern, but automatically appending the test by −or rather than by −and. Further the test itself does never match. So a subsequent test −or will cause its other operand to be performed.
−decision "yes"|"no" : If this test is reached then the evaluation ends immediately and action is performed if the decision is "yes" or "true". See operator −if.
−true and **−false** : Always match or match not, respectively. Evaluation goes on.
−sort_lba : Always match. This causes −find to perform its action in a sequence sorted by the ISO image block addresses of the files. It may improve throughput with actions which read data from optical drives. Action will always get the absolute path as parameter.
Available operators are:
−not : Matches if the next test or sub expression does not match. Several tests do this specifically: −undamaged, −lba_range with negative start_lba, −has_no_acl, −has_no_xattr, −has_no_aaip, −has_no_filter .
−and : Matches if both neighboring tests or expressions match.
−or : Matches if at least one of both neighboring tests or expressions matches.
−sub ... **−subend** or **(** ... **)** : Enclose a sub expression which gets evaluated first before it is processed by neighboring operators. Normal precedence is: −not, −or , −and.
−if ... **−then** ... **−elseif** ... **−then** ... **−else** ... **−endif** : Enclose one or more sub expressions. If the −if expression matches, then the −then expression is evaluated as the result of the whole expression up to −endif. Else the next −elseif expression is evaluated and if it matches, its −then expression. Finally in case of no match, the −else expression is evaluated. There may be more than one −elseif. Neither −else nor −elseif are mandatory. If −else is missing and would be hit, then the result is a non−match.
−if−expressions are the main use case for above test −decision.

Default action is **echo**, i.e. to print the address of the found file. Other actions are certain **xorriso** commands which get performed on the found files. These commands may have specific parameters. See also their particular descriptions.
chown and **chown_r** change the ownership and get the user id as parameter. E.g.: −exec chown thomas −−
chgrp and **chgrp_r** change the group attribute and get the group id as parameter. E.g.: −exec chgrp_r staff −−
chmod and **chmod_r** change access permissions and get a mode string as parameter. E.g.: −exec chmod a−w,a+r −−
alter_date and **alter_date_r** change the timestamps. They get a type character and a timestring as parameters.
E.g.: −exec alter_date "m" "Dec 30 19:34:12 2007" −−
set_to_mtime sets the ctime and atime to the value found in mtime.
lsdl prints file information like shell command ls −dl.
compare performs command −compare with the found file address as iso_rr_path and the corresponding file address below its parameter disk_path_start. For this the iso_rr_path of the −find command gets replaced by the disk_path_start.
E.g.: −find /thomas −exec compare /home/thomas −−
update performs command −update with the found file address as iso_rr_path. The corresponding file address is determined like with above action "compare".
update_merge is like update but does not delete the found file if it is missing on disk. It may be run several times and records with all visited files whether their counterpart on disk has already been seen by one of the update_merge runs. Finally, a −find run with action "rm_merge" may remove all files that saw no counterpart on disk.

Up to the next "rm_merge" or "clear_merge" all newly inserted files will get marked as having a disk counterpart.
rm removes the found iso_rr_path from the image if it is not a directory with files in it. I.e. this "rm" includes "rmdir".
rm_r removes the found iso_rr_path from the image, including whole directory trees.
rm_merge removes the found iso_rr_path if it was visited by one or more previous actions "update_merge" and saw no counterpart on disk in any of them. The marking from the update actions is removed in any case.
clear_merge removes an eventual marking from action "update_merge".
report_damage classifies files whether they hit a data block that is marked as damaged. The result is printed together with the address of the first damaged byte, the maximum span of damages, file size, and the path of the file.
report_lba prints files which are associated to image data blocks. It tells the logical block address, the block number, the byte size, and the path of each file. There may be reported more than one line per file if the file has more than one section. In this case each line has a different extent number in column "xt".
report_sections like report_lba but telling the byte sizes of the particular sections rather than the overall byte size of the file.
getfacl prints access permissions in ACL text form to the result channel.
setfacl attaches ACLs after removing existing ones. The new ACL is given in text form as defined with command −setfacl.
E.g.: −exec setfacl u:lisa:rw,u::rw,g::r,o::−,m::rw −−
getfattr prints xattr name−value pairs to the result channel. The choice of namespaces depends on the setting of command −xattr: "on" or "user" restricts it to the namespace "user", "any" only omits namespace "isofs".
get_any_xattr prints xattr name−value pairs from any namespace except ACL to the result channel. This is mostly for debugging of namespace "isofs".
list_extattr mode prints a script to the result channel, which would use FreeBSD command setextattr to set the file’s xattr name−value pairs of user namespace. Parameter mode controls the form of the output of names and values. Default mode "e" prints harmless characters in shell quotation marks, but represents texts with octal 001 to 037 and 0177 to 0377 by an embedded echo −e command. Mode "q" prints any characters in shell quotation marks. This might not be terminal−safe but should work in script files. Mode "r" uses no quotation marks. Not safe. Mode "b" prints backslash encoding. Not suitable for shell parsing.
E.g. −exec list_extattr e −−
Command −backslash_codes does not affect the output.
get_md5 prints the MD5 sum, if recorded, together with file path.
check_md5 compares the MD5 sum, if recorded, with the file content and reports if mismatch.
E.g.: −find / −not −pending_data −exec check_md5 FAILURE −−
make_md5 equips a data file with an MD5 sum of its content. Useful to upgrade the files in the loaded image to full MD5 coverage by the next commit with −md5 "on".
E.g.: −find / −type f −not −has_md5 −exec make_md5 −−
setfattr sets or deletes xattr name value pairs.
E.g.: −find / −has_xattr −exec setfattr −−remove−all ’’ −−
set_hfs_crtp adds, changes, or removes HFS+ creator and type attributes.
E.g.: −exec set_hfs_crtp YYDN TEXT
E.g.: −find /my/dir −prune −exec set_hfs_crtp −−delete −
get_hfs_crtp prints the HFS+ creator and type attributes together with the iso_rr_path, if the file has such attributes at all.
E.g.: −exec get_hfs_crtp
set_hfs_bless applies or removes HFS+ blessings. 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. The −find run will end as soon as the first blessing is issued. The previous bearer of the blessing will lose it then. No file object can bear more than one blessing.
E.g.: −find /my/blessed/directory −exec set_hfs_bless p
Further there is blessing "none" or "n" which revokes any blessing from the found files. This −find run will not stop when the first match is reached.
E.g.: −find / −has_hfs_bless any −exec set_hfs_bless none
get_hfs_bless prints the HFS+ blessing role and the iso_rr_path, if the file is blessed at all.
E.g.: −exec get_hfs_bless
set_filter applies or removes filters.
E.g.: −exec set_filter −−zisofs −−
mkisofs_r applies the rules of mkisofs −r to the file object:
user id and group id become 0, all r−permissions get granted, all w denied. If there is any x−permission, then all three x get granted. s− and t−bits get removed.
sort_weight attributes a LBA weight number to regular files.
The 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 boot catalog has a hardcoded weight of 1 billion. Normally it should occupy the block with the lowest possible address.
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.
Data files which are added by other commands get an initial weight of 0. Boot image files have a default weight of 2.
E.g.: −exec sort_weight 3 −−
show_stream shows the content stream chain of a data file.
show_stream_id is like show_stream, but also prints between stream type and first ":" in square brackets libisofs id numbers: [fs_id,dev_id,ino_id].
hide brings the file into one of the hide states "on", "iso_rr", "joliet", "hfsplus", "off". They may be combined. E.g.: joliet:hfsplus
E.g.:
```
 −find / −disk_name *_secret −exec hide on
```
print_outname prints in the first line the filename as registered by the program model, and in the second line the filename after conversion forth and back between local character set and one of the namespaces "rockridge", "joliet", "ecma119", or "hfsplus". The third output line is "−−" .
The name conversion does not take into respect the possibility of name collisions in the target namespace. Such collisions are most likely in "joliet" and "ecma119", where they get resolved by automatic file name changes.
E.g.:
```
 −find / −bad_outname joliet −exec print_outname joliet
```
estimate_size prints a lower and an upper estimation of the number of blocks which the found files together will occupy in the emerging ISO image. This does not account for the superblock, for the directories in the −find path, or for image padding.
find performs another run of −find on the matching file address. It accepts the same params as −find, except iso_rr_path.
E.g.:
```
 −find / −name ’???’ −type d −exec find −name ’[abc]*’ −exec chmod a−w,a+r −−
```

Filters for data file content:

Filters may be installed between data files in the ISO image and their content source outside the image. They may also be used vice versa between data content in the image and target files on disk.
Built−in filters are "−−zisofs" and "−−zisofs−decode". The former is to be applied via −set_filter, the latter is automatically applied if zisofs compressed content is detected with a file when loading the ISO image.
Another built−in filter pair is "−−gzip" and "−−gunzip" with suffix ".gz". They behave about like external

gzip and gunzip but avoid forking a process for each single file. So they are much faster if there are many small files.

−external_filter name option[:option] program_path [arguments] --

Register a content filter by associating a name with a program path, program arguments, and some behavioral options. Once registered it can be applied to multiple data files in the ISO image, regardless whether their content resides in the loaded ISO image or in the local filesystem. External filter processes may produce synthetic file content by reading the original content from stdin and writing to stdout whatever they want. They must deliver the same output on the same input in repeated runs.

Options are:

"default" means that no other option is intended.

"suffix=..." sets a file name suffix. If it is not empty then it will be appended to the file name or removed from it.

"remove_suffix" will remove a file name suffix rather than appending it.

"if_nonempty" will leave 0−sized files unfiltered.

"if_reduction" will try filtering and revoke it if the content size does not shrink.

"if_block_reduction" will revoke if the number of 2 kB blocks does not shrink.

"used=..." is ignored. Command −status shows it with the number of files which currently have the filter applied.

Examples:

```
 −external_filter bzip2 suffix=.bz2:if_block_reduction \
              /usr/bin/bzip2 −−
 −external_filter bunzip2 suffix=.bz2:remove_suffix \
              /usr/bin/bunzip2 −−
```

−unregister_filter name

Remove an −external_filter registration. This is only possible if the filter is not applied to any file in the ISO image.

−close_filter_list

Irrevocably ban commands −concat "pipe", −external_filter, and −unregister_filter, but not −set_filter. Use this to prevent external filtering in general or when all intended filters are registered and −concat mode "pipe" shall be disallowed. External filters may also be banned totally at compile time of **xorriso**. By default they are banned if **xorriso** runs under setuid permission.

−set_filter name iso_rr_path [***]

Apply an −external_filter or a built−in filter to the given data files in the ISO image. If the filter suffix is not empty , then it will be applied to the file name. Renaming only happens if the filter really gets attached and is not revoked by its options. By default files which already bear the suffix will not get filtered. The others will get the suffix appended to their names. If the filter has option "remove_suffix", then the filter will only be applied if the suffix is present and can be removed. Name oversize or collision caused by suffix change will prevent filtering.

With most filter types this command will immediately run the filter once for each file in order to determine the output size. Content reading operations like −extract , −compare and image generation will perform further filter runs and deliver filtered content.

At image generation time the filter output must still be the same as the output from the first run. Filtering for image generation does not happen with files from the loaded ISO image if the write method of growing is in effect (i.e −indev and −outdev are identical).

The reserved filter name "−−remove−all−filters" revokes filtering. This will revoke suffix renamings as well. Use "−−remove−all−filters+" to prevent any suffix renaming.

Attaching or detaching filters will not alter the state of −changes_pending. If the filter manipulations shall be the only changes in a write run, then explicitly execute −changes_pending "yes".

−set_filter_r name iso_rr_path [***]
Like −set_filter but affecting all data files below eventual directories.

Writing the result, drive control:

(see also paragraph about settings below)

−rollback
Discard the manipulated ISO image and reload it from −indev. (Use −rollback_end if immediate program end is desired.)

−changes_pending "no"|"yes"|"mkisofs_printed"|"show_status"
Write runs are performed only if a change of the image has been made since the image was loaded or created blank. Vice versa the program will start a write run for pending changes when it ends normally (i.e. not by abort and not by command −rollback_end).
The command −changes_pending can be used to override the automatically determined state. This is mainly useful for setting state "yes" despite no real changes were made. The sequence −changes_pending "no" −end is equivalent to the command −rollback_end. State "mkisofs_printed" is caused by emulation command −as mkisofs if option −print−size is present.
The pseudo−state "show_status" can be used to print the current state to result channel.
Image loading or manipulations which happen after this command will again update automatically the change status of the image.

−commit
Perform the write operation. Afterwards, if −outdev is readable, make it the new −dev and load the image from there. Switch to growing mode. (A subsequent −outdev will activate modification mode or blind growing.) −commit is performed automatically at end of program if there are uncommitted manipulations pending.
So, to perform a final write operation with no new −dev and no new loading of image, rather execute command −end. If you want to go on without image loading, execute −commit_eject "none". To eject after write without image loading, use −commit_eject "all".
To suppress a final write, execute −rollback_end.

Writing can last quite a while. It is not unnormal with several types of media that there is no progress visible for the first few minutes or that the drive gnaws on the medium for a few minutes after all data have been transmitted. **xorriso** and the dri ves are in a client−server relationship. The drives have much freedom about what to do with the media. Some combinations of drives and media simply do not work, despite the promises by their vendors. If writing fails then try other media or another drive. The reason for such failure is hardly ever in the code of the various burn programs but you may well try some of those listed below under SEE ALSO.

−eject "in"|"out"|"all"
Eject the medium in −indev, −outdev, or both drives, respectively. Note: It is not possible yet to effectively eject disk files.

−commit_eject "in"|"out"|"all"|"none"
Combined −commit and −eject. When writing has finished do not make −outdev the new −dev, and load no ISO image. Rather eject −indev and/or −outdev. Give up any non−ejected drive.

−blank mode
Make media ready for writing from scratch (if not −dummy is activated).
This affects only the −outdev not the −indev. If both drives are the same and if the ISO image was altered then this command leads to a FAILURE event. Defined modes are:
  as_needed, fast, all, deformat, deformat_quickest
"as_needed" cares for used CD−RW, DVD−RW and for used overwritable media by applying −blank "fast". It applies −format "full" to yet unformatted DVD−RAM and BD−RE. Other media in blank state are gracefully ignored. Media which cannot be made ready for writing from scratch cause a FAILURE event.
"fast" makes CD−RW and unformatted DVD−RW re−usable, or invalidates overwritable ISO

images. "all" might work more thoroughly and need more time.
"deformat" converts overwritable DVD−RW into unformatted ones.
"deformat_quickest" is a faster way to deformat or blank DVD−RW but produces media which are only suitable for a single session. Some drives announce this state by not offering feature 21h, but some drives offer it anyway. If feature 21h is missing, then**xorriso** will refuse to write on DVD−RW if not command −close is set to "on".
The progress reports issued by some drives while blanking are quite unrealistic. Do not conclude success or failure from the reported percentages. Blanking was successful if no SORRY event or worse occurred.
Mode may be prepended by "force:" in order to override the evaluation of the medium state by libburn. E.g. "force:fast". Blanking will nevertheless only succeed if the drive is willing to do it.

−format mode

Convert unformatted DVD−RW into overwritable ones, "de−ice" DVD+RW, format newly purchased BD−RE or BD−R, re−format DVD−RAM or BD−RE.
Defined modes are:
  as_needed, full, fast, by_index_<num>, fast_by_index_<num>,
  by_size_<num>, fast_by_size_<num>, without_spare
"as_needed" formats yet unformatted DVD−RW, DVD−RAM, BD−RE, or blank unformatted BD−R. Other media are left untouched.
"full" (re−)formats DVD−RW, DVD+RW, DVD−RAM, BD−RE, or blank unformatted BD−R.
"fast" does the same as "full" but tries to be quicker.
"by_index_" selects a format out of the descriptor list issued by command −list_formats. The index number from that list is to be appended to the mode word. E.g: "by_index_3".
"fast_by_index_" does the same as "by_index_" but tries to be quicker.
"by_size_" selects a format out of the descriptor list which provides at least the given size. That size is to be appended to the mode word. E.g: "by_size_4100m". This applies to media with Defect Management. On BD−RE it will not choose format 0x31, which offers no Defect Management.
"fast_by_size_" does the same as "by_size_" but tries to be quicker.
"without_spare" selects the largest format out of the descriptor list which provides no Spare Area for Defect Management. On BD−RE this will be format 0x31.
The formatting action has no effect on media if −dummy is activated.
Formatting is normally needed only once during the lifetime of a medium, if ever. But it is a reason for re−formatting if:
 DVD−RW was deformatted by −blank,
 DVD+RW has read failures (re−format before next write),
 DVD−RAM or BD−RE shall change their amount of defect reserve.
BD−R may be written unformatted or may be formatted before first use. Formatting activates Defect Management which tries to catch and repair bad spots on media during the write process at the expense of half speed even with flawless media.
The progress reports issued by some drives while formatting are quite unrealistic. Do not conclude success or failure from the reported percentages. Formatting was successful if no SORRY event or worse occurred. Be patient with apparently frozen progress.

−list_formats

Put out a list of format descriptors as reported by the output drive for the current medium. The list gives the index number after "Format idx", a MMC format code, the announced size in blocks (like "2236704s") and the same size in MiB.
MMC format codes are manifold. Most important are: "00h" general formatting, "01h" increases reserve space for DVD−RAM, "26h" for DVD+RW, "30h" for BD−RE with reserve space, "31h" for BD−RE without reserve space, "32h" for BD−R.
Smaller format size with DVD−RAM, BD−RE, or BD−R means more reserve space.

−list_speeds

Put out a list of speed values as reported by the drives with the loaded media. The list tells read speeds of the input drive and of the output drive. Further it tells write speeds of the output drive.
The list of write speeds does not necessarily mean that the medium is writable or that these speeds are actually achievable. Especially the lists reported with empty drive or with ROM media obviously advertise speeds for other media.
It is not mandatory to use speed values out of the listed range. The drive is supposed to choose a safe speed that is as near to the desired speed as possible.
At the end of the list, "Write speed L" and "Write speed H" are the best guesses for lower and upper write speed limit. "Write speed l" and "Write speed h" may appear only with CD and eventually override the list of other speed offers.
Only if the drive reports contradicting speed information there will appear "Write speed 0", which tells the outcome of speed selection by command −speed 0, if it deviates from "Write speed H".
"Read speed L" and "Read speed H" tell the minimum and maximum read speeds, as reported by the drive. They would be chosen by −read_speed "min" or "max" if they undercut or surpass the built−in limits. These are "1x", "52xCD", "24xDVD", "20xBD".

−list_profiles "in"|"out"|"all"

Put out a list of media types supported by −indev, −outdev, or both, respectively. The currently recognized type is marked by text "(current)".

−truncate_overwritable entity id adjust

On overwritable medium copy the volume descriptors of an existing session to the overall descriptors at LBA 0 ff. This makes all sessions **inaccessible** which are younger than the activated one. A reason to do this would be read errors in the younger sessions and the wish to re−write or skip them.
This operation is only allowed if no changes to the loaded filesystem are pending. If an −indev is acquired then it is released before the write operation begins and re−acquired only in case of success.
The parameters "entity" and "id" have the same meaning as with command −load. They choose the existing ISO session which shall become the youngest accessible session. Available entity names are "session", "track", "lba", "sbsector", "volid". "auto" makes few sense. id is a number or search text as appropriate for the given entity.
Parameter "adjust" controls the claimed size of the activated session. Text "new" means the size of the newly activated session as it was before this command. I.e. the space of the then inaccessible younger sessions will be re−used when appending more sessions.
"old" means the size up to the end of the previously youngest session. I.e. "old" will not free the space of the then inaccessible younger sessions for re−use.
A number preceded by "+" gives the number of bytes to be added to "new". A number without "+" gives the overall number of bytes. In any case the result may not be smaller than "new". Numbers may have a unit suffix: "d"=512, "k"=1024, "s"=2048, "m"=1024k, "g"=1024m.
Examples:
Activate session 4 and enable overwriting of the blocks of younger sessions:
```
 −truncate_overwritable session 4 new
```
Activate session 4 and claim the blocks of younger sessions as useless part of session 4:
```
 −truncate_overwritable session 4 old
```
Let session 4 claim additional 500 MiB as useless data:
```
 −truncate_overwritable session 4 +500m
```

−close_damaged "as_needed"|"force"

Try to close the upcoming track and session if the drive reported the medium as damaged. This may apply to CD−R, CD−RW, DVD−R, DVD−RW, DVD+R, DVD+R DL, or BD−R media. It is indicated by warning messages when the drive gets acquired, and by a remark "but next track is damaged" with the line "Media status :" of command −toc.
The setting of command −close determines whether the medium stays appendable.
Mode "as_needed" gracefully refuses on media which are not reported as damaged. Mode "force"

attempts the close operation even with media which appear undamaged.
No image changes are allowed to be pending before this command is performed. After closing was attempted, both drives are given up.

Settings for result writing:

Rock Ridge info will be generated by default. ACLs will be written according to the setting of command −acl.

−joliet "on"|"off"
If enabled by "on", generate Joliet tree additional to ISO 9660 + Rock Ridge tree.

−hfsplus "on"|"off"
If enabled by "on", generate a 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 data submitted by −boot_image system_area=. 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.
The HFS+ filesystem is not suitable for add−on sessions produced by the multi−session method of growing. An existing ISO image may nevertheless be the base for a new image produced by the method of modifying. If −hfsplus is enabled when −indev or −dev gets executed, then AAIP attributes get loaded from the input image and checked for information about HFS creator, filetype, or blessing. If found, then they get enabled as settings for the next image production. Therefore it is advisable to perform −hfsplus "on" before −indev or −dev.
Information about HFS creator, type, and blessings gets stored by xorriso if −hfsplus is enabled at −commit time. It is stored as copy outside the HFS+ partition, but rather along with the Rock Ridge information. xorriso does not read any information from the HFS+ meta data.
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.

−rockridge "on"|"off"
Mode "off" disables production of Rock Ridge information for the ISO 9660 file objects. The multi−session capabilities of xorriso depend much on the naming fidelity of Rock Ridge. So it is strongly discouraged to deviate from default setting "on".

−compliance rule[:rule...]
Adjust the compliance to specifications of ISO 9660/ECMA−119 and its contemporary extensions. In some cases it is worth to deviate a bit in order to circumvent bugs of the intended reader system or to get unofficial extra features.
There are several adjustable rules which have a keyword each. If they are mentioned with this command then their rule gets added to the relaxation list. This list can be erased by rules "strict" or "clear". It can be reset to its start setting by "default". All of the following relaxation rules can be revoked individually by appending "_off". Like "deep_paths_off".
Rule keywords are:
"iso_9660_level="number chooses level 1 with ECMA−119 names of the form 8.3 and −file_size_limit <= 4g − 1, or level 2 with ECMA−119 names up to length 32 and the same

−file_size_limit, or level 3 with ECMA−119 names up to length 32 and −file_size_limit >= 400g −200k. If necessary −file_size_limit gets adjusted.
"allow_dir_id_ext" allows ECMA−119 names of directories to have a name extension as with other file types. It does not force dots and it omits the version number, though. This is a bad tradition of mkisofs which violates ECMA−119. Especially ISO level 1 only allows 8 characters in a directory name and not 8.3.
"omit_version" does not add versions (";1") to ECMA−119 and Joliet file names.
"only_iso_version" does not add versions (";1") to Joliet file names.
"deep_paths" allows ECMA−119 file paths deeper than 8 levels.
"long_paths" allows ECMA−119 file paths longer than 255 characters.
"long_names" allows up to 37 characters with ECMA−119 file names.
"no_force_dots" does not add a dot to ECMA−119 file names which have none.
"no_j_force_dots" does not add a dot to Joliet file names which have none.
"lowercase" allows lowercase characters in ECMA−119 file names.
"7bit_ascii" allows nearly all 7−bit characters in ECMA−119 file names. Not allowed are 0x0 and ’/’. If not "lowercase" is enabled, then lowercase letters get converted to uppercase.
"full_ascii" allows all 8−bit characters except 0x0 and ’/’ in ECMA−119 file names.
"untranslated_names" might be dangerous for inadverted reader programs which rely on the restriction to at most 37 characters in ECMA−119 file names. This rule allows ECMA−119 file names up to 96 characters with no character conversion. If a file name has more characters, then image production will fail deliberately.
"untranslated_name_len="number enables untranslated_names with a smaller limit for the length of file names. 0 disables this feature, −1 chooses maximum length limit, numbers larger than 0 give the desired length limit.
"joliet_long_names" allows Joliet leaf names up to 103 characters rather than 64.
"joliet_long_paths" allows Joliet paths longer than 240 characters.
"joliet_utf16" encodes Joliet 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.
"always_gmt" stores timestamps in GMT representation with timezone 0.
"rec_mtime" records with non−RockRidge directory entries the disk file’s mtime and not the creation time of the image. This applies to the ECMA−119 tree (plain ISO 9660), to Joliet, and to ISO 9660:1999. "rec_time" is default. If disabled, it gets automatically re−enabled by −as mkisofs emulation when a pathspec is encountered.
"new_rr" uses Rock Ridge version 1.12 (suitable for GNU/Linux but not for older FreeBSD or for Solaris). This implies "aaip_susp_1_10_off" which may be changed by subsequent "aaip_susp_1_10".
Default is "old_rr" which uses Rock Ridge version 1.10. This implies also "aaip_susp_1_10" which may be changed by subsequent "aaip_susp_1_10_off".
"aaip_susp_1_10" allows AAIP to be written as unofficial extension of RRIP rather than as official extension under SUSP−1.12.
"no_emul_toc" saves 64 kB with the first session on overwritable media but makes the image incapable of displaying its session history.
"iso_9660_1999" causes the production of an additional directory tree compliant to ISO 9660:1999. It can record long filenames for readers which do not understand Rock Ridge.
"old_empty" uses 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.
Default setting is

```
 "clear:only_iso_version:deep_paths:long_paths:no_j_force_dots:
 always_gmt:old_rr".
```

Note: The term "ECMA−119 name" means the plain ISO 9660 names and attributes which get visible if the reader ignores Rock Ridge.

−rr_reloc_dir name

Specify the name of the relocation directory in which deep directory subtrees shall be placed if −compliance is set to "deep_paths_off" or "long_paths_off". A deep directory is one that has a chain of 8 parent directories (including root) above itself, or one that contains a file with an ECMA−119 path of more than 255 characters.
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.
The default relocation directory is the root directory. By giving a non−empty name with −rr_reloc_dir, a directory in the root directory may get this role. If that directory does not already exist at −commit time, 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 and must not be longer than 255 bytes.

−volid text

Specify the volume ID, which most operating systems will consider to be the volume name of the image or medium.
xorriso accepts any text up to 32 characters, but according to rarely obeyed specs stricter rules apply:
ECMA−119 demands 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 the name of the mount point when the medium is inserted into a playful computer system.
If an ISO image gets loaded while the volume ID is set to default "ISOIMAGE" or to "", then the volume ID of the loaded image will become the effective volume id for the next write run. But as soon as command −volid is performed afterwards, this pending ID is overridden by the new setting.
Consider this when setting −volid "ISOIMAGE" before executing −dev, −indev, or −rollback. If you insist in −volid "ISOIMAGE", set it again after those commands.

−volset_id text

Set the volume set ID string to be written with the next −commit. Permissible are up to 128 characters. This setting gets overridden by image loading.

−publisher text

Set the publisher ID string to be written with the next −commit. This may identify the person or organisation who specified what shall be recorded. Permissible are up to 128 characters. This setting gets overridden by image loading.

−application_id text

Set the application ID string to be written with the next −commit. This may identify the specification of how the data are recorded. Permissible are up to 128 characters. This setting gets overridden by image loading.
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.

−system_id text

Set the system ID string to be written with the next −commit. 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. This setting gets overridden by image loading.

−volume_date type timestring

Set one of the four overall timestamps for subsequent image writing. Available types are:
"c" time when the volume was created.
"m" time when volume was last modified.
"x" time when the information in the volume expires.
"f" time since when the volume is effectively valid.

"all_file_dates" sets mtime, atime, and ctime of all files and directories to the given time. 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 "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 "uuid".
"uuid" sets a timestring that overrides "c" and "m" times literally and sets the time of the El Torito boot catalog. 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).
Timestrings for the other types may be given as with command −alter_date. Some of them are prone to timezone computations. The timestrings "default" or "overridden" cause default settings: "c" and "m" will show the current time of image creation. "x" and "f" will be marked as insignificant. "uuid" will be deactivated.
At −commit time, some timestamps get set to the maximum value of effectively written volume creation and modification time: El Torito boot catalog, HFS+ superblock, ECMA−119 file modification time if −compliance "no_rec_mtime". The isohybrid MBR id is computed from "uuid" if given, else from the effective volume modification date.

−copyright_file text
Set the copyright file name to be written with the next −commit. This should be the ISO 9660 path of a file in the image which contains a copyright statement. Permissible are up to 37 characters. This setting gets overridden by image loading.

−abstract_file text
Set the abstract file name to be written with the next −commit. 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. This setting gets overridden by image loading.

−biblio_file text
Set the biblio file name to be written with the next −commit. This should be the ISO 9660 path of a file in the image which contains bibliographic records. Permissible are up to 37 characters. This setting gets overridden by image loading.

−preparer_id text
Set the preparer ID string to be written with the next −commit. 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.
Unlike other ID strings, this setting is not influenced by image loading.

−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.
This setting is not influenced by image loading.

−out_charset character_set_name
Set the character set to which file names get converted when writing an image. See paragraph "Character sets" for more explanations. When loading the written image after −commit the setting of −out_charset will be copied to −in_charset.

−uid uid

User id to be used for all files when the new ISO tree gets written to media.

−gid gid

Group id to be used for all files when the new ISO tree gets written to media.

−zisofs parameter[:parameters]

Set global parameters for zisofs compression. This data format is recognized and transparently uncompressed by some Linux kernels. It is to be applied via command −set_filter with built−in filter "−−zisofs".
Note: This command is only permitted while no −−zisofs filters are applied to any files.
Parameters are:
"level="[0−9] zlib compression: 0=none, 1=fast,..., 9=slow
"block_size="32k|64k|128k sets the size of version 1 compression blocks.
"by_magic=on" enables an expensive test at image generation time which checks files from disk whether they already are zisofs compressed, e.g. by program mkzftree. "by_magic=v2" enables processing of already zisofs2 compressed files additionally to those of zisofs version 1. "by_magic=off" disables both.
"version_2="off|as_needed|on controls compression by experimental version zisofs2 which can encode files of size 4 GiB or larger. The Linux kernel (as of 5.9) does not yet know this format and will complain like
isofs: Unknown ZF compression algorithm: PZ
The files will then appear in their compressed form with zisofs2 header, block pointer list, and compressed data.
zisofs2 is recognized by xorriso in files from loaded images and gets equipped with −−zisofs−decode filters, unless restrictions on the number of block pointers prevent this.
Mode "off" restricts compression to files smaller than 4 GiB uncompressed size. Mode "as_needed" uses zisofs2 for larger files. Mode "on" uses zisofs2 for all zisofs compressed files.
"susp_z2="off|on controls production of SUSP entries "Z2" instead of "ZF" with zisofs2 compressed files. Unaware Linux kernels are supposed to silently ignore "Z2" entries.
"block_size_v2="32k|64k|128k|256k|512k|1m sets the size of compression blocks for zisofs2.
"bpt_target="−1|>0 sets a number of block pointers per file, which is considered low enough to justify a reduction of block size. If this number is larger than 0, then block sizes smaller than the settings of block_size= or block_size_v2= are tried whether they yield not more block pointers than the given number. If so, the smallest suitable block size is applied.
The inavoidable final block pointer counts. E.g. a file of 55 KiB has 3 block pointers if block size is 32k, and 2 block pointers with block size 64k.
bpt_target=−1 disables this automatic block size adjustment.
"max_bpt="1k...128g sets the limit for the overall allocated block pointer memory. Block pointers occupy virtual memory while a file gets uncompressed and while a file, which shall be compressed, waits for ISO filesystem creation.
One pointer occupies 8 bytes of memory and governs block_size or block_size_v2 uncompressed bytes. I.e. with block size 128k, 1m of block pointer memory suffices for at most 16g of uncompressed file size. Each file consumes one end block pointer, independently of the file size. Partially filled end blocks may further reduce the effective payload.
In case of overflow of the max_bpt limit while adding compression filters the program tries to go on by discarding all buffered block pointers of previously added −−zisofs filters. From then on all newly added filters will discard their block pointers immediately after being added. Discarded block pointers cause an additional read and compression run of the input file during the production of the ISO filesystem.
"max_bpt_f="1k...128g sets the limit for the memory size of the block pointer list of a single file. max_bpt_f is never larger than max_bpt. If either is set to violate this rule, the other gets set to the same value. If both values are the same before a change by max_bpt= or max_bpt_f=, then both limits stick together unless the limit is decreased by max_bpt_f=.
"bpt_free_ratio="−1|0.0...1.0 sets a threshold for switching to block pointer discarding during compression. If less than the given fraction of the max_bpt_f= memory is free, then block pointers

of compression filters get discarded immediately after being added. Value −1 disables this feature.
"default" is the same as "level=6:block_size=32k:by_magic=off: version_2=off:block_size_v2=128k:susp_z2=off:max_bpt=256m:max_bpt_f=256m: bpt_free_ratio=−1".

−speed code|number[k|m|c|d|b]
Set the burn speed. Default is "max" (or "0") = maximum speed as announced by the drive. Further special speed codes are:
"min" (or "−1") selects minimum speed as announced by the drive.
"none" avoids to send a speed setting command to the drive before burning begins.
Speed can be given in media dependent numbers or as a desired throughput per second in MMC compliant kB (= 1000) or MB (= 1000 kB). Media x−speed factor can be set explicitly by "c" for CD, "d" for DVD, "b" for BD, "x" is optional.
Example speeds:
 706k = 706kB/s = 4c = 4xCD
 5540k = 5540kB/s = 4d = 4xDVD
If there is no hint about the speed unit attached, then the medium in the −outdev will decide. Default unit is CD = 176.4k.
MMC drives usually activate their own idea of speed and take the speed value given by the burn program only as upper limit for their own decision.

−stream_recording "on"|"off"|"full"|"data"|number
Setting "on" tries to circumvent the management of defects on DVD−RAM, BD−RE, or BD−R. Defect management keeps partly damaged media usable. But it reduces write speed to half nominal speed even if the medium is in perfect shape. For the case of flawless media, one may use −stream_recording "on" to get full speed.
"full" tries full speed with all write operations, whereas "on" does this only above byte address 32s. One may give a number of at least 16s in order to set an own address limit.
"data" causes full speed to start when superblock and directory entries are written and writing of file content blocks begins.

−dvd_obs "default"|"32k"|"64k"
GNU/Linux specific: Set the number of bytes to be transmitted with each write operation to DVD or BD media. A number of 64 KB may improve throughput with bus systems which show latency problems. The default depends on media type, on command −stream_recording , and on compile time options.

−modesty_on_drive parameter[:parameters]
Control whether the drive buffer shall be kept from getting completely filled. Parameter "on" (or "1") keeps the program from trying to write to the burner drive while its buffer is in danger to be filled over a given limit. If this limit is exceeded then the program will wait until the filling reaches a given low percentage value.
This can ease the load on operating system and drive controller and thus help with achieving better input bandwidth if disk and burner are not on independent controllers (like hda and hdb). It may also help with throughput problems of simultaneous burns on different burners with Linux kernels like 3.16, if one has reason not to fix the problem by −scsi_dev_family "sg". On the other hand it increases the risk of buffer underflow and thus reduced write speed.
Some burners are not suitable because they report buffer fill with granularity too coarse in size or time, or expect their buffer to be filled to the top before they go to full speed.
Parameters "off" or "0" disable this feature.
The threshold for beginning to wait is given by parameter "max_percent=". Parameter "min_percent=" defines the threshold for resuming transmission. Percentages are permissible in the range of 25 to 100. Numbers in this range without a prepended name are interpreted as "on:min_percent=".
E.g.: −modesty_on_drive 75
The optimal values depend on the buffer behavior of the drive.
Parameter "timeout_sec=" defines after which time of unsuccessful waiting the modesty shall be

disabled because it does not work.
Parameter "min_usec=" defines the initial sleeping period in microseconds. If the drive buffer appears to be too full for sending more data, the program will wait the given time and inquire the buffer fill state again. If repeated inquiry shows not enough free space, the sleep time will slowly be increased to what parameter "max_usec=" defines.
Parameters, which are not mentioned with a −modesty_on_drive command, stay unchanged. Default is:

```
 −modesty_on_drive off:min_percent=90:max_percent=95:
 timeout_sec=120:min_usec=5000:max_usec=25000
```

−use_immed_bit "on"|"off"|"default"

Control whether several long lasting SCSI commands shall be executed with the Immed bit, which makes the commands end early while the drive operation is still going on. xorriso then inquires progress indication until the drive reports to be ready again. If this feature is turned off, then blanking and formatting will show no progress indication.
It may depend on the operating system whether −use_immed_bit is set to "off" by default. Command −status will tell by appending "/on" or "/off" if a drive has already been acquired and −use_immed_bit is currently set to "default". Command −use_immed_bit tolerates and ignores such appended text.

−stdio_sync "on"|"off"|"end"|number

Set the number of bytes after which to force output to stdio: pseudo drives. This forcing keeps the memory from being clogged with lots of pending data for slow devices. Default "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.

−dummy "on"|"off"

If "on" then simulate burning or refuse with FAILURE event if no simulation is possible, do neither blank nor format.

−fs number["k"|"m"]

Set the size of the fifo buffer which smoothens the data stream from ISO image generation to media burning. Default is 4 MiB, minimum 64 kiB, maximum 1 GiB. The number may be followed by letter "k" or "m" which means unit is kiB (= 1024) or MiB (= 1024 kiB).

−close "on"|"off"|"as_needed"

If −close is set to "on" then mark the written medium as not appendable any more. This will have no effect on overwritable media types. Setting "on" is the contrary of cdrecord option −multi, and is one aspect of growisofs option −dvd−compat.
If set to "off" then keep the medium writable for an appended session.
If set to "as_needed" then use "on" only if "off" is predicted to fail with the given medium and its state.
Not all drives correctly recognize fast−blanked DVD−RW which need "on". If there is well founded suspicion that a burn run failed due to −close "off", then −close "as_needed" causes a re−try with "on".
Note that emulation command −as "cdrecord" temporarily overrides the current setting of −close by its own default −close "on" if its option −multi is missing.

−write_type "auto"|"tao"|"sao/dao"

Set the write type for the next burn run. "auto" will select SAO with blank CD media, DAO with blank DVD−R[W] if −close is "on", and elsewise CD TAO or the equivalent write type of the particular DVD/BD media. Choosing TAO or SAO/DAO explicitly might cause the burn run to fail if the desired write type is not possible with the given media state.

−padding number["k"|"m"]|"included"|"appended"

Append the given number of extra bytes to the image stream. This is a traditional remedy for a traditional bug in block device read drivers. Needed only for CD recordings in TAO mode. Since one can hardly predict on what media an image might end up, **xorriso** adds the traditional 300k of padding by default to all images.

For images which will never get to a CD it is safe to use −padding 0 .
Normally padding is not written as part of the ISO image but appended after the image end. This is −padding mode "appended".
Emulation command −as "mkisofs" and command −jigdo cause padding to be written as part of the image. The same effect is achieved by −padding mode "included".

Bootable ISO images:

Contrary to published specifications many BIOSes will load an El Torito record from the first session on media and not from the last one, which gets mounted by default. This makes no problems with overwritable media, because they appear to inadverted readers as one single session.
But with multi−session media CD−R[W], DVD−R[W], DVD+R, it implies that the whole bootable system has to reside already in the first session and that the last session still has to bear all files which the booted system expects after mounting the ISO image.
If a boot image from ISOLINUX or GRUB is known to be present on media then it is advised to patch it when a follow−up session gets written. But one should not rely on the capability to influence the bootability of the existing sessions, unless one can assume overwritable media.
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 command −append_partition. E.g.:

```
  −append_partition 2 0xef /tmp/efi.img
  −boot_image any efi_path=−−interval:appended_partition_2:all::
```

There are booting mechanisms which do not use an El Torito record but rather start at the first bytes of the image: PC−BIOS MBR or EFI GPT for hard−disk−like devices, APM partition entries for Macs which expect HFS+ boot images, MIPS Volume Header for old SGI computers, DEC Boot Block for old MIPS DECstation, SUN Disk Label for SPARC machines, HP−PA boot sector for HP PA−RISC machines, DEC Alpha SRM boot sector for old DEC Alpha machines.

Several of the following commands 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: command −append_partition, boot specs system_area=, grub2_mbr=, prep_boot_part=, efi_boot_part=.
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 an 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 −boot_image bootspecs which announce El Torito boot image paths: bin_path=, efi_path=. The number gives the partition number as used with the corresponding command −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::"

−boot_image "any"|"isolinux"|"grub"
"discard"|"keep"|"patch"|"replay"|"show_status"|
bootspec|"next"

Define the equipment of the emerging filesystem with boot entry points.
With systems which boot via BIOS or EFI this is a set of El Torito boot images, possibly MBR boot code, and possibly partition tables of type MBR, GPT, or APM. Such file sets get produced by boot loader systems like ISOLINUX or GRUB.

Each −boot_image command has two parameters: type and setting. More than one −boot_image command may be used to define the handling of one or more boot images. Sequence matters.
Types **isolinux** and **grub** care for known peculiarities. Type **any** makes no assumptions about the origin of the boot images.

When loading an ISO filesystem, system area and El Torito boot images get loaded, too. The default behavior is not to write loaded El Torito boot images and to write the loaded system area content without alterations.
discard gives up the El Torito boot catalog and its boot images. regardless whether loaded from an ISO filesystem or defined by commands. Any BIOS or EFI related boot options get revoked. Nevertheless, loaded system area data stay valid. If desired, they have to be erased by
 −boot_image any system_area=/dev/zero
keep keeps or copies El Torito boot images unaltered and writes a new catalog.
patch applies patching to existing El Torito boot images if they seem to bear a boot info table.
A boot info table needs to be patched when the boot image gets newly introduced into the ISO image or if an existing image gets relocated. This is automatically done if type "isolinux" or "grub" is given, but not with "any".
If patching is enabled, then boot images from previous sessions will be checked whether they seem to bear a boot info table. If not, then they stay unpatched. This check is not infallible. So if you do know that the images need no patching, use "any" "keep". "grub" "patch" will not patch EFI images (platform_id=0xef).
replay is a more modern version of "patch", which not only cares for existing El Torito boot equipment but also for the recognizable boot provisions in the System Area. It discards any existing −boot_image setting and executes the commands proposed by command −report_el_torito "cmd".
This action will only succeed if the file objects mentioned in the output of command −report_el_torito "cmd" are still available. Do not remove or rename boot image files after −indev.
Drop unknown El Torito: −boot_image "any" "discard"
Maintain recognizable stuff: −boot_image "any" "replay"
El Torito only for GRUB: −boot_image "grub" "patch"
El Torito only for ISOLINUX: −boot_image "isolinux" "patch"
show_status will print what is known about the loaded boot images and their designated fate.

A **bootspec** is a word of the form name=value. It is used to describe the parameters of a boot feature. The names "dir", "bin_path", "efi_path" lead to El Torito bootable images. Name "system_area" activates a given file as MBR or other disk header.
On all media types this is possible within the first session. In further sessions an existing boot image can get replaced by a new one, but depending on the media type this may have few effect at boot time. See above.
El Torito boot images have to be added to the ISO image by normal means (image loading, −map, −add, ...). In case of ISOLINUX the files should reside either in ISO image directory /isolinux or

in /boot/isolinux . In that case it suffices to use as bootspec the text "**dir=/isolinux**" or "dir=/boot/isolinux". E.g.:
 −boot_image isolinux dir=/boot/isolinux
which bundles these individual settings:
 −boot_image isolinux bin_path=/boot/isolinux/isolinux.bin
 −boot_image isolinux cat_path=/boot/isolinux/boot.cat
 −boot_image isolinux load_size=2048
 −boot_image any boot_info_table=on
An El Torito boot catalog file gets inserted into the ISO image with address **cat_path=** with the first −boot_image "any" "next" or at −commit time. It is subject to normal −overwrite and −reassure processing if there is already a file with the same name. The catalog lists the boot images and is read by the boot facility to choose one of the boot images. But it is not necessary that it appears in the directory tree at all. One may hide it in all trees by **cat_hidden=on**. Other possible values are "iso_rr", "joliet", "hfsplus", and the default "off". The timestamps of the boot catalog file are refreshed at commit time. Command −volume_date "uuid" can be used to set their value.
bin_path= depicts an El Torito boot image file, a binary program which is to be started by the hardware boot facility (e.g. the BIOS) at boot time.
efi_path= depicts an El Torito boot image file that is ready for EFI booting. This is normally a FAT filesystem image not larger than 65535 blocks of 512 bytes (= 32 MiB − 512). Its load_size is determined automatically, no boot info table gets written, no boot medium gets emulated, platform_id is 0xef.
emul_type= can be one of "no_emulation", "hard_disk", "diskette". It controls the boot medium emulation code of a boot image. The default "no_emulation" is suitable for ISOLINUX, GRUB, FreeBSD cdboot.
load_size= is a value which depends on the boot image. Default is 2048 which matches the expectations of most boot images. The special value "full" means the full size of the boot image file rounded up to a multiple of 2048 bytes. Maximum is 33,552,384 bytes.
boot_info_table=on causes address patching to bytes 8 to 63 of the boot image which is given by "any" "bin_path=". "boot_info_table=off" disables this patching.
grub2_boot_info=on causes address patching to byte 2548 of the boot image which is given by "any" "bin_path=". 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. "grub2_boot_info=off" disables this patching.
platform_id= defines by a hexadecimal or decimal number the Platform ID of the boot image. "0x00" is 80x86 PC−BIOS, "0x01" is PowerPC, "0x02" is Mac, "0xef" is EFI (decimal "239").
id_string=text|56_hexdigits defines 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.
sel_crit=hexdigits defines 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.
next ends the definition of a boot image and starts a new one. Any following −bootimage bootspecs will affect the new image. The first "next" discards loaded boot images and their catalog.
system_area=disk_path copies at most 32768 bytes from the given disk file to the very start of the ISO image. This System Area is reserved for system dependent boot software, e.g. an MBR which can be used to boot from USB stick or hard disk.
Other than an El Torito boot image, the file disk_path needs not to be added to the ISO image.
−boot_image isolinux system_area= implies "partition_table=on". In this case, the disk path should lead to one of the SYSLINUX files isohdp[fp]x*.bin or to a file which was derived from one of those files. E.g. to the first 512 bytes from an ISOLINUX isohybrid ISO image.
In this case, El Torito boot images (dir=, bin_path=, efi_path=) may be augmented by **isolinux partition_entry=gpt_basdat** or **isolinux partition_entry=gpt_hfsplus**, and by **isolinux**

partition_entry=apm_hfsplus. The boot image will then be mentioned in an invalid GPT as Basic Data or GPT HFS+ partition, and in a valid APM as HFS+ partition. The first three GPT partitions will also be marked by MBR partitions. The MBR partition of type 0xEF is what actually is used by EFI firmware for booting from USB stick.
In multi−session situations the existing System Area is preserved by default. In in this case, the special disk_path "." prevents reading of a disk file but nevertheless causes adjustments in the loaded system area data. Such adjustments may get ordered by −boot_image commands.
−boot_image any gpt_disk_guid=value controls 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 "volume_date_uuid" produces a low quality GUID from the value set by −volume_date "uuid".
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.
−boot_image any part_like_isohybrid=on enables −boot_image isolinux partition_entry= even if no −boot_image isolinux system_area= is given. 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.
−boot_image any iso_mbr_part_type=number sets the partition type of the MBR 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.
grub2_mbr=disk_path works like "any" system_area= with additional patching for modern GRUB MBRs. 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.
This feature can be revoked either by grub2_mbr= with empty disk path, or by submitting a disk_path via system_area=.
partition_table=on causes a simple partition table to be written into bytes 446 to 511 of the System Area.
With type "isolinux" it shows a partition that begins at byte 0 and it causes the LBA of the first boot image to be written into the MBR. For the first session this works only if also "system_area=" and "bin_path=" or "dir=" is given.
With types "any" and "grub" it shows a single partition which starts at byte 512 and ends where the ISO image ends. This works with or without system_area= or boot image.
Bootspecs chrp_boot_part=, prep_boot_part=, and efi_boot_part= overwrite this entry in the MBR partition table.
If types "isolinux" or "grub" are set to "patch", then "partition_table=on" is activated without new boot image. In this case the existing System Area gets checked whether it bears addresses and sizes as if it had been processed by "partition_table=on". If so, then those parameters get updated when the new System Area is written.
Special "system_area=/dev/zero" causes 32k of NUL−bytes. Use this to discard an MBR which was loaded with the ISO image.
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.
appended_part_as=mbr is the default. Appended partitions get marked in GPT only if GPT is produced because of other settings. If given explicitly, this clears setting "gpt" and "apm". Nevertheless "apm" may be added to "mbr".
appended_part_as=apm marks partitions from −append_partition in APM additionally to "mbr" or "gpt".
By default, appended partitions get marked in APM only if APM is produced because of other options together with part_like_isohybrid="on".
chrp_boot_part=on causes a single partition in MBR which covers the whole ISO image and has type 0x96. This is not compatible with any other feature that produces MBR partition entries. It makes GPT unrecognizable.
prep_boot_part=disk_path inserts the content of a data file into the image and marks it by an MBR partition of type 0x41. The parts of the ISO image before and after this partition will be covered by further MBR partitions. The data file is supposed to contain ELF executable code.
efi_boot_part=disk_path inserts the content of a data file into the image and marks it by a GPT partition. If not chrp_boot_part=on, then the first partition in MBR will have type 0xee to announce the presence of GPT. The data file is supposed to contain a FAT filesystem.
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 bootspec efi_path=. The affected EFI boot image cannot show up in HFS+ because it is stored outside the HFS+ partition.
partition_offset=2kb_block_adr causes 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. 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. So the value defined here is only in effect if a new ISO image gets written.
partition_hd_cyl=number gives the number of heads per cylinder for the partition table. 0 chooses a default value. Maximum is 255.
partition_sec_hd=number gives the number of sectors per head for the 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.
partition_cyl_align=mode controls 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 with "isolinux" "partition_table=on".
Mode "on" causes alignment by padding with "partition_table=on" for any type. Mode "all" is like "on" but also pads up partitions from −append_partition to an aligned size.
Mode "off" disables alignment for any type.
mbr_force_bootable=mode enforces an MBR partition with "bootable/active" flag if options like partition_table= or grub2_mbr= indicate production of a bootable MBR. 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="on" 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 and a partition gets created by −append_partition, then mbr_force_bootable="on" causes a bootflag like it would do with a bootable MBR.
mips_path=iso_rr_path declares a data file in the image to be a MIPS Big Endian boot file and causes 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 system_area=. Up to 15 boot files can be declared by mips_path=.
mipsel_path=iso_rr_path declares 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 system_area=. Only a single boot file can be declared by mipsel_path=.
sparc_label=text causes the production of a SUN Disk Label with the given text as ASCII label. Partitions 2 to 8 may be occupied by appended images. Partition 1 will always be the ISO image. See command −append_partition. The first 512 bytes of any data provided by system_area= will be overwritten.
grub2_sparc_core=iso_rr_path causes the content address and size of the given file 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 sets 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_ bootspecs are mandatory, if any of the hppa_ bootspecs is used. Only hppa_hdrversion= is allowed to be missing.
hppa_bootloader=iso_rr_path designates the given path as HP−PA bootloader file.
hppa_kernel_32=iso_rr_path designates the given path as HP−PA 32 bit kernel file.
hppa_kernel_64=iso_rr_path designates the given path as HP−PA 64 bit kernel file.
hppa_ramdisk=iso_rr_path designates the given path as HP−PA RAM disk file.
hppa_hdrversion=number chooses between PALO header version 5 (default) and version 4. For the appropriate value see in PALO source code: PALOHDRVERSION.
alpha_boot=iso_rr_path declares a data file in the image to be the DEC Alpha SRM Secondary Bootstrap Loader and causes production of a boot sector which points to it. This is mutually exclusive with production of other boot blocks like MBR.
mips_discard, **sparc_discard**, **hppa_discard**, **alpha_discard** revoke any boot file declarations made for mips/mipsel, sparc, hppa, or alpha, respectively. This removes the ban on production of other boot blocks.
hfsplus_serial=hexstring sets 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 sets 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 sets 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.

−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.
Partitions may be appended with boot block type MBR and with SUN Disk Label.
With MBR:
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 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 MBR partition type codes search the Internet for "Partition Types" or run fdisk command "L".
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.
The disk_path must provide the necessary data bytes at commit time. An empty disk_path disables this feature for the given partition number.
With SUN Disk Label (selected by −boot_image any sparc_label=):
partition_number may be 2 to 8. Number 1 will always be the ISO image. Partition start addresses are aligned to 320 KiB. The type_code does not matter. Submit 0x0.
Partition image name "." causes the partition to become a copy of the next lower valid one.

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."
xorriso can produce a .jigdo and a .template file together with a single−session ISO image. 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 **xorriso** session on a blank −outdev, 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 **xorriso** will refuse to write to non−blank targets, it will disable multi−session emulation, and padding will be counted as part of the ISO image.

−jigdo parameter_name value

Clear Jigdo Template Extraction parameter list or add a parameter to that list. The alias names are the corresponding genisoimage options. They are accepted as parameter names as well. Especially they are recognized by the −as mkisofs emulation command.
Parameter **clear** with any value empties the whole list. No .jigdo and .template file will be produced.
checksum_algorithm chooses the checksum algorithm which shall be used for the data file entries in the .jigdo file and is expected in the checksum file. Permissible are "md5" or "sha256". Default is "md5".
Alias: −jigdo−checksum−algorithm
template_path sets the disk_path for the .template file with the holed and compressed ISO image copy.
Alias: −jigdo−template
jigdo_path sets the disk_path for the .jigdo file with the checksums and download addresses for filling the holes in .template.
Alias: −jigdo−jigdo
checksum_path sets the disk_path where to find the checksum file with symbolic file addresses and checksums according to **checksum_algorithm**.
Alias: md5_path
Alias: −checksum−list
Alias: −md5−list
min_size sets the minimum size for a data file to be listed in the .jigdo file and being a hole in the .template file.

Alias: −jigdo−min−file−size
exclude adds 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.
Alias: −jigdo−exclude
demand_checksum 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 list file as of "checksum_path". A match causes a MISHAP event.
Alias: demand_md5
Alias: −jigdo−force−checksum
Alias: −jigdo−force−md5
mapping adds 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.
Alias: −jigdo−map
compression chooses one of "bzip2" or "gzip" for the compression of the template file. The jigdo file is put out uncompressed.
Alias: −jigdo−template−compress
checksum_iso chooses one or more of "md5", "sha1", "sha256", "sha512" for the auxiliary "# Image Hex" checksums in the jigdo file. The value may e.g. look like "md5,sha1,sha512". Value "all" chooses all available algorithms. Note that MD5 stays always enabled.
Alias: −checksum_algorithm_iso
checksum_template is like checksum_iso but for "# Template Hex".
Alias: −checksum_algorithm_template

Character sets:

File names are strings of non−zero bytes with 8 bit each. Unfortunately the same byte string may appear as different peculiar national characters on differently nationalized terminals. The meanings of byte codes are defined in **character sets** which have names. Shell command iconv −l lists them.
The file names on hard disk are assumed to be encoded by the **local character set** which is also used for the communication with the user. Byte codes 32 to 126 of the local character set must match the US−ASCII characters of the same code. ISO−8859 and UTF−8 fulfill this demand.
By default, **xorriso** uses the character set as told by shell command "locale" with argument "charmap". This may be influenced by environment variables LC_ALL, LC_CTYPE, or LANG and should match the expectations of the terminal. In some situations it may be necessary to set it by command −local_charset.
Local 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 media use the same local character set. Outside these constraints it may be necessary to let **xorriso** convert byte codes from and to other character sets.
The Rock Ridge file names in ISO filesystems are assumed to be encoded by the **input character set**. The Rock Ridge file names which get written with ISO filesystems will be encoded by the **output character set**.
The sets can be defined independently by commands −in_charset and −out_charset. Normally one will have both identical, if ever. Other than the local character set, these two character sets may deviate from US−ASCII.
The output character sets for Joliet and HFS+ are not influenced by these commands. Joliet uses output character set UCS−2 or UTF−16. HFS+ uses UTF−16.
The default output charset is the local character set of the terminal where **xorriso** runs. So by default no conversion happens between local filesystem names and emerging Rock Ridge names in the image. The situation stays ambiguous and the reader has to riddle what character set was used.
By command −auto_charset it is possible to attribute the output charset name to the image. This makes the situation unambiguous. But if your terminal character set does not match the character set of the local file names, then this attribute can become plainly wrong and cause problems at read time. To prevent this it is necessary to check whether the terminal properly displays all intended filenames. Check especially the exotic national characters.

To enforce recording of a particular character set name without any conversion at image generation time, set −charset and −local_charset to the desired name, and enable −backslash_codes to avoid evil character display on your terminal.

−charset character_set_name
Set the character set from which to convert file names when loading an image and to which to convert when writing an image.

−local_charset character_set_name
Override the system assumption of the local character set name. If this appears necessary, one should consider to set −backslash_codes to "on" in order to avoid dangerous binary codes being sent to the terminal.

Exception processing:

Since the tasks of **xorriso** are manifold and prone to external influence, there may arise the need for **xorriso** to report and handle problem events.
Those events get classified when they are detected by one of the software modules and forwarded to reporting and evaluation modules which decide about reactions. Event classes are sorted by severity:
"NEVER" The upper end of the severity spectrum.
"ABORT" The program is being aborted and on its way to end.
"FATAL" The main purpose of the run failed or an important resource failed unexpectedly.
"FAILURE" An important part of the job could not be performed.
"MISHAP" A FAILURE which can be tolerated during ISO image generation.
"SORRY" A less important part of the job could not be performed.
"WARNING" A situation is suspicious of being not intended by the user.
"HINT" A proposal to the user how to achieve better results.
"NOTE" A harmless information about noteworthy circumstances.
"UPDATE" A pacifier message during long running operations.
"DEBUG" A message which would only interest the program developers.
"ALL" The lower end of the severity spectrum.

−abort_on severity
Set the severity threshold for events to abort the program.
Useful: "NEVER", "ABORT", "FATAL", "FAILURE" , "MISHAP", "SORRY"
It may become necessary to abort the program anyway, despite the setting by this command. Expect not many "ABORT" events to be ignorable.
A special property of this command is that it works preemptive if given as program start argument. I.e. the first −abort_on setting among the start arguments is in effect already when the first operations of **xorriso** begin. Only "−abort_on" with dash "−" is recognized that way.

−return_with severity exit_value
Set the threshold and exit_value to be returned at program end if no abort has happened. This is to allow **xorriso** to go on after problems but to get a failure indicating exit value from the program, nevertheless. Useful is a value lower than the −abort_on threshold, down to "WARNING".
exit_value may be either 0 (indicating success to the starter of the program) or a number between 32 and 63. Some other exit_values are used by **xorriso** if it decides to abort the program run:
1=abort due to external signal
2=no program arguments given
3=creation of **xorriso** main object failed
4=failure to start libburnia−project.org libraries
5=program abort during argument processing
6=program abort during dialog processing

−report_about severity
Set the threshold for events to be reported.
Useful: "SORRY", "WARNING", "HINT", "NOTE", "UPDATE", "DEBUG", "ALL"
Regardless what is set by −report_about, messages get always reported if they reach the severity threshold of −abort_on .

Event messages are sent to the info channel "I" which is usually stderr but may be influenced by command −pkt_output. Info messages which belong to no event get attributed severity "NOTE".
A special property of this command is that the first −report_about setting among the start arguments is in effect already when the first operations of **xorriso** begin. Only "−report_about" with dash "−" is recognized that way.

−signal_handling mode

Control the installation of a signal handler which shall react on external signals (e.g. from program "kill" or from keys Ctrl+C) or on signals caused by severe program errors.
Mode "on" is the default. It uses the signal handler of libburn which produces ugly messages but puts much effort in releasing optical drives before **xorriso** ends.
Mode "off" as first −signal_handling among the start arguments prevents all own signal precautions of **xorriso**. Inherited signal handler settings stay as they are.
It works like "sig_dfl" if given after other signal handling was already established at program start.
Mode "sig_dfl" uses the system provided default handling of signals, which is normally a sudden abort of the program. To prevent stuck drives, the libburn handler is used during burning, blanking, and formatting on MMC drives.
Mode "sig_ign" tries to ignore as many signal types as possible. This imposes the risk that **xorriso** refuses to end until externally kill −9 if performed. kill −9 then imposes the risk that the drive is left in unusable state and needs poweroff to be reset. So during burning, blanking, and formatting wait for at least their normal run time before killing externally.
A special property of this command is that the first −signal_handling setting among the start arguments is in effect already when the first operations of **xorriso** begin. Only "−signal_handling" with dash "−" is recognized that way.

−error_behavior occasion behavior

Control the program behavior at problem event occasions. For now this applies to occasions "image_loading" which is given while an image tree is read from the input device, and to "file_extraction" which is given with osirrox commands like −extract.
With "image_loading" there are three behaviors available:
"best_effort" goes on with reading after events with severity below FAILURE if the threshold of command −abort_on allows this.
"failure" aborts image tree reading on first event of at least SORRY. It issues an own FAILURE event. This is the default.
"fatal" acts like "failure" but issues the own event as FATAL.
With occasion "file_extraction" there are three behaviors:
"keep" maintains incompletely extracted files on disk. This is the default.
"delete" removes files which encountered errors during content extraction.
"best_effort" starts a revovery attempt by means of −extract_cut if the file content stems from the loaded ISO image and is not filtered.

Dialog mode control:

−dialog "on"|"off"|"single_line"

Enable or disable to enter dialog mode after all program arguments are processed. In dialog mode input lines get prompted via readline or from stdin.
If no −abort_on severity was set when dialog starts, then "NEVER" is set to avoid abort in most cases of wrong input or other problems. Before dialog begins, the default is "FAILURE" which e.g. aborts on unknown commands.
Mode "on" supports input of newline characters within quotation marks and line continuation by trailing backslash outside quotation marks. Mode "single_line" does not.

−page length width

Describe terminal to the text pager. See also above, paragraph Result pager.
If parameter length is nonzero then the user gets prompted after that number of terminal lines. Zero length disables paging.
Parameter width is the number of characters per terminal line. It is used to compute the number of

terminal lines which get occupied by an output line. A usual terminal width is 80.

−use_readline "on"|"off"
If "on" then use readline for dialog. Else use plain stdin.
See also above, paragraph Dialog, Readline, Result pager.

−reassure "on"|"tree"|"off"
If "on" then ask the user for "y" or "n":
before deleting or overwriting any file in the ISO image,
before overwriting any disk file during restore operations,
before rolling back pending image changes,
before committing image changes to media,
before changing the input drive,
before blanking or formatting media,
before ending the program.
With setting "tree" the reassuring prompt will appear for an eventual directory only once and not for each file in its whole subtree.
Setting "off" silently kills any kind of image file object and performs above irrevocable actions.
To really produce user prompts, command −dialog needs to be set to "on". Note that the prompt does not appear in situations where file removal is forbidden by command −overwrite. −reassure only imposes an additional curb for removing existing file objects.
Be aware that file objects get deleted from the ISO image immediately after confirmation. They are gone even if the running command gets aborted and its desired effect gets revoked. In case of severe mess−up, consider to use −rollback to revoke the whole session.

Drive and media related inquiry actions:

−devices
Show list of available MMC drives with the addresses of their libburn standard device files.
This is only possible when no ISO image changes are pending. After this command was executed, there is no drive current and no image loaded.
In order to be visible, a device has to offer rw−permissions with its libburn standard device file. Thus it might be only the **superuser** who is able to see all drives.
Drives which are occupied by other processes get not shown.

−device_links
Like −devices, but presenting the drives with addresses of symbolic links which point to the actual device files.
Modern GNU/Linux systems may shuffle drive addresses from boot to boot. The udev daemon is supposed to create links which always point to the same drive, regardless of its system address. The command −device_links shows the addresses of such links if they begin by "/dev/dvd" or "/dev/cd". Precedence is: "dvdrw", "cdrw", "dvd", "cdrom", "cd".

−toc
Show media specific tables of content. This is the session history of the medium, not the ISO image directory tree.
In case of overwritable media holding a valid ISO image, it may happen that only a single session gets shown. But if the first session on the overwritable media was written by **xorriso** then a complete session history can be emulated.
A drive which is incapable of writing may show any media as CD−ROM or DVD−ROM with only one or two sessions on it. The last of these sessions is supposed to be the most recent real session then.
Some read−only drives and media show no usable session history at all. Command −rom_toc_scan might help.
If input device and output device are both acquired and not the same, then both tables−of−content get shown.

−toc_of "in"|"out"|"all"[":short"]

Like command −toc but explicitly choosing which drive's table−of−content to show. "in" shows −indev or −dev, "out" shows −outdev or −dev, "all" shows the same as −toc.
If ":short" is appended to the drive choosing word, then only a short summary of drive state and medium content is printed.
As further difference to −toc, this command does not emit FAILURE events if the desired drive is not acquired.

−mount_cmd drive entity id path

Emit an appropriate command line for mounting the ISO session indicated by drive, entity and id. The result will be different on GNU/Linux and on FreeBSD or NetBSD.
drive can be "indev" or "outdev" to indicate already acquired drives, or it can be the path of a not yet acquired drive. Prefix "stdio:" for non−MMC drives is not mandatory.
For entity and id, see also command −load. They must be either "sbsector" with the superblock sector address as id, or "track" with a track number as id, or "session" with a session number, or "volid" with a search pattern for the volume id, or "auto" with which any text as id mounts the first track of the last session.
path will be used as mount point and must already exist as a directory on disk.
The command gets printed to the result channel. See command −mount for direct execution of this command.

−mount_opts option[:option...]

Set options which influence −mount and −mount_cmd. Currently there is only option "exclusive" which is default and its counterpart "shared". The latter causes **xorriso** not to give up the affected drive with command −mount. On GNU/Linux it adds mount option "loop" which may enable mounting of several sessions of the same block device at the same time. One should not write to a mounted optical medium, of course. Take care to umount all sessions before ejecting.

−session_string drive entity id format

Print to the result channel a text which gets composed according to format and the parameters of the addressed session.
Formats "linux:"path or "freebsd:"path produce the output of −mount_cmd for the given operating systems.
In other texts **xorriso** will substitute the following parameter names. An optional prefix "string:" will be removed.
"%device%" will be substituted by the mountable device path of the drive address.
"%sbsector%" will be substituted by the session start sector.
"%track%", "%session%", "%volid%" will be substituted by track number, session number, or volume id of the depicted session.

−print_size

Print the foreseeable consumption of 2048 byte blocks by next −commit. This can last a while as a −commit gets prepared and only in last moment is revoked by this command. The result depends on several settings and also on the kind of output device. If no −jigdo options are set and not command −as "mkisofs" was used, then −padding (300 kB by default) is not counted as part of the image size.
If an El Torito boot image file is already depicted, then command −print_size automatically executes −boot_image "any" "next". This means that the properties of that boot image cannot be edited by subsequent commands.

−tell_media_space

Print available space on the output medium and the free space after subtracting already foreseeable consumption by next −commit.
Note that the title of the prediction "After commit :" is misleading. It is rather the space that may still be filled in this session without making the next −commit fail from medium overflow.
The free space after the next −commit might be smaller by several MB. This depends on medium type, number of recorded sessions, and drive habits.

−pvd_info

Print various ID strings and timestamps which can be found in loaded ISO images. Some of the IDs may be changed by commands like −volid or −publisher. For these IDs −pvd_info reports what would be written with the next −commit. The timestamps get not automatically propagated from loaded image to newly written image. The ones for new images may be set by command −volume_date. See there for the meaning of the particular timestamps.

−report_el_torito mode

With mode **plain** print a report about the information found in the El Torito boot catalog of the loaded ISO image.
With mode **help** print a text which explains the meaning of the lines put out by "plain".
Mode **cmd** tries to print the **xorriso** commands which are necessary to produce the found boot equipment: disk identifiers, El Torito boot images, and System Area. Disk identifiers are strings which the booting operating system might use to find the ISO filesystem from where it comes. Currently known is the use of volume id and modification date.
The intended use case is modification of the filesystem by having −indev and −outdev pointing to different images or drives. The result might be insufficient, if the found equipment cannot be produced by xorriso. Various SORRY events may arise in this case, but it is not guaranteed that xorriso recognizes all its insufficiencies.
Mode **as_mkisofs** tries to print the **xorriso −as mkisofs** options, which are necessary to produce the found equipment. The intended use case is to use the mounted filesystem as input tree together with the printed options.

−report_system_area mode

With mode **plain** print a report about the information found in the System Area of the loaded ISO image. The report consists of zero to many lines with a header text, a colon, and information text.
With mode **help** print a text which explains the meaning of the lines put out by "plain". You probably will have to look for more documentation which explains the technical details of the mentioned boot facilities.
Modes **cmd** and **as_mkisofs** work like with command −report_el_torito. See above.
With mode **gpt_disk_guid** print the GPT disk GUID of the loaded ISO in RFC 4122 text format to result channel. It is not considered an error if no GPT is present. In this case nothing is printed to result channel.
With mode **gpt_crc_of:**disk_path read up to 32 KiB from the disk file with the path given after the colon. Compute the GPT compliant CRC number and print it to the result channel. The number is shown like "0x690fd979". The special disk_path "−" causes reading from standard input.
With mode **make_guid** print a pseudo−random GUID in RFC 4122 text format to result channel.

Navigation in ISO image and disk filesystem:

−cd iso_rr_path

Change the current working directory in the ISO image. This is prepended to iso_rr_paths which do not begin with '/'.
It is possible to set the working directory to a path which does not exist yet in the ISO image. The necessary parent directories will be created when the first file object is inserted into that virtual directory. Use −mkdir if you want to enforce the existence of the directory already at first insertion.

−cdx disk_path

Change the current working directory in the local filesystem. To be prepended to disk_paths which do not begin with '/'.

−pwd

Tell the current working directory in the ISO image.

−pwdx

Tell the current working directory in the local filesystem.

−ls iso_rr_pattern [***]
List files in the ISO image which match shell patterns (i.e. with wildcards ’*’ ’?’ ’[a−z]’). If a pattern does not begin with ’/’ then it is compared with addresses relative to −cd.
Directories are listed by their content rather than as single file item.
Pattern expansion may be disabled by command −iso_rr_pattern.

−lsd iso_rr_pattern [***]
Like −ls but listing directories as themselves and not by their content. This resembles shell command ls −d.

−lsl iso_rr_pattern [***]
Like −ls but also list some of the file attributes. The output format resembles shell command ls −ln.
File type ’e’ indicates the El Torito boot catalog.
If the file has non−trivial ACL, then a ’+’ is appended to the permission info. If the file is hidden, then ’I’ for "iso_rr", ’J’ for "joliet", ’A’ for "hfsplus", ’H’ for multiple hiding gets appended. Together with ACL it is ’i’, ’j’, ’a’, ’h’.

−lsdl iso_rr_pattern [***]
Like −lsd but also list some of the file attributes. The output format resembles shell command ls −dln.

−lsx disk_pattern [***]
List files in the local filesystem which match shell patterns. Patterns which do not begin with ’/’ are used relative to −cdx.
Directories are listed by their content rather than as single file item.
Pattern expansion may be disabled by command −disk_pattern.

−lsdx disk_pattern [***]
Like −lsx but listing directories as themselves and not by their content. This resembles shell command ls −d.

−lslx disk_pattern [***]
Like −lsx but also listing some of the file attributes. Output format resembles shell command ls −ln.

−lsdlx disk_pattern [***]
Like −lsdx but also listing some of the file attributes. Output format resembles shell command ls −dln.

−getfacl iso_rr_pattern [***]
Print the access permissions of the given files in the ISO image using the format of shell command getfacl. If a file has no ACL then it gets fabricated from the −chmod settings. A file may have a real ACL if it was introduced into the ISO image while command −acl was set to "on".

−getfacl_r iso_rr_pattern [***]
Like −gefacl but listing recursively the whole file trees underneath eventual directories.

−getfattr iso_rr_pattern [***]
Print the xattr of the given files in the ISO image. If a file has no such xattr then noting is printed for it. The choice of namespaces depends on the setting of command −xattr: "on" or "user" restricts it to namespace "user", "any" only omits namespace "isofs".

−getfattr_r iso_rr_pattern [***]
Like −gefattr but listing recursively the whole file trees underneath of directories.

−du iso_rr_pattern [***]
Recursively list size of directories and files in the ISO image which match one of the patterns. similar to shell command du −k.

−dus iso_rr_pattern [***]

List size of directories and files in the ISO image which match one of the patterns. Similar to shell command du −sk.

−dux disk_pattern [***]

Recursively list size of directories and files in the local filesystem which match one of the patterns. Similar to shell command du −k.

−dusx disk_pattern [***]

List size of directories and files in the local filesystem which match one of the patterns. Similar to shell command du −sk.

−findx disk_path [-name pattern] [-type t] [-exec action [params]] --

Like −find but operating on local filesystem and not on the ISO image. This is subject to the settings of −follow.
−findx accepts the same −type parameters as −find. Additionally it recognizes type "mountpoint" (or "m") which matches subdirectories which reside on a different device than their parent. It never matches the disk_path given as start address for −findx.
−findx accepts the −exec actions as does −find. But except the following few actions it will always perform action "echo".
in_iso reports the path if its counterpart exists in the ISO image. For this the disk_path of the −findx command gets replaced by the iso_rr_path given as parameter.
E.g.: −findx /home/thomas −exec in_iso /thomas_on_cd −−
not_in_iso reports the path if its counterpart does not exist in the ISO image. The report format is the same as with command −compare.
add_missing iso_rr_path_start adds the counterpart if it does not yet exist in the ISO image and marks it for "rm_merge" as non−removable.
E.g.: −findx /home/thomas −exec add_missing /thomas_on_cd −−
is_full_in_iso reports if the counterpart in the ISO image contains files. To be used with −type "m" to report mount points.
empty_iso_dir deletes all files from the counterpart in the ISO image. To be used with −type "m" to truncate mount points.
estimate_size prints a lower and an upper estimation of the number of blocks which the found files together will occupy in the emerging ISO image. This does not account for the superblock, for the directories in the −findx path, or for image padding.
list_extattr mode prints a script to the result channel, which would use FreeBSD command setextattr to set the file's xattr name−value pairs of user namespace. See −find for a description of parameter mode.
E.g. −exec list_extattr e −−

−compare disk_path iso_rr_path

Compare attributes and eventual data file content of a fileobject in the local filesystem with a file object in the ISO image. The iso_rr_path may well point to an image file object which is not yet committed, i.e. of which the data content still resides in the local filesystem. Such data content is prone to externally caused changes.
If iso_rr_path is empty then disk_path is used as path in the ISO image too.
Differing attributes are reported in detail, differing content is summarized. Both to the result channel. In case of no differences no result lines are emitted.

−compare_r disk_path iso_rr_path

Like −compare but working recursively. I.e. all file objects below both addresses get compared whether they have counterparts below the other address and whether both counterparts match.

−compare_l disk_prefix iso_rr_prefix disk_path [***]

Perform −compare_r with each of the disk_path parameters. iso_rr_path will be composed from disk_path by replacing disk_prefix by iso_rr_prefix.

−show_stream iso_rr_path [***]

Display the content stream chain of data files in the ISO image. The chain consists of the iso_rr_name and one or more streams, separated by " < " marks. A stream description consists of one or more texts, separated by ":" characters. The first text tells the stream type, the following ones, if ever, describe its individual properties. Frequently used types are:
 disk:’disk_path’ for local filesystem objects.
 image:’iso_rr_path’ for ISO image file objects.
 cout:’disk_path offset count’ for −cut_out files.
 extf:’filter_name’ for external filters.
 −−zisofs:algorithm:block_size for zisofs compression filters.
 −−zisofs−decode:algorithm:block_size for zisofs uncompression filters.
 −−gzip for internal gzip compression filters.
 −−gunzip for internal gzip uncompression filters.
Example:
 ’/abc/xyz.gz’ < extf:’gzip’ < disk:’/home/me/x’

−show_stream_r iso_rr_path [***]

Like −show_stream but working recursively.

Evaluation of readability and recovery:

It is not uncommon that optical media produce read errors. The reasons may be various and get obscured by error correction which is performed by the drives and based on extra data on the media. If a drive returns data then one can quite trust that they are valid. But at some degree of read problems the correction will fail and the drive is supposed to indicate error.
xorriso can scan a medium for readable data blocks, classify them according to their read speed, save them to a file, and keep track of successfully saved blocks for further tries on the same medium.
By command −md5 checksums may get recorded with data files and whole sessions. These checksums are reachable only via indev and a loaded image. They work independently of the media type and can detect transmission errors.

−check_media [option [option ...]] --

Try to read data blocks from the indev drive, optionally copy them to a disk file, and finally report about the encountered quality. Several options may be used to modify the default behavior.
The parameters given with this command override the default settings which may have been changed by command −check_media_defaults. See there for a description of available options.
The result list tells intervals of 2 KiB blocks with start address, number of blocks and quality. Qualities which begin with "+" are supposed to be valid readable data. Qualities with "−" are unreadable or corrupted data. "0" indicates qualities which are not covered by the check run or are regularly allowed to be unreadable (e.g. gaps between tracks).
Alternatively it is possible to report damaged files rather than blocks.
If −md5 is "on" then the default mode what=tracks looks out for libisofs checksum tags for the ISO session data and checks them against the checksums computed from the data stream.

−check_media_defaults [option [option ...]] --

Preset options for runs of −check_media, −extract_cut and best_effort file extraction. Options given with −check_media will override the preset options. −extract_cut will override some options automatically.
An option consists of a keyword, a "=" character, and a value. Options may override each other. So their sequence matters.
The default setting at program start is:
use=indev what=tracks min_lba=−1 max_lba=−1 retry=default
time_limit=28800 item_limit=100000 data_to=’’ event=ALL
abort_file=/var/opt/xorriso/do_abort_check_media
sector_map=’’ map_with_volid=off patch_lba0=off report=blocks
bad_limit=invalid slow_limit=1.0 chunk_size=0s async_chunks=0
Option "reset=now" restores these startup defaults.

Non−default options are:
report="files" lists the files which use damaged blocks (not with use=outdev). The format is like with find −exec report_damage. Note that a MD5 session mismatch marks all files of the session as damaged. If finer distinction is desired, perform −md5 off before −check_media.
report="blocks_files" first lists damaged blocks and then affected files.
use="outdev" reads from the output drive instead of the input drive. This avoids loading the ISO image tree from media.
use="sector_map" does not read any media but loads the file given by option sector_map= and processes this virtual outcome.
what="disc" scans the payload range of a medium without respecting track gaps.
what="image" similar to "disc", but restricts scanning to the range of the ISO 9660 image, if present.
min_lba=limit omits all blocks with addresses lower than limit.
max_lba=limit switches to what=disc and omits all blocks above limit.
chunk_size=size sets the number of bytes to be read in one low−level read operation. This gets rounded down to full blocks of 2048 bytes. 0 means automatic size.
retry="on" forces read retries with minimal senseful chunk size when the normal read chunk produces a read error. This size is 1s with CD and stdio files, 16s with DVD (1 ECC Block), and 32s with BD (1 Cluster). By default, retries are only enabled with CD media. "retry=off" forbits retries for all media types.
abort_file=disk_path gives the path of the file which may abort a scan run. Abort happens if the file exists and its mtime is not older than the start time of the run. Use shell command "touch" to trigger this. Other than an aborted program run, this will report the tested and untested blocks and go on with running **xorriso**.
time_limit=seconds gives the number of seconds after which the scan shall be aborted. This is useful for unattended scanning of media which may else overwork the drive in its effort to squeeze out some readable blocks. Abort may be delayed by the drive gnawing on the last single read operation. Value −1 means unlimited time.
item_limit=number gives the number of report list items after which to abort. Value −1 means unlimited item number.
data_to=disk_path copies the valid blocks to the given file, which must support random access writing, unless disk_path is "−" which means standard output.
In the latter case, patch_lba0= settings other than "off" yield failure. Further the usual result messages of −check_media get redirected to the info channel. But beware of result messages from other commands. Beware of −*dev "−" which redirect standard output to standard error. Keep the run simple:

```
  xorriso −indev /dev/sr0 −check_media data_to=− −− | md5sum
  xorriso −outdev /dev/sr0 −check_media data_to=− use=outdev \
          what=disc min_lba=0 max_lba=999999 −− | sha256sum
```

event=severity sets the given severity for a problem event which shall be issued at the end of a check run if data blocks were unreadable or failed to match recorded MD5 checksums. Severity "ALL" disables this event.
sector_map=disk_path tries to read the file given by disk_path as sector bitmap and to store such a map file after the scan run. The bitmap tells which blocks have been read successfully in previous runs. It is the persistent memory for several scans on the same medium, even with intermediate eject, in order to collect readable blocks whenever the drive is lucky enough to produce them. The stored file contains a human readable TOC of tracks and their start block addresses, followed by binary bitmap data.
By default, untested blocks are not considered bad, but rather as intentionally unread. If you expect time_limit= or item_limit= to abort the run, then consider to use bad_limit="untested".
map_with_volid="on" examines tracks whether they are ISO images and prints their volume IDs into the human readable TOC of sector_map=.
patch_lba0="on" transfers within the data_to= file a copy of the currently loaded session head to the start of that file and patches it to be valid at that position. This makes the loaded session the

last valid session of the image file when it gets mounted or loaded as stdio: drive. New sessions will be appended after this last session and will overwrite any sessions which have followed it.
patch_lba0="force" performs patch_lba0="on" even if **xorriso** believes that the copied data are not valid.
patch_lba0= may also bear a number. If it is 32 or higher it is taken as start address of the session to be copied. In this case it is not necessary to have an −indev and a loaded image. ":force" may be appended after the number.
bad_limit=threshold sets the highest quality which shall be considered as damage. Choose one of "good", "md5_match", "slow", "partial", "valid", "untested", "md5_mismatch", "invalid", "tao_end", "off_track", "unreadable".
"valid" and "invalid" are qualities imported from a sector_map file. "tao_end" and "off_track" are intentionally not readable, but not bad either. "partial" are blocks retrieved from a partially readable chunk. They are supposed to be ok but stem from a suspicious neighborhood.
"md5_match" and "md5_mismatch" regions overlap with regions of other quality. The former is a strong confirmation for quality, the latter only tells that one or more blocks of the region must be wrong.
By default bad_limit is set higher than md5_mismatch, so that mismatches are classified as quality class "0" rather than "−". This means that the sectors of a MD5 mismatch range are recorded in the sector_map as successfully read, if the drive handed them out at all. Set "bad_limit=md5_mismatch" to let the sector_map record the whole mismatching range as yet not retrieved.
slow_limit=threshold sets the time threshold for a single read chunk to be considered slow. This may be a fractional number like 0.1 or 1.5.
async_chunks=number enables asynchronous MD5 processing if number is 2 or larger. In this case the given number of read chunks is allocated as fifo buffer. On very fast MMC drives try: chunk_size=64s async_chunks=16.

−check_md5 severity iso_rr_path [***]
Compare the data content of the given files in the loaded image with their recorded MD5 checksums, if there are any. In case of any mismatch an event of the given severity is issued. It may then be handled by appropriate settings of commands −abort_on or −return_with which both can cause non−zero exit values of the program run. Severity ALL suppresses that event.
This command reports match and mismatch of data files to the result channel. Non−data files cause NOTE events. There will also be UPDATE events from data reading.
If no iso_rr_path is given then the whole loaded session is compared with its MD5 sum. Be aware that this covers only one session and not the whole image if there are older sessions.

−check_md5_r severity iso_rr_path [***]
Like −check_md5 but checking all data files underneath the given paths. Only mismatching data files will be reported.

osirrox ISO-to-disk restore commands:

Normally **xorriso** only writes to disk files which were given as stdio: pseudo−drives or as log files. But its alter ego osirrox is able to extract file objects from ISO images and to create, overwrite, or delete file objects on disk.
Disk file exclusions by −not_mgt, −not_leaf, −not_paths apply. If disk file objects already exist then the settings of −overwrite and −reassure apply. But −overwrite "on" only triggers the behavior of −overwrite "nondir". I.e. directories cannot be deleted.
Access permissions of files in the ISO image do not restrict restoring. The directory permissions on disk have to allow rwx.

−osirrox setting[:option:...]
Setting **off** disables disk filesystem manipulations. This is the default unless the program was started with leafname **osirrox**. Elsewise the capability to restore files can be enabled explicitly by −osirrox **on**. It can be irrevocably disabled by −osirrox **banned**.
The setting **blocked** is like **off**. But it can only be revoked by setting **unblock**, which elsewise is

like **on**. This can be used to curb command scripts which might use **on** undesiredly.
To enable restoring of special files by **device_files** is potentially dangerous. The meaning of the number st_rdev (see man 2 stat) depends much on the operating system. Best is to restore device files only to the same system from where they were copied. If not enabled, device files in the ISO image are ignored during restore operations.
Due to a bug of previous versions, device files from previous sessions might have been altered to major=0, minor=1. So this combination does not get restored.
Option **concat_split_on** is default. It enables restoring of split file directories as data files if the directory contains a complete collection of −cut_out part files. With option **concat_split_off** such directories are handled like any other ISO image directory.
Option **auto_chmod_off** is default. If **auto_chmod_on** is set then access restrictions for disk directories get circumvented if those directories are owned by the effective user who runs **xorriso**. This happens by temporarily granting rwx permission to the owner.
Option **sort_lba_on** may improve read performance with optical drives. It can restore large numbers of hard links without exhausting −temp_mem_limit. It does not preserve directory mtime and it needs −osirrox option auto_chmod_on in order to extract directories which offer no write permission. Default is **sort_lba_off**.
Option **o_excl_on** is the default unless the program was started with leafname "osirrox". On GNU/Linux it tries to avoid using drives which are mounted or in use by other libburn programs. Option **o_excl_off** on GNU/Linux enables access to such drives by the equivalent of −drive_access "shared:readonly". I.e. drives which get acquired while **o_excl_off** will refuse to get blanked, formatted, written, or ejected. But be aware that even harmless inquiries can spoil ongoing burns of CD−R[W] and DVD−R[W].
Option **strict_acl_off** is default. It tolerates on FreeBSD the presence of directory "default" ACLs in the ISO image. With **strict_acl_on** these GNU/Linux ACLs cause on FreeBSD a FAILURE event during restore with −acl "on".
Option **check_md5_off** disables MD5 checking during copy to disk. The default option **check_md5_on** enables it if −md5 is "on". If a data file with recorded MD5 is copied as a whole to the disk filesystem, then the MD5 of the copied content gets computed and compared with the recorded MD5. A mismatch causes an error message of severity SORRY. Option **check_md5_force** causes an error message if −md5 is "on" but no MD5 is recorded for the data file.
Option **sparse=** controls production of sparse files during extraction of files from the ISO filesystem. Default is **sparse=off**.
A positive number like in **sparse=1m** sets the minimum requirement for the length of a sequence of 0−bytes which shall be represented by a gap. This saves disk space if the disk filesystem supports sparse files. A gap gets created by help of lseek(2) if a sequence of read buffers, which contain only 0−bytes, bears at least the minimum amount of bytes. Expect read buffers to be in the size range of 32k or 64k.
Command −paste_in creates gaps only if the writing begins at or after the end of the existing disk file. So the sequence of −paste_in commands matters. Command −concat does not create sparse files.

−extract iso_rr_path disk_path
Copy the file objects at and underneath iso_rr_path to their corresponding addresses at and underneath disk_path. This is the inverse of −map or −update_r.
If iso_rr_path is a directory and disk_path is an existing directory then both trees will be merged. Directory attributes get extracted only if the disk directory is newly created by the copy operation. Disk files get removed only if they are to be replaced by file objects from the ISO image.
As many attributes as possible are copied together with restored file objects.

−extract_single iso_rr_path disk_path
Like −extract, but if iso_rr_path is a directory then its sub tree gets not restored.

−extract_l iso_rr_prefix disk_prefix iso_rr_path [***]

Perform −extract with each of the iso_rr_path parameters. disk_path will be composed from iso_rr_path by replacing iso_rr_prefix by disk_prefix.

−extract_cut iso_rr_path byte_offset byte_count disk_path

Copy a byte interval from a data file out of an ISO image into a newly created disk file. The main purpose for this is to offer a way of handling large files if they are not supported by mount −t iso9660 or if the target disk filesystem cannot store large files.

If the data bytes of iso_rr_path are stored in the loaded ISO image, and no filter is applied, and byte_offset is a multiple of 2048, then a special run of −check_media is performed. It may be quicker and more rugged than the general reading method.

−cpx iso_rr_path [***] disk_path

Copy single leaf file objects from the ISO image to the address given by disk_path. If more then one iso_rr_path is given then disk_path must be a directory or non−existent. In the latter case it gets created and the extracted files get installed in it with the same leafnames.

Missing directory components in disk_path will get created, if possible.

Directories are allowed as iso_rr_path only with −osirrox "concat_split_on" and only if they actually represent a complete collection of −cut_out split file parts.

−cpax iso_rr_path [***] disk_path

Like −cpx but restoring mtime, atime as in ISO image and trying to set ownership and group as in ISO image.

−cp_rx iso_rr_path [***] disk_path

Like −cpx but also extracting whole directory trees from the ISO image.

The resulting disk paths are determined as with shell command cp −r : If disk_path is an existing directory then the trees will be inserted or merged underneath this directory and will keep their leaf names. The ISO directory "/" has no leaf name and thus gets mapped directly to disk_path.

−cp_rax iso_rr_path [***] disk_path

Like −cp_rx but restoring mtime, atime as in ISO image and trying to set ownership and group as in ISO image.

−paste_in iso_rr_path disk_path byte_offset byte_count

Read the content of a ISO data file and write it into a data file on disk beginning at the byte_offset. Write at most byte_count bytes. This is the inverse of command −cut_out.

−concat mode [target | lim prog [args [...]] lim] iso_rr_path [***]

Copy the data content of one or more data files of the ISO image into a disk file object, into a file descriptor, or start a program and copy the data into its standard input. The latter is subject to the security restrictions for external filters.

Modes **overwrite** and **append** write into the target which is given by the second parameter. This may be the path to a disk file object, or "−" which means standard output, or a text of the form /dev/fd/number, where number is an open file descriptor (e.g. standard error is /dev/fd/2). An existing target file is not removed before writing begins. If it is not able to take content data, then this command fails. Mode overwrite truncates regular data files to 0 size before writing into them. Example:

```
 −concat append /home/me/accumulated_text /my/iso/text −−
```

Mode **pipe** expects as second parameter a delimiter word which shall mark the end of the program argument list. The third argument is the disk_path to the program. It must contain at least one '/'. $PATH is not applied. Further parameters up to the announced delimiter word are used as arguments with the program start. Example:

```
 −iso_rr_pattern on \
 −concat pipe + /usr/bin/wc + "/my/iso/files*" −−
```

The further parameters in all modes are the iso_rr_paths of data files. Their content gets concatenated in the copy.

−extract_boot_images disk_path

Copy boot equipment to disk, which is not necessarily represented as data files in the ISO filesystem. The data get written into various files in a disk directory, which may already exist or of which the parent must exist so that it can get created.
Files may be missing if their corresponding information is not present in the ISO filesystem. Existing files do not get overwritten but rather cause a failure event.
The same data may appear in different files. E.g. the El Torito boot image for EFI is often the same data as the EFI partition in MBR or GPT.
File "eltorito_catalog.img" contains the El Torito Boot Catalog.
Files "eltorito_img*_*.img" contain El Torito Boot images. The first "*" gives the image number, the second "*" gives the type: "bios", "mac", "ppc", "uefi", or a hex number.
File "mbr_code_isohybrid.img" contains the ISOLINUX MBR template.
File "mbr_code_grub2.img" contains the GRUB2 MBR template.
File "systemarea.img" contains the whole 32 KiB of System Area if not all zero.
Files "mbr_part*_efi.img" contain EFI partition images from the MBR partition table. The "*" text part gives the partition number.
Files "mbr_part*_prep.img" contain PReP partition images.
Files "gpt_part*_efi.img" contain EFI partition images from GPT.
Files "gpt_part*_hfsplus.img" contain HFS+ partition images from GPT. To avoid extracting the whole HFS+ aspect of hybrid ISO filesystems, the partition image is extracted only if it has less than half of the size of the ISO filesystem or if the partition is outside the ISO filesystem.

−mount drive entity id path

Produce the same line as −mount_cmd and then execute it as external program run after giving up the depicted drive. See also −mount_opts. This demands −osirrox to be enabled and normally will succeed only for the superuser. For safety reasons the mount program is only executed if it is reachable as /bin/mount or /sbin/mount.

Command compatibility emulations:

Writing of ISO 9660 on CD is traditionally done by program mkisofs as ISO 9660 image producer and cdrecord as burn program. **xorriso** does not stri ve for their comprehensive emulation. Nevertheless it is ready to perform some of its core tasks under control of commands which in said programs trigger comparable actions.

−as personality option [options] --

Perform the variable length option list as sparse emulation of the program depicted by the personality word.

Personality "**mkisofs**" accepts the options listed with:

```
 −as mkisofs −help −−
```

Among them: −R (always on), −r, −J, −o, −M, −C, −dir−mode, −file−mode, −path−list, −m, −exclude−list, −f, −print−size, −pad, −no−pad, −V, −v, −version, −graft−points, −z, −no−emul−boot, −b, −c, −boot−info−table, −boot−load−size, −input−charset, −G, −output−charset, −U, −hide, −hide−joliet, −hide−list, −hide−joliet−list, file paths and pathspecs.
A lot of options are not supported and lead to failure of the mkisofs emulation. Some are ignored, but better do not rely on this tolerance.
The supported options are documented in detail in xorrisofs.info and in man xorrisofs. The description here is focused on the effect of mkisofs emulation in the context of a **xorriso** run.
Other than with the "cdrecord" personality there is no automatic −commit at the end of a "mkisofs" option list. Verbosity settings −v (= "UPDATE") and −quiet (= "SORRY") persist. The output file persists until things happen like −commit, −rollback, −dev, or end of **xorriso**.
Options which affect all file objects in the ISO image, like −r or −dir−mode, will be applied only to files which are present in the ISO image when the command −as ends. If you use several −as mkisofs commands in the same run, then consider to put such options into the last −as command.
If files are added to the image, then −pacifier gets set to "mkisofs" and −stdio_sync is defaulted to "off" if no such setting was made yet.

−graft−points is equivalent to −pathspecs on. Note that pathspecs without "=" are interpreted differently than with **xorriso** command −add. Directories get merged with the root directory of the ISO image, other filetypes get mapped into that root directory.
If pathspecs are given and if no output file was chosen before or during the "mkisofs" option list, then standard output (−outdev "−") will get into effect. If −o points to a regular file, then it will be truncated to 0 bytes when finally writing begins. This truncation does not happen if the drive is chosen by **xorriso** commands before −as mkisofs or after its list delimiter. Directories and symbolic links are no valid −o targets.
Writing to stdout is possible only if −as "mkisofs" was among the start arguments or if other start arguments pointed the output drive to standard output.
−print−size inhibits automatic image production at program end. This ban is lifted only if the pending image changes get discarded.
Padding is counted as part of the ISO image if not option −−emul−toc is given.
If no −iso−level is given, then level 1 is chosen when the first file or directory is added to the image. At the same occasion directory names get allowed to violate the standard by −compliance option allow_dir_id_ext. This may be avoided by option −disallow_dir_id_ext.
Option −root is supported. Option −old−root is implemented by **xorriso** commands −mkdir, −cp_clone, −find update_merge, and −find rm_merge. −root and −old−root set command −disk_dev_ino to "ino_only" and −md5 to "on", by default. −disk_dev_ino can be set to "off" by −−old−root−no−ino or to "on" by −−old−root−devno . −md5 can be set to "off" by −−old−root−no−md5 .
Not original mkisofs options are −−quoted_path_list , −−hardlinks , −−acl , −−xattr , −−md5 , −−stdio_sync . They work like the **xorriso** commands with the same name and hardcoded parameter "on", e.g. −acl "on". Explicit parameters are expected by −−stdio_sync and −−scdbackup_tag.
The capability to preserve multi−session history on overwritable media gets disabled by default. It can be enabled by using −−emul−toc with the first session. See −compliance no_emul_toc.
−−sort−weight gets as parameters a number and an iso_rr_path. The number becomes the LBA sorting weight of regular file iso_rr_path or of all regular files underneath directory iso_rr_path. (See −find −exec sort_weight).
Adopted from grub−mkisofs are −−protective−msdos−label (see −boot_image grub partition_table=on) and −−modification−date=YYYYMMDDhhmmsscc (see −volume_date uuid).
For EFI bootable GRUB boot images use −−efi−boot. It performs −boot_image grub efi_path= surrounded by two −boot_image "any" "next". Alternative option −e from Fedora genisoimage sets bin_path and platform_id for EFI, but performs no "next".
For MBR bootable ISOLINUX images there is −isohybrid−mbr FILE, where FILE is one of the Syslinux files mbr/isohdp[fp]x*.bin . Use this instead of −G to apply the effect of −boot_image isolinux partition_table=on.
−−boot−catalog−hide is −boot_image any cat_hidden=on.
−mips−boot is the same as −boot_image any mips_path= .
−mipsel−boot leads to mipsel_path= .
−partition_offset number is −boot_image any partition_offset=number.
Command −append_partition is supported.
−untranslated_name_len number is −compliance untranslated_name_len=number.
−−old−empty is −compliance old_empty.
The options of genisoimage Jigdo Template Extraction are recognized and performed via **xorriso** command −jigdo. See the "Alias:" names there for the meaning of the genisoimage options.

Personalities "**xorrisofs**", "**genisoimage**", and "**genisofs**" are aliases for "mkisofs".
If **xorriso** is started with one of the leafnames "xorrisofs", "genisofs", "mkisofs", or "genisoimage", then it performs −read_mkisofsrc and prepends −as "genisofs" to the program arguments. I.e. all arguments will be interpreted mkisofs style until "−−" is encountered. From then on, arguments are interpreted as **xorriso** commands.
−−no_rc as first argument of such a program start prevents interpretation of startup files. See

section FILES below.

Personality "**cdrecord**" accepts the options listed with:
 −as cdrecord −help −−
Among them: −v, dev=, speed=, blank=, fs=, −eject, −atip, padsize=, tsize=, −isosize, −multi, −msinfo, −−grow_overwriteable_iso, write_start_address=, track source file path or "−" for standard input as track source.
It ignores most other options of cdrecord and cdrskin but refuses on −audio, −scanbus, and on blanking modes unknown to **xorriso**.
The scope is only a single data track per session to be written to blank, overwritable, or appendable media. The medium gets closed if closing is applicable and not option −multi is present.
If an input drive was acquired, then it is given up. This is only allowed if no image changes are pending.
dev= must be given as **xorriso** device address. Addresses like 0,0,0 or ATA:1,1,0 are not supported.
If a track source is given, then an automatic −commit happens at the end of the "cdrecord" option list.
−−grow_overwriteable_iso enables emulation of multi−session on overwritable media. To enable emulation of a TOC, the first session needs −C 0,32 with −as mkisofs (but no −M) and −−grow_overwriteable_iso write_start_address=32s with −as cdrecord.
A much more elaborate libburn based cdrecord emulator is the program cdrskin.
Personalites "**xorrecord**", "**wodim**", and "**cdrskin**" are aliases for "cdrecord".
If **xorriso** is started with one of the leafnames "xorrecord", "cdrskin", "cdrecord", or "wodim", then it automatically prepends −as "cdrskin" to the program arguments. I.e. all arguments will be interpreted cdrecord style until "−−" is encountered. From then on, arguments are interpreted as **xorriso** commands.
−−no_rc as first argument of such a program start prevents interpretation of **xorriso** startup files. See section FILES below.

−read_mkisofsrc

Try one by one to open for reading:
 ./.mkisofsrc , $MKISOFSRC , $HOME/.mkisofsrc , $(dirname $0)/.mkisofsrc
On success interpret the file content as of man mkisofs CONFIGURATION, and end this command. Do 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 (−application_id) , PUBL (−publisher) , SYSI (−system_id) , VOLI (−volid) , VOLS (−volset_id)
Any other lines will be silently ignored.

−pacifier behavior_code

Control behavior of UPDATE pacifiers during write operations. The following behavior codes are defined:
"xorriso" is the default format:
Writing: sector XXXXX of YYYYYY [fifo active, nn% fill]
"cdrecord" looks like:
X of Y MB written (fifo nn%) [buf mmm%]
"mkisofs"
nn% done, estimate finish Tue Jul 15 20:13:28 2008
The frequency of the messages can be adjusted by
"interval=number"
where number gives the seconds between two messages. Permissible settings are 0.1 to 60.0.

−scdbackup_tag list_path record_name

Set the parameter "name" for a scdbackup checksum record. It will be appended in an scdbackup checksum tag to the −md5 session tag if the image starts at LBA 0. This is the case if it gets

written as first session onto a sequential medium, or piped into a program, named pipe or character device.
If list_path is not empty then the record will also be appended to the data file given by this path.
Program scdbackup_verify will recognize and verify tag and file record.
An empty record_name disables this feature.

Scripting, dialog and program control features:

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

−options_from_file fileaddress
Read quoted input from fileaddress and execute it like dialog lines. Empty lines and lines which begin by # are ignored. Normally one line should hold one **xorriso** command and all its parameters. Nevertheless lines may be concatenated by a trailing backslash.
See also section "Command processing", paragraph "Quoted input".

−help
Print helptext.

−version
Print program name and version, component versions, license.

−list_extras code
Tell whether certain extra features were enabled at compile time. Code "all" lists all features and a headline. Other codes pick a single feature. Code "codes" lists them. They share names with related commands (see also there):
"acl" tells whether xorriso has an adapter for local filesystems ACLs.
"xattr" tells whether xorriso has an adapter for local filesystems EA.
"jigdo" tells whether production of Jigdo files is possible.
"zisofs" tells whether zisofs and built−in gzip filters are enabled.
"external_filter" tells whether external filter processes are allowed and whether they are allowed if real user id and effective user id differ.
"dvd_obs" tells whether 64 kB output to DVD media is default.
"use_readline" tells whether readline may be enabled in dialog mode.

−history textline
Copy textline into libreadline history.

−status mode|filter
Print the current settings of **xorriso**. Modes:

```
  short... print only important or altered settings
  long ... print all settings including defaults
  long_history  like long plus history lines
```

Filters begin with '−' and are compared literally against the output lines of −status:long_history. A line is put out only if its start matches the filter text. No wildcards.

−status_history_max number
Set maximum number of history lines to be reported with −status "long_history".

−list_delimiter word
Set the list delimiter to be used instead of "−−". It has to be a single word, must not be empty, not longer than 80 characters, and must not contain quotation marks.
For brevity the list delimiter is referred as "−−" throughout this text.

−sh_style_result "on"|"off"
Make the result output of some filesystem inspection commands look more like the output of equivalent shell commands. The most important effect is to prevent the wrapping of file addresses into quotation marks with commands

```
  −pwd −pwdx −ls −lsd −lsl −lsdl −lsx −lsdx −lslx −lsdlx
```

−du −dus −dux −dusx −findx −find
This will make ambiguous the representation of file names which contain newline characters. On the other hand it should facilitate integration of xorriso into shell scripts which already use the corresponding shell commands.

−backslash_codes "on"|"off"|mode[:mode]
Enable or disable the interpretation of symbolic representations of special characters with quoted input, or with program arguments, or with program text output. If enabled the following translations apply:
\a=bell(007) \b=backspace(010) \e=Escape(033) \f=formfeed(014)
\n=linefeed(012) \r=carriage_return(015) \t=tab(011)
\v=vtab(013) \\=backslash(134) \[0−7][0−7][0−7]=octal_code
\x[0−9a−f][0−9a−f]=hex_code \cC=control−C
Translations can occur with quoted input in 3 modes:
"in_double_quotes" translates only inside " quotation.
"in_quotes" translates inside " and ’ quotation.
"with_quoted_input" translates inside and outside quotes.
With the start program arguments there is mode:
"with_program_arguments" translates program arguments.
Mode "encode_output" encodes output characters. It combines "encode_results" with "encode_infos". Inside single or double quotation marks encoding applies to 8−bit characters octal 001 to 037 , 177 to 377 and to backslash(134). Outside quotation marks some harmless ASCII control characters stay unencoded: bell(007), backspace(010), tab(011), linefeed(012), formfeed(014), carriage_return(015).
Mode "off" is default and disables any translation. Mode "on" is "with_quoted_input:with_program_arguments:encode_output".

−temp_mem_limit number["k"|"m"]
Set the maximum size of temporary memory to be used for image dependent buffering. Currently this applies to pattern expansion, LBA sorting, restoring of hard links.
Default is 16m = 16 MiB, minimum 64k = 64 kiB, maximum 1024m = 1 GiB.

−print text
Print a text line to the result channel which is by default stdout.

−print_info text
Print a text line to the info channel which is by default stderr.

−print_mark text
Print a text line to the mark channel which is by default directed to both, result and info channel. An empty text will cause no output at all.

−prompt text
Show text at beginning of output line and wait for the user to hit the Enter key or to send a line via stdin.

−sleep seconds
Wait for the given number of seconds before performing the next command. Expect coarse granularity no better than 1/100 seconds.

−errfile_log mode path|channel
If problem events are related to input files from the filesystem, then their disk_paths can be logged to a file or to output channels R or I.
Mode can either be "plain" or "marked". The latter causes marker lines which give the time of log start, burn session start, burn session end, log end or program end. In mode "plain", only the file paths are logged.
If path is "−" or "−R" then the log is directed to the result channel. Path "−I" directs it to the info message channel. Any text that does not begin with "−" is used as path for a file to append the log lines.

Problematic files can be recorded multiple times during one program run. If the program run aborts then the list might not be complete because some input files might not have been processed at all.
The errfile paths are transported as messages of very low severity "ERRFILE". This transport becomes visible with −report_about "ALL".

−session_log path
If path is not empty it gives the address of a plain text file where a log record gets appended after each session. This log can be used to determine the start_lba of a session for mount options −o sbsector= (on GNU/Linux) or −s (on FreeBSD) from date or volume ID.
Record format is: timestamp start_lba size volume−id
The first three items are single words, the rest of the line is the volume ID.

−scsi_log "on"|"off"
Mode "on" enables very verbose logging of SCSI commands and drive replies. Logging messages get printed to stderr, not to any of the **xorriso** output channels.
A special property of this command is that the first −scsi_log setting among the start arguments is in effect already when the first operations of **xorriso** begin. Only "−scsi_log" with dash "−" is recognized that way.

−end
End program after writing pending changes.

−rollback_end
Discard pending changes. End program immediately.

any text
Only in dialog or file execution mode, and only as first non−whitespace in line: Do not execute the line but store it in readline history.

Support for frontend programs via stdin and stdout:

−pkt_output "on"|"off"
Consolidate text output on stdout and classify each line by a channel indicator:
 ’R:’ for result lines,
 ’I:’ for notes and error messages,
 ’M:’ for −mark texts.
Next is a decimal number of which only bit 0 has a meaning for now. 0 means no newline at end of payload, 1 means that the newline character at the end of the output line belongs to the payload. After another colon and a blank follows the payload text.
Example:
 I:1: enter option and parameters :

−logfile channel fileaddress
Copy output of a channel to the given file. Channel may be one of: "." for all channels, "I" for info messages, "R" for result lines, "M" for −mark texts.

−mark text
If text is not empty it will get put out on "M" channel each time **xorriso** is ready for the next dialog line or before **xorriso** performs a command that was entered to the pager prompt.

−msg_op opcode parameter_text
This command shall facilitate extraction of particular information from the message output of other commands. It gives access to the C API function Xorriso_parse_line() and to the message sieve that is provided by the C API. Please refer to their descriptions in file xorriso.h. Further it helps to interpret the severity codes of info messages.
Intended users are frontend programs which operate xorriso in dialog mode.
The result output of this command is not caught by the message sieve.
The following opcodes are defined:
start_sieve

Install the message sieve as of Xorriso_sieve_big() and start watching program messages. The parameter_text has no meaning.
show_sieve
Show a list of filter rule names. The parameter_text has no meaning. The list begins by a line with the return value of Xorriso_sieve_get_result() with flag bit3. If this value is larger than 0, then the next line tells the number of names. The following lines show one name each.
read_sieve
Use the parameter_text as name of a filter rule and inquire its next recorded result. See Xorriso_sieve_big() for a list of names and reply strings.
The recorded strings are put out on result channel. They get wrapped into lines which tell their structure. The first line tells the return value of Xorriso_sieve_get_result(). The next line tells the number of strings. Each string begins by a line that tells the number of lines of the string. Then follow these lines. They are to be concatenated with a newline character between each of them. Finally the number of still available recorded results of the given name is put out.
clear_sieve
Dispose all recorded strings and continue watching program messages. The parameter_text has no meaning.
end_sieve
Dispose the sieve with its filter rules and stop watching program messages. The parameter_text has no meaning.
parse
Read a text from dialog input and submit it to Xorriso_parse_line(). The parameter_text word shall consist of several words separated by blanks. It will be necessary to use both kinds of quotation marks.
E.g. "'ISO session :' '' 0 0 1"
The five parameter words are: prefix, separators, max_words, flag, number_of_input_lines. The former four are handed over to Xorriso_parse_line(). The number of input lines minus one tells xorriso how many newline characters are part of the input text.
The announced number of text lines will be read from dialog input, concatenated with a newline character between each of them, and submitted to Xorriso_parse_line() as parameter line. Note that newlines outside of quotation marks are interpreted as separators if the separators parameter is empty.
The parsed strings are put out on result channel. They get wrapped into lines which tell their structure. The first line tells the return value of Xorriso_parse_line(). The next line tells the number of strings. Each string begins by a line that tells the number of lines of the string. Then follow these lines. They are to be concatenated with a newline character between each of them.
If −backslash_codes "encode_output" is enabled, then the strings undergo encoding as if they were enclosed in quotes. Escpecially each string will be put out as a single result line.
parse_bulk
Like "parse", but with the fifth parameter word being number_of_input_texts rather than number_of_input_lines. Each input text has to be preceded by a line that tells number_of_input_lines as with "parse". Then come the announced number of text lines.
All input texts will be read before printing of result lines begins. This consumes memory in xorriso. So the number_of_input_texts should not be extremely high. On the other hand, large transactions of command, input texts, and results are desirable if connection latency is an issue.
parse_silently
Like "parse" but not issuing a prompting message. Confusing to humans.
parse_bulk_silently
Like "parse_bulk" but not issuing a prompting message. Confusing to humans.
compare_sev
The parameter_text should contain two comma separated severity texts as issued by this program. Like "SORRY,UPDATE". See also paragraph "Exception processing".
These two severity texts get compared and a number gets printed to the result channel. This number is 0 if both severities are equal. It is −1 if the first severity is lower than the second one. It

is 1 is the first severity is higher than the second one.
Above example "SORRY,UPDATE" will yield 1.
list_sev
Print to the result channel a blank separated list of all severity names. Sorted from low to high severity.

−named_pipe_loop mode[:mode] disk_path_stdin disk_path_stdout disk_path_stderr
Temporarily replace standard input, standard output and standard error by named pipes. Enter dialog mode without readline.
Defined modes are:
"cleanup" removes the submitted pipe files when the loop ends.
"keep" does not delete them. This is the default.
"buffered" reads all lines from the input pipe until EOF before it opens the output pipes and processes the input lines.
"direct" opens the output pipes after the first input line was read. Each line is executed directly after it is read. This is the default.
The other three parameters must either be disk paths to existing named pipes, or be "−" to leave the according standard i/o channel unreplaced.
xorriso will open the stdin pipe, read and execute dialog lines from it until the sender closes the pipe. The output pipes get opened depending on mode "buffered" or "direct". After all lines are executed, xorriso will close its side of the pipes and enter a new cycle of opening, reading and executing.
If an input line consists only of the word "end_named_pipe_loop" then −named_pipe_loop will end and further xorriso commands may be executed from other sources.

−launch_frontend program [arguments ...] --
Start the program that is given as first parameter. Submit the other parameters as program arguments. Enable xorriso dialog mode.
Two nameless pipe objects are created. xorriso standard input gets connected to the standard output of the started program. xorriso standard output and standard error get connected to the standard input of that program.
xorriso will abort when the started program ends or if it cannot be started at all. In both cases it will return a non−zero exit value. The exit value will be zero if the frontend sends −end or −rollback_end before ending itself.
This command may be totaly banned at compile time. It is banned by default if xorriso runs under setuid permissions.
The program name will not be searched in the $PATH directories. To make this clear, it must contain at least one /−character. Best is an absolute path.
Example:

```
  xorriso −launch_frontend "$(which xorriso−tcltk)" −stdio −−
```

The frontend program should first send via its standard output:

```
  −mark 0 −pkt_output on −msg_op start_sieve − −reassure off
```

It should be ready to decode −pkt_output and to react on −mark messages. Best is to increment the −mark number after each sent command sequence and then to wait for the new number to show up in a mark message:

```
  ...some...commands... −mark <incremented_number>
```

Further are advised:

```
  −report_about UPDATE −abort_on NEVER
  −iso_rr_pattern off −disk_pattern off
```

A check of the xorriso version should be done, in order to make sure that all desired features are present.
Command −launch_frontend will only work once per xorriso run. If no command parameters are submitted or if program is an empty text, then no program will be started but nevertheless −launch_frontend will be irrevocably disabled.

−prog text
Use text as name of this program in subsequent messages

−prog_help text
Use text as name of this program and perform −help.

EXAMPLES

Overview of examples:

As superuser learn about available drives
Blank medium and compose a new ISO image as batch run
A dialog session doing about the same
Manipulate an existing ISO image on the same medium
Copy modified ISO image from one medium to another
Bring a prepared ISOLINUX tree onto medium and make it bootable
Change existing file name tree from ISO-8859-1 to UTF-8
Operate on storage facilities other than optical drives
Burn an existing ISO image file to medium
Perform multi-session runs as of cdrtools traditions
Let xorriso work underneath growisofs
Adjust thresholds for verbosity, exit value and program abort
Examples of input timestrings
Incremental backup of a few directory trees
Restore directory trees from a particular ISO session to disk
Try to retrieve blocks from a damaged medium

As superuser learn about available drives

On Linux, FreeBSD or NetBSD consider to give rw−permissions to those users or groups which shall be able to use the drives with **xorriso**. On Solaris use pfexec. Consider to restrict privileges of **xorriso** to "base,sys_devices" and to give r−permission to user or group.

```
$ xorriso −device_links
1  −dev '/dev/cdrom1' rwrw−− :  'TSSTcorp' 'DVD−ROM SH−D162C
1  −dev '/dev/cdrw'   rwrw−− :  'TSSTcorp' 'CDDVDW SH−S223B'
2  −dev '/dev/cdrw3'  rwrw−− :  'HL−DT−ST' 'BDDVDRW_GGC−H20L'
```

Blank medium and compose a new ISO image as batch run

Acquire drive /dev/sr2, make medium ready for writing a new image, fill the image with the files from hard disk directories /home/me/sounds and /home/me/pictures.
Because no −dialog "on" is given, the program will then end by writing the session to the medium.

```
$ xorriso −outdev /dev/sr2 \
 −blank as_needed \
 −map /home/me/sounds /sounds \
 −map /home/me/pictures /pictures
```

The ISO image may be shaped in a more elaborate way like the following: Omit some unwanted stuff by removing it from the image directory tree. Reintroduce some wanted stuff.

```
$ cd /home/me
$ xorriso −outdev /dev/sr2 \
 −blank as_needed \
 −map /home/me/sounds /sounds \
 −map /home/me/pictures /pictures \
 −rm_r \
   /sounds/indecent \
   '/pictures/*private*' \
   /pictures/confidential \
   −− \
 −cd / \
 −add pictures/confidential/work* −−
```

Note that '/pictures/*private*' is a pattern for iso_rr_paths while pictures/confidential/work* gets expanded by the shell with addresses from the hard disk. Commands −add and −map have different parameter rules but finally the same effect: they put files into the image.

A dialog session doing about the same

Some settings are already given as start argument. The other activities are done as dialog input. The pager gets set to 20 lines of 80 characters.
The drive is acquired by command −dev rather than −outdev in order to see the message about its current content. By command −blank this content is made ready for being overwritten and the loaded ISO image is made empty.
In order to be able to eject the medium, the session needs to be committed explicitly.

```
$ xorriso -dialog on -page 20 80 -disk_pattern on
enter option and arguments :
−dev /dev/sr2
enter option and arguments :
−blank as_needed
enter option and arguments :
−map /home/me/sounds /sounds -map /home/me/pictures /pictures
enter option and arguments :
−rm_r /sounds/indecent /pictures/*private* /pictures/confidential
enter option and arguments :
−cdx /home/me/pictures -cd /pictures
enter option and arguments :
−add confidential/office confidential/factory
enter option and arguments :
−du /
enter option and arguments :
−commit_eject all -end
```

Manipulate an existing ISO image on the same medium

Load image from drive. Remove (i.e. hide) directory /sounds and its subordinates. Rename directory /pictures/confidential to /pictures/restricted. Change access permissions of directory /pictures/restricted. Add new directory trees /sounds and /movies. Burn to the same medium, check whether the tree can be loaded, and eject.

```
$ xorriso −dev /dev/sr2 \
 −rm_r /sounds −− \
 −mv \
   /pictures/confidential \
   /pictures/restricted \
   −− \
 −chmod go−rwx /pictures/restricted −− \
 −map /home/me/prepared_for_dvd/sounds_dummy /sounds \
 −map /home/me/prepared_for_dvd/movies /movies \
 −commit −eject all
```

Copy modified ISO image from one medium to another

Load image from input drive. Do the same manipulations as in the previous example. Acquire output drive and blank it. Burn the modified image as first and only session to the output drive.

```
$ xorriso −indev /dev/sr2 \
 −rm_r /sounds −− \
 ...
 −outdev /dev/sr0 −blank as_needed \
 −commit −eject all
```

Bring a prepared ISOLINUX tree onto medium and make it bootable

The user has already created a suitable file tree on disk and copied the ISOLINUX files into subdirectory ./boot/isolinux of that tree. Now **xorriso** can burn an El Torito bootable medium:

```
$ xorriso −outdev /dev/sr0 −blank as_needed \
  −map /home/me/ISOLINUX_prepared_tree / \
  −boot_image isolinux dir=/boot/isolinux
```

Change existing file name tree from ISO-8859-1 to UTF-8

This example assumes that the existing ISO image was written with character set ISO−8859−1 but that the readers expected UTF−8. Now a new session gets added with converted file names. Command −changes_pending "yes" enables writing despite the lack of any manipulation command.
In order to avoid any weaknesses of the local character set, this command pretends that it uses already the final target set UTF−8. Therefore strange file names may appear in messages, which will be made terminal−safe by command −backslash_codes.

```
$ xorriso −in_charset ISO−8859−1 −local_charset UTF−8 \
  −out_charset UTF−8 −backslash_codes on −dev /dev/sr0 \
  −changes_pending yes −commit −eject all
```

Operate on storage facilities other than optical drives

Full read−write operation is possible with regular files and block devices:

```
$ xorriso −dev /tmp/regular_file ...
```

Paths underneath /dev normally need prefix "stdio:"

```
$ xorriso −dev stdio:/dev/sdb ...
```

If /dev/sdb is to be used frequently and /dev/sda is the system disk, then consider to place the following lines in a **xorriso** Startup File. They allow you to use /dev/sdb without prefix and protect disk /dev/sda from **xorriso**:

```
 −drive_class banned   /dev/sda*
 −drive_class harmless /dev/sdb
```

Other writeable file types are supported write−only:

```
$ xorriso −outdev /tmp/named_pipe ...
```

Among the write−only drives is standard output:

```
$ xorriso −outdev − \
 ...
 | gzip >image.iso.gz
```

Burn an existing ISO image file to medium

Actually this works with any kind of data, not only ISO images:

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

Perform multi-session runs as of cdrtools traditions

Between both processes there can be performed arbitrary transportation or filtering.
The first session is written like this:

```
$ xorriso −as mkisofs 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
$ xorriso −as mkisofs −M /dev/sr0 −C $m 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.

This example works for multi−session media only. Add cdrskin option −−grow_overwriteable_iso to all −as cdrecord runs in order to enable multi−session emulation on overwritable media.

Let xorriso work underneath growisofs

growisofs expects an ISO formatter program which understands options −C and −M. If **xorriso** gets started by name "xorrisofs" then it is suitable for that.

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

growisofs has excellent burn capabilities with DVD and BD. It does not emulate session history on overwritable media, though.

Adjust thresholds for verbosity, exit value and program abort

Be quite verbose, exit 32 if severity "FAILURE" was encountered, do not abort prematurely but forcibly go on until the end of commands.

```
$ xorriso ... \
 −report_about UPDATE \
 −return_with FAILURE 32 \
 −abort_on NEVER \
 ...
```

Examples of input timestrings

As printed by program date: **’Thu Nov 8 14:51:13 CET 2007’**
The same without ignored parts: **’Nov 8 14:51:13 2007’**
The same as expected by date: **110814512007.13**
Four weeks in the future: **+4w**
The current time: **+0**
Three hours ago: **−3h**
Seconds since Jan 1 1970: **=1194531416**

Incremental backup of a few directory trees

This changes the directory trees /projects 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. Accelerated comparison is enabled at the expense of potentially larger backup size. Only media with the expected volume ID or blank media are accepted. Files with names matching *.o or *.swp get excluded explicitly.

When done with writing the new session gets checked by its recorded MD5.

```
$ xorriso \
 −abort_on FATAL \
 −for_backup −disk_dev_ino on \
 −assert_volid ’PROJECTS_MAIL_*’ FATAL \
 −dev /dev/sr0 \
 −volid PROJECTS_MAIL_"$(date ’+%Y_%m_%d_%H%M%S’)" \
 −not_leaf ’*.o’ −not_leaf ’*.swp’ \
 −update_r /home/thomas/projects /projects \
 −update_r /home/thomas/personal_mail /personal_mail \
 −commit −toc −check_md5 FAILURE −− −eject all
```

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 the run fails gracefully due to lack of remaining space on the old one.
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. To apply zisofs compression to those data files which get newly copied from the local filesystem, insert these commands immediately before −commit :

```
 −hardlinks perform_update \
 −find / −type f −pending_data −exec set_filter −−zisofs −− \
```

Commands −disk_dev_ino and −for_backup depend on stable device and inode numbers on disk. Without them, an update run may use −md5 "on" to match recorded MD5 sums against the current file content on hard disk. This is usually much faster than the default which compares both contents directly.
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** command −toc. Used after −commit the following command prints the matching mount command for the newly written session (here for mount point /mnt):

```
 −mount_cmd "indev" "auto" "auto" /mnt
```

Commands −mount_cmd and −mount are also able to produce the mount commands for older sessions in the table−of−content. E.g. as superuser:

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

Above example produces a result similar to −root / −old−root / with mkisofs. For getting the session trees accumulated in the new sessions, let all −update commands use a common parent directory and clone it after updating is done:

```
 −update_r /home/thomas/projects /current/projects \
 −update_r /home/thomas/personal_mail /current/personal_mail \
 −clone /current /"$(date ’+%Y_%m_%d_%H%M%S’)" \
```

The cloned tree will have a name like /2011_02_12_155700.

Sessions on multi−session media are separated by several MB of unused blocks. So with small sessions the payload capacity can become substantially lower than the overall media capacity. If the remaining space on a medium does not suffice for the next gap, the drive is supposed to close the medium automatically.

Better do not use your youngest backup for −update_r. 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.

Restore directory trees from a particular ISO session to disk

This is an alternative to mounting the medium and using normal file operations.
First check which backup sessions are on the medium:

```
$ xorriso −outdev /dev/sr0 −toc
```

Then enable restoring of ACL, xattr and hard links. Load the desired session and copy the file trees to disk. Avoid to create /home/thomas/restored without rwx−permission.

```
$ xorriso −for_backup \
 −load volid ’PROJECTS_MAIL_2008_06_19*’ \
 −indev /dev/sr0 \
 −osirrox on:auto_chmod_on \
 −chmod u+rwx / −− \
 −extract /projects /home/thomas/restored/projects \
 −extract /personal_mail /home/thomas/restored/personal_mail \
 −rollback_end
```

The final command −rollback_end prevents an error message about the altered image being discarded.

Try to retrieve blocks from a damaged medium

```
$ xorriso −abort_on NEVER −indev /dev/sr0 \
 −check_media time_limit=1800 report=blocks_files \
 data_to="$HOME"/dvd_copy sector_map="$HOME"/dvd_copy.map −−
```

This can be repeated several times, if necessary with −eject or with other −indev drives. See the human readable part of "$HOME"/dvd_copy.map for addresses which can be used on "$HOME"/dvd_copy with mount option −o sbsector= or −s.

FILES

Program alias names:

Normal installation of **xorriso** creates three links or copies which by their program name pre−select certain settings:
xorrisofs starts **xorriso** with −as mkisofs emulation.
xorrecord starts **xorriso** with −as cdrecord emulation.
osirrox starts with −osirrox "on:o_excl_off" which allows further commands to copy files from ISO image to disk and to apply command −mount to one or more of the existing ISO sessions.

Startup files:

If not −no_rc is given as the first argument then **xorriso** 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 above, but none of them is required to exist. The line format is described with command −options_from_file.
If mkisofs emulation was enabled by program name "xorrisofs", "mkisofs", "genisoimage", or "genisofs", then afterwards −read_mkisofsrc is performed, which reads .mkisofsrc files. See there.

Runtime control files:

The default setting of −check_media abort_file= is:

```
/var/opt/xorriso/do_abort_check_media
```

ENVIRONMENT

The following environment variables influence the program behavior:
HOME is used to find startup files of xorriso and mkisofs.
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 −volume date "uuid", sets −boot_image "any" "gpt_disk_guid=" to "volume_date_uuid", −volume_date "all_file_dates" to "set_to_mtime", and −iso_nowtime to "=$SOURCE_DATE_EPOCH".
Startup files and program options can override the effect of SOURCE_DATE_EPOCH.

SEE ALSO

For the mkisofs emulation of xorriso
xorrisofs(1)

For the cdrecord emulation of xorriso
xorrecord(1)

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

Libreadline, a comfortable input line facility
readline(3)

Other programs which produce ISO 9660 images
mkisofs(8), genisoimage(1)

Other programs which burn sessions to optical media
growisofs(1), cdrecord(1), wodim(1), cdrskin(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) 2007 − 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

xorriso 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. Thanks to Andy Polyakov who invented emulated growing, to Derek Foreman and Ben Jansens who once founded libburn.
Compliments towards Joerg Schilling whose cdrtools served me for ten years.