NAME

gio - GIO commandline tool

SYNOPSIS

gio help [*COMMAND*]

gio version

gio cat *LOCATION*...

gio copy [*OPTION*...] *SOURCE*... *DESTINATION*

gio info [*OPTION*...] *LOCATION*...

gio launch *DESKTOP-FILE* [*FILE-ARG*...]

gio list [*OPTION*...] [*LOCATION*...]

gio mime *MIMETYPE* [*HANDLER*]

gio mkdir [*OPTION*...] *LOCATION*...

gio monitor [*OPTION*...] [*LOCATION*...]

gio mount [*OPTION*...] [*LOCATION*...]

gio move [*OPTION*...] *SOURCE*... *DESTINATION*

gio open *LOCATION*...

gio rename *LOCATION NAME*

gio remove [*OPTION*...] *LOCATION*...

gio save [*OPTION*...] *DESTINATION*

gio set [*OPTION*...] *LOCATION ATTRIBUTE VALUE*...

gio trash [*OPTION*...] [*LOCATION*...]

gio tree [*OPTION*...] [*LOCATION*...]

DESCRIPTION

gio is a utility that makes many of the GIO features available from the commandline. In doing so, it provides commands that are similar to traditional utilities, but let you use GIO locations instead of local files: for example you can use something like smb://server/resource/file.txt as a location.

Plain filenames which contain a colon will be interpreted as URIs with an unknown protocol. To avoid this, prefix them with a path such as ./, or with the file: protocol.

COMMANDS

help [*COMMAND*]

Displays a short synopsis of the available commands or provides detailed help on a specific command.

version

Prints the GLib version to which **gio** belongs.

cat *LOCATION*...

Concatenates the given files and prints them to the standard output.

The **cat** command works just like the traditional **cat** utility.

Note: just pipe through **cat** if you need its formatting options like **-n**, **-T** or other.

copy [*OPTION*...] *SOURCE... DESTINATION*

Copies one or more files from *SOURCE* to *DESTINATION*. If more than one source is specified, the destination must be a directory.

The **copy** command is similar to the traditional **cp** utility.

Options

-T, **--no-target-directory**

Don’t copy into *DESTINATION* even if it is a directory.

-p, **--progress**

Show progress.

-i, **--interactive**

Prompt for confirmation before overwriting files.

--preserve
Preserve all attributes of copied files.

-b, **--backup**
Create backups of existing destination files.

-P, **--no-dereference**
Never follow symbolic links.

--default-permissions
Use the default permissions of the current process for the destination file, rather than copying the permissions of the source file.

info [*OPTION*...] *LOCATION*...
Shows information about the given locations.

The **info** command is similar to the traditional **ls** utility.

Options

-w, **--query-writable**
List writable attributes.

-f, **--filesystem**
Show information about the filesystem that the given locations reside on.

-a --attributes=*ATTRIBUTES*
The attributes to get.

Attributes can be specified with their GIO name, e.g. standard::icon, or just by namespace, e.g. unix, or by *, which matches all attributes. Several attributes or groups of attributes can be specified, separated by comma.

By default, all attributes are listed.

-n, **--nofollow-symlinks**
Don’t follow symbolic links.

launch *DESKTOP-FILE* [*FILE-ARG*...]

Launch a desktop file from any location given.

The **launch** command extends the behavior of the **open** command by allowing any desktop file to be launched, not only those registered as file handlers.

list [*OPTION*...] [*LOCATION*...]

Lists the contents of the given locations. If no location is given, the contents of the current directory are shown.

The **list** command is similar to the traditional **ls** utility.

Options

-a --attributes=*ATTRIBUTES*

The attributes to get.

Attributes can be specified with their GIO name, e.g. standard::icon, or just by namespace, e.g. unix, or by *, which matches all attributes. Several attributes or groups of attributes can be specified, separated by comma.

By default, all attributes are listed.

-h, **--hidden**

Show hidden files.

-l, **--long**

Use a long listing format.

-n, **--nofollow-symlinks**

Don’t follow symbolic links.

-d, **--print-display-names**

Print display names.

-u, **--print-uris**

Print full URIs.

mime *MIMETYPE* [*HANDLER*]

If no handler is given, the **mime** command lists the registered and recommended applications for

the mimetype. If a handler is given, it is set as the default handler for the mimetype.

Handlers must be specified by their desktop file name, including the extension. Example: org.gnome.gedit.desktop.

mkdir [*OPTION*...] *LOCATION*...
Creates directories.

The **mkdir** command is similar to the traditional **mkdir** utility.

Options

-p, **--parent**
Create parent directories when necessary.

monitor [*OPTION*...] [*LOCATION*...]
Monitors files or directories for changes, such as creation deletion, content and attribute changes, and mount and unmount operations affecting the monitored locations.

The **monitor** command uses the GIO file monitoring APIs to do its job. GIO has different implementations for different platforms. The most common implementation on Linux uses inotify.

Options

-d, **--dir=***LOCATION*
Monitor the given location as a directory. Normally, the file type is used to determine whether to monitor a file or directory.

-f, **--file=***LOCATION*
Monitor the given location as a file. Normally, the file type is used to determine whether to monitor a file or directory.

-D, **--direct=***LOCATION*
Monitor the file directly. This allows changes made via hardlinks to be captured.

-s, **--silent=***LOCATION*
Monitor the file directly, but don’t report changes.

-n, **--no-moves**
Report moves and renames as simple deleted/created events.

-m, **--mounts**
Watch for mount events.

mount [*OPTION*...] [*LOCATION*...]
Provides commandline access to various aspects of GIO’s mounting functionality.

Mounting refers to the traditional concept of arranging multiple file systems and devices in a single tree, rooted at /. Classical mounting happens in the kernel and is controlled by the mount utility. GIO expands this concept by introducing mount daemons that can make file systems available to GIO applications without kernel involvement.

GIO mounts can require authentication, and the **mount** command may ask for user IDs, passwords, and so on, when required.

Options

-m, **--mountable**
Mount as mountable.

-d, **--device=***ID*
Mount volume with device file, or other identifier.

-u, **--unmount**
Unmount the location.

-e, **--eject**
Eject the location.

-t, **--stop=***DEVICE*
Stop drive with device file.

-s, **--unmount-scheme=***SCHEME*
Unmount all mounts with the given scheme.

-f, **--force**
Ignore outstanding file operations when unmounting or ejecting.

-a, **--anonymous**
Use an anonymous user when authenticating.

-l, **--list**
List all GIO mounts.

-o, **--monitor**
Monitor mount-related events.

-i, **--detail**
Show extra information.

--tcrypt-pim
The numeric PIM when unlocking a VeraCrypt volume.

--tcrypt-hidden
Mount a TCRYPT hidden volume.

--tcrypt-system
Mount a TCRYPT system volume.

move [*OPTION*...] *SOURCE*... *DESTINATION*
Moves one or more files from *SOURCE* to *DESTINATION*. If more than one source is specified, the destination must be a directory.

The **move** command is similar to the traditional **mv** utility.

Options

-T, **--no-target-directory**
Don’t copy into *DESTINATION* even if it is a directory.

-p, **--progress**
Show progress.

-i, **--interactive**
Prompt for confirmation before overwriting files.

-b, **--backup**
Create backups of existing destination files.

-C, **--no-copy-fallback**
Don’t use copy and delete fallback.

open *LOCATION*...

Opens files with the default application that is registered to handle files of this type.

GIO obtains this information from the shared-mime-info database, with per-user overrides stored in **$XDG_DATA_HOME**/applications/mimeapps.list.

The **mime** command can be used to change the default handler for a mimetype.

Environment variables will not be set on the application, as it may be an existing process which is activated to handle the new file.

rename *LOCATION NAME*

Renames a file.

The **rename** command is similar to the traditional **rename** utility.

remove [*OPTION*...] *LOCATION*...

Deletes each given file.

This command removes files irreversibly. If you want a reversible way to remove files, see the **trash** command.

Note that not all URI schemes that are supported by GIO may allow deletion of files.

The **remove** command is similar to the traditional **rm** utility.

Options

-f, **--force**

Ignore non-existent and non-deletable files.

save [*OPTION*...] *DESTINATION*

Reads from standard input and saves the data to the given location.

This is similar to just redirecting output to a file using traditional shell syntax, but the **save** command allows saving to location that GIO can write to.

Options

-b, **--backup**

Back up existing destination files.

-c, **--create**
Only create the destination if it doesn’t exist yet.

-a, **--append**
Append to the end of the file.

-p, **--private**
When creating, restrict access to the current user.

-u, **--unlink**
When replacing, replace as if the destination did not exist.

-v, **--print-etag**
Print the new ETag in the end.

-e, **--etag=***ETAG*
The ETag of the file that is overwritten.

set *LOCATION ATTRIBUTE VALUE...*
Sets a file attribute on a file.

File attributes can be specified with their GIO name, e.g standard::icon. Note that not all GIO file attributes are writable. Use the **--query-writable** option of the **info** command to list writable file attributes.

If the *TYPE* is unset, *VALUE* does not have to be specified. If the *TYPE* is stringv, multiple values can be given.

Options

-t, **--type=***TYPE*
Specifies the type of the attribute. Supported types are string, stringv, bytestring, boolean, uint32, int32, uint64, int64 and unset.

If the type is not specified, string is assumed.

-d, **--delete**
Unsets an attribute (same as setting it’s type to unset).

-n, **--nofollow-symlinks**
Don’t follow symbolic links.

trash [*OPTION*...] [*LOCATION*...]
Sends files or directories to the ‘Trashcan’ or restore them from ‘Trashcan’. This can be a different folder depending on where the file is located, and not all file systems support this concept. In the common case that the file lives inside a user’s home directory, the trash folder is **$XDG_DATA_HOME**/Trash.

Note that moving files to the trash does not free up space on the file system until the ‘Trashcan’ is emptied. If you are interested in deleting a file irreversibly, see the **remove** command.

Inspecting and emptying the ‘Trashcan’ is normally supported by graphical file managers such as Nautilus, but you can also see the trash with the command: **gio trash --list** or **gio list trash://**.

Options

-f, **--force**
Ignore non-existent and non-deletable files.

--empty
Empty the trash.

--list
List files in the trash with their original locations

--restore
Restore a file from trash to its original location. A URI beginning with trash:// is expected here. If the original directory doesn’t exist, it will be recreated.

tree [*OPTION*...] [*LOCATION*...]
Lists the contents of the given locations recursively, in a tree-like format. If no location is given, it defaults to the current directory.

The **tree** command is similar to the traditional **tree** utility.

Options

-h, **--hidden**
Show hidden files.

-l, **--follow-symlinks**
Follow symbolic links.

EXIT STATUS

On success 0 is returned, a non-zero failure code otherwise.

SEE ALSO

cat(1), **cp**(1), **ls**(1), **mkdir**(1), **mv**(1), **rm**(1), **tree**(1).