The gPhoto2 Manual

by The gPhoto2 Team, Tim Waugh, and Hans Ulrich Niedermann

The gPhoto2 Manual

by The gPhoto2 Team, Tim Waugh, and Hans Ulrich Niedermann

Table of Contents

List of Figures

Chapter 1. Quick start

Abstract

How you quickly get your pictures to your computer, assuming somebody has already set up everything correctly.

This chapter assumes that somebody has set up your system correctly for use with libgphoto2. This is something the packages from your system vendor (RPM packages, DEB packages or BSD ports) and/or your system administrator should already have done for you. If not, follow the instructions in Chapter 2 first.

Currently, please take a look at the command line examples from gphoto2(1) and the examples described on www.gphoto.org [http://www.gphoto.org/].

FIXME: We will add more examples here in the future.

Chapter 2. Setting up your system for use with libgphoto2 and gphoto2

Table of Contents

Abstract

This chapter aims to help you set up your system such that you can use libgphoto2 with any frontend. However, we will have some examples using the gphoto2(1) command line frontend, as this is the frontend which is always provided.

System Overview

gPhoto2 consists of two libraries libgphoto2 (see gphoto2(3)), and libgphoto2_port (see gphoto2_port(3)), which is used by the former, and a command line frontend (**gphoto2**, see gphoto2(1)). Other (GUI) frontends (like e.g. gtkam) are available as separate packages.

In order to get access to the camera, your frontend process requires write permissions to the respective device special file, e.g. to `/dev/ttyS3` or `/proc/bus/usb/1/012`.

For security reasons, we strongly recommend not to run any gphoto2(3) frontend as root. So you have to set up the permissions of your camera device accordingly. This is described in chapter 2 - Setting up permissions.

Then you can run your frontend. For the command line gphoto2(3) frontend gphoto2(1), this is described in the gphoto2(1) man page.

Setting up permissions for serial (RS232) ports

Note
I don't have any experience with serial ports for user processes. So take this section with a grain of salt.

If you have a serial port reserved for the camera, you may just chown the device file to the user you want to run gphoto2 as.

Setting up permissions for USB ports

As USB allows hotplugging of devices, there is a mechanism that dynamically creates the device files for the devices currently connected and switched on.

The operating system has to determine which users may access a device dynamically. As the operating system cannot determine this by itself, there have to be some helper applications.

The configuration of these helper applications is explained in the following section.

USB ports on Linux

As gphoto2 provides a user space driver, in order to have gphoto2 access your camera, you have to disable all kernel drivers which want to handle the camera themselves (e.g. the Linux dc2xx or stv680 drivers). You can check whether these modules are loaded by executing **lsmod**.

On Linux systems, you have basically two options to allow user access to USB devices:

1. allow a certain user/group or the world access to *allall* USB devices by mounting `/proc/bus/usb` with adequate user and/or group permissions (default is world-readable and root-only-writable, which is good)
2. use hotplug (http://linux-hotplug.sourceforge.net/) and allow access only to the USB devices you want to be accessible (you need `/proc/bus/usb` mounted here as well, but not mounted writable by anybody else than root)

Solution 2 has a huge advantage over 1: It doesn't allow the user/group to interfere with or eavesdrop on any other USB devices which might be attached, such as USB keyboards, fingerprint reader or similar. The following paragraphs thus describe setting up 2.

On Linux systems, from the 2.4 kernel series on, the kernel supports hotplugging. You may have to compile a kernel with hotplug support if you're not already running one. You may have to install the hotplug package (http://linux-hotplug.sourceforge.net/) if you don't have it installed already.

You can find out if your kernel has hotplug support by looking for the file `/proc/sys/kernel/hotplug`. If it exists, you have a hotplug enabled kernel. If

```
cat
/proc/sys/kernel/hotplug
```

prints the path to your hotplug binary (usually **/sbin/hotplug**) and this binary exists, you are ready to rock.

Also note that the following solution does *notnot* provide absolute security and that you know the security implications of the respective **usbcam** script you are going to use.

1. You must have the files `devices` and `drivers` in your `/proc/bus/usb` directory. If not, check the following paragraph for hints.
   Load your USB driver and mount the USB device filesystem, i.e. e.g.

   ```
   # modprobe usb-uhci
   # modprobe usb-ohci
   # mount
   -t usbdevfs
   /proc/bus/usb
   ```

   Modern distributions like Redhat 7.2 handle this automatically if you have your USB hardware enabled. Check your BIOS settings if **lspci** doesn't list any USB hardware.

2. Add the output of

   ```
   # gphoto2 --print-usb-usermap
   ```

   to `/etc/hotplug/usb.usermap` after removing all lines beginning with "usbcam".
   This makes hotplug recognise all USB cameras which your version of gphoto2(3) supports and makes **hotplug** run the **usbcam** script you choose in step 3whenever one of these cameras is attached.

3. Choose the right `/etc/hotplug/usb/usbcam` script for you. Example scripts are found in `packages/linux-hotplug` in the source tree and in `linux-hotplug/` after installation and in the doc dir of the binary package. Choose one, adapt it for your needs, and copy it to `/etc/hotplug/usb/usbcam`. The directory `/etc/hotplug` should already exists, whereas it may be that you have to create the directory `/etc/hotplug/usb`.
   All three scripts shipped with gPhoto2 also have extensive commentary explaining their usage in more detail.

`usbcam.console`	The most simple solution is using usbcam.console. This changes the permissions so that the user owning the console according to the pam_console access the camera. This works only if you're logging in with pam_console, i.e. e.g. using gdm on Redhat Linux.
`usbcam.user`	If you want only one user to have access to the camera, use `usbcam.user` and change it accordingly. There is a specially marked line in the script you have to change.
`usbcam.group`	If you want multiple users to have access to the camera, add all of these users to one group - either a special group camera or a generic group users will do - and use that group in `usbcam.group`. There is a specially marked line in the script you have to change.

4. Make your script file `/etc/hotplug/usb/usbcam` executable.

5. Plug in the camera and switch it on. If you already did so, please unplug and/or switch off first. The kernel will now notice that your camera has been connected and, hopefully finding no kernel driver for the device, will ask hotplug to do something about it.
   Hotplug will then look into `/etc/hotplug/usb.usermap` and find that the **usbcam** script is to be called for the newly attached device. Thus **/etc/hotplug/usb/usbcam** is executed, hopefully setting the device permissions correctly.

   Your `/var/log/messages` syslog file will contain some messages to that effect.

6. Run gphoto2(1) or any other gphoto2(3) frontend and enjoy:

```
[user@home ~]$ gphoto2 --list-ports
[user@home ~]$ gphoto2 --auto-detect
[user@home ~]$ gphoto2 --summary
[user@home ~]$ gphoto2 --list-files
[user@home ~]$ gphoto2 --get-all-images
```

Specifying the port and camera you use

Abstract

libgphoto2 identifies a camera by two values:

a. the port it is connected to

b. the name of the camera

How these may be specified is discussed in this chapter.

Port names

Serial ports	are named like serial:/dev/ttyS2 if you want to use the serial device `/dev/ttyS2`.
USB ports	As USB works with auto detection, you do not have to specify a device file. Therefore you just use the gphoto2 port usb:.

Camera name

The model name of the camera does not have to be specified when using the usb: port. Otherwise you can specify a camera model like Canon PowerShot G2. You will be better off choosing the model from the list of supported models rather than just trying to type your camera model.

```
[me@home ~]$ gphoto2 --camera "Canon PowerShot G2" --list-files
```

Chapter 3. Developer Documentation: The Inner Workings

Table of Contents

Abstract

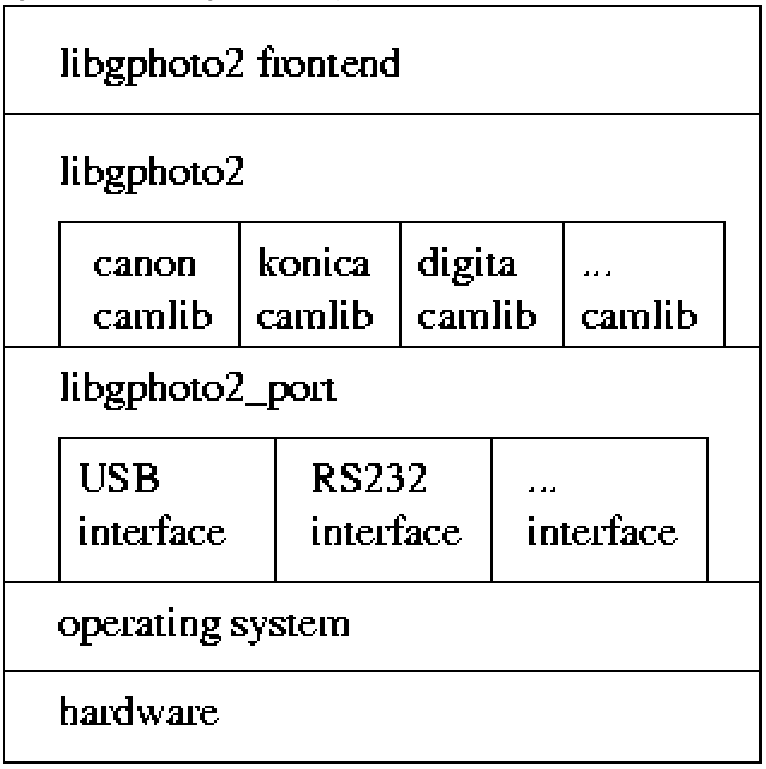
How it works internally, how you can work on it and how you can use it in your own software.

The APIs defined here are described in more detail by the autogenerated documentation at FIXME. Eventually, they should be included here, but as they currently are in Docbook SGML and this was written in Docbook XML, this isn't trivial.

Anyway, we provide you with the architecture context here and will let you read up on the API details in the respective external documentation.

The gPhoto2 software architecture

Figure 3.1. The gPhoto2 system architecture



$Id: architecture.fig,v 1.3 2001/11/27 21:58:40 hun Exp $

Diagram describing how frontends, libgphoto2, camlibs, libgphoto2_port and your Operating System work together.

The libgphoto2 API

Figure 3.2. The gPhoto2 system architecture

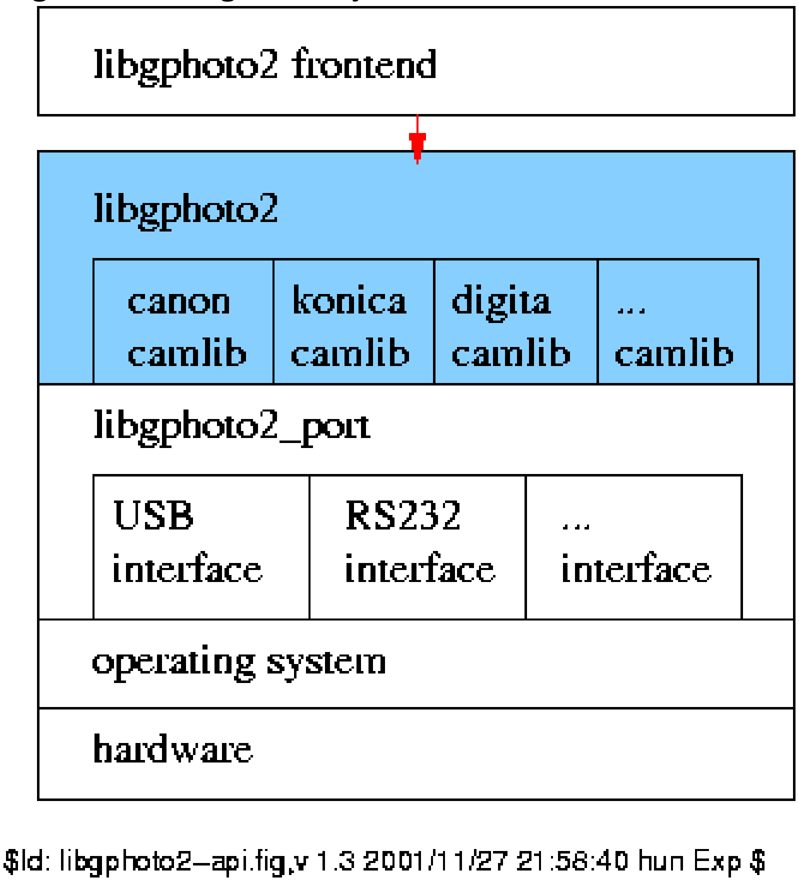


Diagram describing where the libgphoto2 API is located within the gPhoto2 software architecture

The camlib API

Figure 3.3. The gPhoto2 system architecture

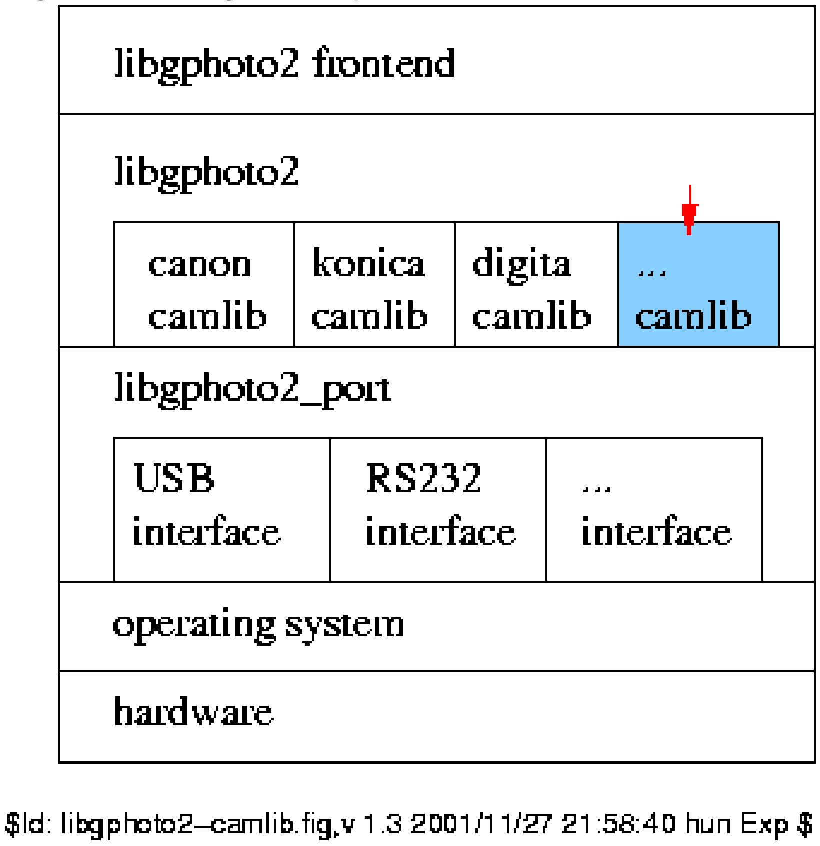
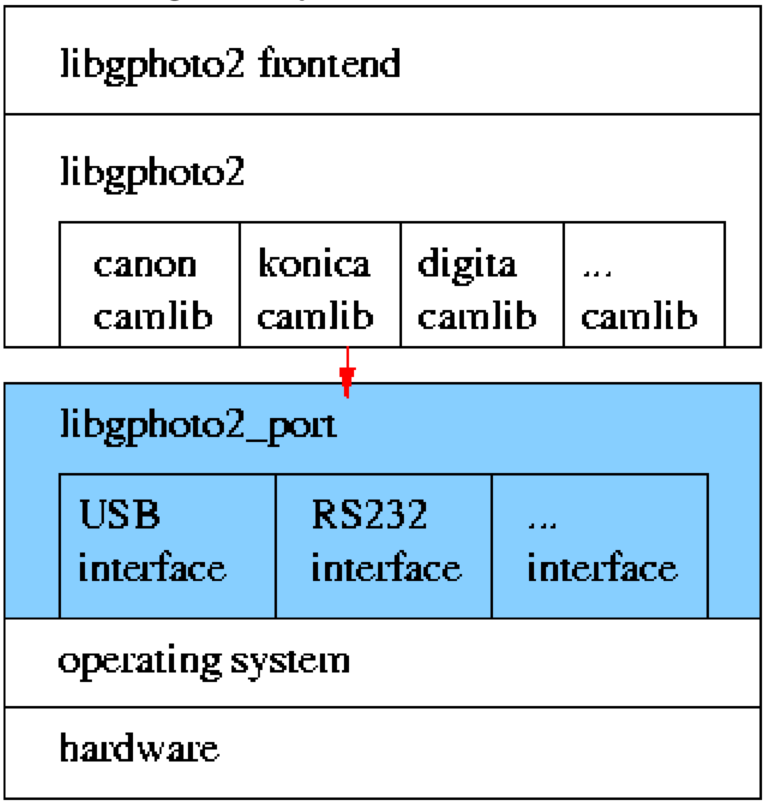


Diagram describing where the camlib API is located within the gPhoto2 software architecture

The libgphoto2_port API

Figure 3.4. The gPhoto2 system architecture



$Id: libgphoto2_port.fig,v 1.3 2001/11/27 21:58:40 hun Exp $

Diagram describing where the libgphoto2_port API is located within the gPhoto2 software architecture

Chapter 4. Utopia: A look into the possible future

Table of Contents

Abstract

Things that may or may not be implemented in the future. Usefulness and feasability of these things may vary considerably.

Language Bindings

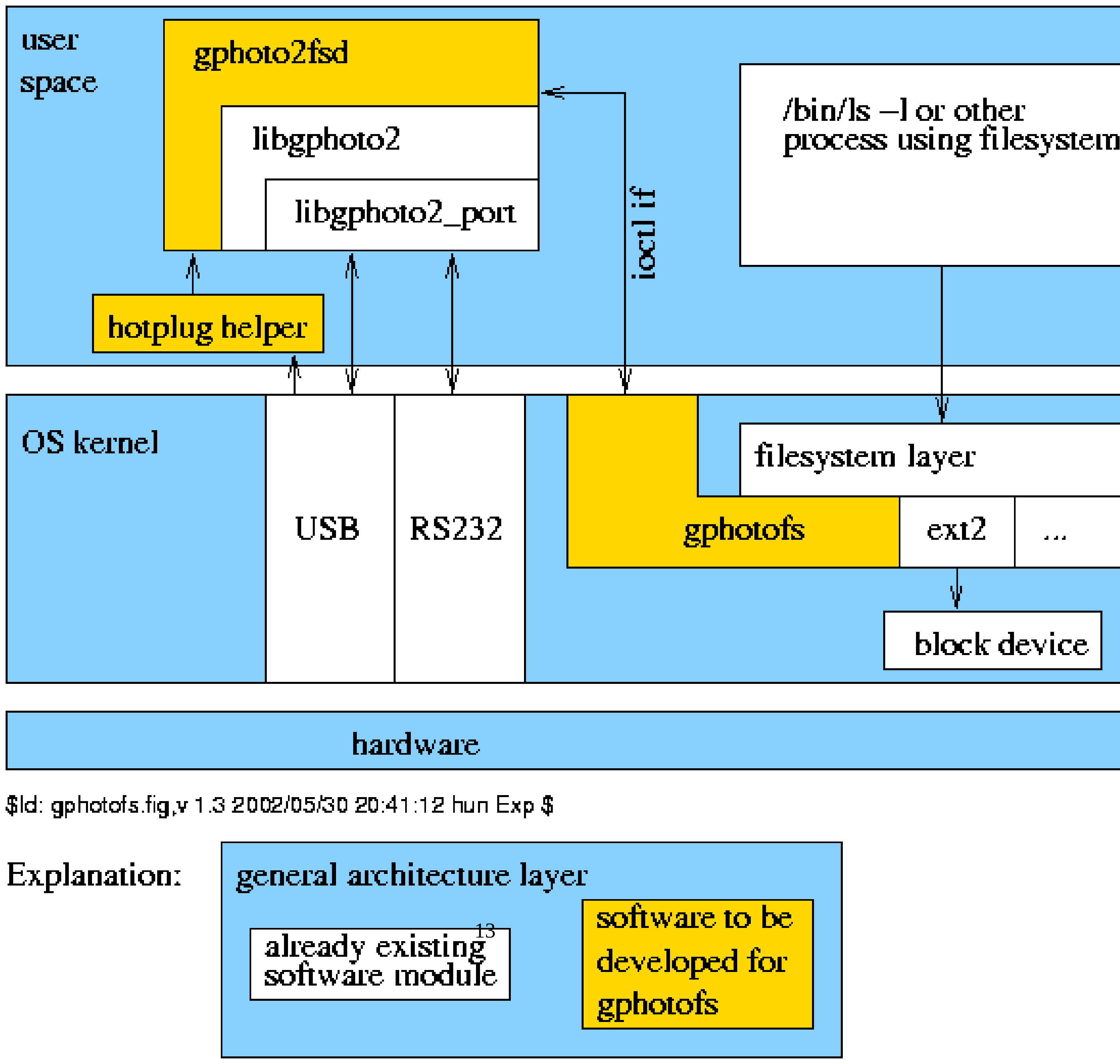
If somebody is interested in writing a libgphoto2 frontend in another programming language, it would be nice to have language bindings for that language. Perl, Python and Java (JNI) come to mind...

The gPhoto2 file system

Chapter 4. Utopia: A look into the possible future

Figure 4.1. The gPhoto2 file system architecture

NOTE:
The following is the result of a session of Lutz and Uli in the Wichtel.
We thought about what would be nice to have and came up with a gphotofs which allows you to mount any camera in the filesystem.
Some prototype code has been written by Lutz, but there is nothing else yet, especially nothing which actually does something.



$Id: gphotofs.fig,v 1.3 2002/05/30 20:41:12 hun Exp $

Diagram describing where the gPhoto2 file system is located within the gPhoto2 and system software architecture

The gPhoto2 Reference (the man pages)

<refentryinfo>
<author>The gPhoto2Team</author>
<editor>TimWaugh
<affiliation>

`<twaugh@redhat.com>`
</affiliation>
</editor>
<date>February 2002</date>
</refentryinfo>

Name

gphoto2 -- command-line gphoto2 client
gphoto2

Synopsis

`gphoto2` [--debug] [-q | --quiet] [-v | --verbose] [-h | --help] [--list-cameras] [--usb-usermap-script *NAME*] [--print-usb-usermap] [--list-ports] [--stdout] [--stdout-size] [--auto-detect] [--port *PATH*] [--speed *SPEED*] [--camera *MODEL*] [--filename *FILENAME*] [--usbid *USBID*] [-a | --abilities] [--folder *FOLDER*] [[-R | --recurse] | --norecurse] [-l | --list-folders] [-L | --list-files] [-m *NAME* | --mkdir *NAME*] [-r *NAME* | --rmdir *NAME*] [-n | --num-files] [-p *RANGE* | --get-file *RANGE*] [-P | --get-all-files] [-t *RANGE* | --get-thumbnail *RANGE*] [-T | --get-all-thumbnails] [-r *RANGE* | --get-raw-data *RANGE*] [--get-all-raw-data] [--get-audio-data *RANGE*] [--get-all-audio-data] [-d *RANGE* | --delete-file *RANGE*] [-D | --delete-all-files] [-u *FILENAME* | --upload-file *FILENAME*] [--capture-preview] [--capture-image] [--capture-movie] [--capture-sound] [--capture-show-info *RANGE*] [--summary] [--manual] [--about] [--shell]

Description

gphoto2(3) is a cross-platform digital camera library, and gphoto2(1) is a command-line client for it.

Where an option takes a range of files, thumbnails, or other data, they are numbered beginning at 1. A range is a comma-separated list of numbers or spans ('*`first-last`*'). Ranges are XOR (exclusive or), so that '1-5,3,7' is equivalent to '1,2,4,5,7'.

`--debug`
: Turn on debugging.

`-q`, `--quiet`
: Quiet output (default=verbose).

`-v`, `--version`
: Display version and exit.

`-h`, `--help`
: Display a short usage message.

`--list-cameras`
: List supported camera models.

`--usb-usermap-script` *NAME*
: Use *NAME* as the hotplug usb script used in the output of `--print-usb-usermap`. Default is "usbcam".

`--print-usb-usermap` For every camera supported by your version of **gphoto2**, print the corresponding line for inclusion into the `usb.usermap` hotplug config file. The script name used in these lines is specified with the `--usb-usermap-script` option.

`--list-ports` List supported port devices.

`--stdout` Send file to stdout.

`--stdout-size` Print filesize before data.

`--auto-detect` List auto-detected cameras.

`--port` *`PATH`* Specify port device.

`--speed` *`SPEED`* Specify serial transfer speed.

`--camera` *`MODEL`* Specify camera model. Most model names contain spaces: remember to enclose the name in quotes so that the shell knows it is one parameter. For example: **`--camera "Kodak DC240"`**.

`--filename` *`FILENAME`* Specify a filename.

`--usbid` *`USBID`* (Expert only) Override USB IDs.

`-a`, `--abilities` Display camera abilities.

`-f`, `--folder` *`FOLDER`* Specify camera folder (default="/").

`-R`, `--recurse` Recursion (default for download).

`--no-recurse` No recursion (default for deletion).

`-l`, `--list-folders` List folders in folder.

`-L`, `--list-files` List files in folder.

`-m`, `--mkdir` *`NAME`* Create a directory.

`-r`, `--rmdir` *`NAME`* Remove a directory.

`-n`, `--num-files` Display number of files.

`-p`, `--get-file` *`RANGE`* Get files given in range.

`-P`, `--get-all-files` Get all files from folder.

`-t`, `--get-thumbnail` *`RANGE`* Get thumbnails given in range.

`-T`, `--get-all-thumbnails` Get all thumbnails from folder.

`-r`, `--get-raw-data` *`RANGE`*	Get raw data given in range.
`--get-all-raw-data`	Get all raw data from folder.
`--get-audio-data` *`RANGE`*	Get audio data given in range.
`--get-all-audio-data`	Get all audio data from folder.
`--delete-files` *`RANGE`*	Delete files given in range.
`--delete-all-files`	Delete all files in folder.
`-u`, `--upload-file` *`FILENAME`*	Upload a file to camera.
`--capture-preview`	Capture a quick preview.
`--capture-image`	Capture an image.
`--capture-movie`	Capture a movie.
`--capture-sound`	Capture an audio clip.
`--show-info` *`RANGE`*	Show info.
`--summary`	Summary of camera status.
`--manual`	Camera driver manual.
`--about`	About the camera driver.
`--shell`	Start the gphoto2 shell, an interactive environment. See SHELL MODE for a detailed description.

Shell Mode

The following commands are available:

cd	Change to a directory on the camera.
lcd	Change to a directory on the local machine.
exit, quit, q	Exit the gphoto2 shell.
get	Download the file to the current directory.
get-thumbnail	Download the thumbnail to the current directory.
get-raw	Download raw data to the current directory.

show-info	Show information.
delete	Delete a file or directory.
show-exif	Show EXIF information (only if compiled with EXIF support).
help, ?	Displays command usage.
ls	List the contents of the current directory on the camera.

See also

gphoto2(3), `gphoto2.txt`, `gphoto2-cli.txt`, http://www.gphoto.org/

Examples

`gphoto2 --list-files`	List files on camera.
`gphoto2 --get-file` ***`7-13`***	Get files number 7 through 13 from the list output by **`gphoto2 --list-files`**.
`gphoto2 --usb-usermap-script` ***`gphoto2cam`*** **`--print-usb-usermap`** >> `/etc/hotplug/usb.usermap`	Append a line to the **hotplug** config file `/etc/hotplug/usb.usermap` for every supported camera, using `/etc/hotplug/usb/gphoto2cam` for setting up permissions correctly.

<refentryinfo>
<author>The gPhoto2Team</author>
<editor>Hans UlrichNiedermann
<affiliation>

`<gp@n-dimensional.de>`
</affiliation>
</editor>
</refentryinfo>

Name

gphoto2 -- cross-platform digital camera library
gphoto2

Synopsis

#include <gphoto2.h>

Description

The gphoto2 library provides applications with access to a variety of cameras.

This man page will be extended with autogenerated documentation of the interface types and methods used for communication between the gphoto2 library and a frontend.

Files

`~/.gphoto/settings`	Here gphoto2 applications may store their configuration used to access gphoto2.

See also

gphoto2(1), gphoto2_port(3), `gphoto2.txt`, http://www.gphoto.org/

<refentryinfo>
<author>The gPhoto2Team</author>
<editor>Hans UlrichNiedermann
<affiliation>

`<gp@n-dimensional.de>`
</affiliation>
</editor>
</refentryinfo>

Name

gphoto2_port -- cross-platform port access library
gphoto2_port

Synopsis

#include <gphoto2_port.h>

Description

The libgphoto2_port library was written to provide gphoto2(3) with a generic way of accessing ports. In this function, libgphoto2_port is the successor of the libgpio library.

Currently, libgphoto2_port supports serial (RS-232) and USB connections, the latter requiring libusb to be installed.

The autogenerated API docs will be added here in the future.

See also

gphoto2(3), `gphoto2.txt`, http://www.gphoto.org/, http://libusb.sourceforge.net/

Appendix A. Resources: Where to find related information

http://www.gphoto.org/	The home page of the gPhoto project.
http://sourceforge.net/projects/gphoto/	The gPhoto project page.
http://libusb.sourceforge.net/	The libusb home page.
http://sourceforge.net/projects/libexif/	The libexif project page.