The gPhoto2 Manual

by The gPhoto2 Team, Tim Waugh, Hans Ulrich Niedermann, and Michael J. Rensing

The gPhoto2 Manual

by The gPhoto2 Team, Tim Waugh, Hans Ulrich Niedermann, and Michael J. Rensing

Published $Date: 2003/03/10 22:05:39 $

Table of Contents

List of Figures

List of Examples

About this document

Mission statement, history and options for the future

The purpose of writing is to inflate weak ideas, obscure poor reasoning, and inhibit clarity.
--Calvin, in one of Bill Watterson's "Calvin and Hobbes" comic strips

This document is called "The gPhoto2 Manual". You can get it from the gphoto homepage, found in at least one of these places: http://gphoto.net/, http://gphoto.org/, http://gphoto.com/, http://gphoto.sf.net/.

Mission statement. The gPhoto2 Manual intends to be a comprehensive source of information helping users and developers to get more value out of the gPhoto2 software suite (i.e. libgphoto2, gphoto2 CLI interface, gtkam).

Document history. Hans Ulrich Niedermann first wrote a few text files of user documentation for gphoto2. Then shortly before the 2.0 release, Tim Waugh wrote the man pages in Docbook XML. Then Hans Ulrich Niedermann migrated the text files into the XML file and began with general improvements. Then after the 2.1.0 release, Michael J. Rensing joined us with gtkam documentation.

Ideas for the future. The following ideas are just ideas, not mandatory policy.

1. Add the FAQ, a gtkam manual, the older developer docs and the current autogenerated source code documentation in convenient places.
2. Improve the document structure.
3. Create a pure ASCII text version of specific parts (like e. g. installation FAQs, compilation FAQs etc.) for inclusion into the respective distributions.

The editors want to thank all those who gave us feedback and suggestions about this manual .

Chapter 1. Finding what you need

Abstract

This chapter lists information about what you need to run gphoto2. This information is intended to assist you in setting up your computer so that you can connect a camera and use libgphoto2 software to download images.

What you need:

1. A camera and whatever cables (serial RS232, USB, ...) you need to connect your camera to your computer

   A list of currently supported cameras is maintained at http://www.teaser.fr/~hfiguiere/linux/digicam.html

   ### Important

   If you want to use a camera using the USB mass storage protocol, libgphoto2 will not help you. Your operating system already contains an USB mass storage driver, so there is no need to write another one.

   For information on how to use a USB mass storage camera with Linux, see the USB Digital Camera HOWTO [http://www.tldp.org/HOWTO/USB-Digital-Camera-HOWTO/] (which is more a USB mass storage camera HOWTO).

2. A computer with the right kind of port (serial RS232, USB, ...) running a Unix-like operating system:

   - BSD (FreeBSD, OpenBSD, NetBSD, ...)
   - Linux (RedHat 7.3, SuSE 8.x, Mandrake 8.x, Debian 3.0 Woody, Debian Unstable Sid, Gentoo, ...)
   - OS/2 (yes, there is an OS/2 port, and there are people who use gphoto2 2.1.0 under OS/2)
   - MacOS X (several people are working on that, exact status is unknown as of 2002-10-16)
   - FIXME: Is there a list somewhere, or can we get contributions of verified OS's?

   The hardware architecture (x86, PowerPC, Sparc) doesn't matter. Or at least, it shouldn't.

   For USB cameras, your Unix OS must be supported by `libusb`. At 2002-08-29, this is the case for Linux 2.2 and 2.4, FreeBSD and OpenBSD. FIXME: Is the USB support for OS/2 independent of libusb?

3. libgphoto2, some libgphoto2 frontend and (if you want to use USB cameras) libusb

   ### libgphoto2 frontends

gphoto2	Official command line interface (CLI) frontend. Simple command line interface and kind of a reference implementation. This libgphoto2 frontend i very useful for debugging camera drivers and other problems.
gtkam	Official GUI frontend using GTK+ 2. Shows thumbnails of pictures on the camera, and supports upload, download, capturing of pictures.
gnocam	The Gnome VFS interface. Still has to be ported to Gnome 2. Volunteers are welcome.
kamera	The KDE IO slave interface. FIXME: URL, words, info
digikam	3rd party software available from http://digikam.sourceforge.net/. Digikam is a standalone KDE frontend.

flPhoto	3rd party software available from http://www.easysw.com/~mike/flphoto/. FLTK-based frontend for gPhoto called "flphoto" which provides some nice functionality beyond the current GTK-based frontend. Aside from downloading images from a camera, you can also "import" you existing files on disk, adjust/transform the images, print them, and do slideshows. The final version will also offer a web export to a directory or uploaded to a web server.

Available software formats

source tarballs	Building requires mainly make and a C compiler.
CVS sources	Building requires a certain set of build tools (automake, autoconf, gettext, libtool), and possibly even a combination of certain releases.
system-specific packages	As packages are specifically created for your system, this is probably the easiest way to install the software. Packages for your system can be source packages or binary packages, depending on your system. They have names like RPM (Redhat, SuSE), DEB (Debian), port (BSD), ebuild (Gentoo).
FIXME	Need improved descriptions and lists of various forms of source code and binaries.

Chapter 2. 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 the chapter on permission setup first.

FIXME: We need examples using RS232 cameras and the gphoto2 shell.

2.1. Using gtkam

FIXME: gtkam docs under construction. Please report errors, omissions, and constructive suggestions about the gtkam documentation to michael.rensing@shaw.ca

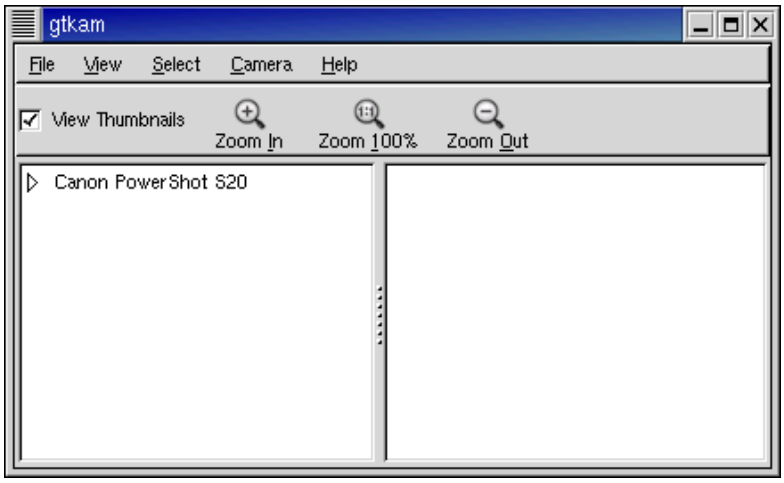
Abstract

gtkam is a graphical front end for the gphoto2 library. It does not communicate directly with the camera, but uses gphoto2 to do so. As a result, it is necessary to have gphoto2 installed and running correctly before gtkam will work.

If everything is configured correctly, the following steps should work:

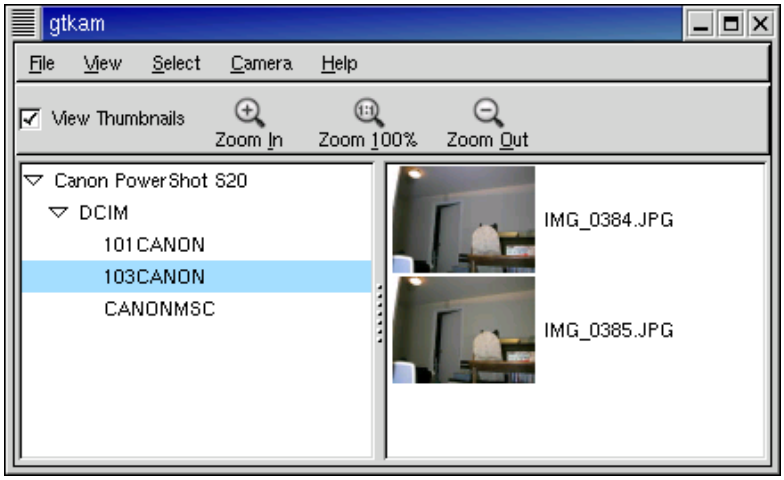
1. Plug in the connecting cable to your camera and to the computer.
2. Turn on your camera and switch to the mode which enables communication with a computer.
3. Run gtkam. This may be from a menu (in RedHat, it's Main Menu->Programs->Graphics->gtKam)(in Debian, it's Debian Menu->Viewers->gtKam). Otherwise, typing **gtkam** from the command line should start the application.

If all is well, and your camera has been added previously, you should see your camera listed in the left frame with a plus sign in a box to the left.



Click on the plus sign (v.0.1.3) or the arrow (v.0.1.9) to expand the listings of directories. The number in brackets to the right of the directory name is the number of images detected in that directory.

Click on directory containing images, and you should see a listing of the images in the right frame.



Troubleshooting

If you do not see any images listed, be sure that there are images in the camera. If there are, then you need to do some troubleshooting. Start with the FAQ .

2.2. Using the gphoto2 command line interface (CLI)

OK, we begin with finding out how you can connect a camera to your computer:

```
alice@host:~$ gphoto2 --list-ports
Devices found: 4
Path                      Description
-------------------------------------------------------------
serial:/dev/ttyS0            Serial Port 0
serial:/dev/ttyS2            Serial Port 2
serial:/dev/ttyS3            Serial Port 3
usb:                      Universal Serial Bus
alice@host:~$
```

In this example, we see that we have the three serial devices `/dev/ttyS0`, `/dev/ttyS2` and `/dev/ttyS3` which are configured properly and to which **gphoto2** has write permissions for. Additionally, there is also a USB bus. However, we cannot tell yet whether we will have write access to the USB device file your camera will be assigned by your operating system.

The next step is to connect you camera and find out whether **gphoto2** can find it. This may only work with USB. (FIXME: can any of the people with RS232 cameras help me on this issue?)

```
alice@host:~$ gphoto2 --auto-detect
Model                    Port
----------------------------------------------------------
Canon PowerShot G2          usb:
alice@host:~$
```

In this case, a camera called “Canon PowerShot G2” is connected to your system’s USB bus.

OK, so now let’s see what we can find out about the camera:

```
alice@host:~$ gphoto2 --summary
Detected a 'Canon PowerShot G2'.
Camera Summary:

Camera identification:
  Model: Canon PowerShot G2
  Owner: Alice Smith

Power status: on battery (power OK)

Flash disk information:
  Drive D:
   31'885'312 bytes total
   27'668'480 bytes available

Time: 2002-07-09 20:45:15 (host time +0 seconds)

alice@host:~$
```

The actual result of **gphoto2 *--summary*** may be different from camera to camera. However, we can see that **gphoto2** obviously can talk to the camera, and we get some information about the camera.

OK, now that we have seen that there is some space occupied on the storage medium of the camera, let's have a look at what is stored there:

```
alice@host:~$ gphoto2 --list-files
Detected a 'Canon PowerShot G2'.
There are no files in folder '/'.
There are no files in folder '/DCIM'.
There are 4 files in folder '/DCIM/154CANON':
#1    CRW_5417.CRW           rd  2334 KB image/x-canon-raw
#2    IMG_5415.JPG           rd  1044 KB image/jpeg
#3    IMG_5416.JPG           rd   31 KB image/jpeg
#4    MVI_5418.AVI           rd  682 KB video/x-msvideo
There are no files in folder '/DCIM/CANONMSC'.
alice@host:~$
```

There are multiple folders on the camera, and there are several files of different types and sizes on stored in one of them. One is a raw file of the type this camera produces, two are JPEG files (one large and one small) and one seems to be a short video sequence.

Now that we have an impression of what expects us when we actually download the images from the camera, we are going to do that:

```
alice@host:~$ gphoto2 --get-all-files
Detected a 'Canon PowerShot G2'.
Downloading 'CRW_5417.CRW' from folder '/DCIM/154CANON'...
Saving file as CRW_5417.CRW
Downloading 'IMG_5415.JPG' from folder '/DCIM/154CANON'...
Saving file as IMG_5415.JPG
Downloading 'IMG_5416.JPG' from folder '/DCIM/154CANON'...
Saving file as IMG_5416.JPG
Downloading 'MVI_5418.AVI' from folder '/DCIM/154CANON'...
Saving file as MVI_5418.AVI
alice@host:~$
```

And now we have all the files in our current directory (in this case in directory ~) and can do something with them.

Well, that's about it. You may also want to have a look at the command line examples from the gphoto2(1) man page.

2.3. Using Gnome VFS

FIXME: We need to write this section

This still has to be ported to the new Gnome 2 VFS. If you can help, please contact us.

2.4. Using kioslave

FIXME: We need to write this section

Chapter 3. Frequently Asked Questions (FAQ)

Generic libgphoto2 FAQs (frontend-independent)

3.3.1.1. I just downloaded the libgphoto2-2.1.1 distribution off sourceforge and it builds "libgphoto2.so.2.0.3". Has anyone else reported this?

Yes, many times. The release number ("2.1.1") and the library version ("2.0.3") are by their respective definitions *completely unrelated*. As we won't "fix" a problem which is not there, please stop bugging us with that "problem".

3.3.1.2. What can I do about the error message "Could not find USB device"?

The exact error message looks like

```
Could not find USB device (vendor 0x0000, product 0x0000). Make
sure this device is connected to the computer
```

You have to set up the permissions on your USB device correctly. See Section 4.3, "Setting up permissions for USB ports".

3.3.1.3. gphoto2 does not find my camera. What shall I do?

First try to run **gphoto2 *--list-ports*** and look whether the ports you want to use are listed there:

- For serial devices, the port with the appropriate device must be listed
- For USB devices, you need the "usb:" port in the list. If you don't see it, check whether you compiled gphoto2 with libusb support.

Then try to run **gphoto2 *--auto-detect*** and look whether the camera is detected.

If your gphoto2 cannot access the USB device as non-root users, you have to set up hotplugging correctly. See Section 4.3, "Setting up permissions for USB ports" for details.

AND DO NOT RUN gphoto2 AS ROOT. And no other libgphoto2 frontend either.

3.3.1.4. Why do I get the error message "Could not claim the USB device"?

The exact message looks like

```
*** Error ('Could not claim the USB device') ***

Could not claim interface 0 (Operation not permitted). Make sure no
other program or kernel module (e.g. dc2xx or stv680) is using the
device and you have read/write access to the device.
```

You have to make sure that no such kernel module is loaded and that you have set up the permissions on your USB device correctly, such that you have (non-root) write access to the camera device. How to set this up, is described at Section 4.3, "Setting up permissions for USB ports".

3.3.1.5. Why does my Canon PowerShot G1/G2/Pro 90 IS not work with gphoto2 when using the AC adapter?

This is a longstanding bug with the G1 and Pro 90 IS, and a relatively new bug with the G2. With the G2, it was introduced with a new camera firmware (G2 1.0.0.0 worked, 1.1.0.0 does not). We haven't fixed this bug yet since none of the active gphoto2 developers owned one of these cameras until recently (i.e. updated his G2's firmware).

Your options are as follows, in descending order of preference:

- Fix it and send us a patch.
- Donate a camera (G3 anyone? :-) to one of the active Canon driver developers.
- Run the camera on battery instead of the AC adapter.

There is some more info regarding this at http://krixor.xy.org/fredrik/gp/canon-g1-ac-problem/. Good luck.

3.3.1.6. The PTP driver crashes on my Linux 2.4.18 system. What should I do?

There is a bug in the Linux 2.4.18 USB driver which may result in random crashes of the libgphoto2 PTP driver. Please upgrade to a more recent kernel.

3.3.1.7. I've got a serial camera, but gphoto2 doesn't work on SuSE 7.0 (or greater). You told me that my camera is supported! Fix it!

Cool down, Joe, and listen to the wise words of Linda MacPhee-Cobb:

[If you are not getting a response from a camera plugged into the serial port, check your `/var/log/boot.msg` file for the following entry near the end:]

```
*This package use the ttyS0 device by default.
The installation of this package only makes sense
if you use a braille display.
```

[This is from blinx, and you will need to turn it off, or better, uninstall it.]

3.3.1.8. Dudes, Windows is the way to go. How do I compile gphoto2 for that environment?

We don't have the money to buy Windows and something like Visual C. Therefore, we can't provide you a gphoto2 for Windows. You have two choices (at least):

- Buy a fast computer, buy Windows and Visual C and some manuals on how to access USB or serial ports on Windows. Send all that including a blank cheque to us (ask for the address on `<gphoto-devel@gphoto.org>`).
- Try to figure out how to get gphoto2 working on Windows and send us the patches.

3.3.1.9. I wanna have CVS write access. How do I get one?

We provide CVS write access to persons that show *continued* interest in a *specific part* of gphoto2. If there is already a person maintaining this specific part of gphoto2 (i.e. a camera driver), we prefer you submit patches to the maintainer who will then check them in or request modifications.

3.3.1.10. You took away my CVS write access! What did I do wrong?

If your name is not listed in `libgphoto2/MAINTAINERS`, and if we haven't heard from you lately, we assume that you finished your work on gphoto2 and that you don't need write access any longer. If this is not the case, please contact one of the project admins.

3.3.1.11. I have a special problem with CameraXYZ. Where can I find information about it?

You can look into the `README` file of your driver *`driver`*. It is located in `camlibs/`*`driver/`* in your source tree. If you do not have a source tree, that is bad luck up to and including release 2.1.1 - we will install the `README` files in a future release, though.

3.3.1.12. I run Linux 2.4.x and I can't get USB to work at all. I keep getting this error message in the syslog:

```
kernel: usb_control/bulk_msg: timeout
kernel: usbdevfs: USBDEVFS_BULK failed dev 3 ep 0x81 len 64 ret -110
```

How can I make USB work?

No idea. Sometimes it works and sometimes it doesn't. If you know a solution, please let us know.

3.3.1.13. What's the best way to rotate downloaded JPEG images without losses?

```
alice@host:~$ # If you do not care about preserving EXIF information
alice@host:~$ jpegtran -copy all -rotate 90
alice@host:~$
alice@host:~$ # If you want to preserve EXIF information
alice@host:~$ jhead -cmd 'jpegtran -rotate 90 -outfile &o &i' my-pic.jpg
```

Installation FAQs

3.3.2.1. I use SuSE Linux 8.0/8.1 and I'm having problems compiling your gphoto2 2.1.1 release. What should I do?

The easiest solution is probably to use the RPM packages from ftp://ftp.suse.com/pub/people/meissner/gphoto/2.1.1/8.0-i386/ or ftp://ftp.suse.com/pub/people/meissner/gphoto/2.1.1/8.1-i386/, respectively.

3.3.2.2. gphoto2 does not work, the driver reports some error messages. What shall I do?

First of all make sure, that you have only one instance of gphoto2 and libgphoto2 installed on your system. Unless you know what you're doing you should remove all gphoto2 installation, then you may do (as root):

```
host:~# rm -rf /usr/local/lib/gphoto2* /usr/local/lib/libgphoto2*
host:~# rm -rf /usr/lib/gphoto2* /usr/lib/libgphoto2*
```

Above applies especially if you used packaged and CVS versions.

After that, install the newest version of libgphoto2 and gphoto2 and test whether it works now.

3.3.2.3. I want to use GNU stow, but that fails.

I don't know what stow is, but have a look at #553647 [].

3.3.2.4. I just sucessfully compiled and installed libgphoto2-2.2.1. But when configuring gphoto2-2.1.1 I get `Library requirements (libgphoto2 >= 2.1.1) not met`

If you don’t have pkg-config installed yet, then please install it.

Otherwise, your PKG_CONFIG_PATH variable doesn’t contain the `pkgconfig/` directory in the `lib/` directory of your libgphoto2 installation.

For a default installation, run

```
alice@host:~$ PKG_CONFIG_PATH=/usr/lib/pkgconfig:/usr/local/lib/pkgconfig
alice@host:~$ export PKG_CONFIG_PATH
```

and now run the gphoto2 **`./configure`** again.

Chapter 4. Setting up your system for use with `libgphoto2` and gphoto2

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.

4.1. System Overview

gPhoto2 consists of two libraries `libgphoto2` (see libgphoto2(3)), and `libgphoto2_port` (see libgphoto2_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 libgphoto2(3) frontend as root. So you have to set up the permissions of your camera device accordingly. This is described in the following two sections, Section 4.2, “Setting up permissions for serial (RS232) ports” and Section 4.3, “Setting up permissions for USB ports”.

Then you can run your frontend. For the command line libgphoto2(3) frontend gphoto2(1), this is described in the gphoto2(1) man page.

4.2. Setting up permissions for serial (RS232) ports

On Unix systems, serial ports usually are represented by device files. The serial port device files are e.g.:

for Linux	`/dev/ttyS?` starting with `/dev/ttyS0`
for FreeBSD	`/dev/cuaa?` starting with `/dev/cuaa0`

If you have a serial port reserved for the camera, you can just use **chown**, **chgrp**, **chmod** on its device file to provide a certain user or group or everybody with access to it.

You can use the **gphoto2 *--list-ports*** command to check whether you have permissions to access a certain device:

```
alice@host:~$ gphoto2 --list-ports
Devices found: 2
Path                       Description
----------------------------------------------------------
serial:/dev/ttyS0             Serial Port 0
usb:                       Universal Serial Bus

alice@host:~$
```

This shows that user alice (who seems to be running the command) has access to the serial device `/dev/ttyS0`.

4.2.1. A few examples

Let's make a few examples with the first serial port on a Linux system, called `/dev/ttyS0`. You start with having a look at the device file:

```
host:~# ls -l /dev/ttyS0
crw-rw----   1 root    dialout   4,  64 Jul  5 12:05 /dev/ttyS0
host:~#
```

OK, so user root and all users of the group dialout can access the device /dev/ttyS0. You now have a few choices of what to do.

Example 4.1. Exclusive access to a serial camera for *one* user

You want to give user alice exclusive access to the first serial port on your Linux machine:

```
host:~# # give alice exclusive access to the device file (you must be root)
host:~# chown alice /dev/ttyS0
host:~# chmod 0600 /dev/ttyS0
host:~# ls -l /dev/ttyS0
crw-------   1 alice   dialout   4,  64 Jul  5 12:05 /dev/ttyS0
host:~#
```

Test for access by running **`gphoto2 --list-ports`** as the respective user. Look for `/dev/ttyS0` in the output.

Example 4.2. Access to serial camera for a group of users

You could just add user alice to the group dialout, but this will almost certainly cause confusion between dialing and using the camera. So you'd better create a group named "camera", add alice to that group and give the group access to the camera:

```
host:~# # give users in group "camera" access to this device (you must be root)
host:~# groupadd camera
host:~# chgrp camera /dev/ttyS0
host:~# chmod 0060 /dev/ttyS0
host:~# ls -l /dev/ttyS0
c---rw----   1 root    camera    4,  64 Jul  5 12:05 /dev/ttyS0
host:~#
```

Test for access by running **`gphoto2 --list-ports`** as the respective user. Look for `/dev/ttyS0` in the output.

BTW, it makes sense to also set the permissions for USB camera devices to allow the "camera" group access. Then any user in the "camera" group will have access to any camera.

Example 4.3. Access to serial camera for *anybody*

Allowing *anybody* access to something is considered bad security. However, for quick testing or for non-networked single-user systems, this can also make sense:

```
host:~# # give anybody access to this device (you must be root)
host:~# chmod a+rw /dev/ttyS0
host:~# ls -l /dev/ttyS0
crw-rw-rw-   1 root    dialout   4,  64 Jul  5 12:05 /dev/ttyS0
host:~#
```

Test for access by running **gphoto2 `--list-ports`** as the respective user. Look for `/dev/ttyS0` in the output.

4.3. Setting up permissions for USB ports

As USB is designed for 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.

If you are using pre-built packages (such as SuSE or Redhat RPMs), these packages will already have done this configuration for you.

If you are installing and setting up libgphoto2 yourself, then the configuration of these helper applications is explained for you the following section.

4.3.1. 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**.

We will not cover basic USB setup here. For how to get USB working on your hardware and your system at all, we'd like to refer you to http://www.linux-usb.org/USB-guide/ and especially http://www.linux-usb.org/USB-guide/c122.html . This page explains the USB basics in a better way than we could do.

Now that you've got your basic USB system working, you have basically two options to allow user access to USB devices on your Linux system:

a. allow a certain user and/or group or the whole world access to *all* 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)

b. 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 b has a huge advantage over solution a: 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 b.

Note

The fact that we are using the name "*`usbcam`*" for setting up permissions for gphoto2 has a reason. In fact, the permission setup you're doing here has nothing to do with gphoto2 specifically - any user space software wanting to access your USB camera will be able to make use of your camera only if the permissions are correctly set up. So I (hun) chose the identifier and script name "*`usbcam`*" and not "gphoto2cam" or somethink similar.

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

```
alice@host:~$ 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 *not* provide absolute security and that you should definitely 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 (e.g. OHCI or UHCI) and mount the USB device filesystem, i.e. e.g.

   ```
   host:~# modprobe usb-uhci
   host:~# modprobe usb-ohci
   host:~# mount -t usbdevfs usb /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. In the file `/etc/hotplug/usb.usermap` remove all lines beginning with "*usbcam*". We are going to add new lines there and don't want to have the old ones get in the way.

   Add the output of **/usr/lib/libgphoto2/print-usb-usermap** to the `/etc/hotplug/usb.usermap` file:

   ```
   # EITHER this
   host:~# /usr/lib/libgphoto2/print-usb-usermap >> /etc/hotplug/usb.usermap
   ```

   ## Note

   In older versions of gphoto2 (gphoto2 2.0, gphoto2 2.1.0, and the CVS versions in between and shortly after 2.1.0), the functionality of **print-usb-usermap** was contained in the command **gphoto2 *--print-usb-usermap***.

   Debian peculiarity. Debian uses a well-structured method for setting up linux-hotplug. You should use this mechanism in order to enjoy its benefits. Instead of modifying `/etc/hotplug/usb.usermap` directly, we set up a special file `/usr/lib/hotplug/libgphoto2/usb.usermap` and let Debian's **update-usb.usermap** script update the `/etc/hotplug/usb.usermap` file:

   ```
   # This is only for Debian, OK?
   host:~# mkdir /usr/lib/hotplug/libgphoto2
   host:~# ./path/to/libgphoto2-build/packaging/linux-hotplug/print-usb-usermap >
   /usr/lib/hotplug/libgphoto2/usb.usermap
   host:~# update-usb.usermap
   ```

   This makes hotplug recognise all USB cameras which your version of libgphoto2(3) supports and makes **hotplug** run the ***usbcam*** script you choose in step 3 whenever one of these cameras is attached.

3. Choose the right `/etc/hotplug/usb/`*`usbcam`* script for you.

   Example scripts are found in `packaging/linux-hotplug/` in the source tree and in the doc dir (usually /usr/share/doc/gphoto2/ or something similar) under `linux-hotplug/` after installation.

   Choose a script which fits your requirements best, adapt it for your needs, and copy it to the file `/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`. The script *must* be called `/etc/hotplug/usb/`*`usbcam`*, not `/etc/hotplug/usb/usbcam.user` or something similar.

   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. It won't work on Debian GNU/Linux at all (they claim security reasons, but I didn't investigate further into that matter).

   `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.

   `usbcam.x11-app` — If you want only one user to have access to the camera and your favourite X11 libgphoto2 frontend launched automatically, use `usbcam.x11-app` and change it accordingly. There are a few specially marked lines in the script you have to change.

4. Make the script file you just created and possibly adapted (`/etc/hotplug/usb/`*`usbcam`*, *not* `/etc/hotplug/usb/usbcam.whatever`!) executable:

   ```
   host:~# chmod +x /etc/hotplug/usb/usbcam
   ```

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.

   You will probably want to check whether the respective device file has its permissions set up correctly. Have a look at `/proc/bus/usb` with **`ls -lR /proc/bus/usb`**. There should be at least one device file with the permissions set according to your wishes.

6. Run gphoto2(1) or any other libgphoto2(3) frontend and enjoy:

```
alice@host:~$ gphoto2 --list-ports
alice@host:~$ gphoto2 --auto-detect
alice@host:~$ gphoto2 --summary
alice@host:~$ gphoto2 --list-files
alice@host:~$ gphoto2 --get-all-images
```

4.3.2. USB ports on other systems (BSD, MacOS X, OS/2)

FIXME: Still to be written

4.4. Specifying the port and camera you use

Abstract

`libgphoto2` identifies a camera by two values: the port it is connected to and the name of the camera. How these may be specified is discussed in this chapter.

4.4.1. 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:`.

The port used is stored in `~/.gphoto/settings`, so you won't have to specify it the second time you run your frontend, if you're still using the same port.

4.4.2. 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.

```
alice@host:~$ gphoto2 --camera "Canon PowerShot G2" --list-files
```

The camera used is stored in `~/.gphoto/settings`, so you won't have to specify it the second time you run your frontend, if you're still using the same camera.

Chapter 5. Compiling and installing

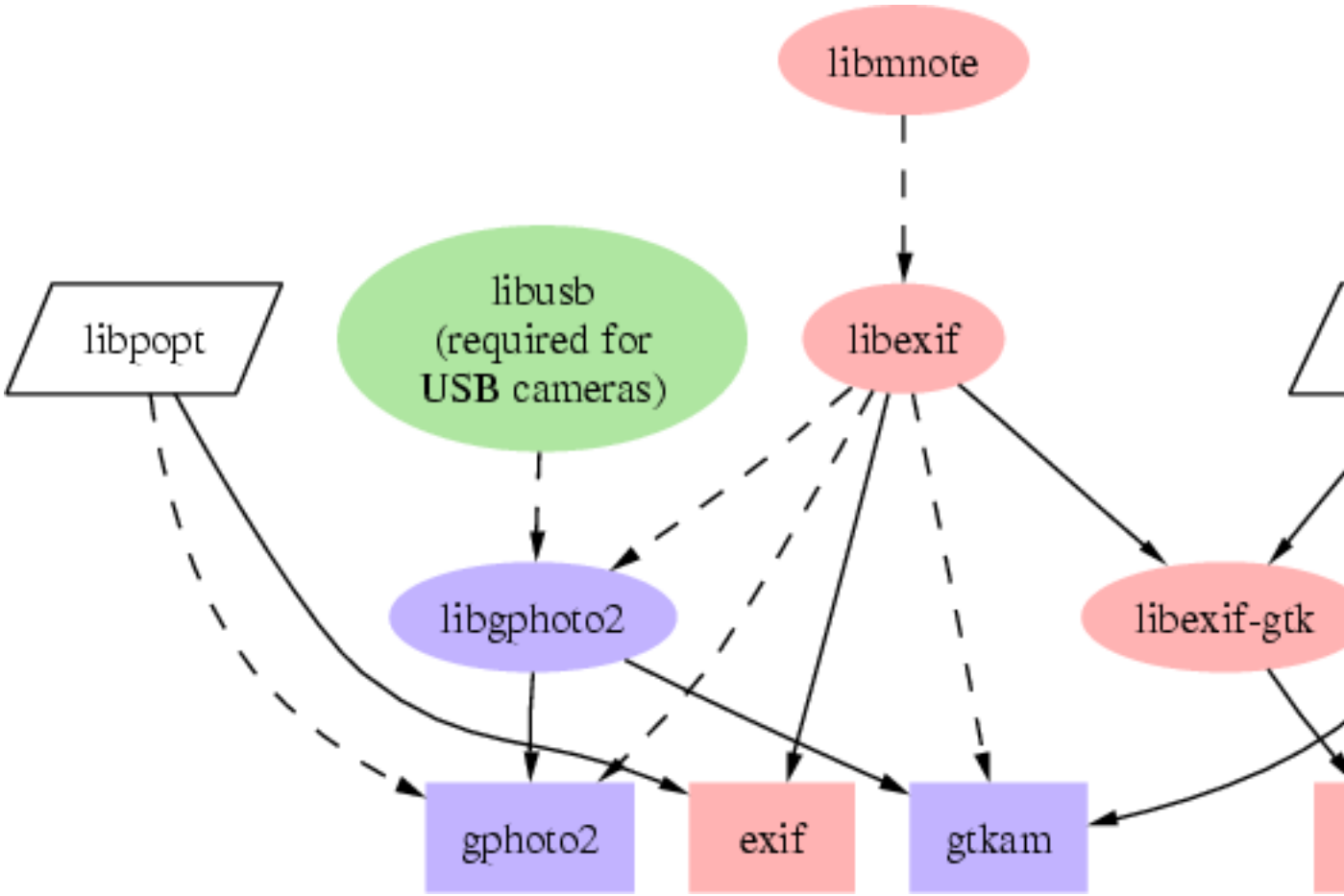
5.1. Overview, requirements and dependencies

Abstract

This section wants to give you an overview about software packages, software requirements and the dependencies between them. It should prepare you for the decision what packages you want to install and what sequence.

As you can see in Figure 5.1, multiple software packages are required for using digital cameras. If you find this picture a little complicated, then you may want to consider using pre-compiled binary packages.

Figure 5.1. Software dependencies




```
Arrows mean “required for”, dashed lines mean
“can optionally be used with”.
```

Nowadays, most people will want to use gphoto2 on the command line and the gtkam GUI with contemporary USB cameras delivering EXIF images. So you’ll have to get and compile libexif, libusb, libgphoto2, gphoto2, and gtkam.

You can see that this quite some work to do. This is why we created gphoto-meta, which lets you build everything in just one step. Continue reading in Section 5.2, “Using CVS sources via gphoto-meta” for a quick demonstration of gphoto-meta. If you want to do everything manually, skip that chapter and continue with Section 5.3, “Starting from CVS”.

5.2. Using CVS sources via gphoto-meta

Abstract

Fed up with typing **cvs up && ./autogen.sh && ./configure && make install** for each and every package? No problem. Use gphoto-meta. You just have to run one script which fetches and updates all required packages from CVS, fetches some build tools if necessary and then builds and installs all software packages in the correct sequence.

```
alice@host:~$ cvs -d :pserver:anonymous@cvs.sourceforge.net:/cvsroot/gphoto login
Logging in to :pserver:anonymous@cvs.sourceforge.net:2401/cvsroot/gphoto
CVS password:ENTER
alice@host:~$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/gphoto co gphoto-meta
co gphoto-meta
cvs server: Updating gphoto-meta
U gphoto-meta/.htaccess
U gphoto-meta/.htfixupload
U gphoto-meta/AUTHORS
U gphoto-meta/ChangeLog
U gphoto-meta/Makefile.am
U gphoto-meta/NEWS
U gphoto-meta/README
U gphoto-meta/autogen.sh
U gphoto-meta/bootstrap.sh
U gphoto-meta/build-tool-list
U gphoto-meta/clean.sh
U gphoto-meta/compileinstall.sh
U gphoto-meta/configure.in
U gphoto-meta/cvs-module-list
U gphoto-meta/cvs-module-list-candidates
U gphoto-meta/gettextize.patch
U gphoto-meta/upload-dist
cvs server: Updating gphoto-meta/utils
U gphoto-meta/utils/Makefile.am
U gphoto-meta/utils/common.sh
alice@host:~$ cd gphoto-meta
alice@host:~/gphoto-meta$ ./bootstrap.sh --tools
a few 10000 lines deleted
########################################################################
# Installed test versions of everything into $HOME/gphoto-meta/dist-root
# You may want to set the following variables to use the gphoto suite
# installed there:
########################################################################
export PATH="$HOME/gphoto-meta/dist-root/bin:$HOME/gphoto-meta/tool-root/bin:$PATH"
export LD_LIBRARY_PATH="$HOME/gphoto-meta/dist-root/lib:$HOME/gphoto-meta/tool-root/lib:"
export PKG_CONFIG_PATH="$HOME/gphoto-meta/dist-root/lib/pkgconfig:/usr/lib/pkgconfig"
########################################################################
#> cd $HOME/gphoto-meta/dist-files
Note: To be sure that the packages in $HOME/gphoto-meta/dist-files
 really work, run
    $HOME/gphoto-meta/compileinstall.sh
alice@host:~/gphoto-meta$
```

That’s it :-) Read the last few lines. If you just want to use what you just compiled, then just execute the three export statements given:

```
alice@host:~/gphoto-meta$ export
PATH="$HOME/gphoto-meta/dist-root/bin:$HOME/gphoto-meta/tool-root/bin:$PATH"
alice@host:~/gphoto-meta$ export
LD_LIBRARY_PATH="$HOME/gphoto-meta/dist-root/lib:$HOME/gphoto-meta/tool-root/lib:"
alice@host:~/gphoto-meta$ export
PKG_CONFIG_PATH="$HOME/gphoto-meta/dist-root/lib/pkgconfig:/usr/lib/pkgconfig"
```

However, if you want to install the software into another directory (called *$HOME/root* here), then use **compileinstall.sh**:

```
alice@host:~/gphoto-meta$ ./compileinstall.sh $HOME/root
another few 10000 lines deleted
alice@host:~/gphoto-meta$ export PATH=$HOME/root/bin:$PATH
alice@host:~/gphoto-meta$ export LD_LIBRARY_PATH=$HOME/root/lib:$LD_LIBRARY_PATH
alice@host:~/gphoto-meta$
```

You now have installed everything into the *$HOME/root* directory.

If you want to update an existing installation of gphoto-meta, you run the following to update your packages:

```
alice@host:~/gphoto-meta$ cvs up
output snipped
alice@host:~/gphoto-meta$ ./bootstrap.sh --tools --update
output snipped
alice@host:~/gphoto-meta$
```

Run **compileinstall.sh** if you want to install the newly created packages somewhere.

5.3. Starting from CVS

First find out what software packages you want. Fetch all these software packages with the packages they depend on, according to Figure 5.1.

You can get these programs via CVS. Detailed instructions are available from the project pages (libusb [http://sf.net/projects/libusb], libexif [http://sf.net/projects/libexif], gphoto [http://sf.net/projects/gphoto]).

All the software modules libusb, libexif, libexif-gtk, exif, gexif, libgphoto2, gphoto2, and gtkam use an automake-based build system. This build system has to be initialized before you can use it in the **./configure && make && make install** manner you know.

This initialization requires a few specialized build tools like **libtoolize**, **gettextize** and **pkg-config**. Make sure you have these installed. Then run the **./autogen.sh** script contained in each of the CVS modules which does the build system initialization for you.

Note

If **autogen.sh** (or rather the **gettextize** script it calls) asks you to acknowledge a few paragraphs and press enter, just press enter. **autogen.sh** already does the setup for you.

When the build system of all modules is initialized, you can start configuring and building the source, as described in Section 5.4, “Configuring, building, and installing”

5.4. Configuring, building, and installing

WRITEME :-P (PKG_CONFIG_PATH, ./configure, make, make install, /etc/ld.so.conf, LD_LIBRARY_PATH, PATH, yadda yadda)

Chapter 6. Developer Documentation: The Inner Workings

Abstract

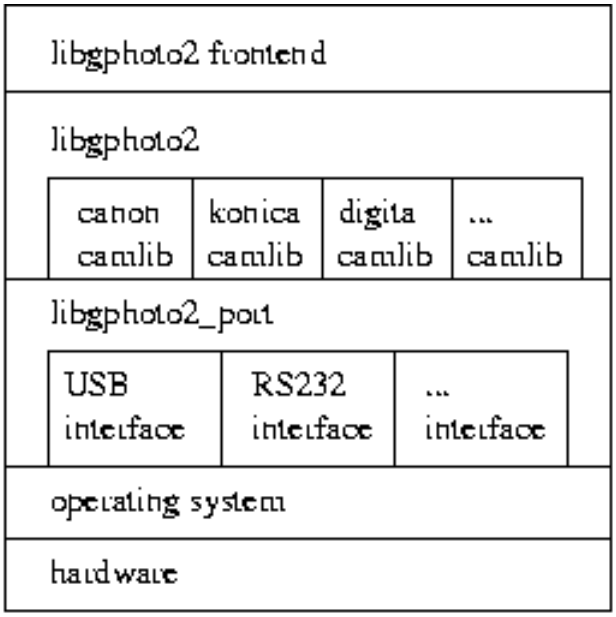
How it works internally, how you can work on it and how you can use it in your own software.

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.

6.1. The gPhoto2 software architecture

Figure 6.1. The gPhoto2 system architecture

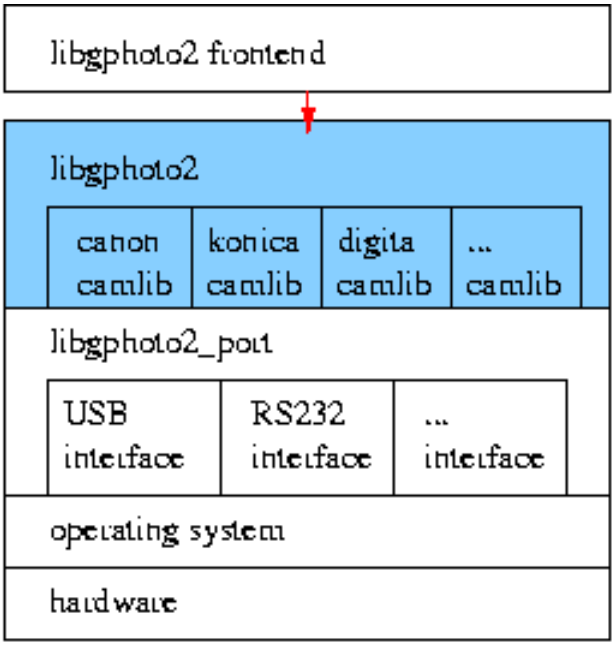


$Id: architecture.fig,v 1.1 2002/08/17 23:09:55 hun Exp $

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

6.2. The `libgphoto2` API

Figure 6.2. The libgphoto2 API within the gPhoto2 system architecture

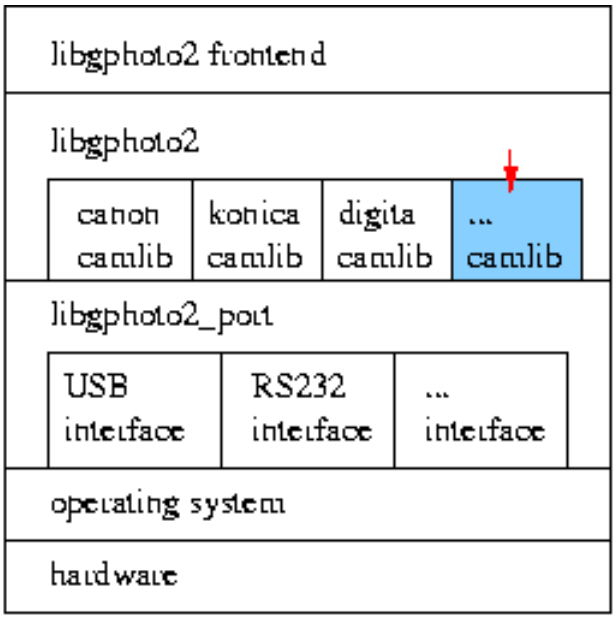


$Id: libgphoto2-api.fig,v 1.1 2002/08/17 23:09:55 hun Exp $

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

6.3. The `camlib` API

Figure 6.3. The camlib API within the gPhoto2 system architecture

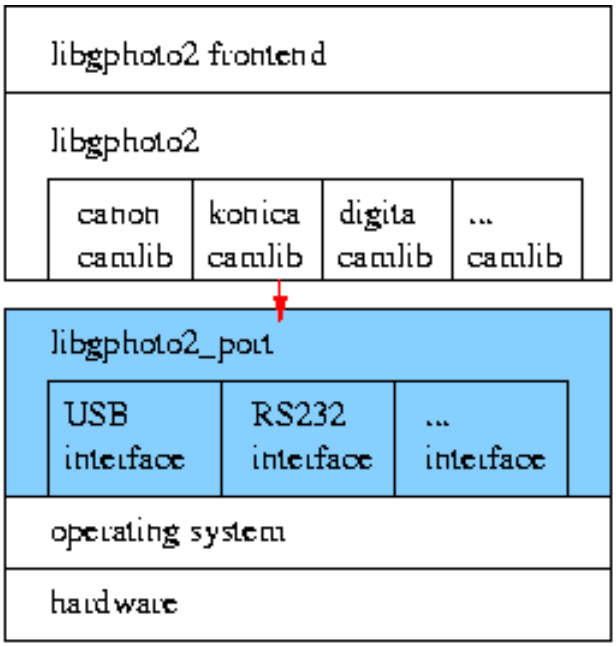


$Id: libgphoto2-camlib.fig,v 1.1 2002/08/17 23:09:55 hun Exp $

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

6.4. The `libgphoto2_port` API

Figure 6.4. The libgphoto2_port API within the gPhoto2 system architecture



$Id: libgphoto2_port.fig,v 1.1 2002/08/17 23:09:55 hun Exp $

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

Chapter 7. Utopia: A look into the possible future

Abstract

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

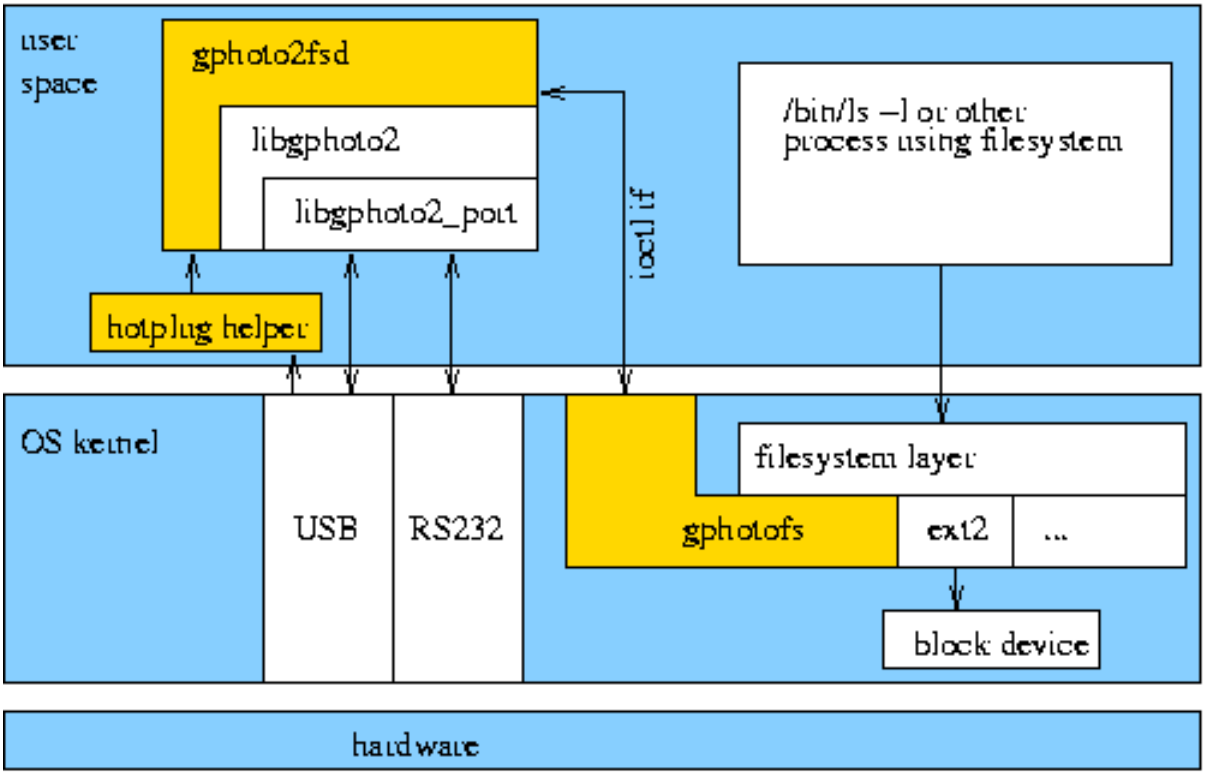
7.1. Language Bindings

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...

7.2. The gPhoto2 file system

Figure 7.1. 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 1.1 2002/08/17 23:09:55 hun Exp $

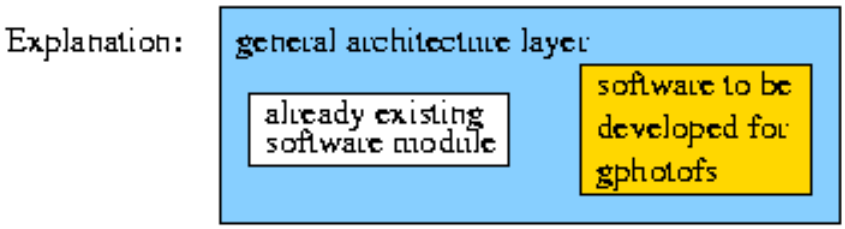


Diagram describing where the gPhoto2 file system is located within the gPhoto2 and system software architecture

The gPhoto2 Reference (the man pages)

Name

gphoto2 -- command-line `gphoto2` client

gphoto2
`gphoto2` [--debug] [[-q] | [--quiet]] [[-v] | [--verbose]] [[-h] | [--help]]
[--list-cameras] [--list-ports] [--stdout]
[--stdout-size] [--auto-detect] [--port *PATH*]
[--speed *SPEED*] [--camera *MODEL*] [--filename *FILENAME*]
[--usbid *USBIDS*] [[-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

libgphoto2(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.
`--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 the filename to use when saving downloaded files. The `--filename` option accepts %a, %A, %b, %B, %d, %H, %k, %I, %l, %j, %m, %M, %S, %y, %%, (see date(1)) and, in addition, %n for the number, %C for the filename suffix, and %f for the filename without suffix.

`--usbid` *`USBIDS`* (Expert only) Override USB IDs. *`USBIDS`* must be of the form *`DetectedVendorID`*:*`DetectedProductID`*=*`TreatAsVendorID`*:*`TreatAsProductID`* to treat any USB device detected as *`DetectedVendorID`*:*`DetectedProductID`* as *`TreatAsVendorID`*:*`TreatAsProductID`* instead. All the VendorIDs and ProductIDs should be hexadecimal numbers beginning in C notation, i.e. beginning with '0x'.

Example: *`--usbid 0x4a9:0x306b=0x4a9:0x306c`*

`-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

libgphoto2(3), The gPhoto2 Manual, http://www.gphoto.org/

Examples

`gphoto2 --list-ports`	Shows what kinds of ports (USB and serial) you have.
`gphoto2 --auto-detect`	Shows what camera(s) you have connected.
`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`**.

Name

libgphoto2 -- cross-platform digital camera library

libgphoto2

```
#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), libgphoto2_port(3), The gPhoto2 Manual, the automatically generated API docs, http://www.gphoto.org/

Name

libgphoto2_port -- cross-platform port access library

libgphoto2_port

```
#include <gphoto2_port.h>
```

Description

The `libgphoto2_port` library was written to provide libgphoto2(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

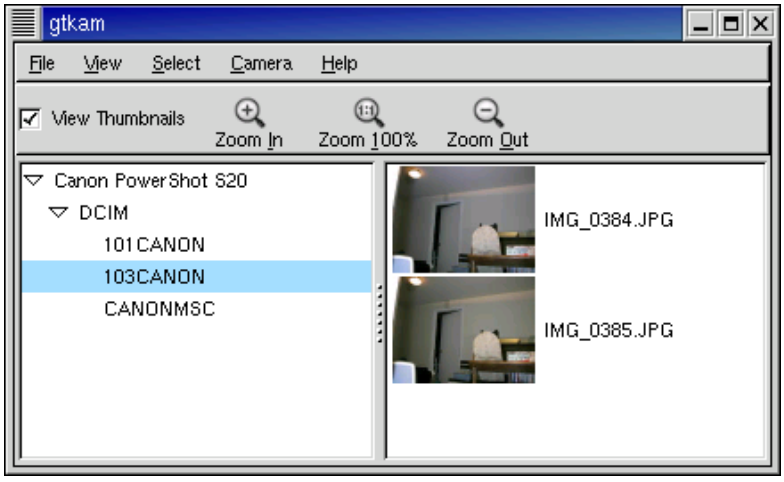
libgphoto2(3), The gPhoto2 Manual, http://www.gphoto.org/, the automatically generated API docs, http://libusb.sourceforge.net/

Name

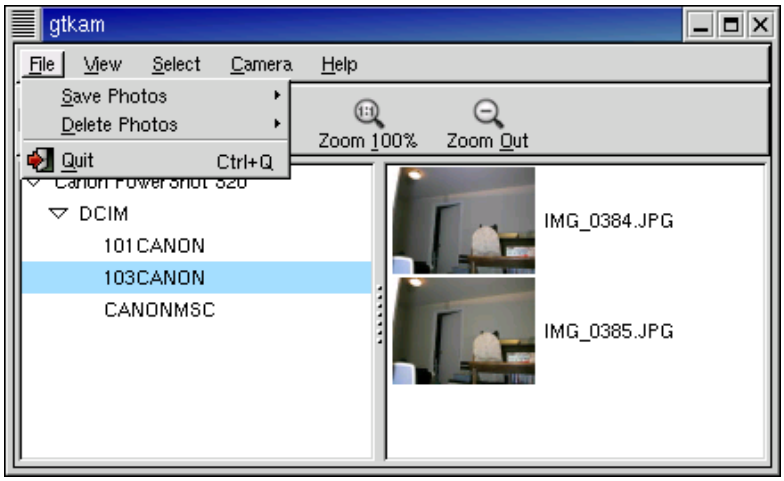
gtkam -- Graphical front end for gphoto2

gtkam

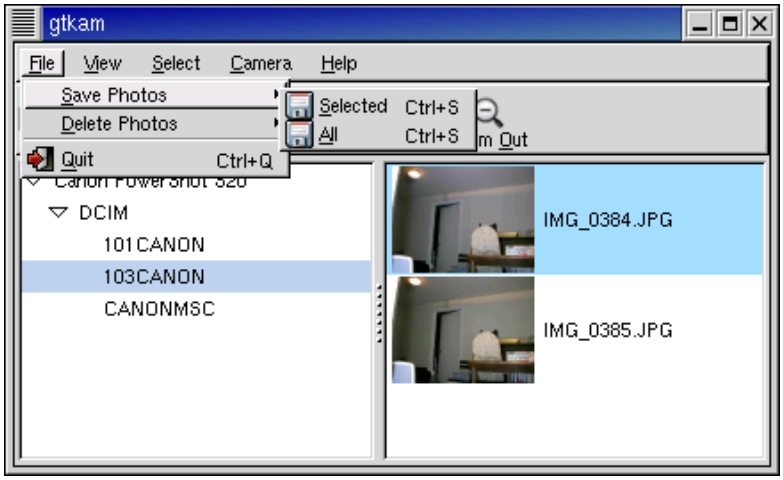
Commands



File Menu



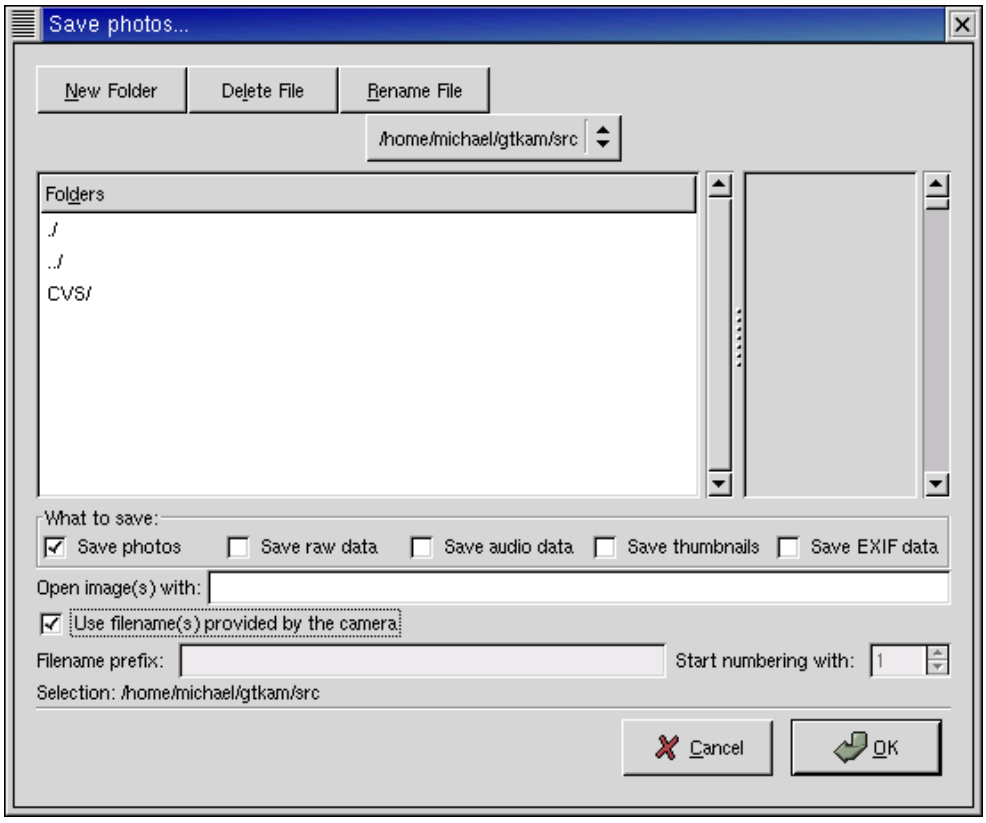
File->Save Photos Gives options to save photos to the computer.



File->Save Photos->Selected — Saves the selected photos from the camera to the computer.

File->Save Photos->All — Saves all photos in the camera to the computer.

Both menu items open a "Save photos..." dialog box which allows the user to select the folder to save the images in, as well as a number of other options.



What to save: — This is a set of checkboxes which allows the user to select the forms in which data for the selected image(s) will be saved. Any or all of these options may be selected at once.

Save photos	This is the default. It causes the photos being saved to be downloaded from the camera and saved to the identified folder.
Save raw data	FIXME: What to say?
Save audio data	Causes any audio data associated with the image to be saved to the selected folder.
Save thumbnails	Saves a small version of the selected images to the identified folder.
Save EXIF data	Saves the EXIF data associated with the images. FIXME: should we say more about the EXIF format?

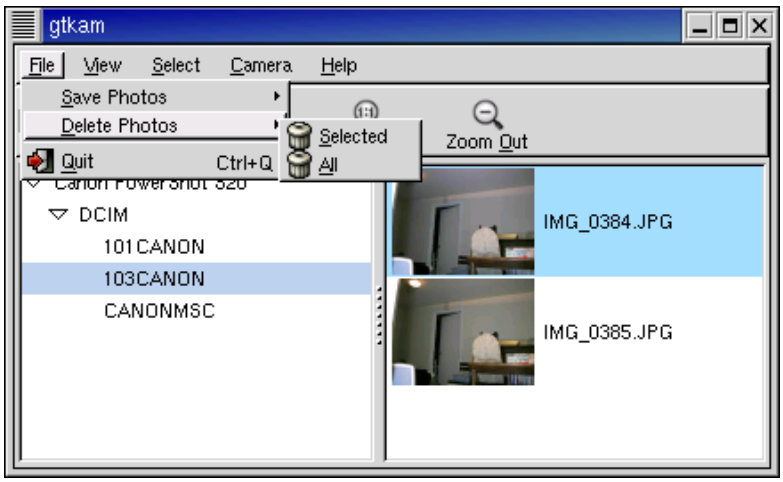
Open image(s) with: This allows the user to specify FIXME ??

Use filename(s) provided by the camera. Causes the saved files to have the names provided by the camera.

Filename prefix: Defines a prefix to be used when creating filenames. This prefix is combined with the number starting with the number defined in:

Start numbering with: Defines the first number of the sequence of photos to be stored.

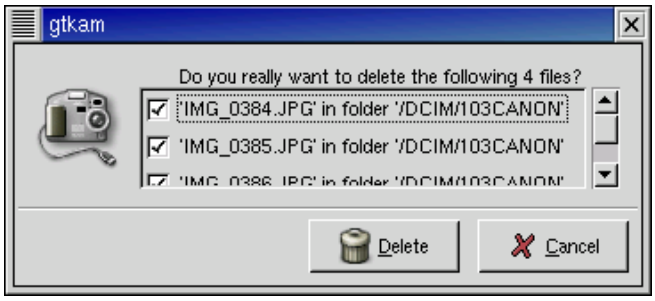
File->Delete Photos Gives options to delete photos from the camera.



File->Delete Photos->Selected Deletes selected photos from the camera.

File->Delete Photos->All — Deletes all photos in the camera.

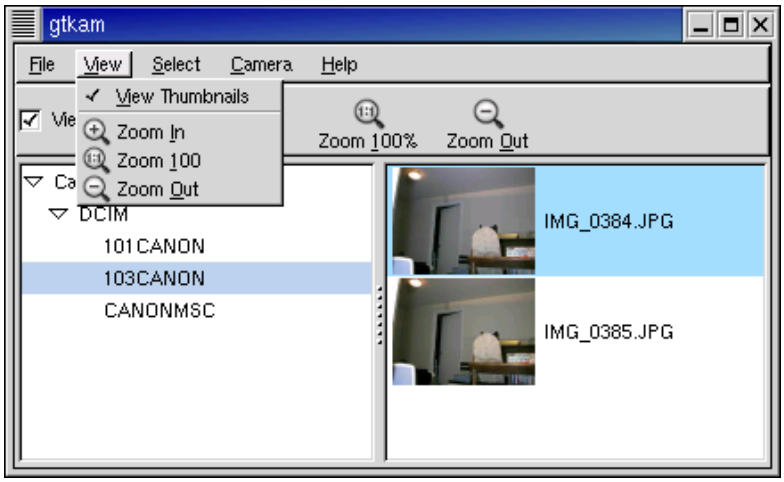
Both of these menu items open a dialog to confirm the deletion of the photos. A list of photos is presented with a checkbox beside each photo name. Uncheck the box to prevent a specific file from being deleted.



File->Quit — Quits gtkam.

View Menu

Gives options for previewing photos before downloading them from the camera.



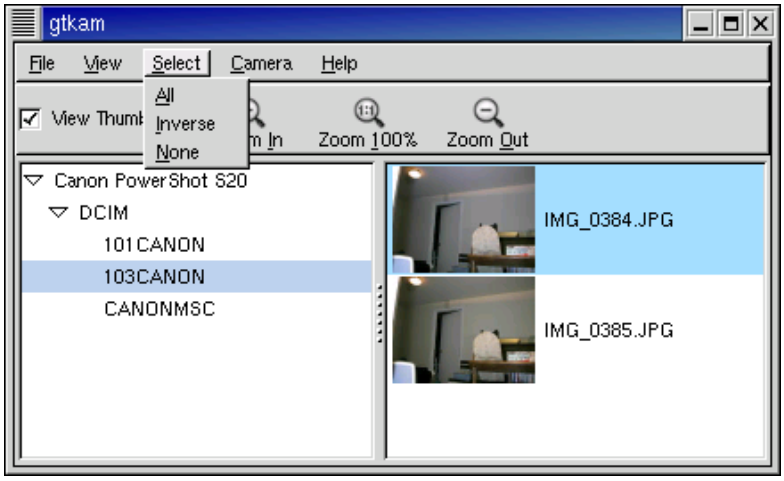
View->View Thumbnails — This is the same command as the checkbox on the main gtkam window. It allows the user to select whether thumbnails of the photos are displayed, or just the file names.

View->Zoom In — Enlarges the thumbnails.

View->Zoom 100 — Returns the thumbnails to default size.

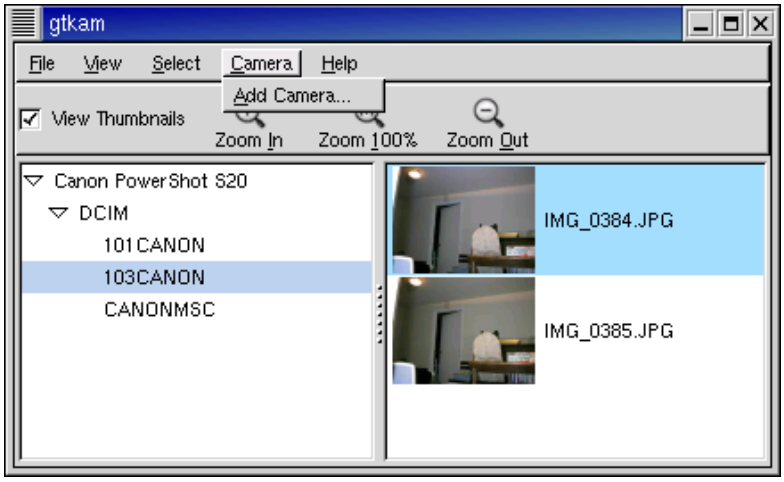
View->Zoom Out — Reduces the thumbnails.

Select Menu

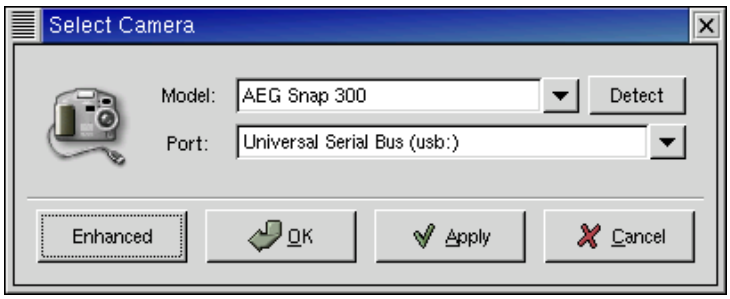


Select->All	Selects all pictures in the camera.
Select->Inverse	Inverts the selections already made so that previously unselected images are now selected, and previously selected images are now unselected.
Select->None	Undoes all selections so that no images are selected.

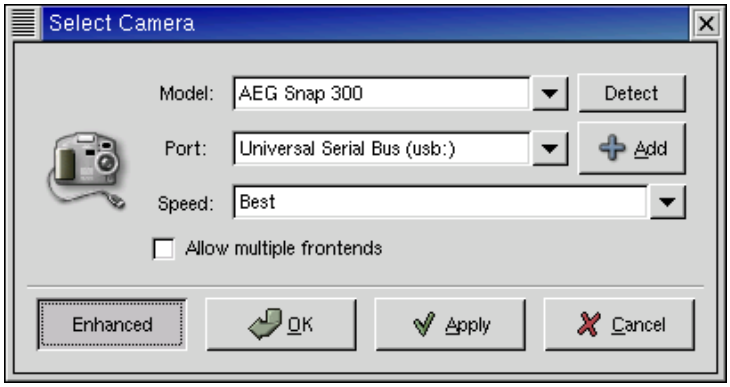
Camera Menu



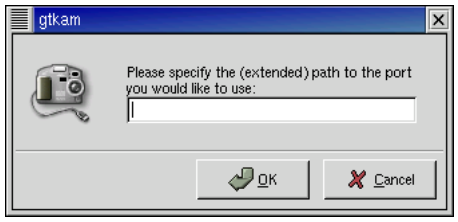
Camera->Add Camera... Opens a window to allow the user to define the camera and interface being used.



"Simple" Select Camera Dialog.

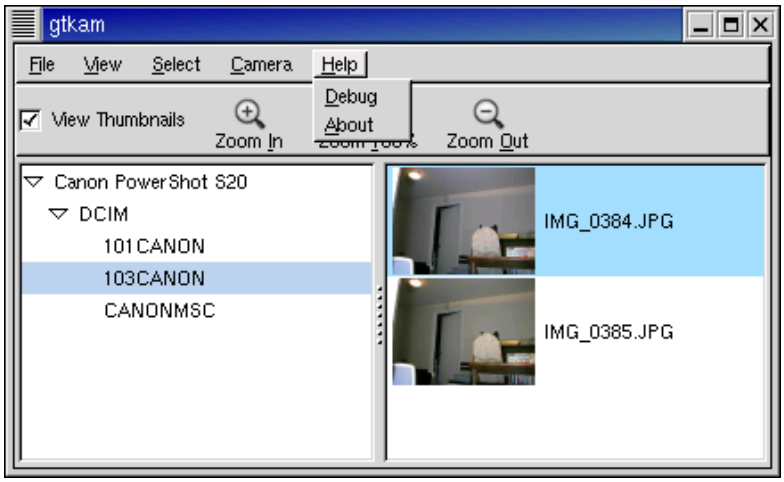


"Enhanced" Select Camera Dialog.

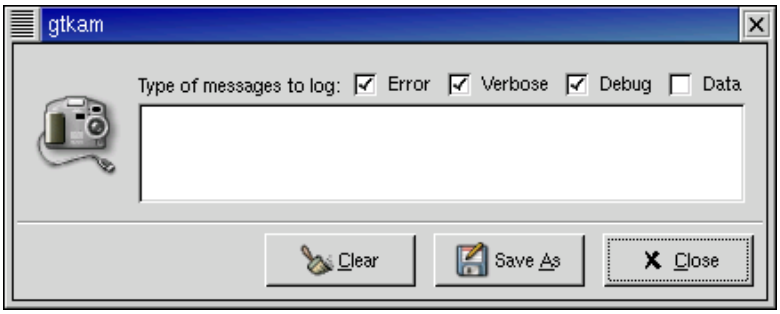


Dialog for explicitly adding a path to the port to which the camera is attached.

Help Menu

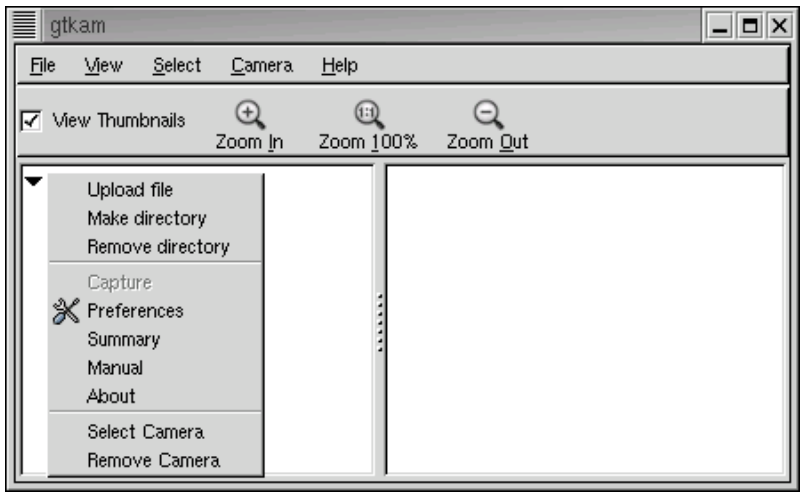


Help->Debug... Opens the logging window which allows the user to select the type of messages to log.

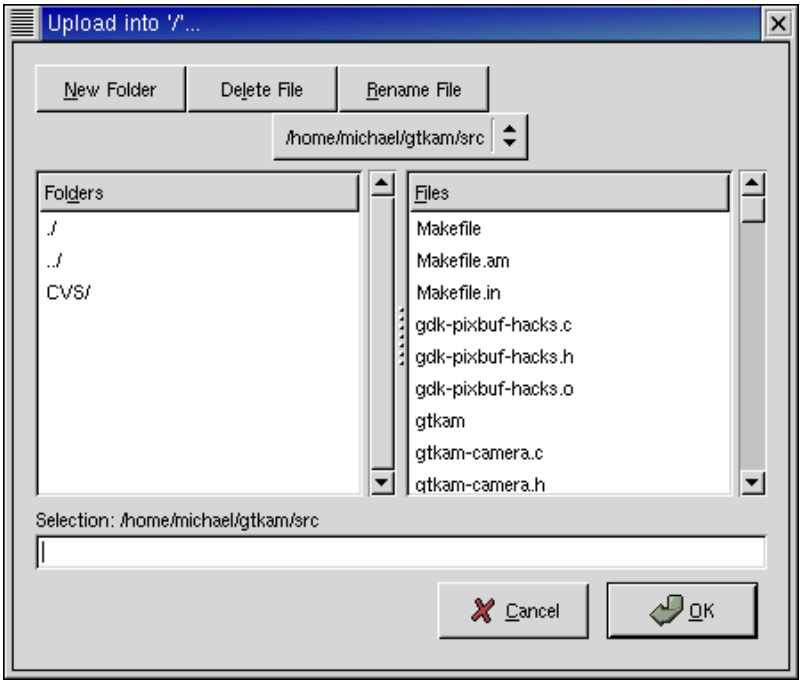


	Error	causes the logging of any error messages.
	Verbose	Increases the amount of information being logged. (I can't see what effect this has. - mjr)
	Debug	Causes the logging of progress reports as the software communicates with the camers.
	Data	Causes the logging of all data being transferred from the camera. This will generate huge quantities of hexadecimal information.
	Save As...	opens a dialog to allow the user to save the information logged in the debug window.
	Close	Closes the debug window
Help->About...	Opens the about gtKam window. This gives the current version, author credits, and contact information.	

Right Click (over directory pane) Menu



Right Click->Upload File	Opens a dialog to identify an image file for uploading to the camera.

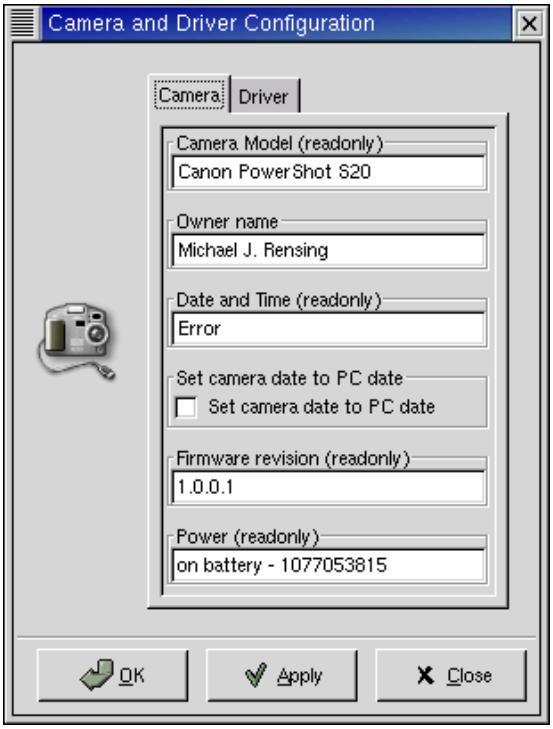


Right Click->Make directory Opens dialog to create new directory.

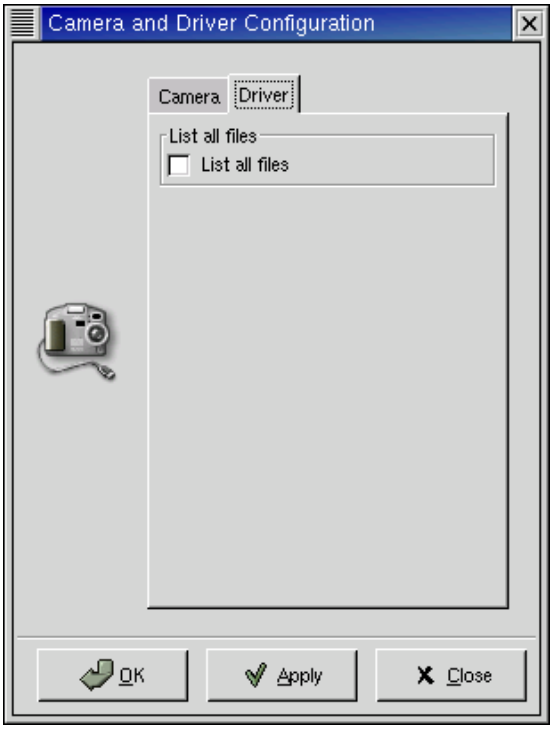
Right Click->Remove directory Opens a dialog to remove (delete) an existing folder (directory).

Right Click->Capture Opens a window to allow the user to trigger the camera to acquire an image if the camera and its gphoto2 driver supports this feature.. (This feature is not available on my camera, so I would appreciate some input on this item. -mjr)

Right Click->Preferences



Camera configuration tab.



Driver configuration tab.

Right Click->Summary — Displays a window with information about the camera and its memory.

Right Click->Manual — I'm not sure what this should contain. Anyone? - mjr

FIXME: need useful image

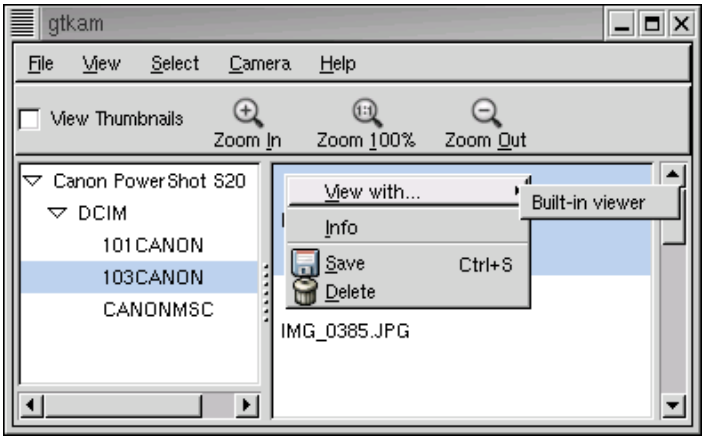
Right Click->About — Displays a window with information about the camera driver and its authors.

This should include contact information for the maintainers of the drivers for your camera.

Right Click->Select Camera — Opens the Add Camera dialog.

Right Click->Remove Camera — Removes the camera from the directory pane.

Right Click (over list pane) Menu



Right Click->View with...->Built-in viewer — Downloads the image under the cursor and displays it in a new window. The window automatically resizes the image to fill the window.

Right Click->Info — Presents file name, type, size, date and permissions information about the image under the cursor.

Right Click->Save — Downloads the image under the cursor and opens a dialog to allow it to be saved to the computer.

Right Click->Delete — Opens a dialog to allow the user to delete the image under the cursor.

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. `libusb` allows C programs to access the USB interface of all supported operating systems in an OS independent manner.

http://sourceforge.net/projects/libexif/ The `libexif` project page. `libexif` allows programs written in C to access the metainformation from the EXIF tags in the JPEG files most modern cameras produce.

http://www.teaser.fr/~hfiguiere/linux/digicam.html Hubert Figuiere's digicam support list

http://n-dimensional.de/projects/digicam/ Hans Ulrich Niedermann's home page (one of the developers), non-daily CVS snapshot tarballs [http://n-dimensional.de/projects/digicam/software/snapshots/] and patches [http://n-dimensional.de/projects/digicam/software/snapshots/patches/], and packages [http://n-dimensional.de/projects/digicam/software/] which are updated from time to time.

http://aamot.org/ole/photography.html Ole Aamot's page about "Using Digital Still Photography Devices with GNU/Linux". The next two links were stolen from there.

http://www.deater.net/weave/vmwprod/agfa/ Vince's page about support of Agfa cameras

http://www.jedi.com/obiwan/linux-digicam.html Obi-Wan's page about "Supporting a Canon S110 Elph Digital Camera under Debian Linux (with a review of the camera and sample photos)"

Glossary

This is not a real glossary (yet), it’s just an example.

EXIF: Exchangeable Image File	Most current digital camera store images using Exif compressed files. Exif compressed files use the baseline JPEG DCT format specified in ISO/IEC 10918-1. This means the image data can be read by any application supporting "JPEG", including essentially all web browsers and image editing, desktop presentation, and document creation applications. In addition, Exif/JPEG stores metadata within application segments at the beginning of the file, and uses sRGB as the default color space.
gphoto	The glorious application by Scott Fritzinger which started everything.
gphoto2	Glorious successor of gphoto. Can make use of EXIF.