photopc

Picture Mining

There are lots of GUI tools available for accessing pictures stored on digital cameras, but we are going to take a look at a tool for the command line, photopc, which is useful for automating tasks in a scripted environment.

BY HEIKE JURZIK

GUI tools allow you easy access via preview functions, but if you regularly produce a large number of picures that you will be processing on your computer, you will appreciate a tool that you can use for shell scripting – a tool such as *photopc*.

Installing the Source Code

The subscription CD in this issue includes the files *photopc_3.05.tar.gz* [1] and *photopc-3.05J23.tar.gz* (USB support) [2] in the *LinuxUser/photopc/* directory. You should install only one of these packages, depending on whether you will be attaching your camera to the USB port. After mounting the CD, ensure that you are the superuser, *root*, and follow these steps:

```
asteroid:~# cd /usr/local/src/↵
asteroid:/usr/local/src# tar ↵
xzvf /cdrom/LinuxUser/photopc/↵
photopcXY.tar.gz
```

This creates a new directory called *photopcXY*. Now change to the directory (*cd photopcXY*) and type *./configure*. If everything works out correctly, you should see the following:

```
creating ./config.status
creating Makefile
creating config.h
config.h is unchanged
creating dos/version.h
dos/version.h is unchanged
creating win32/version.h
win32/version.h is unchanged
```

You can then go on to complete the final two steps:

```
asteroid:/usr/local/src/↵
photopcXY# make
[...]
asteroid:/usr/local/src/↵
```

KNOW HOW

Although GUIs such as KDE or GNOME are useful for various tasks, if you intend to get the most out of your Linux machine, you will need to revert to the good old command line from time to time. Apart from that, you will probably be confronted with various scenarios where some working knowledge will be extremely useful in finding your way through the command line jungle.

```
photopcXY# make install
[...]
```

If everything worked out, and no error messages were displayed, you will find that the program has been installed to */usr/local/bin*.

Before You Start

Your camera will either be attached to your USB port via a USB lead, or to a serial port using a serial lead – this depends on your computer and the type of camera you are using. If the camera is attached to a serial port, you will need to access the port explicitly, using the *-l* flag (that is a lower-case "l" as in "Lima") and the device name, each time you launch the program. To simplify this, you can create a **symbolic link** for the device. If the camera is attached to the first serial port, ensure that you are the superuser, *root*, and then enter the following command:

```
asteroid:~# ln -s ↵
/dev/ttyS0 /dev/photopc
```

The superuser, *root*, can now use the tool without any trouble, but "mere mortal" users will need access privileges before they can access the program as planned. As already mentioned, the symlink */dev/photopc* points to the serial interface to which the camera is attached. In order to communicate with

the camera without *root* privileges, your users will need write privileges for the device. You can check the access privileges for the interface using the *ls -l* command (see also Box 1):

```
asteroid:~# ls -l /dev/ttyS0
crw-rw----   1 root   dialout
4,  64 Jun 30 16:30 /dev/ttyS0
```

The first character represents the file type – in this case it is a "c" for "character device". The "r" refers to read privileges and the "w" to write privileges. They could also be followed by an "x", for executable, i.e. the right to launch the file. The first group of three characters refers to the owner of the file, the next three to the group and the last three represent any other users on the system. For the serial interface in our example this means that *root* (the owner of the file), and the members of the *dialout* group, which was created by the Debian Woody distribution for this device, have read and write privileges. (This would still apply if the group had a different name). To check whether you are a member of this group, use your normal user account and type *groups*:

```
huhn@asteroid:~$ groups
users cdrom floppy sudo audio ➥
video dos cdwrite
```

To add the user *huhn* to the *dialout* group, ensure that you are the superuser, *root*, and edit the */etc/group* file. Look for the *dialout* group in this file and add the user as required. (To add multiple users simply user a comma separated list). The entry in the */etc/group* file will thus read:

```
dialout:x:20:huhn
```

If you use shadow passwords for groups (file */etc/gshadow*), you will need to edit this file and add the user to the group:

```
dialout:*::huhn
```

GLOSSARY

Symbolic link: *A link to another file that is treated by the application program exactly as the file would be. If you delete the file the symlink points to, any commands using the link will be pointing into empty space. Symlinks are created using the "ln -s" command.*

To apply these changes type *newgrp dialout* while logged in as a normal user. Alternatively, just log on again to enable the new group membership.

Depending on your distribution there are varying approaches to working with USB. Some systems may allow you to simply attach your camera to the USB port and power the camera on. If this does not work, you can refer to the approach shown here to access your USB camera via *photopc*. Make sure that you check your access privileges for */proc/bus/usb* first. Then attach your camera to the USB port, fire up the camera and type:

```
asteroid:~# ls -l ➥
/proc/bus/usb/001
total 0
-rw-rw-r--   1 root   root   ➥
18 Jul 31 17:19  001
-rw-rw-r--   1 root   root   ➥
18 Jul 31 17:19  002
```

Mere mortal users are apparently not allowed to talk to the camera as both devices belong to the *root* user and group. But changing that requires only a few steps. As previously seen in the example with the serial port, first ensure that you are the superuser, *root*, and then edit the */etc/group* file. Add a new group called *usb*. The numbers for system groups are usually in the range 0 through 99. User groups follow from 100 upward – although exceptions are possible. Locate an unused number for the group (we used 51 on our test system) and enter the group number with your account name:

```
usb:x:51:huhn
```

If shadow passwords are used, you will need to add an entry to */etc/gshadow*:

```
usb:*::huhn
```

Now all we need is an entry in */etc/fstab*:

```
none /proc/bus/usb usbdevfs ➥
auto,devmode=0664,devgid=➥
51   0   0
```

This modifies the file rights for USB devices to allow members of group 51, that is *usb*, read and write access. There is also a way to provide a shortcut for launching the program. You would normally need to type the *-u* parameter explicitly in order to tell *photopc* to use the Universal Serial Bus. However, the bash command *alias* can save you a lot of typing: Add the following entry to your *.bashrc*:

```
alias photopc='photopc -u'
```

Now call *source ~/.bashrc* to update the system with the modified file.

Off to Work!

Assuming that everything is configured right, contacting the camera should be no problem:

Box 1: Interface names in Linux

If you take a look at your */dev* directory, you will find a number of entries that appear somewhat cryptic at first sight. Linux uses device files to communicate with hardware devices (such as hard disks, floppy drives, mice, sound cards and so on). The first character is either *b* (for "block device") or *c* (for "character device"), and represents the access mode. Important device names are:

- */dev/hd** - IDE drives
- */dev/lp** - parallel ports (printers etc.)
- */dev/sd** - SCSI drives
- */dev/scd** - SCSI CD ROM drives
- */dev/tty** - virtual terminals
- */dev/ttyS** - serial ports

Some of these device entries are represented by links, for example you can access the */dev/cdrom* entry on */dev/hdc* (a CD ROM drive attached to the second IDE bus) and */dev/mouse* on */dev/ttyS1* (the second serial port).

The USB device file system is generated dynamically in a similar fashion to the */proc* file system, and is normally to be found under */proc/bus/usb*. Directories following the "oon" pattern contain the ports for active USB devices. The files *devices* and *drivers* contain an overview of the devices currently attached and any drivers assigned to them. The kernel is responsible for creating these directories (provided it can support USB). As the data in */proc/bus/usb/devices* is quite extensive, you will probably want to use an X application, such as *usbview*, to keep track of all the attached USB devices.

```
huhn@asteroid:~$ photopc query
Found usb device id 0x100 by
vendor 0x7b4
Found usb camera: Olympus
Optical Co., Ltd. C-2100/C3000/
C3040 Camera
Starting in folder "\DCIM\
1000LYMP"
Resolution: 7 - SQ1-1280x960
-Normal
Camera time: Wed Jul 31
22:22:19 2002 CEST
[...]
```

You can launch *photopc* with the *-h* (help) flag set to display a complete overview of the available parameters and command options. You will need to use the *less* pager to prevent the output simply scrolling off screen:

```
huhn@asteroid:~$ photopc
-h | less
```

The *count* command will count the pictures on the camera. If you need a more precise overview, you can try *list* instead (Listing 1).

The file names shown here might seem a little cryptic – these are the camera's internal names for the image files. When you download the images *photopc* uses a default format "MMDD_NNN.jpg" (month, day and number) for storing the images on your hard disk. Launch the *image* to do so:

```
huhn@asteroid:~$ photopc
image 1 .
Found usb device id 0x100 by
vendor 0x7b4
Found usb camera: Olympus
Optical Co., Ltd. C-2100/C3000/
C3040 Camera
Starting in folder "\DCIM\
1000LYMP"
1: 279907 of 279907
taken Fri Jul 05 20:05:44 2002
 CEST
file "./0705_001.jpg"
```

This syntax downloads the first image on the camera to the current working directory (represented by the period, "."). If you want to store multiple images, you can designate a range (e.g. *photopc image 1-5 .*), or supply a comma-separated list of images (e.g. *photopc image 1,2,5 .*). The *thumbnail* command is extremely useful – instead of downloading the full images, you can create miniatures.

Fully Automatic

The options shown so far are available in other programs, of course. What makes command-line tools so special is the fact that you can integrate them neatly into shell scripts. This allows you to combine single steps effectively. The script given in Listing 2 shows an interactive script for *photopc*. ■

Listing 1: Output of the list command

```
huhn@asteroid:~$ photopc list
No.     Size    R P        Date and Time                 Filename    [...]
45  45  262400 83887040 - Mon Jul 29 23:24:43 2002 CEST P7292878.JPG
46  46  276982 83887040 - Wed Jul 31 16:39:24 2002 CEST P7312879.JPG
47  47  275218 83887040 - Wed Jul 31 18:41:30 2002 CEST P7312880.JPG
```

Listing 2: photopc script

```
#!/bin/bash

# define photopc call type (for USB in this example, use "photopc" for
serial)
PHOTOPC="photopc -u"

# Create target directory (if not already created)
echo "Type the name of the directory:"
read mydir
mkdir -p $mydir || exit 1

# Count pictures -- uses the last line of output
number=`$PHOTOPC count | tail -1`
echo "There are $number pictures on the camera."

# If a list is required, pipe output to more
echo "Would you like a list of pictures?"
read answ
if [ $answ = "y" ]
then
  $PHOTOPC list | more || exit 1
fi

# Selecting pictures images
echo "Which pictures would you like to download?
(poss. entries e.g.: 1 or 1-30 or 1,2,10)"
read range
$PHOTOPC image $range $mydir || exit 1

# Thumbnail selection
echo "Would you like thumbnails of the pictures? (y/n)"
read answ
if [ $answ = "y" ]
then
  $PHOTOPC thumbnail $range $mydir || exit 1
else
  echo "No thumbnails requested."
fi
```

INFO

[1] *http://photopc.sourceforge.net/*

[2] *http://www.math.ualberta.ca/imaging/*

[3] *http://www.lightner.net/lightner/bruce/ppc_use.html*