Logitech

- Logitech Quickcam Messenger unter Linux

Logitech Quickcam Messenger unter Linux

Eine Kurzanleitung um den Logitech Quickcam Messenger USB unter Linux zu betreiben.

- Einen modifizierten Treiber für die Quickcam Messenger Modelle gibts hier: http://home.mag.cx/messenger/
- Installation des Treibers:

```
tar xvfz qc-usb-messenger-<version>.tar.gz
cd qc-usb-messenger-<version>
make install
modprobe quickcam
```

- im dmesg sollte jetzt etwa folgendes auftauchen:

```
Linux video capture interface: v1.00
quickcam [16.302906]: ----------LOADING QUICKCAM MODULE------------
quickcam [16.302919]: struct quickcam size: 4196
quickcam: QuickCam USB camera found (driver version QuickCam Messenger/Communicate USB 1.3 $Date
quickcam: Kernel:2.6.16-1.2133_FC5smp bus:1 class:FF subclass:FF vendor:046D product:08F0
quickcam [16.304103]: poisoning qc in qc_usb_init
quickcam [16.307867]: E00A contains 08F0
quickcam: Sensor VV6450 detected
input: Quickcam snapshot button as /class/input/input3
quickcam [16.310227]: Quickcam snapshot button registered on usb-0000:00:1f.2-1/input0
quickcam: Registered device: /dev/video0
usbcore: registered new driver quickcam
```

- nun können wir die Kamera verwenden, im Beispiel auf /dev/video0

für meine Homepage verwende ich das Programm camE:

- Installation:

```
yum -y install camE
```

- Konfigurationsübersicht:
  - ich verwende kein FTP (camE kann FTP und scp)
  - Bilder werden nur lokal verarbeitet)
  - jede Sekunde wird ein Bild aufgenommen
  - Bilder werden nicht archiviert
  - folgende Pfade müssen in der config angepasst werden: infofile, archive, archive_thumbnails_dir, temp_file
  - temp_file = <httpdocroot>/webcam/webcam.jpg – dies ist das eigentliche Bild, das übertragen wird

~/.camErc

```
[ftp]
host = cam.super.tasty.men.com
user = giblet
pass = NO_WAY_BUDDY
dir  = public_html/images
```

```
# where should the file end up? Also, this extension determines the file
# type the image is saved as. Try image.png for a png.
file = webcam.jpg
# camE uploads to a temp file, and moves it across when done
# this way people don't view half-uploaded images
tmp  = uploading.jpg
# keep the connection open (1) or reopen it for each shot (0)
keepalive = 1
# do passive ftp?
passive = 1
#an interface to use for non-passive ftp.  use "-" to let libcurl choose, or
#use a real interface name.  (libcurl often chooses incorrectly)
interface = -
# ftp debugging? (noisy)
debug = 0
# Actually do the upload? If do = 0, just take and archive pics.
do = 0
# Some servers require us to explicitly delete the previous image
# In that case, enable this option
delete_first = 0
# Determines how many shots are taken before an image is uploaded.
# (1 == every picture is uploaded, 10 would be every 10th image)
# (Defaults to 1 if not present)
upload_every = 1

# you can set ftp->do to 0 above and use scp instead - you still need
# the dir, file and tmp settings in the ftp section for this to work.
# scp also honors the upload_every setting from the ftp section, and
# will also default to a value of 1 if not present.

[scp]
# target = user@host

[grab]
device = /dev/video0
# store temp image on local machine
temp_file = <httpdocroot>/webcam/webcam.jpg
# lag reduction, takes 5 shots, discards the first 4, thus clearing mmap
# buffers
lag_reduce = 5
# This goes at the bottom left, with the message from "infofile" appended.
# It is run through strftime, so date vars are expanded.
text   = %d/%m/%Y %T %Z -
#width  = 352
#height = 288
# delay between uploading one shot and starting the next
delay  = 1
# do we want to correct the delay for a slow connect?
# (keeps the perpetually updating clients in sync)
correct = 1
# scale image resolution dynamically based on bandwidth?
# percentage of the delay to spend uploading the image,
# 100 disables, useful values are < 40
percent = 100

############################################################
# PWC specific features (only for philps cams right now)

# framerate of cam capture, lower fps -> less grainy image, more chance or
# blurred motion
framerate = 20

# image settings (0-100)
colour = 50
brightness = 50
contrast = 50
hue = 50
whiteness = 50

# White balance mode
# can be "auto", "indoor", "outdoor", "fluorescent" or "manual"
pwc_wb_mode = auto
# if _mode is set to manual, these two controls affect the balance
# (0-100)
pwc_wb_red = 50
pwc_wb_blue = 50

############################################################
```

```
# where to log activity. comment out this line to disable logging
logfile = /dev/null
# gets the message text from here. one line allowed only. means you can do
# stuff like echo "sleeping and stuff" > ~/.caminfo
infofile = <homedir>/.caminfo
# directory to archive pics in. They are datestamped and saved in here.
archive = <httpdocroot>/webcam
# archive pics in datestamped subdirs
# (1 == with subdirs, 0 == without subdirs)
archive_subdirs = 0
# extension (determines type) of archived images.
archive_ext = jpg
# determines how many shots are taken before a pic is archived
# (1 == every pic, 0 == don't archive)
archive_shot_every = 0
# create archive thumbnails enable/disable flag and give width/height
archive_thumbnails_dir    = <httpdocroot>/webcam/thumbnails
archive_thumbnails_create = 0
archive_thumbnails_width  = 120
archive_thumbnails_height = 90
# jpeg quality (you can save as png etc too, but then quality does squat)
quality = 80
input  = 0
# 0 for PAL, 1 for NTSC
norm   = 0
# Goes in the top right. strftime() is run on this too, so put date stuff in
# if you like
title_text = WebCAM
# color/transparency of title text
title_r = 255
title_g = 255
title_b = 0
title_a = 255
# font for title text. fontname/size
title_font = luxisb/8
# fancy font styles
# title_style = /path/to/title.style
# color/transparency of message text
text_r = 255
text_g = 255
text_b = 0
text_a = 255
# font for message text. fontname/size
text_font = luxisb/8
# fancy font styles
# text_style = /path/to/text.style
# color/transparency of rectangle behind text
# make it 0,0,0,0 to disable.
bg_a = 0
bg_b = 0
bg_g = 0
bg_a = 100
# directory to look for ttf fonts in
ttf_dir = /usr/share/X11/fonts/TTF/
```

~/.caminfo (Text der im Bild eingebettet wird)

```
www.magenbrot.net
```

das Programm kann dann mit „camE“ gestartet werden. Nun legt es jede Sekunde das aktuelle Bilder unter <httpdocroot>/webcam/webcam.jpg ab. Jetzt fehlt noch die Anwendung, die die aktuellen Bilder an den Client „pusht“.

Erstmal eine schöne Verpackung für das Webcam-Bild: <httpdocroot>/webcam/index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="expires" content="0">
<title>magenbrot.net -- webcam</title>
<link href="../styles.css" rel="stylesheet" type="text/css">
```

```
</head>

<body>

<br><br>
<center>
my Webcam...<br><br>
<IMG SRC="/cgi-bin/nph-webgrab.cgi/0.jpg" ALT="WebCam Picture-"><br>
<br>
Standort: Betriebsb&uuml;ro OnlineDienst Nordbayern<br>
</center>
</body>
</html>
```

und hier die CGI-Push Anwendung: <httpdocroot>/cgi-bin/nph-webgrab.cgi

```
#!/usr/bin/perl

print "HTTP/1.0 200 Document follows\n";
print "Content-type: multipart/x-mixed-replace;boundary=goober\n\n";

while(true){
  print "\n--goober\n";
  print "Content-type: image/jpeg\n\n";

  open(IN, "../webcam/webcam.jpg");

  while (read(IN, $buffer, 4096)) {
    print $buffer;
  }
  close("../webcam/webcam.jpg");
  sleep(1);
}
print "\n--goober--\n";

exit 0;
```

fertig.