```
/* Simple Video4Linux image grabber. */
/*
 *      Video4Linux Driver Test/Example Framegrabbing Program
 *
 *      Compile with:
 *              gcc -s -Wall -Wstrict-prototypes v4lgrab.c -o v4lgrab
 *      Use as:
 *              v4lgrab >image.ppm
 *
 *      Copyright (C) 1998-05-03, Phil Blundell <philb@gnu.org>
 *      Copied from http://www.tazenda.demon.co.uk/phil/vgrabber.c
 *      with minor modifications (Dave Forrest, drf5n@virginia.edu).
 *
 *
 *      For some cameras you may need to pre-load libv4l to perform
 *      the necessary decompression, e.g.:
 *
 *      export LD_PRELOAD=/usr/lib/libv4l/v4l1compat.so
 *      ./v4lgrab >image.ppm
 *
 *      see http://hansdegoede.livejournal.com/3636.html for details.
 *
 */

#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <stdlib.h>

#include <linux/types.h>
#include <linux/videodev.h>

#define VIDEO_DEV "/dev/video0"

/* Stole this from tvset.c */

#define READ_VIDEO_PIXEL(buf, format, depth, r, g, b)                   \
{                                                                       \
        switch (format)                                                 \
        {                                                               \
                case VIDEO_PALETTE_GREY:                                \
                        switch (depth)                                  \
                        {                                               \
                                case 4:                                 \
                                case 6:                                 \
                                case 8:                                 \
                                        (r) = (g) = (b) = (*buf++ << 8);\
                                        break;                          \
                                                                        \
                                case 16:                                \
                                        (r) = (g) = (b) =               \
                                                *((unsigned short *) buf);
\
```

```
                        buf += 2;                                       \
                        break;                                          \
                }                                                       \
                break;                                                  \
                                                                        \
                                                                        \
        case VIDEO_PALETTE_RGB565:                                      \
        {                                                               \
                unsigned short tmp = *(unsigned short *)buf;            \
                (r) = tmp&0xF800;                                       \
                (g) = (tmp<<5)&0xFC00;                                  \
                (b) = (tmp<<11)&0xF800;                                 \
                buf += 2;                                               \
        }                                                               \
        break;                                                          \
                                                                        \
        case VIDEO_PALETTE_RGB555:                                      \
                (r) = (buf[0]&0xF8)<<8;                                 \
                (g) = ((buf[0] << 5 | buf[1] >> 3)&0xF8)<<8;            \
                (b) = ((buf[1] << 2 ) & 0xF8)<<8;                       \
                buf += 2;                                               \
                break;                                                  \
                                                                        \
        case VIDEO_PALETTE_RGB24:                                       \
                (r) = buf[0] << 8; (g) = buf[1] << 8;                   \
                (b) = buf[2] << 8;                                      \
                buf += 3;                                               \
                break;                                                  \
                                                                        \
        default:                                                        \
                fprintf(stderr,                                         \
                        "Format %d not yet supported\n",                \
                        format);                                        \
        }                                                               \
}

static int get_brightness_adj(unsigned char *image, long size, int *brightness)
{
  long i, tot = 0;
  for (i=0;i<size*3;i++)
    tot += image[i];
  *brightness = (128 - tot/(size*3))/3;
  return !((tot/(size*3)) >= 126 && (tot/(size*3)) <= 130);
}

int main(int argc, char ** argv)
{
  int fd = open(VIDEO_DEV, O_RDONLY), f;
  struct video_capability cap;
  struct video_window win;
  struct video_picture vpic;

  unsigned char *buffer, *src;
  int bpp = 24, r = 0, g = 0, b = 0;
  unsigned int i, src_depth = 16;
```

```
if (fd < 0) {
  perror(VIDEO_DEV);
  exit(1);
}

if (ioctl(fd, VIDIOCGCAP, &cap) < 0) {
  perror("VIDIOGCAP");
  fprintf(stderr, "(" VIDEO_DEV " not a video4linux device?)\n");
  close(fd);
  exit(1);
}

if (ioctl(fd, VIDIOCGWIN, &win) < 0) {
  perror("VIDIOCGWIN");
  close(fd);
  exit(1);
}

if (ioctl(fd, VIDIOCGPICT, &vpic) < 0) {
  perror("VIDIOCGPICT");
  close(fd);
  exit(1);
}

if (cap.type & VID_TYPE_MONOCHROME) {
  vpic.depth=8;
  vpic.palette=VIDEO_PALETTE_GREY;    /* 8bit grey */
  if(ioctl(fd, VIDIOCSPICT, &vpic) < 0) {
    vpic.depth=6;
    if(ioctl(fd, VIDIOCSPICT, &vpic) < 0) {
      vpic.depth=4;
      if(ioctl(fd, VIDIOCSPICT, &vpic) < 0) {
        fprintf(stderr, "Unable to find a supported capture format.\n");
        close(fd);
        exit(1);
      }
    }
  }
} else {
  vpic.depth=24;
  vpic.palette=VIDEO_PALETTE_RGB24;

  if(ioctl(fd, VIDIOCSPICT, &vpic) < 0) {
    vpic.palette=VIDEO_PALETTE_RGB565;
    vpic.depth=16;

    if(ioctl(fd, VIDIOCSPICT, &vpic)==-1) {
      vpic.palette=VIDEO_PALETTE_RGB555;
      vpic.depth=15;

      if(ioctl(fd, VIDIOCSPICT, &vpic)==-1) {
        fprintf(stderr, "Unable to find a supported capture format.\n");
        return -1;
      }
    }
  }
```

```
  }

  buffer = malloc(win.width * win.height * bpp);
  if (!buffer) {
    fprintf(stderr, "Out of memory.\n");
    exit(1);
  }

  do {
    int newbright;
    read(fd, buffer, win.width * win.height * bpp);
    f = get_brightness_adj(buffer, win.width * win.height, &newbright);
    if (f) {
      vpic.brightness += (newbright << 8);
      if(ioctl(fd, VIDIOCSPICT, &vpic)==-1) {
        perror("VIDIOSPICT");
        break;
      }
    }
  } while (f);

  fprintf(stdout, "P6\n%d %d 255\n", win.width, win.height);

  src = buffer;

  for (i = 0; i < win.width * win.height; i++) {
    READ_VIDEO_PIXEL(src, vpic.palette, src_depth, r, g, b);
    fputc(r>>8, stdout);
    fputc(g>>8, stdout);
    fputc(b>>8, stdout);
  }

  close(fd);
  return 0;
}
```