CCS-ECL Library

Macros, Header Files, and Table Format

Macros

ABS(v) return absolute value of (v)

DrawArc(Image *img, int min_pattern, int func)
DrawCrop(Image *img, int min_pattern, int func)
DrawLine(Image *img, int min_pattern, int func)

- img — image structure pointer
- min_pattern — minimum pattern size to draw; if drawing pattern is smaller than given min_pattern, then no thing will be drawed.
- func — X logical function. See Xlib Programming Manual Vol-1.

These three macros draw (re-draw) a pattern described and stored by sub-image in image structure.

ImageEvent(Image *img, long EventMasks)

- img — image structure pointer
- EventMasks — X event masks

True if EventMasks match any event in the event queue associated with image window, and removed that event from the event queue. False otherwise.

RemoveImageEvent(Image *img, long EventMasks)

- img — image structure pointer
- EventMasks — X event masks

Remove all events match EventMasks in image event queue.

Loop looping in following module

MIN(a, b) return the less value in a and b

MAX(a, b) return the greater value in a and b

Sign(v) return the sign of value (v)

bound_check(int size, int bottom, int top)

It checks bottom (or left) and size fit in 0 and top (right) boundaries.

```
if    (bottom < 0) then    b = 0;
else if (size + bottom > top)    size = top - bottom;
```

io_test(int io_fd, statements then_what)

- io_fd — file descriptor
- then_what — C statement(s)

If a file I/O described by io_fd is not a file (or is a terminal mode I/O), then, execute the then_what statement. The then_what can be any single C statement or combinations, such as goto, exit, etc.

isfloat(int c) return True if c is a floating point character

init_readpipe(U_IMAGE img)
initial pipe read handler in img structure

NZALLOC(int nblocks, int blocksize, char* caller)
return allocated non-clean memory address, or NULL in failure.

pointer_buffer_size(void* ***p***) return buffer size associated with pointer ***p***.

pushpipe(char *buf, int size, FILE *fp)
put data back to pipe stream

reset_readpipe(U_IMAGE *img)
reset I/O routine to regular C functions

str_save(char *s) return buffer pointer to saved string (s)

ZALLOC(int nblocks, int blocksize, char* caller)
return allocated clean memory address, or NULL in failure.

Library Header Files and Major Structures

The CCS-ECL has many header files in its "include" directory, but only a few of them are useful to user. They are:

header.def
imagedef.h
net_need.h
rtp.h
stdef.h
function.h
panel.h

The first five are general purpose header files, and the last two are used for X programs.

The header.def defines all image formats, such as IFMT_BYTE, IFMT_VFFT3D, and color formats, e.g. CFM_SGF, CFM_ILC, etc.

The imagedef.h defines basic image structure U_IMAGE and image type, such as JPEG, HIPS, PNM, RLE, TiFF, etc.

The net_need.h is used for network on socket layer, and the rtp.h is designed for real time video and movie data transmission on network.

The stdef.h is a kernel definition file and used for low level program. If any other header files above is used, the stdef.h is automatically included.

Function.h defines macros for X image and panel. It includes panel.h, which defines panel and its part (widget) structures, such as image, window, and panel structures, as well the Image structure. A few compiling directives are important for X definitions. (1) to invoke panel.h in function.h, the X_WINDOW_DEP is needed. (2) The EXTENDED_COLOR directive is used for Image structure to handle color dithering, color gamma adjusting, fast zooming and window refreshing.

The major structures U_IMAGE and Image will be explained in the following section, and some window and panel structures will be described after it.

<1> U_IMAGE structure:

```
typedef struct  {
        char    *name,              /* file or image name*/
                *history, *desc;    /* descriptions*/
        long    colormap;           /* color map for X*/
        void    *dpy, *gc, *igc;
        long    frame, win, icon;
        void    *event;             /* from colormap to here, all for X*/
        int     x0, y0,             /* image origin*/
                width, height,      /* image size*/
                icon_width, icon_height,    /* icon size*/
                font_w, font_h, ascent;     /* fonts information*/
        VType   *image, *icon_image;        /* X images*/
        int     curve, linearlow, linearup, bgrd, fgrd,/* elastic curve */
                scale, tmp_offset,          /* used for anything */
                mark_x, mark_y,             /* marks position in X window*/
                resize_w, resize_h,         /* reduced window size*/
                sub_img_x, sub_img_y,       /* sub-image origin*/
                sub_img_w, sub_img_h,       /* sub-image size*/
                mag_fact, mag_mode,         /* display magnified image */
                mag_x, mag_y, mag_w, mag_h,/* subimage currently being
viewed */
                save_mag_x, save_mag_y, save_mag_w, save_mag_h,
                save_mag_fact,      /* save_* for fast switch magnified images*/
                dpy_depth,          /* visual (monitor) type*/
                frames, fn,         /* Z-size and Z-pointer*/
                channels, dpy_channels,     /* original and current channels*/
                *hist, *histp;      /* histogram buffer & pointer */
        VType   *src, *dest, *cnvt, *img_buf, *lkt;     /* buffers*/
        bool    sub_img, sub_img_enh,       /* sub window and enhanced */
                active,             /* active window*/
                rw_cmap,            /* writable colormap*/
                color_dpy,          /* False if black/white */
                dither_img,         /* no true color monitor*/
                update,             /* image has been modified*/
                logscale, setscale, /* use unique / own maxcnt */
                load_all, save_all; /* if not true, do 1 frame at a time */
        Mregister  mmm, *marray;    /* min, max info*/
                /* standard interfaces*/
        StdInterface    *errors, *header_handle, *std_swif;
        TableInterface  (*table_if)(), **table, tables;
        VType   **superimpose,      /* superimposes draws & texts*/
                *parts;             /* scroll bars and others*/
        short   draws, texts,       /* number of draws & texts in si stacks */
                stack_num;                  /* parts in part stack*/
```

```
        int      in_type, mid_type, o_type,  /* image types*/
                 in_color, color_form,       /* color formats*/
                 in_form, o_form,            /* image formats*/
                 pxl_in, pxl_out;            /* pixel sizes*/
        FILE     *IN_FP, *OUT_FP;            /* file pointers*/
        void     (*map_scanline)(), (*MAG_scanline)();
        bool     binary_img, mono_img,       /* same info as color_form*/
                 mono_color, sep_colors,
                 update_header;     /* if true, allow to update header again */
        int      entries,           /* color entries are allocated*/
                 ncmap, cmaplen;    /* colormap channels and size*/
        cmap_t   **cmap;            /* color maps*/
#if defined EXTENDED_U_IMAGE | defined X_WINDOW_DEP
        int      lvls, lvls_squared,
                 visual_class;
        VType    *dpy_visual;
#endif
        } U_IMAGE;      /* standard Union Image structure for filters */
```

Description:

name	structure name
history	history record
desc	program description
colormap	X color map ID
dpy	X display pointer
gc	X graphic content pointer
igc	icon gc
frame	image canvas if SCROLLBAR_on_CANVAS is defined; otherwise, it will be equal to win.
win	X ID of window on canvas
icon	icon X window ID
event	X event pointer
x0, y0	image X window origin related to canvas
width, height	image dimensions
icon_width, icon_height	icon dimensions
font_w, font_h, ascent	font properties used on window
image	X image structure
icon_image	icon X image structure
curve	image elastic tuning curve
linearlow, linearup	minimum and maximum pixel values for display
bgrd, fgrd	elastic tuning on background or foreground
scale	elastic tuning scale factor
tmp_offset	temporary register can be used for any thing
mark_x, mark_y	mark position
resize_w, resize_h	window resize dimension
sub_img_x, sub_img_y	sub-image origin
sub_img_w, sub_img_h	sub-image dimension

mag_fact	zooming factor (negative is zooming down)
mag_mode	zooming, zoom shift, or zooming swap
mag_x, mag_y	zooming area origin
mag_w, mag_h	zooming area dimension
save_mag_x, save_mag_y, save_img_w, save_img_h, save_mag_fact	saved zooming parameter for zooming shift or swap
dpy_depth	display color depth (1, 4, 8, 16, 24, or 32-bit display)
frames	z dimension of images
fn	z position
channels	input image channels (color planes)
dpy_channels	output or display image channels
hist	histogram buffer
histp	histogram pointer
src	source data buffer
dest	destination data buffer
cnvt	working (converting) buffer
img_buf	X image data buffer (may equal to img->image->data)
lkt	mapping lookup table
sub_img	if sub-window exists
sub_img_enh	if sub-window is modified
active	if working cursor is in current image window
rw_cmap	if color map is writable
color_dpy	if display device is a color one or output should be color
dither_img	if image need to be dithered for display
update	if image data has been modified
logscale, setscale	how histogram be scaled for plotting
load_all, save_all	True for loading or saving an entire image. False for loading or saving one frame at a time.
mmm	minimum, maximum pixel values, and maximum pixel count register
marray	mmm array for each frame
(*errors)(), (*header_handle)(), (*read)(), (*seek)(), (*std_swif)(), (*write)() (*table)()	interfaces
table	table array for table_if
tables	number of tables in the table array
superimpose	superimposed draw elements array
parts	scrollbar or other widget stack
draws	number of draws in superimposed draw elements array
texts	number of text strings in superimposed draw elements array
stack_num	number elements in parts stack
in_type	input image type, such as JPEG, HIPS, ...
mid_type	image type to be converted to
o_type	output image type
in_color	input image color format
color_form	image color format to be converted to for working or outputting
in_form	input image format, such as IFMT_BYTE, IFMT_LONG, IFMT_VFFT2D, ...
o_form	output image format

pxl_in	number of bytes for each pixel in input images
pxl_out	number of bytes for each pixel in output images
IN_FP	input file stream pointer
OUT_FP	output file stream pointer
(*map_scanline)()	map scanline from original data to display data interface
(*MAG_scanline)()	zooming interface
binary_img	if the input image is a binary image
mono_img	if the input image is a monochrome image (gray scale)
mono_color	if the input image is a pseudo color image
sep_colors	if the input image is a true color image
update_header	update header description when writing image header
entries	number of color cells are used in its color map
ncmap	number of color map channels
cmaplen	color map size in each channel
cmap	color map
lvls	maximum number of color cells required
lvls_squared	root of lvls
visual_class	visual class
dpy_visual	display visual structure

<2> Image structure

```
typedef struct  {
        char    *name, *history, *desc;
        Colormap colormap;
        Display *dpy;
        GC      gc, igc;
        Window  frame, win, icon;
        XEvent  *event;
        int     x0, y0, width, height,
                icon_width, icon_height,
                font_w, font_h, ascent;
        XImage  *image, *icon_image;
        int     curve, linearlow, linearup, bgrd, fgrd,/* elastic curve */
                scale, tmp_offset,          /* this 2 fields used for anything */
                mark_x, mark_y, resize_w, resize_h,
                sub_img_x, sub_img_y, sub_img_w, sub_img_h,
                mag_fact, mag_mode,         /* display magnified image*/
                mag_x, mag_y, mag_w, mag_h,/* sub-image currently being
viewed */
                save_mag_x, save_mag_y, save_mag_w, save_mag_h,
                save_mag_fact, dpy_depth,
                frames, fn, channels, dpy_channels,
                *hist, *histp;      /* histogram buffer & pointer*/
        byte    *data, *scan_data,/* original data & scan buffers*/
                *cnvt,
                *img_buf, *icon_buf;        /* map buffers for image & icon*/
        bool    sub_img, sub_img_enh,       /* sub window and enhance*/
                active, rw_cmap,            /* writable colormap*/
                color_dpy,                  /* False if black/white*/
                dither_img, update,
                logscale, setscale, /* use unique / own max histogram count */
                load_all, save_all;
        Mregister mmm, *marray;
        StdInterface    *errors, *header_handle, *read, *seek, *std_swif, *write;
        TableInterface (*table_if)(), **table, tables;
        superimpose_elems   **superimpose;
        PanelParts *parts;
        short   draws, texts, stack_num;    /* parts in parts stack*/
        int     in_type, mid_type, o_type, in_color, color_form,
                in_form, o_form, pxl_in, pxl_out;
        FILE    *IN_FP, *OUT_FP;
        void    (*map_scanline)(), (*MAG_scanline)();
        bool    binary_img,
                mono_img,
                mono_color,
```

```
                sep_colors,
                pixmap_failed;/* as update_header flag in U_IMAGE*/
        int     entries,        /* number of colors or gray levels*/
                ncmap, cmaplen; /* in regular length (not 2 ** cmaplen)*/
        cmap_t  **in_cmap;
        int     lvls, lvls_squared, visual_class;
        Visual  *dpy_visual;
#if     defined EXTENDED_COLOR | defined C_TUNER
        Window  pix_info_window;
        Pixmap  pixmap, icn_pixmap, mag_pixmap, refresh_pixmap;
        float   gamma, dpy_gamma;
        char    *title;
        int     cmlen,  /* color_map-length in Comment*/
                *modN, *divN, **dm16;
        unsigned int    *pixel_table;
#endif
        } Image;
```

Basically, It is same as U_IMAGE structure. It is different from U_IMAGE at two places: 1) buffer names; 2) more items

1) The Image buffers are named data and scan_data, and they are corresponding to src and dest in U_IMAGE structure.

2) EXTENDED_COLOR items are used for gamma correction, fast zooming shift and swap, color dither and map.

pix_info_window	virtual window for displaying pixel information
pixmap	store extra copy of window content for fast recovering
icn_pixmap	store extra copy of icon content for fast icon recovering
mag_pixmap	store extra copy of zooming image for fast zooming changing
refresh_pixmap	pixel map working pointer to pixel maps above
gamma, dpt_gamma	gamma value for gamma correction
cmlen	RLE color map length (in_cmap.c)
modN, divN, dm16	dither information arrays
pixel_table	a table contains dithered color map pixel values for fast color mapping

Table Format for T able Interface

The CCS table interface file has a simple format:

Keyword *data-length* *format* *value/position* *options*

CCS table reserved key-words are:

header/magic describes what type of table is in this table file. Its format is:

header magic-length header-length [magic] [mode]

If magic-length is non-zero, then *magic* string must be present and should be no more than four characters. The modes are addr, line, or skip. In addr mode, the *value/position* field is for position (absolute data address offset from the beginning of image files). In line mode, the *value/position* field is the line number, and the key string option is needed. In skip mode, the *value/position* field is for immediate value, and the *data-length* field should be zero. This rule is used for the all key-words except Marks. The header length usually is variable and can indicated by the header length being equal to zero

mark symbol string

Two mark symbols are available in current table interface:

end of header / eoh
end of line/item

to indicate end of header or end of items.

colormap size type [position]

The color map formats known by CCS are:

RAS

RRRRRRRRRRR.......
GGGGGGGGG.......
BBBBBBBBBBB....... in binary format.

HEX

RGB RGB RGB ... ... in hex format.

GIF

RGBRGBRGB...... in binary format.

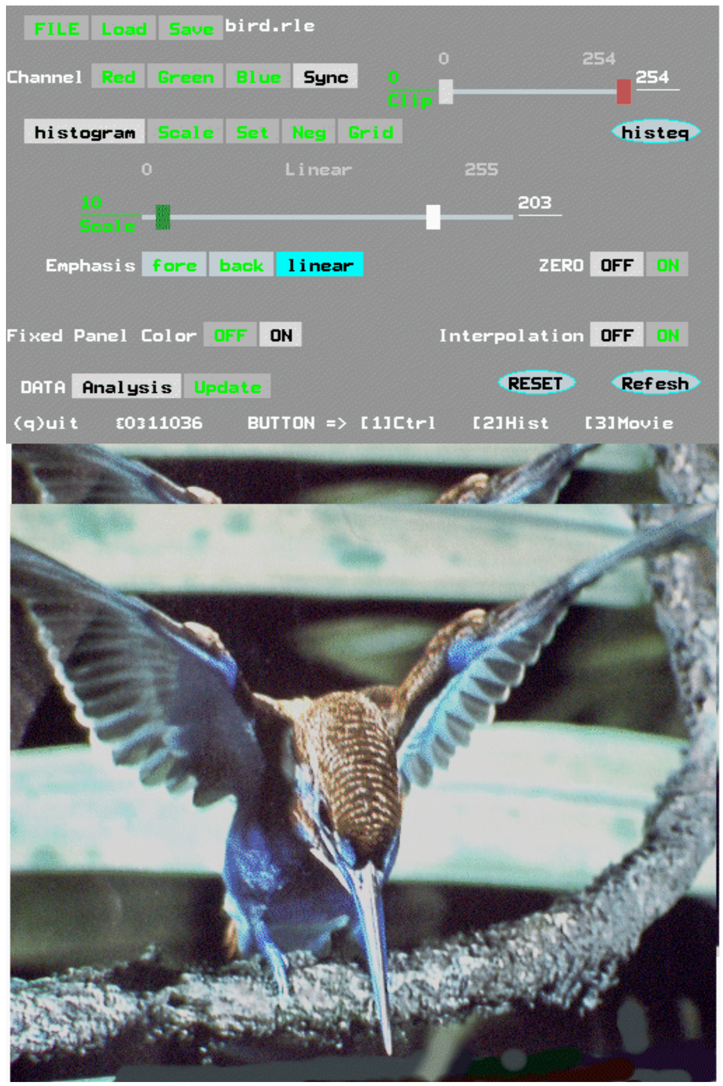
DEF

default format

The encode values are any image types defined in imagedef.h, such as: HIPS, RAS, ...
The format values are any image format defined in header.def, such as: SGF, ILC, ALPHA, ...

The format field for keyword "encode" and "format" should be either "#" or "%s" (immediate value or string format). The format field for keyword "width", "height", "frames", and "depth" can be "#" or reasonable C format, such %i or %h, etc.

Program Guide



Two programs are used as examples in this guide. The first one, toany.c, is a conversion filter to convert different type of images to other different type of images. The second one, getx.c, is an image displayer and editor. We will use a few steps to build these programs from a very simple but very powerful program to a very fancy program but not very complicated.

Example 1:

toany.c - torle and tohips

```
/*
%	TORLE . C
%
*/
#include "header	.def"
#include "imagedef.h"

U_IMAGE		uimg;
char		usage[]="torle in.any > out.rle\n";

main()
{
	format_init(&uimg, IMAGE_INIT_TYPE, RLE, RLE, *argv	, "ver -
1.0");
	if ((*uimg.header_handle)(HEADER_READ, &uimg, 0, 0, T	rue) < 0)
		syserr("Unknown image type");
	(*uimg.std_swif)(FI_LOAD_FILE, &uimg, 0, 0),
	(*uimg.std_swif)(FI_SA	VE_FILE, &uimg, 0, 0);
}
```

Is this very simple? Let's do one more for HIPS:

```
/*
%	TOHIPS . C
%
*/
#include "header	.def"
#include "imagedef.h"

U_IMAGE		uimg;
char		usage[]="tohips in.any > out.hips\n";

main()
{
	format_init(&uimg, IMAGE_INIT_TYPE, HIPS, -1, *argv	, "ver -1.0");
	if ((*uimg.header_handle)(HEADER_READ, &uimg, 0, 0, T	rue) < 0)
		syserr("Unknown image type");
	(*uimg.std_swif)(FI_LOAD_FILE, &uimg, 0, 0),
	(*uimg.std_swif)(FI_SA	VE_FILE, &uimg, 0, 0);
}
```

The output type in tohips.c is -1, and it means the output type is same as the middle type. Now, let's build a toany .c.

Example 1.1:

toany . c — enhanced program

```
/*          ToAny . C
%
%           Copyright (c)      1991, 1993      Jin Guojun
*/
#include "header    .def"
#include "imagedef.h"

U_IMAGE uimg;
char        usage[]="toany [-hips] [-rast] [-rle] [-color] in.any > out.other\n";

main(argc, argv)
int         argc;
char*       argv[];
{
char        info[128];
int         i, mid_type, out_header=T      rue, o_type=HIPS;

/*          initial input any type, and output OTHER type            */

   for (i=1; i < argc; i++)
            if (*argv[i] == '-') {
                     if (strcmp(argv[i], "-rle") == 0)
                              o_type = RLE;
                     else     if (strcmp(argv[i], "-rast") == 0)
                              o_type = RAS;
                     else     if (strcmp(argv[i], "-color") == 0)
                              uimg.color_dpy--;
            }
            else     if ((in_fp=fopen(argv[i], "rb") == NULL)
                     syserr("open input %s", argv[i]);

            /* use RLE format for other conversions */
   mid_type = o_type==HIPS ? HIPS : RLE;
   format_init(&uimg, IMAGE_INIT_TYPE, mid_type, o_type, *argv            , "ver -
2.0");

   io_test(fileno(in_fp), usage_n_options(usage, i, argv[i]));

   if (in_header)
            if ((*uimg.header_handle)(HEADER_READ, &uimg, 0, 0, T           rue) < 0)
                     syserr("Unknown image type");
   uimg.pxl_out = uimg.pxl_in;              /* output size = input size, for HIPS */

            /* example to add description */
   sprintf(info, "%s: e%d*%d_%d_%d",
            i_name, enlg_row    , enlg_cln, bgn_row    ,bgn_cln);
```

```
    (*uimg.header_handle)(ADD_DESC, &uimg, info);

    uimg.update_header = (*uimg.header_handle)(HEADER_TO, &uimg, 0);
/* don't update again while HEADER_WRITE */

    if (uimg.o_type == HIPS)            /* HIPS is a single header type image */
    if (out_header)
            (*uimg.header_handle)(HEADER_WRITE, &uimg, argc, argv          , True);
    else
            (*uimg.header_handle)(HEADER_FWRITE, &uimg, argc, argv          ,
stderr);

    for (f=0; f<uimg.frames; f++)
            (*uimg.std_swif)(FI_LOAD_FILE, &uimg, 0, 0),
            (*uimg.std_swif)(FI_SA    VE_FILE, &uimg, 0, 0);
exit(0);
}
```

So far , toany can convert images to HIPS, RASTER, and RLE images, and you can add more output image types to it. Next, we are going to add parse_argus() function to make program generate help page, rotation and dither to 8-bit color functions.

Example 1.2:
toany . c — completed program

```
/*        ToAny . C
%
%         Copyright (c)    1991,1993    Jin Guojun
*/
#include "header   .def"
#include "imagedef.h"

U_IMAGE     uimg;
arg_fmt_list_string        argfmt[] = {
          {"-c", "%-", No, 1, 0,            "output color image"},
          {"-t", "%s", NULL, 2, 1,          "type of output image"},
          {"-d", "%b", No, 1, 0,            " dither to 8-bit"},
          {"-k", "%b", True, 1, 0,          " keep going when errors hanppen"},
          {"-l", "%-", No, 1, 0,            " rotate left 90"},
          {"-r", "%+", No, 1, 0,            " rotate right 90"},
{"        [<] input [[> |] output]", No, 0, 0, 0, "end of usage"},
          NULL     };

main(argc, argv)
int       argc;
char*     argv[];
{
bool      dither8=False, rotate90=0;
char      info[128], **filelist, *type_str;
int       f, num_files, mid_type, out_header=T       rue, o_type=HIPS;

  if ((num_files = parse_argus(&filelist, argc, argv         , argfmt,
          &uimg.color_dpy  ,
          &type_str  ,
          &dither8,
          &rotate90, &rotate90)) < 0)
  exit     (num_files);

  if (type_str && (o_type=available_type(type_str)) < 0)
          prgmerr('o', "wrong output image type %s", type_str);
  mid_type = o_type==HIPS ? HIPS : RLE;
  format_init(&uimg, IMAGE_INIT_TYPE, mid_type, o_type, *argv           , "ver -
3.0");

  while (num_files--)      {
    if ((in_fp=fopen(filelist[num_files], "rb")) == NULL)
          syserr("open input %s", filelist[num_files]);
    if (num_files)      {
          sprintf(info, "%s.%s", filelist[num_files], type_str);
          if (!(out_fp=freopen(info, "w", stdout)))
                    syserr("open output %s", info);
```

```
        }

        io_test(fileno(in_fp), {parse_usage(argfmt); exit(0);});

        if (in_header)
                if ((*uimg.header_handle)(HEADER_READ, &uimg, 0, 0, True) < 0)
                                syserr("Unknown image type");
        uimg.pxl_out = uimg.pxl_in;             /* output size = input size, for HIPS */

                /* example to add description */
        sprintf(info, "%s: e%d*%d_%d_%d",
                i_name, enlg_row, enlg_cln, bgn_row ,bgn_cln);

        (*uimg.header_handle)(ADD_DESC, &uimg, info);

        uimg.update_header = (*uimg.header_handle)(HEADER_TO, &uimg, 0);

        if (uimg.o_type == HIPS)                /* HIPS is a single header type image */
        if (out_header)
                (*uimg.header_handle)(HEADER_WRITE, &uimg, argc, argv, True);
        else
                (*uimg.header_handle)(HEADER_FWRITE, &uimg, argc, argv,
stderr);

        for (f=0; f<uimg.frames; f++)   {
                (*uimg.std_swif)(FI_LOAD_FILE, &uimg, 0, 0);

                if (to8 && uimg.channels > 1)
                                To_8(&uimg, reg_cmap, to8==1, 256),
                                uimg.dpy_channels = uimg.channels;
                if (rotate90) {
                    uimg.dest = uimg.src;
                    uimg.src = nzalloc(uimg.channels*uimg.width, uimg.height, "rot");

                    row = uimg.width;
                    uimg.width = uimg.height;
                    uimg.height = row;
                    color_rotate_90(uimg.dest, uimg.src, row, uimg.width,
                                uimg.color_form, rotate90+1);
                    free(uimg.dest);
                }

                (*uimg.std_swif)(FI_SAVE_FILE, &uimg, 0, 0);

exit(0);
}
```

The second example is to build an image displayer and editor. First, let's build a displayer for displaying different images.

Example 2:

display-image - a simple image display program

```
/* display-image . c
*/

#include "function.h"


main()
{
XEvent     xe;
char        *display_name = getenv("DISPLAY");
Image       *ximg = zalloc(1, sizeof(*ximg), "ximg");

        ximg->color_dpy = True

        format_init(ximg, IMAGE_INIT_TYPE, RLE, -1,
                        ximg->name="display-image", "ver-1.0");

        if (!(Dpy=XOpenDisplay(display_name)))
          prgmerr(1, "Cant open display %s\n", display_name);

        Screen = XDefaultScreen(Dpy);
        GetVctEntry(Dpy, Screen, DefaultColormap(Dpy, Screen));
        ximg->dpy = Dpy;
        ximg->visual_class = -1;
        ximg->lvls = specified_levels = 240;
        find_appropriate_visual(ximg);

        ximg->event = &xe;
        get_pic(0, NULL, NULL, &ximg, 0);

        do       XNextEvent(Dpy, ximg->event);
        while (!ImageEvent(ximg, KeyPressMask));

        DestroyColorImage(ximg);
}
```

This display-image program can display any images handled by CCS from standard input. Above example uses all basic statements that initialize those necessary variables, and blocks must be written in that order. Next, we will add argument processing for multiple image display and gamma correction.

Example 2.1:
getx . c — prototype

```
/* getx .c
#
%       Copyright (c)    Jin Guojun
%
% Author:       Jin Guojun - LBL
*/

#include "function.h"

Image   **pic;
arg_fmt_list_string       argfmt[] = {
        {"-d", "%s", No, 1, 0, "display name <host[:x.y]>"},
        {"-v", "%i", -1, 1, 1,
                "vis_type: specify Visual type to be used\n\
                        (integer 0 to 5)"},
        {"-b", "%b", True, 1, 0, "binary (black and white)"},
        {"-B", "%b", True, 1, 0, "monochrome"},
        {"-g", "%D", 1.5, 1, 1,
                "gamma for display. (default 1.5)"},
        {"-n", "%d", DEFAULT_LEVELS, 1, 1,
                "number of colors to dither colormaps"},
        {"input: Any Image file. Use stdin if none specified"
        "0", 0, 0, 0, "\nEnd of options"}, NULL };


main(argc, argv)
int     argc;
char*   argv[];
{
char      *display_name = getenv("DISPLAY"), **flst;
Image     *img;
int     i, nf, visual_type;

        specified_levels = 240;

        if ((nf=parse_argus(&fl, argc, argv, argfmt,
                &display_name, &visual_type,
                &bin_img, &mono_img,
                &display_gamma, &specified_levels)) < 0)
        exit    (nf);

        if (!(Dpy=XOpenDisplay(display_name)))
                prgmerr(1, "Cant open display %s\n",
                        display_name);
        Screen = XDefaultScreen(Dpy);
```

```
        GetVctEntry(Dpy, Screen,
                        DefaultColormap(Dpy, Screen));
        pic = zalloc(nf, sizeof(*pic), “pic”);

        for (i=0; i< nf; i++)       {
                img = pic[i] = zalloc(1, sizeof(*img), “g-img”);
                img->color_dpy = True
                format_init(img, IMAGE_INIT_TYPE, RLE, -1,
                          img->name=nf[i], “ver-2.0”);

                if (!(img->IN_FP=fopen(img->name, “w”))) {
                        syserr(“open %s for input”, img->name);


                img->dpy = Dpy;
                img->visual_class = visual_type;
                img->lvls = specified_levels;
                find_appropriate_visual(img);

                get_pic(i, NULL, NULL, pic, 0);
        }
        maintain_pic(i, pic);
}

maintain_pic(num_imgs, pic)
int     num_imgs;
Imgae   **pic;
{
int     cur=0;          /* current active image index           */
Image     img = pic[cur];
XEvent    xe;

        while (num_imgs)       {
                XNextEvent(img->dpy, img->event);
                switch (xe.type)       {
                case EnterNotify:
                  cur = WhichImage(xe.xany.window, pic, num_imgs);
                break;
                case Expose:
                if (img && xe.xexpose.window==img->window)
                        handle_exposure(img, Draws,
                                xe.xexpose.x, xe.xexpose.y,
                                xe.xexpose.width, xe.xexpose.height,
                                img->h, No);
                case KeyPress: {
                char    string[256], *symstr, *XKeysymToString();
                KeySym keysym;
                XComposeStatus      stat;
                bool    shifted_key;
                int     handled_key = keysym, c,
```

```
                        length = XLookupString(&event, string, 256,
                                                &keysym, &stat);
                        string[length] = 0;
                        symstr = XKeysymToString(keysym);
                        shifted_key = event.xkey.state & ShiftMask;

                        if (length == 1) {
                                c = string[0];
                                if ((toupper(c) == 'C'))      { /* close */
                                        DestroyColorImage(img);
                                        if (--num_imgs)      {
                                          for (;cur < n; cur_img++)
                                                pic[cur] = pic[cur + 1];
                                          pic[cur] = NULL;
                                        }
                                } else if ((toupper(c) == 'Q')
                                                break;          /* quit          */
                        }
                        /* others are not used yet */
                }
        }
}
```

This program is much better that display-image.c because can handle exposure, gamma correction, multiple image display, and send images to different displays, as well give a usage description at run time. In next, we will add a control panel to getx.c.

Example 2.2:

getx . c — a color image tuning and analysis system - ETA

```
/
* getx .c
#
%	Copyright (c)	Jin Guojun
%
% Author:	Jin Guojun
*/

#include “function.h”

Image	**pic, parent;
bool	tuner_flag, movie_mode;
arg_fmt_list_string	argfmt[] = {
	{“-T”, “%b”, True, 1, 0, “use Tuner control panel”},
	{“-d”, “%s”, No, 1, 0, “display name <host[:x.y]>”},
	{“-v”, “%i”, -1, 1, 1,
		“vis_type: specify Visual type to be used\n\
			(integer 0 to 5)”},
	{“-b”, “%b”, True, 1, 0, “binary (black and white)”},
	{“-B”, “%b”, True, 1, 0, “monochrome”},
	{“-g”, “%D”, 1.5, 1, 1,
		“gamma for display. (default 1.5)”},
	{“-m”, “%# %d”, 30, 2, 0,
		“frames/sec: run movie [set speed]”},
	{“-n”, “%d”, DEFAULT_LEVELS, 1, 1,
		“number of colors to dither colormaps”},
	{“input: Any Image file. Use stdin if none specified”
	“0”, 0, 0, 0, “\nEnd of options”}, NULL };


main(argc, argv)
int	argc;
char*	argv[];
{
char	*display_name = getenv(“DISPLAY”), **flst;
Image	*img;
int	i, nf, visual_type;

	specified_levels = 240;

	if ((nf=parse_argus(&fl, argc, argv, argfmt,
		&tuner_flag,
		&display_name, &visual_type,
		&bin_img, &mono_img,
		&display_gamma,
		&movie_mode,
		&specified_levels)) < 0)
```

```
        exit    (nf);

        if (!(Dpy=XOpenDisplay(display_name)))
                prgmerr(1, "Cant open display %s\n",
                        display_name);

        pic = zalloc(nf, sizeof(*pic), "pic");
        parent.visual_class = visual_type;
        init_img_info(&parent, Dpy, RLE, -1);
        find_appropriate_visual(&parent);
        CreateColorTuner(Dpy, NULL, &parent,
                                tuner_flag, No, No);

        for (i=0; i< nf; i++)   {
          if (!using_stdin && strcmp(fl[i], "-")) {
                parent.name = fl[i];
                parent.IN_FP = rle_open_f_noexit("getx",
                                                parent.name, "r");
                if (!parent.IN_FP) {
                        prgmerr(nf==1, parent.name);
                        continue;
                }
          } else parent.name = DEF_TITLE;
          if (tuner_flag && i)
                LoadImage(parent.IN_FP, pic+num_images+i-1,
                        parent.name);
          else LoadGXImage(&num_images, &parent, &pic,
                                movid_mode, NULL);
        }
        if (!tuner_flag) {
                InfoWin = CreatePanel(10, 10, 512, 256,
                                "Information", Monitor, NULL);
                MGray = GetGray(Dpy, parent.cmap, ncolors, 64);
                XSetWindowBackground(Dpy, InfoWin->win, MGray);
        }
        maintain_pic(i, pic);
}

maintain_pic(num_imgs, pic)
int     num_imgs;
Imgae   **pic;
{
int     cur=0;          /* current active image index       */
Image     img = pic[cur];
XEvent    xe;

        while (num_imgs)        {
                XNextEvent(img->dpy, img->event);
                switch (xe.type)        {
                case EnterNotify:
```

```
        cur = WhichImage(xe.xany.window, pic, num_imgs);
      break;
      case Expose:
      if (img && xe.xexpose.window==img->window)
            handle_exposure(img, Draws,
                  xe.xexpose.x, xe.xexpose.y,
                  xe.xexpose.width, xe.xexpose.height,
                  img->height, No);
      case KeyPress: {
      char    string[256], *symstr, *XKeysymToString();
      KeySym keysym;
      XComposeStatus      stat;
      bool    shifted_key;
      int     handled_key = keysym, c,
            length = XLookupString(&event, string, 256,
                        &keysym, &stat);
            string[length] = 0;
            symstr = XKeysymToString(keysym);
            shifted_key = event.xkey.state & ShiftMask;

            if (length == 1) {
                  c = string[0];
                  if ((toupper(c) == 'C'))     { /* close */
                        DestroyColorImage(img);
                        if (--num_imgs)     {
                          for (;cur < n; cur_img++)
                              pic[cur] = pic[cur + 1];
                          pic[cur] = NULL;
                        }
                  } else if ((toupper(c) == 'Q')
                              break;          /* quit          */
            }
            /* others are not used yet */
      }
   }
}
```

We made some changes in main() routine to handle the image tuning. The major change is rewrited some codes which do the same thing as before, but use interfaces instead of regular C codes. The real one difference is that we added parent Image structure, initialize parent and passed it to children, and used CreateColorTuner() instead of GetVctEntry(). The initial routine format_init() is replaced by init_img_info(), which does more than format_init() for X initialization. In the last case, we are going to make getx as a extended X server by adding a few more line codes in the begining maintain_pic() subroutine.

Example 2.3:

getx .c — ETA + Extended X server

```
/* getx .c
#
%       Copyright (c)     Jin Guojun
%
% Author:       Jin Guojun - LBL
*/

#include "function.h"

Image   **pic, parent;
bool    tuner_flag, movie_mode, bin_img, mono_img;
arg_fmt_list_string       argfmt[] = {
        {"-T", "%b", True, 1, 0, "use Tuner control panel"},
        {"-d", "%s", No, 1, 0, "display name <host[:x.y]>"},
        {"-v", "%i", -1, 1, 1,
                "vis_type: specify Visual type to be used\n\
                        (integer 0 to 5)"},
        {"-b", "%b", True, 1, 0, "binary (black and white)"},
        {"-B", "%b", True, 1, 0, "monochrome"},
        {"-g", "%D", 1.5, 1, 1,
                "gamma for display. (default 1.5)"},
        {"-m", "%# %d", 30, 2, 0,
                "frames/sec: run movie [set speed]"},
        {"-n", "%d", DEFAULT_LEVELS, 1, 1,
                "number of colors to dither colormaps"},
        {"input: Any Image file. Use stdin if none specified"
        "0", 0, 0, 0, "\nEnd of options"}, NULL };


main(argc, argv)
int     argc;
char*   argv[];
{
char      *display_name = getenv("DISPLAY"), **flst;
Image     *img;
int     i, nf, visual_type;

        specified_levels = 240;

        if ((nf=parse_argus(&fl, argc, argv, argfmt,
                &tuner_flag,
                &display_name, &visual_type,
                &bin_img, &mono_img,
                &display_gamma,
                &movie_mode,
                &specified_levels)) < 0)
```

```
        exit    (nf);

        if (!(Dpy=XOpenDisplay(display_name)))
                prgmerr(1, “Cant open display %s\n”,
                        display_name);

        pic = zalloc(nf, sizeof(*pic), “pic”);
        parent.visual_class = visual_type;
        init_img_info(&parent, Dpy, RLE, -1);
        find_appropriate_visual(&parent);
        CreateColorTuner(Dpy, NULL, &parent,
                                tuner_flag, No, No);

        for (i=0; i< nf; i++)   {
          if (!using_stdin && strcmp(fl[i], “-”)) {
                parent.name = fl[i];
                parent.IN_FP = rle_open_f_noexit(“getx”,
                                                parent.name, “r”);
                if (!parent.IN_FP) {
                        prgmerr(nf==1, parent.name);
                        continue;
                }
          } else parent.name = DEF_TITLE;
          if (tuner_flag && i)
                LoadImage(parent.IN_FP, pic+num_images+i-1,
                        parent.name);
          else LoadGXImage(&num_images, &parent, &pic,
                                movid_mode, NULL);
        }
        if (!tuner_flag) {
                InfoWin = CreatePanel(10, 10, 512, 256,
                                “Information”, Monitor, NULL);
                MGray = GetGray(Dpy, parent.cmap, ncolors, 64);
                XSetWindowBackground(Dpy, InfoWin->win, MGray);
        }
        maintain_pic(i, pic);
}

maintain_pic(num_imgs, pic)
int     i, num_imgs;
Imgae   **pic;
{
bool    udp=0;          /* no udp */
int     cur=0;          /* current active image index       */
Image     img = pic[cur];
XEvent    xe;
        if (tuner_flag)
                histinfo.histp = rpic[0]->hist, DrawPanel();
#ifdef X_Extender
        {
```

```
	char pn[16];
		sprintf(pn, "port # = %d", x_extender_init(0, 0));
		img = pic[cur];
		XDrawString(img->dpy,img->win,img->gc,
		   img->width-120, img->height-8, pn, strlen(pn));
		if (udp)	x_extender_init(0, udp);
	}
#endif

	while (num_imgs)	{
#ifdef X_Extender
	 while (!XEventsQueued(rpic[0]->dpy, QueuedAfterFlush))
		for (i=0; i<udp; i++) {
		register FILE* sfp = x_extender(i);
		   if (sfp > 0) {
			if (i = get_arg_list(&fl))
			   parse_argus(NULL, i, fl, argfmt,
				basic_alist);
			parent.name = ExTITLE;
			if (tuner_flag) {
				if (LoadImage(sfp, pic+num_imgs,
					parent.name) == FileLoad)	{
					if (img) img->active = False;
					img = rpic[n++];
					img->active++;
				}
			} else LoadGXImage(&num_imgs, &parent, &pic,
						movie_mode, NULL);
		   }
		}
#endif

		XNextEvent(img->dpy, img->event);
		switch (xe.type)	{
		case EnterNotify:
		   cur = WhichImage(xe.xany.window, pic, num_imgs);
		break;
		case Expose:
		if (img && xe.xexpose.window==img->window)
			handle_exposure(img, Draws,
				xe.xexpose.x, xe.xexpose.y,
				xe.xexpose.width, xe.xexpose.height,
				img->height, No);
		case KeyPress: {
		char	string[256], *symstr, *XKeysymToString();
		KeySym keysym;
		XComposeStatus	stat;
		bool	shifted_key;
```

```
                int      handled_key = keysym, c,
                         length = XLookupString(&event, string, 256,
                                        &keysym, &stat);
                         string[length] = 0;
                         symstr = XKeysymToString(keysym);
                         shifted_key = event.xkey.state & ShiftMask;

                         if (length == 1) {
                                 c = string[0];
                                 if ((toupper(c) == 'C'))     { /* close */
                                         DestroyColorImage(img);
                                         if (--num_imgs)     {
                                           for (;cur < n; cur_img++)
                                                 pic[cur] = pic[cur + 1];
                                           pic[cur] = NULL;
                                         }
                                 } else if ((toupper(c) == 'Q')
                                                 break;          /* quit          */
                         }
                         /* others are not used yet */
                }
        }
}
```

So far, the *getx* is much powerful and the program example section is going to be finished. Now, you are able to understand how to read the formal *getx.c* code in lbl/x11/tuner directory and to build your own program by following these examples and looking at the formal *getx.c* or other programs under lbl/hips/sources directory.

Good lock!