Complete xmandelf Source Code [A]

The xmandelf program is a Mandelbrot set generator that uses Fortran and the X Window System.

xmandelf.c

A.1

The following xmandelf.c program is the main xmandelf program. It creates the window and all of the buttons for the Mandelbrot process.

```
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xaw/Cardinals.h>
#include <X11/Xaw/Label.h>
#include <X11/Xaw/Command.h>
#include <X11/Xaw/Logo.h>
#include <X11/Xaw/Paned.h>
#include <X11/Xaw/Box.h>
#include <X11/cursorfont.h>
#include <stdio.h>
#include <sys/utsname.h>
#include <unistd.h>

Widget drawform;
Widget detaillabel, mflopslabel, ncpulabel;      /* in widget ibox */
Widget xlabel, ylabel;   /* in widget bbox */
Cursor crosshair_xcr;
int depth;
int iter;
```

(continued)

```
int maxcpus = 8;
int ncpus = 4;

char ncputext[12];
void mandel();

void clear_callback(w, closure, call_data)
    Widget w;
    XtPointer closure;
    XtPointer call_data;
{
    Display *draw_d;
    Window draw_win;

    draw_d = XtDisplay(drawform);
    draw_win = XtWindow(drawform);

    XClearWindow(draw_d, draw_win);
}
void quit_callback(w, closure, call_data)
    Widget w;
    XtPointer closure;
    XtPointer call_data;
{
    exit(0);
}
void ncpu_callback(w, closure, call_data)
    Widget w;
    XtPointer closure;
    XtPointer call_data;
{
    Arg arg;

    ncpus++;
    if (ncpus > maxcpus) ncpus = 1;
```

(continued)

```
    sprintf(ncputext, "%d cpus", ncpus);
    arg.name = XtNlabel; arg.value = (XtArgVal) ncputext;
    XtSetValues(ncpulabel, &arg, 1);
}

XtCallbackRec clear_callbacks[] = {
  { clear_callback, NULL },
  { NULL, NULL },
};

void julia();
XtCallbackRec julia_callbacks[] = {
  { julia, NULL },
  { NULL, NULL },
};
void zoom();
XtCallbackRec zoom_callbacks[] = {
  { zoom, NULL },
  { NULL, NULL },
};
void mooz();
XtCallbackRec mooz_callbacks[] = {
  { mooz, NULL },
  { NULL, NULL },
};
void redo();
XtCallbackRec redo_callbacks[] = {
  { redo, NULL },
  { NULL, NULL },
};
void resetmandel();
XtCallbackRec mandel_callbacks[] = {
  { resetmandel, NULL },
  { NULL, NULL },
};
```

(continued)

```
XtCallbackRec quit_callbacks[] = {
  { quit_callback, NULL },
  { NULL, NULL },
};

void detail();
XtCallbackRec detail_callbacks[] = {
  { detail, NULL },
  { NULL, NULL },
};

void reset();
XtCallbackRec reset_callbacks[] = {
  { reset, NULL },
  { NULL, NULL },

};
void ncpu_callback();
XtCallbackRec ncpu_callbacks[] = {
  { ncpu_callback, NULL },
  { NULL, NULL },
};

main(argc,argv)
int argc;
char **argv;
{
    Widget toplevel, pane;
    Widget box, ibox, bbox;                      /* in widget pane */
    Widget hostlabel;                            /* in widget box  */
    Widget juliabutton, quitbutton, mandelbutton; /* in widget box  */
```

(continued)

```
    Widget moozbutton, zoombutton, redobutton, clearbutton;/*in box */
    Widget detailbutton, resetbutton, ncpubutton; /* in widget ibox */


    Display *dp;
    int i;
    char siter[20];
Arg argies[10];
struct utsname U;

toplevel = XtInitialize("Anyname","Xmandelf",NULL,0,&argc,argv);
dp = XtDisplay(toplevel);

depth = DisplayPlanes(dp,DefaultScreen(dp));

pane = XtCreateManagedWidget("pane", panedWidgetClass, toplevel,
    NULL, ZERO);

box = XtCreateManagedWidget("box", boxWidgetClass, pane, NULL,
    ZERO);
ibox = XtCreateManagedWidget("ibox", boxWidgetClass, pane, NULL,
    ZERO);

    uname (&U);
    i = 0;
XtSetArg( argies[i], XtNborderWidth, (XtArgVal) 0); i++;
    hostlabel = XtCreateManagedWidget(U.nodename, labelWidgetClass,
          box, argies, i);

i = 0;
XtSetArg( argies[i], XtNcallback, (XtArgVal) mandel_callbacks); i++;
    mandelbutton = XtCreateManagedWidget("mandel", commandWidgetClass,
          box, argies, i);
```

(continued)

```
  i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) zoom_callbacks); i++;
    zoombutton = XtCreateManagedWidget("mandelzoom", commandWidget-
Class,
          box, argies, i);

    i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) mooz_callbacks); i++;
    moozbutton = XtCreateManagedWidget("unzoom", commandWidgetClass,
          box, argies, i);

    i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) redo_callbacks); i++;
    redobutton = XtCreateManagedWidget("redo", commandWidgetClass,
          box, argies, i);

  i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) julia_callbacks); i++;
    juliabutton = XtCreateManagedWidget("julia", commandWidgetClass,
          box, argies, i);

  i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) clear_callbacks); i++;
    clearbutton = XtCreateManagedWidget("clear", commandWidgetClass,
          box, argies, i);

  i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) quit_callbacks); i++;
    quitbutton = XtCreateManagedWidget("quit", commandWidgetClass,
          box, argies, i);

    i = 0;
  XtSetArg( argies[i], XtNborderWidth, (XtArgVal) 0); i++;
    mflopslabel = XtCreateManagedWidget("   0 megaflops",
    labelWidgetClass, ibox, argies, i);
```

(continued)

```
  i = 0;
    if (depth > 1) iter = 256; else iter = 64;
    sprintf(siter, "%d iterations", iter);
  XtSetArg( argies[i], XtNborderWidth, (XtArgVal) 0); i++;
    detaillabel = XtCreateManagedWidget(siter, labelWidgetClass,
          ibox, argies, i);

    i = 0;
    ncpus = maxcpus = sysconf(_SC_CRAY_NCPU);
    sprintf(ncputext, "%d cpus", maxcpus);
  XtSetArg( argies[i], XtNborderWidth, (XtArgVal) 0); i++;
    ncpulabel = XtCreateManagedWidget(ncputext, labelWidgetClass,
          ibox, argies, i);

  i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) detail_callbacks); i++;
detailbutton = XtCreateManagedWidget("increase iterations",
commandWidgetClass,ibox, argies, i);

  i = 0;
  XtSetArg( argies[i], XtNcallback, (XtArgVal) reset_callbacks); i++;
resetbutton = XtCreateManagedWidget("reset iterations",
commandWidgetClass, ibox, argies, i);

    i = 0;
    XtSetArg( argies[i], XtNcallback, (XtArgVal) ncpu_callbacks); i++;
       ncpubutton = XtCreateManagedWidget("cpus", commandWidgetClass,
          ibox, argies, i);

       i = 0;
       XtSetArg( argies[i], XtNheight, (XtArgVal) 512); i++;
       XtSetArg( argies[i], XtNwidth, (XtArgVal) 512); i++;
       XtSetArg( argies[i], XtNbackground, (XtArgVal)
          BlackPixel(dp, DefaultScreen(dp))); i++;
```

(continued)

```
  drawform = XtCreateManagedWidget("DrawForm", logoWidgetClass, pane,
          argies, i);

  bbox = XtCreateManagedWidget("box", boxWidgetClass, pane, NULL,
ZERO);

  i = 0;
  XtSetArg( argies[i], XtNwidth, (XtArgVal) 512); i++;
  XtSetArg( argies[i], XtNresize, (XtArgVal) False); i++;
XtSetArg( argies[i], XtNborderWidth, (XtArgVal) 0); i++;
  xlabel = XtCreateManagedWidget("", labelWidgetClass,
          bbox, argies, i);

  i = 0;
  XtSetArg( argies[i], XtNwidth, (XtArgVal) 512); i++;
  XtSetArg( argies[i], XtNresize, (XtArgVal) False); i++;
XtSetArg( argies[i], XtNborderWidth, (XtArgVal) 0); i++;
  ylabel = XtCreateManagedWidget("", labelWidgetClass,
          bbox, argies, i);

    i = 0;
    XtSetArg( argies[i], XtNcallback, (XtArgVal) reset_callbacks);
i++;
  resetbutton = XtCreateManagedWidget("reset iterations",
commandWidgetClass,
          ibox, argies, i);

  XtRealizeWidget(toplevel);
  crosshair_xcr = XCreateFontCursor(dp, XC_crosshair);
XDefineCursor(dp, XtWindow(drawform), crosshair_xcr);

  XtMainLoop();
}
```

`mandel.c`
A.2

The following `mandel.c` routines handle all actions that result from the pushing of buttons. One routine calls the Fortran `mandelbrot` routine.

```
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <stdio.h>
extern Widget detaillabel, drawform, mflopslabel;
extern Widget xlabel, ylabel;
struct zoomd {
     struct zoomd *zp;
     double lx, ux;
     double ly, uy;
};
struct zoomd *zoomp = NULL, *zoompn;
char *malloc();
double GENERATE();

extern int depth;
extern long iter;
extern int ncpus;
long mwidth;
long bias = 1;                /* bias into color table */
long wwidth = 0;
long wheight = 0;

Arg arg;
XImage *xip = NULL;
char *dp = NULL;
Window draw_win;
GC draw_gc;
Screen *draw_Screen;
Display *draw_d;
Visual *draw_v;
```

(continued)

```
#define MIN(a,b)((a) < (b) ? (a) : (b))
#define MAX(a,b)((a) < (b) ? (b) : (a))

  void resetmandel(w, closure, call_data)
       Widget w;
       caddr_t closure;
       caddr_t call_data;
  {
       if (zoomp) { /* reset to beginning by popping zps off stack */
          while (zoompn = zoomp->zp) {
          free(zoomp);
          zoomp = zoompn;
       }
  } else {                                    /* This is first time */
          zoomp = (struct zoomd *) malloc(sizeof (struct zoomd));
       zoomp->zp = NULL;  /* NULL means last in stack - don't pop */
       zoomp->lx = -2.25; zoomp->ux = .75;
       zoomp->ly = -1.5; zoomp->uy = 1.5;
  }
  setxylabels();
  setiter();
  mandel(w, closure, call_data);
}

mandel(w, closure, call_data)
    Widget w;
    caddr_t closure;
    caddr_t call_data;
{
    double wx, wy, oldwx, oldwy;
    double x, y, x0, y0;
    double incrx, incry;
    double mflops;
    long ix, iy;
    long i;
    long zero = 0;
```

(continued)

```
    draw_d = XtDisplay(drawform);
    draw_win = XtWindow(drawform);
    draw_Screen = XtScreen(drawform);
    draw_gc = draw_Screen->default_gc;
    draw_v = draw_Screen->root_visual;

    if (dp == NULL) {
       mwidth = (depth > 1)? 64: 8;
       dp = malloc(mwidth*64);
       if (dp == NULL) {printf("malloc failed\n"); return; }
       if (xip) XDestroyImage(xip);
       xip = XCreateImage(draw_d, draw_v, depth, ZPixmap, 0, dp,
          64, 64, 8, 0);
       xip->byte_order = MSBFirst;
       xip->bitmap_bit_order = MSBFirst;
       xip->bits_per_pixel = 8;

    }
    XtSetArg(arg, XtNwidth, &wwidth);
    XtGetValues(drawform, &arg, 1);
    XtSetArg(arg, XtNheight, &wheight);
    XtGetValues(drawform, &arg, 1);

    incrx = (zoomp->ux - zoomp->lx)/wwidth;
    incry = (zoomp->uy - zoomp->ly)/wheight;

    y0 = zoomp->uy-incry;
    x0 = zoomp->lx+incrx;

    if (closure == 0) XClearWindow(draw_d, draw_win);

    mflops =
    GENERATE(&wheight,&y0,&incry,&wwidth,&x0,&incrx,&iter,&ncpus);
    setmflops(mflops);
  }
```

(continued)

```
  FPUTI(pix,piy,index)
  int *pix,*piy;
  int index[64*64];
  {
    int i,j;

  #define VSIZE 64

    if (depth == 1) {
       int *idp = (int *) dp;
       long shiftw;
       for (i = 0; i < VSIZE; i++)
          *(idp + i) = 0;
       for (i = 0; i < VSIZE; i++, idp++) {
          shiftw = 0x8000000000000000;
          for (j = 0; j < VSIZE; j++) {
             if ((index[i* VSIZE + j] & 7)  != 0)
                *idp |= shiftw;
             shiftw >>= 1;
          }
       }
  #if 0
    } else if (depth == 4) {
       for (i = 0; i < VSIZE; i+=2) {
         for (j = 0; j < VSIZE; j++) {
         dp[i * VSIZE + j/2] = ( ((index[i * VSIZE + j] & 0x0f) <<4)
|                               (index[i * VSIZE + j + 1] & 0x0f)  );
          }
       }
  #endif
    } else {                          /* depth == 8 */
       for (i = 0; i < VSIZE; i++) {
       for (j = 0; j < VSIZE; j++) {
```

(continued)

```
            dp[i * VSIZE + j] = index[i * VSIZE + j] + bias;
         }
      }
   }

   XPutImage(draw_d,draw_win,draw_gc,xip,0,0,*pix,*piy,VSIZE,VSIZE);
}

void zoom(w, closure, call_data)
     Widget w;
     caddr_t closure;
     caddr_t call_data;
{
     Window draw_win;
     GC draw_gc;
     Screen *draw_Screen;
     Display *draw_d;
     Window Root;
     int S;

     draw_d = XtDisplay(drawform);
     if (zoomp == NULL) { XBell(draw_d, 0); return; }
     draw_win = XtWindow(drawform);
     draw_Screen = XtScreen(drawform);
     draw_gc = draw_Screen->default_gc;
     S = DefaultScreen(draw_d);
     Root = RootWindow(draw_d, S);

     XSetForeground(draw_d, draw_gc, 0x55);
     XSetSubwindowMode(draw_d, draw_gc, IncludeInferiors);
     XSetFunction(draw_d, draw_gc, GXxor);

     XSelectInput(draw_d, draw_win,
        ButtonPressMask | ButtonReleaseMask | PointerMotionMask);
     while (1) {
```

(continued)

```
        static int rubberband = 1;
        XEvent report;
        XNextEvent(draw_d, &report);
        switch(report.type) {
           int winx0,winy0,winx1,winy1,width,height;
           int x0, y0, x1, y1;
           double tw, th;

           case ButtonPress:
              XGrabServer(draw_d);
              x0 = winx0 = report.xbutton.x;
              y0 = winy0 = report.xbutton.y;
                 rubberband = 0;
                 width = height = 0;
                 break;

              case ButtonRelease:
                 winx1 = report.xbutton.x;
                 winy1 = report.xbutton.y;
        XDrawRectangle(draw_d,draw_win, draw_gc, x0, y0, width, width);
                 if ((width = winx1 – winx0) < 0)
                         width = – width;
                 if ((height = winy1 – winy0) < 0)
                         height = – height;
                 if (width < height)
                         width = height;
                 x0 = (winx1 > winx0) ? winx0 : winx0 – width;
                 y0 = (winy1 > winy0) ? winy0 : winy0 – width;
                 x1 = x0 + width;
                 y1 = y0 + width;
        XDrawRectangle(draw_d,draw_win, draw_gc, x0, y0, width, width);
                 rubberband = 1;
```

(continued)

```
            XSetFunction(draw_d, draw_gc, GXcopy);
            XUngrabServer(draw_d);
            XFlush(draw_d);
            tw = zoomp->ux - zoomp->lx;
            th = zoomp->uy - zoomp->ly;
            zoompn = (struct zoomd *) malloc(sizeof (struct zoomd));
            zoompn->zp = zoomp;                /* push onto stack */
            zoompn->ux = zoomp->lx + ((double)x1/(double)wwidth) *tw;
            zoompn->lx = zoomp->lx + ((double)x0/(double)wwidth) *tw;
            zoompn->ly = zoomp->uy - ((double)y1/(double)wheight)*th;
            zoompn->uy = zoomp->uy - ((double)y0/(double)wheight)*th;
               zoomp = zoompn;        /* zoomp is current pointer */
               setxylabels();
               mandel(w, closure, call_data);
               return;       /* call mandel with new width, height */

         case MotionNotify:
            if (rubberband) break;
            while (XCheckTypedEvent(draw_d, MotionNotify, &report));
      XDrawRectangle(draw_d,draw_win, draw_gc, x0, y0, width, width);
               winx1 = report.xbutton.x;
               winy1 = report.xbutton.y;
               if ((width = winx1 - winx0) < 0)
                        width = - width;
               if ((height = winy1 - winy0) < 0)
                        height = - height;

               if (width < height)
                        width = height;
               x0 = (winx1 > winx0) ? winx0 : winx0 - width;
               y0 = (winy1 > winy0) ? winy0 : winy0 - width;
      XDrawRectangle(draw_d,draw_win, draw_gc, x0, y0, width, width);
```

(continued)

```
                break;
             default:
                break;
          }
     }
}

void mooz(w, closure, call_data)
     Widget w;
     caddr_t closure;
     caddr_t call_data;
{
     draw_d = XtDisplay(drawform);
     if (zoomp == NULL) { XBell(draw_d, 0); return; }
     if (zoomp->zp == NULL) { XBell(draw_d, 0); return; }

     if (zoompn = zoomp->zp) {                    /* last on stack */
          free(zoomp);
          zoomp = zoompn;
     }
     setxylabels();
     mandel(w, closure, call_data);
}
void redo(w, closure, call_data)
     Widget w;
     caddr_t closure;
     caddr_t call_data;
{
     draw_d = XtDisplay(drawform);
     if (zoomp == NULL) { XBell(draw_d, 0); return; }
     mandel(w, 1, call_data);
}

void detail(w, closure, call_data)
     Widget w;
```

(continued)

```
    caddr_t closure;
    caddr_t call_data;
{

    if (depth > 1) iter += 256; else iter += 64;
    setiter();
    /*mandel(w, 1, call_data);*/
}
void reset(w, closure, call_data)
    Widget w;
    caddr_t closure;
    caddr_t call_data;
{
    if (depth > 1) iter = 256; else iter = 64;
    setiter();
}

setiter()
{
    char siter[20];
    sprintf(siter, "%d iterations", iter);
    arg.name = XtNlabel; arg.value = (XtArgVal) siter;
    XtSetValues(detaillabel, &arg, 1);
}

setmflops(r)
float r;
{
    char string[20];
    sprintf(string, "%.1f megaflops", r);
    arg.name = XtNlabel; arg.value = (XtArgVal) string;
    XtSetValues(mflopslabel, &arg, 1);
}
setxylabels()
{
```

(continued)

```
    char xtext[80], ytext[80];
    sprintf(xtext,"%15.10f < x < %15.10f", zoomp->lx, zoomp->ux);
    arg.name = XtNlabel; arg.value = (XtArgVal) xtext;
    XtSetValues(xlabel, &arg, 1);
    sprintf(ytext,"%15.10f < y < %15.10f", zoomp->ly, zoomp->uy);
    arg.name = XtNlabel; arg.value = (XtArgVal) ytext;
    XtSetValues(ylabel, &arg, 1);
}
```

julia.c
A.3

The following julia.c routines handle the julia button by generating a julia set.

```
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <stdio.h>
extern Widget drawform;
struct zoomd {
    struct zoomd *zp;
    double lx, ux;
    double ly, uy;
};
extern int depth;
extern struct zoomd *zoomp;
extern long iter;
XWindowAttributes draw_wattr;

void julia(w, closure, call_data)
    Widget w;
```

(continued)

```
    caddr_t closure;
    caddr_t call_data;
{

    double a,b;
    double wx, wy, oldwx, oldwy;
    double sx, x, y;
    double incrx, incry;
    long ix, iy;
    static long width = 0;
    static long mwidth = 0;
    static long bias = 1;

    Arg arg;
    int wwidth, wheight;
    static XImage *xip1 = NULL, *xip2 = NULL;
    static char *dp1 = NULL, *dp2 = NULL;
    char *malloc();
    char ab[80];
    Window draw_win;
    static Window s_win = NULL;
    GC draw_gc;
    Screen *draw_Screen;
    Display *draw_d;
    Visual *draw_v;
    Window Root;
    int S;                                          /* Screen */

       draw_d = XtDisplay(drawform);
       if (zoomp == NULL)                { XBell(draw_d, 0); return;}
       draw_win = XtWindow(drawform);
       draw_Screen = XtScreen(drawform);
       draw_gc = draw_Screen->default_gc;
       draw_v = draw_Screen->root_visual;
```

(continued)

```
        XtSetArg(arg, XtNwidth, &wwidth);
        XtGetValues(drawform, &arg, 1);
        XtSetArg(arg, XtNheight, &wheight);
        XtGetValues(drawform, &arg, 1);

    XSelectInput(draw_d,draw_win,ButtonPressMask|ButtonReleaseMask);
        while (1) {
            int winx, winy;
            XEvent report;
            XNextEvent(draw_d, &report);
            if (report.type == ButtonRelease) {
  /* if (report.xbutton.x <= 0 || report.xbutton.y <= 0) {  */
  /* make sure button release is in drawform window         */
               if (report.xbutton.window != draw_win ||
                  report.xbutton.x & 0x8000 ||
                  report.xbutton.y & 0x8000 ||
                  (report.xbutton.x == 0 && report.xbutton.y == 0) ||
                  report.xbutton.x > wwidth ||
                  report.xbutton.y > wheight) {
                     XBell(draw_d, 0);
                     continue;
            }
            a = ( (double)report.xbutton.x/ (double)wwidth);
            b = ( (double)report.xbutton.y/ (double)wheight);
            a = (a * (zoomp->ux - zoomp->lx)) + zoomp->lx;
            b = zoomp->uy - (b * (zoomp->uy - zoomp->ly));
            /*printf("julia: a = %f, b = %f\n", a, b);*/
            break;
        }
        continue;
    }
    S = DefaultScreen(draw_d);
    Root = RootWindow(draw_d, S);
    if (s_win == NULL) {
```

(continued)

```
      s_win = XCreateSimpleWindow(draw_d,Root,0,0,256,256,1,1,0);
      XSelectInput(draw_d, s_win, ExposureMask);
      XMapWindow(draw_d, s_win);
      XSync(draw_d, 0);
      while (1) {
         XEvent report;
      XNextEvent(draw_d, &report);
      if (report.type == Expose) break;
   }
}
sprintf(ab, "%f + %fi", a,b);
XStoreName(draw_d, s_win, ab);
XClearWindow(draw_d, s_win);

XGetWindowAttributes(draw_d, s_win, &draw_wattr);
if (width != draw_wattr.width) {
      width = draw_wattr.width;
      mwidth = (depth > 1)? width: (1 + width/8);
      /* XDestroyImage should free dp1 and dp2 as well */
      if (xip1) {XDestroyImage(xip1); XDestroyImage(xip2); }
      dp1 = malloc(mwidth);
      dp2 = malloc(mwidth);
      if (dp2 == NULL) {printf("malloc failed\n"); return; }

      xip1 = XCreateImage(draw_d, draw_v, depth, ZPixmap, 0, dp1,
            width, 1, 8, mwidth);
      xip2 = XCreateImage(draw_d, draw_v, depth, ZPixmap, 0, dp2,
            width, 1, 8, mwidth);
      xip1->byte_order = xip2->byte_order = MSBFirst;
      xip1->bitmap_bit_order = xip2->bitmap_bit_order = MSBFirst;
}
sx = x = -1.5; y = -1.5;
incrx = -x/draw_wattr.width * 2;
incry = -y/draw_wattr.height * 2;
```

(continued)

```
for (iy = 0; iy < (1 + draw_wattr.height/2); iy++, y+= incry) {
  long i;
  x = sx;
  for (ix = 0; ix < draw_wattr.width; ix++, x+= incrx) {
       wx = x; wy = y;
       for (i = 0; i < iter; i++) {
                oldwx = wx;
                wx = wx * wx – wy * wy + a;
                wy = 2 * oldwx * wy + b;
                if (wx * wx + wy * wy > 4) break;
       }
       if (depth > 1) {                          /* color */
             *(dp1 + ix) = i + bias;
             *(dp2 + width – ix – 1) = i + bias;
       } else {
          if (i == iter) {          /* could also test if (i & 2) */
                dp1[ix/8] |= 1 << (7 – ix&7);
                dp2[(width – ix – 1)/8] |= 1 << (8 – (width – ix)&7);
          }
       }
    }

  XPutImage(draw_d, s_win, draw_gc, xip1, 0,0, 0, iy, width, 1);
if (iy != draw_wattr.height – iy)
  XPutImage(draw_d,s_win,draw_gc,xip2,0,0,0, draw_wattr.height – iy,
    width, 1);
  if (depth == 1) for (i = 0 ; i < mwidth; i++) *(dp1 + i) = *(dp2 +
i) = 0;
}

}
```

`generate.f` A.4

The following `generate.f` routine is a multitasked Fortran program that generates a Mandelbrot set.

```
      function generate(wheight,y0,incry,wwidth,x0,incrx,iter,n)
      parameter (incx=64,incy=64)
      real x0,incrx,y0,incry
      integer wwidth,wheight,index(0:incx-1,0:incy-1),flops
      complex w0(0:incx-1,0:incy-1), w(0:incx-1,0:incy-1)

cmic$ numcpus n
      flops = 0
      t0 = timef()

cmic$ do all autoscope private(index,w0,w)
ccmic$ do all shared(wheight,y0,incry,wwidth,x0,incrx,
ccmic$1              iter,dp,drawd,drawwin,drawgc,xip,flops)
ccmic$1 private(ix,iy,x,y,i,ssum,ixlo,iylo,ixhi,iyhi,index,w,w0,
findex,
ccmic$1          indexsum)

      do 500 iylo = 0,wheight-1,incy
        iyhi = min0(incy,wheight-1+1-iylo)
        do 500 ixlo = 0,wwidth-1,incx
          ixhi = min0(incx,wwidth-1+1-ixlo)

          do 100 iy=0,iyhi-1
            y = y0-iylo*incry - iy*incry
cdir$ shortloop
            do 100 ix=0,ixhi-1
              x = x0+ixlo*incrx + ix*incrx
              w0(ix,iy) = cmplx(x,y)
              w(ix,iy) = w0(ix,iy)*w0(ix,iy) + w0(ix,iy)
              index(ix,iy)=cvmgt(0,1,(real(w(ix,iy))*real(w(ix,iy))
     +                  + aimag(w(ix,iy))*aimag(w(ix,iy)).le.4.))
   100         continue
```

(continued)

```
          do 300 i=1,iter-1
            do 200 iy=0,iyhi-1
cdir$ shortloop
              do 200 ix=0,ixhi-1
                if ( index(ix,iy) .eq. 0) then
                  w(ix,iy) = w(ix,iy)*w(ix,iy) + w0(ix,iy)
                index(ix,iy)=cvmgt(0,i,(real(w(ix,iy))*real(w(ix,iy))
     +                    + aimag(w(ix,iy))*aimag(w(ix,iy)).le.4.))
                end if
   200           continue
   300         continue

               indexsum = 0
               do 400 iy=0,iyhi-1
cdir$ shortloop
                 do 400 ix=0,ixhi-1
                   if (index(ix,iy).eq.0) index(ix,iy)=iter
                   indexsum = indexsum + index(ix,iy)
   400         continue

cmic$ guard
      flops = flops + 13*incx*incy + 11*indexsum
      call FPUTI(ixlo,iylo,index)
cmic$ end guard

   500 continue

      telapse = timef() - t0
      generate = 1.e-3*flops/telapse
c     print 990, 1.e-3*flops/telapse
c 990 format(' Mflops: ',f7.2)

          return
          end
```

`makefile` A.5

The following `makefile` program compiles the `xmandelf` program.

```
F=cf77
CC=cc
CFLAGS = $(INCLUDES) –DUSG –DSYSV
FFLAGS =  –ZP
SHELL=/bin/sh

xmandelf: xmandelf.o mandel.o julia.o generate.o
        $(CF) $(FFLAGS) –o $@ xmandelf.o mandel.o julia.o  generate.o \
                –lXaw –lXt –lXmu –lX11 –lXext

generate.o:        generate.f
        $(CF) $(FFLAGS) –c generate.f
clean:
        @–rm xmandelf *.o

install:        xmandelf
        if [ ! –d $$HOME/bin ] ; \
        then \
                echo mkdir $$HOME/bin;\
                mkdir $$HOME/bin;\
        fi
        cp xmandelf $$HOME/bin
```