mandel-fork2.c

```c
/*
 * mandel.c
 *
 * A program to draw the Mandelbrot Set on a 256-color xterm.
 *
 */

#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

#include <sys/mman.h>
#include <sys/wait.h>

/*TODO header file for m(un)map*/

#include "mandel-lib.h"

#define MANDEL_MAX_ITERATION 100000

/***************************
 * Compile-time parameters *
 ***************************/

/*
 * Output at the terminal is is x_chars wide by y_chars long
*/
int y_chars = 50;
int x_chars = 90;

/*
 * The part of the complex plane to be drawn:
 * upper left corner is (xmin, ymax), lower right corner is (xmax, ymin)
*/
double xmin = -1.8, xmax = 1.0;
double ymin = -1.0, ymax = 1.0;

/*
 * Every character in the final output is
 * xstep x ystep units wide on the complex plane.
 */
double xstep;
double ystep;

int **color_val;

int safe_atoi(char *s, int *val)
{
        long l;
        char *endp;

        l = strtol(s, &endp, 10);
        if (s != endp && *endp == '\0') {
                *val = l;
```

```
58                  return 0;
59          } else
60                  return -1;
61  }
62
63  void usage(char *argv0)
64  {
65          fprintf(stderr, "Usage: %s procedure_count\n"
66                  "Exactly one argument required:\n"
67                  "    procedure_count: The number of procedures to create.\n",
68                  argv0);
69          exit(1);
70  }
71
72  /*
73   * This function computes a line of output
74   * as an array of x_char color values.
75   */
76  void compute_mandel_line(int line)
77  {
78      /*
79       * x and y traverse the complex plane.
80       */
81      double x, y;
82
83      int n;
84      int val;
85
86      /* Find out the y value corresponding to this line */
87      y = ymax - ystep * line;
88
89      /* and iterate for all points on this line */
90      for (x = xmin, n = 0; n < x_chars; x+= xstep, n++) {
91
92          /* Compute the point's color value */
93          val = mandel_iterations_at_point(x, y, MANDEL_MAX_ITERATION);
94          if (val > 255)
95              val = 255;
96
97          /* And store it in the color_val[] array */
98          val = xterm_color(val);
99          color_val[n][line] = val;
100     }
101 }
102
103 /*
104  * This function outputs an array of x_char color values
105  * to a 256-color xterm.
106  */
107 void output_mandel_line(int fd, int line)
108 {
109     int i;
110
111     char point ='@';
112     char newline='\n';
113
114     for (i = 0; i < x_chars; i++) {
115         /* Set the current color, then output the point */
116         set_xterm_color(fd, color_val[i][line]);
117         if (write(fd, &point, 1) != 1) {
```

```
118             perror("compute_and_output_mandel_line: write point");
119             exit(1);
120         }
121     }
122 
123     /* Now that the line is done, output a newline character */
124     if (write(fd, &newline, 1) != 1) {
125         perror("compute_and_output_mandel_line: write newline");
126         exit(1);
127     }
128 }
129 
130 /*
131  * Create a shared memory area, usable by all descendants of the calling
132  * process.
133  */
134 void *create_shared_memory_area(unsigned int numbytes)
135 {
136     int pages;
137     void *addr;
138 
139     if (numbytes == 0) {
140         fprintf(stderr, "%s: internal error: called for numbytes == 0\n", __func__);
141         exit(1);
142     }
143 
144     /*
145      * Determine the number of pages needed, round up the requested number of
146      * pages
147      */
148     pages = (numbytes - 1) / sysconf(_SC_PAGE_SIZE) + 1;
149 
150     /* Create a shared, anonymous mapping for this number of pages */
151     /* TODO:*/
152     addr = mmap(NULL, pages * sysconf(_SC_PAGE_SIZE), PROT_READ | PROT_WRITE, MAP_SHARED |
    MAP_ANONYMOUS, -1, 0);
153     if(addr == MAP_FAILED) {perror("mmap"); exit(1);}
154 
155     return addr;
156 }
157 
158 void destroy_shared_memory_area(void *addr, unsigned int numbytes) {
159     int pages;
160 
161     if (numbytes == 0) {
162         fprintf(stderr, "%s: internal error: called for numbytes == 0\n", __func__);
163         exit(1);
164     }
165 
166     /*
167      * Determine the number of pages needed, round up the requested number of
168      * pages
169      */
170     pages = (numbytes - 1) / sysconf(_SC_PAGE_SIZE) + 1;
171 
172     if (munmap(addr, pages * sysconf(_SC_PAGE_SIZE)) == -1) {
173         perror("destroy_shared_memory_area: munmap failed");
174         exit(1);
175     }
176 }
```

```c

int main(int argc, char *argv[])
{
    pid_t p;
    int procs;

    /*
         * Parse the command line
         */
        if (argc != 2)
                usage(argv[0]);
        if (safe_atoi(argv[1], &procs) < 0 || procs <= 0) {
                fprintf(stderr, "`%s' is not valid for `procedures_count'\n", argv[1]);
                exit(1);
        }

    xstep = (xmax - xmin) / x_chars;
    ystep = (ymax - ymin) / y_chars;

    // Initialize 2-D array
    color_val = create_shared_memory_area(x_chars * sizeof(int));

    for(int i = 0; i < x_chars; i++)
        color_val[i] = create_shared_memory_area(y_chars * sizeof(int));


    /*
     * draw the Mandelbrot Set, one line at a time.
     * Output is sent to file descriptor '1', i.e., standard output.
     */
    for (int init_line = 0; init_line < procs; init_line++) {
        p = fork();
        if(p < 0) {perror("fork"); exit(1);}
        if(p == 0) {
            for(int line = init_line; line < y_chars; line += procs)
                        compute_mandel_line(line);
                    exit(0);
        }
    }

    // Parent waits
    for(int i=0; i<procs; i++)
        wait(NULL);

    // Print output
    for(int line = 0; line < y_chars; line++)
        output_mandel_line(1, line);

    // Destroy memory
    for(int i = 0; i < y_chars; i++)
        destroy_shared_memory_area(color_val, y_chars * sizeof(int));
    destroy_shared_memory_area(color_val, x_chars * sizeof(int));

    reset_xterm_color(1);
    return 0;
}

```