FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

66.20 – Organización de Computadoras

1º Cuatrimestre 2018

Trabajo Práctico 1: Infraestructura básica

Integrantes grupo:

Eliana Diaz - Padrón: 89.324 - Email: diazeliana09@gmail.com

Lucas Verón - Padrón: 89.341 - Email: lucasveron86@gmail.com

Bruno Mangiafave - Padrón: 96.420 - Email: mangiafave.bruno@gmail.com

Fecha entrega: 24/04/2018

Nro. de entrega: 1

1. Introducción

A continuación se detallará el diseño e implementación de un programa en lenguaje C y MIPS, que permita dibujar el conjunto de Julia y sus vecindades, como así también la forma de ejecución del mismo y los resultados obtenidos en las distintas pruebas ejecutadas.
El programa recibe, por línea de comando, una serie de parámetros describiendo la región del plano complejo y las características del archivo imagen a generar. Al finalizar la ejecución, y volver al sistema operativo, el programa habrá dibujado el fractal en el archivo de salida. El formato gráfico a usar es pgm.

2. Diseño

Para poder desarrollar la funcionalidad requerida, dibujo del conjunto de Julia y sus vecindades, se especifican los parámetros permitidos, junto a sus limitaciones:

- **-r, o --resolution**
  Permite cambiar la resolución de la imagen generada. El valor por defecto será de 640x480 puntos.
  Formatos aceptados:
  <<número>>x<<número>>
  <<número>>X<<número>>

- **-c, o --center**
  Permite especificar las coordenadas correspondientes al punto central de la porción del plano complejo dibujada, expresado en forma binómica (por ejemplo, a+bi). Por defecto, se usará 0+0i.
  Formato aceptado:
  <<parte real, con signo negativo de ser necesario>>+/-<<parte imaginaria>>i

- **-w, o --width**
  Permite especificar el ancho de la región del plano complejo a dibujar. Valor por defecto: 2
  Formato aceptado: se permiten tanto números enteros como decimales.

- **-H, o --height**
  Permite especificar el ancho de la región del plano complejo a dibujar. Valor por defecto: 2
  Formato aceptado: se permiten tanto números enteros como decimales.

- **-m, o --method**
  Permite configurar el método de ejecución de la application. Los valores posibles son generic y mips32. generic es el algoritmo por defecto cuando no se especifica el parámetro. El algoritmo usado es el correspondiente a generic.c . El método mips32 es la implementación en ensamblador mips32. La misma implementa un buffer.

- **-o, o --output**
  Permite especificar el archivo en donde se colo. No tiene valor por defecto.

Para especificar la salida estándar -cout- se indica con el signo -.
Formatos aceptados:
-
<<nombre del archivo>>.pgm
Sólo se permite letras y números, la barra / para indicar directorio, puntos y se debe de indicar extensión del archivo. La extensión puede estar tanto en mayúscula como en minúscula.

3. Implementación

3.1. Código fuente en lenguaje C

constants.h

```
/*
 * constants.h
 *
 */

#ifndef CONSTANTS_H_
#define CONSTANTS_H_


#define MAX_LENGTH_CHARACTER          11
#define MAX_BUFFER                    100

#define TRUE                          1
#define FALSE                         0

/* State */
#define    OKEY                       0
#define ERROR_FILE                    1
#define ERROR_MEMORY                  2
#define ERROR_WRITE                   3

#endif /* CONSTANTS_H_ */
```

debug.h

```
#ifndef _DEBUG_H_INCLUDED_
#define _DEBUG_H_INCLUDED_

#include <stdio.h>
#include <stdlib.h>

#ifdef DEBUG
#define ASSERT(expr)                                    \
        do {                                            \
                if (!(expr)) {                          \
                        fprintf(stderr,                 \
                            "panic: assertion %s failed: %s() %s:%d.\n", \
                            (#expr),                    \
                            __FUNCTION__,               \
                            __FILE__,                   \
                            __LINE__);                  \
                        fflush(stderr);                 \
                        abort();                        \
```

```
                        }                                               \
        } while (0)
#else
#define ASSERT(expr) do { } while (0)
#endif

#endif
```

defs.h

```
#ifndef _DEFS_H_INCLUDED_
#define _DEFS_H_INCLUDED_

#ifdef SUNOS5
/*
 * Solaris 9 no tiene <stdint.h>
 */
#define MISSING_STDINT_H

/*
 * Tampoco parece estar <getopt.h>
 */
#define MISSING_GETOPT

/*
 * Includes complementarios.
 */
#include <sys/types.h>
#endif

#ifdef MISSING_GETOPT
#include <mygetopt.h>
#else
#include <getopt.h>
#endif

#ifndef MISSING_STDINT_H
#include <stdint.h>
#endif

#include <ctype.h>

/*
 * Turn on format string argument checking. This is more accurate than
 * printfck, but it misses #ifdef-ed code. XXX I am just guessing at what
 * gcc versions support this. In order to turn this off for some platforms,
 * specify #define PRINTFLIKE and #define SCANFLIKE in the system-dependent
 * sections above.
 */
#ifndef PRINTFLIKE
#if (__GNUC__ == 2 && __GNUC_MINOR__ >= 7) || __GNUC__ == 3
#define PRINTFLIKE(x, y) __attribute__ ((format (printf, (x), (y))))
#else
#define PRINTFLIKE(x, y)
#endif
#endif

/*
 * Need to specify what functions never return, so that the compiler can
 * warn for missing initializations and other trouble. However, OPENSTEP4
 * gcc 2.7.x cannot handle this so we define this only if NORETURN isn't
 * already defined above.
 *
 * Data point: gcc 2.7.2 has __attribute__ (Wietse Venema) but gcc 2.6.3 does
```

```
 * not (Clive Jones). So we'll set the threshold at 2.7.
 */
#ifndef NORETURN
#if (__GNUC__ == 2 && __GNUC_MINOR__ >= 7) || __GNUC__ >= 3
#define NORETURN void __attribute__((__noreturn__))
#endif
#endif

#ifndef NORETURN
#define NORETURN void
#endif

/*
 * Safety. On some systems, ctype.h misbehaves with non-ASCII or negative
 * characters. More importantly, Postfix uses the ISXXX() macros to ensure
 * protocol compliance, so we have to rule out non-ASCII characters.
 */
#define _UCHAR_(c) ((unsigned char)(c))
#define ISASCII(c) (isascii(_UCHAR_(c)))
#define ISPRINT(c) (ISASCII(c) && isprint(_UCHAR_(c)))
#define ISSPACE(c) (ISASCII(c) && isspace(_UCHAR_(c)))

#endif
```

mygetopt.h

```
/*		$NetBSD: getopt.h,v 1.7 2005/02/03 04:39:32 perry Exp $	*/

/*-
 * Copyright (c) 2000 The NetBSD Foundation, Inc.
 * All rights reserved.
 *
 * This code is derived from software contributed to The NetBSD Foundation
 * by Dieter Baron and Thomas Klausner.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *        This product includes software developed by the NetBSD
 *        Foundation, Inc. and its contributors.
 * 4. Neither the name of The NetBSD Foundation nor the names of its
 *    contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */
```

```
#ifdef MISSING_GETOPT
#ifndef _GETOPT_H_
#define _GETOPT_H_

#if 0
#include <sys/cdefs.h>
#include <sys/featuretest.h>
#endif
#include <unistd.h>

/*
 * Gnu like getopt_long() and BSD4.4 getsubopt()/optreset extensions
 */
#define no_argument        0
#define required_argument  1
#define optional_argument  2

struct option {
        /* name of long option */
        const char *name;
        /*
         * one of no_argument, required_argument, and optional_argument:
         * whether option takes an argument
         */
        int has_arg;
        /* if not NULL, set *flag to val when option found */
        int *flag;
        /* if flag not NULL, value to set *flag to; else return value */
        int val;
};

int getopt_long(int, char * const *, const char *,
   const struct option *, int *);

#endif /* !_GETOPT_H_ */
#endif /* MISSING_GETOPT */
```

param.h

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
        float UL_re; /* upper left point, real part */
        float UL_im; /* upper left point, imaginary part */
        float LR_re; /* lower right point, real part */
        float LR_im; /* lower right point, imaginary part */
        float d_re;  /* pixel step, real part */
        float d_im;  /* pixel step, imaginary part */
        float s_re;  /* seed of julia set, real part */
        float s_im;  /* seed of julia set, imaginary part */

        size_t x_res;  /* horizontal resolution, e.g. 640 */
        size_t y_res;  /* vertical resolution, e.g. 480 */
        size_t shades; /* amount of shades of gray, e.g. 255 */

        FILE *fp;
} param_t;
```

makedefs

```
#! /bin/sh
```

```
# Ugly function to make our error message more visible among the
# garbage that is output by some versions of make(1).
#
# By now all shells must have functions.
#
error() {
        # Alas, tput(1) is not portable so we can't use visual effects.
        echo "ATTENTION:" 1>&2;
        echo "ATTENTION:" $* 1>&2;
        echo "ATTENTION:" 1>&2;
        exit 1
}

SYSTEM=`(uname -s) 2>/dev/null`
RELEASE=`(uname -r) 2>/dev/null`
MACHINE=`(uname -m) 2>/dev/null`
VERSION=`(uname -v) 2>/dev/null`
SYSLIBS=-lm

case "$SYSTEM.$RELEASE" in
NetBSD.3*)
        SYSTYPE=NETBSD3
        ;;
Linux.2*)
        SYSTYPE=LINUX2
        ;;
Linux.3*)
        SYSTYPE=LINUX3
        ;;
Linux.4*)
        SYSTYPE=LINUX4
        ;;
*)
        error "Unknown system type: $SYSTEM $RELEASE"
        ;;
esac

: ${CC='gcc $(WARN)'} ${OPT='-O'} ${DEBUG='-g'} ${WARN='-Wall'}

export SYSTYPE SYSLIBS CC OPT DEBUG OPTS

sed 's/  / /g' <<EOF
SYSTYPE         = $SYSTYPE
SYSLIBS         = $AUXLIBS $SYSLIBS
CC      = $CC $CCARGS
OPT     = $OPT
DEBUG   = $DEBUG
EXPORT = AUXLIBS='$AUXLIBS' CCARGS='$CCARGS' OPT='$OPT' DEBUG='$DEBUG'
WARN    = $WARN
EOF
```

makedefs.out

```
# Do not edit -- this file documents how this program was built for your machine.
SYSTYPE         = LINUX4
SYSLIBS         = -lm
CC      = gcc $(WARN)
OPT     = -O
DEBUG   = -g
EXPORT = AUXLIBS='' CCARGS='' OPT='-O' DEBUG='-g'
WARN    = -Wall
```

Makefile

```
# Do not edit -- this file documents how this program was built for your machine.
SYSTYPE             = LINUX4
SYSLIBS             = -lm
CC          = gcc $(WARN)
OPT         = -O
DEBUG  = -g
EXPORT = AUXLIBS='' CCARGS='' OPT='-O' DEBUG='-g'
WARN    = -Wall
SHELL    = /bin/sh
DEFS    = -I. -D$(SYSTYPE)
CFLAGS  = $(DEBUG) $(OPT) $(DEFS)
OPTS      = 'CC=$(CC)'
HDRS    = debug.h defs.h mygetopt.h param.h
SRCS      = fileFunctions.S plotFunctions.S utilityFunctions.S generic.c main.c mygetopt_long.c
PROG     = tp1

$(PROG): $(SRCS) $(HDRS)
            $(CC) $(CFLAGS) -o $@ $(SRCS) $(SYSLIBS)

update: $(PROG)
            :

makefiles Makefiles:
            (echo "# Do not edit -- this file documents how this program was built for your machine."; $(SHELL) makedefs)
>makedefs.tmp
            set +e;                             \
            if cmp makedefs.tmp makedefs.out; then \
               rm makedefs.tmp;                 \
            else                                \
               mv makedefs.tmp makedefs.out;      \
            fi >/dev/null 2>/dev/null
            rm -f Makefile; (cat makedefs.out Makefile.in) >Makefile

clean:
            rm -f Makefile
            cp Makefile.init Makefile
            rm -f bin/[!CRS]* lib/[!CRS]* include/[!CRS]* libexec/[!CRS]*     \
                 junk *core .nfs* .pure *.out *.out.* *.tmp *.a *~ *- *.orig \
                 *.t *.o *.bak make.err *.gmon $(PROG)
```

Makefile.in

```
SHELL    = /bin/sh
DEFS    = -I. -D$(SYSTYPE)
CFLAGS  = $(DEBUG) $(OPT) $(DEFS)
OPTS      = 'CC=$(CC)'
HDRS    = debug.h defs.h mygetopt.h param.h
SRCS      = fileFunctions.S plotFunctions.S utilityFunctions.S generic.c main.c mygetopt_long.c
PROG     = tp1

$(PROG): $(SRCS) $(HDRS)
            $(CC) $(CFLAGS) -o $@ $(SRCS) $(SYSLIBS)

update: $(PROG)
            :

makefiles Makefiles:
            (echo "# Do not edit -- this file documents how this program was built for your machine."; $(SHELL) makedefs)
>makedefs.tmp
```

```
	set +e;                             \
	if cmp makedefs.tmp makedefs.out; then \
	   rm makedefs.tmp;                 \
	else                                \
	   mv makedefs.tmp makedefs.out;    \
	fi >/dev/null 2>/dev/null
	rm -f Makefile; (cat makedefs.out Makefile.in) >Makefile

clean:
	rm -f Makefile
	cp Makefile.init Makefile
	rm -f bin/[!CRS]* lib/[!CRS]* include/[!CRS]* libexec/[!CRS]*    \
	     junk *core .nfs* .pure *.out *.out.* *.tmp *.a *~ *- *.orig \
	     *.t *.o *.bak make.err *.gmon $(PROG)
```

Makefile.init

```
# Usage:
#	make makefiles [CC=compiler] [OPT=compiler-flags] [DEBUG=debug-flags]
#
# The defaults are: CC=gcc, OPT=-O, and DEBUG=-g. Examples:
#
#	make makefiles
#	make makefiles CC="purify cc"
#	make makefiles CC=cc OPT=
#
SHELL   = /bin/sh

default: update

update depend clean: Makefiles
	$(MAKE) MAKELEVEL= $@

makefiles Makefiles:
	$(MAKE) -f Makefile.in MAKELEVEL= Makefiles
```

main.c

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

#include <defs.h>
#include <debug.h>
#include <param.h>

#ifndef VERSION
#define VERSION "0.0.1-cvs"
#endif

#ifndef no_argument
#define no_argument 0
#endif

#ifndef required_argument
#define required_argument 1
#endif

#ifndef optional_argument
#define optional_argument 2
#endif
```

```
static void do_plot(void);
extern void mips32_plot(param_t *);
extern void generic(param_t *);

/*
 * Parametros globales.
 */

int x_res = 640;                /* Ancho de imagen por defecto. */
int y_res = 480;                /* Alto de imagen, por defecto. */
float upper_left_re = -1.0f;    /* Extremo superior izquierzo (re). */
float upper_left_im = +1.0f;   /* Extremo superior izquierzo (im). */
float lower_right_re = +1.0f;  /* Extremo inferior derecho (re). */
float lower_right_im = -1.0f;  /* Extremo inferior derecho (im). */
float seed_re = -0.72689535f;   /* Semilla (re). */
float seed_im = +0.18888713f;   /* Semilla (im). */
void (*plot)(param_t *) = NULL;
FILE *output = NULL;

static void parse_cmdline(int, char * const []);
static void do_usage(const char *, int);
static void do_version(const char *);
static void do_resolution(const char *, const char *);
static void do_geometry(const char *, const char *);
static void do_center(const char *, const char *);
static void do_width(const char *, const char *);
static void do_height(const char *, const char *);
static void do_method(const char *, const char *);
static void do_output(const char *, const char *);

int
main(int argc, char * const argv[], char * const envp[])
{
        parse_cmdline(argc, argv);
        do_plot();

        return 0;
}

static void
parse_cmdline(int argc, char * const argv[])
{
        int ch;
        int index = 0;

        struct option options[] = {
                {"help", no_argument, NULL, 'h'},
                {"version", no_argument, NULL, 'V'},
                {"geometry", required_argument, NULL, 'g'},
                {"resolution", required_argument, NULL, 'r'},
                {"center", required_argument, NULL, 'c'},
                {"width", required_argument, NULL, 'w'},
                {"height", required_argument, NULL, 'H'},
                {"output", required_argument, NULL, 'o'},
        };

        while ((ch = getopt_long(argc, argv,
                        "hc:H:m:o:r:w:g:V", options, &index)) != -1) {
                switch (ch) {
                case 'h':
                        do_usage(argv[0], 0);
                        break;
                case 'V':
                        do_version(argv[0]);
                        break;
                case 'g':
```

```
                        do_geometry(argv[0], optarg);
                        break;
                case 'r':
                        do_resolution(argv[0], optarg);
                        break;
                case 'c':
                        do_center(argv[0], optarg);
                        break;
                case 'w':
                        do_width(argv[0], optarg);
                        break;
                case 'H':
                        do_height(argv[0], optarg);
                        break;
                case 'm':
                        do_method(argv[0], optarg);
                        break;
                case 'o':
                        do_output(argv[0], optarg);
                        break;
                default:
                        do_usage(argv[0], 1);
                }
        }

        if (plot == NULL)
                plot = &generic;

        if (output == NULL)
                output = stdout;
}

static void
do_usage(const char *name, int status)
{
        fprintf(stderr, "Usage:\n");
        fprintf(stderr, "  %s -h\n", name);
        fprintf(stderr, "  %s -V\n", name);
        fprintf(stderr, "  %s [options]\n", name);
        fprintf(stderr, "Options:\n");
        fprintf(stderr, "  -r, --resolution "
                " Set bitmap resolution to WxH pixels.\n");
        fprintf(stderr, "  -c, --center     "
                " Set coordinates for the center of the image.\n");
        fprintf(stderr, "  -w, --width      "
                " Change the width of the spanned region.\n");
        fprintf(stderr, "  -H, --height     "
                " Change the height of the spanned region.\n");
        fprintf(stderr, "  -o, --output     "
                " Path to output file.\n");
        fprintf(stderr, "Examples:\n");
        fprintf(stderr, "  %s -o output.pgm\n", name);
        fprintf(stderr, "  %s -r 1600x1200 -o output.pgm\n", name);
        fprintf(stderr, "  %s -c +0.282+0.01i -o output.pgm\n", name);
        exit(status);
}

static void
do_version(const char *name)
{
        fprintf(stderr, "%s\n", VERSION);
        exit(0);
}

static void
do_resolution(const char *name, const char *spec)
```

```
{
        int x;
        int y;
        char c;
        char d;

        if (sscanf(spec, "%d%c%d %c", &x, &c, &y, &d) != 3
           || x <= 0
           || c != 'x'
           || y <= 0)
                do_usage(name, 1);

        /* Set new resolution. */
        x_res = x;
        y_res = y;
}

static void
do_geometry(const char *name, const char *spec)
{
        double re_1, im_1;
        double re_2, im_2;
        char comma;
        char sg_1;
        char sg_2;
        char ii_1;
        char ii_2;
        char ch;

#define PLUS_OR_MINUS(c)  ((c) == '+' || (c) == '-')
#define IMAGINARY_UNIT(x) ((x) == 'i' || (x) == 'j')

        if (sscanf(spec,
                "%lf %c %lf %c %c %lf %c %lf %c %c",
                &re_1,
                &sg_1,
                &im_1,
                &ii_1,
                &comma,
                &re_2,
                &sg_2,
                &im_2,
                &ii_2,
                &ch) != 9
           || !PLUS_OR_MINUS(sg_1)
           || !PLUS_OR_MINUS(sg_2)
           || !IMAGINARY_UNIT(ii_1)
           || !IMAGINARY_UNIT(ii_2)
           || comma != ',') {
                fprintf(stderr, "invalid geometry specification.\n");
                exit(1);
        }

#define MINIMUM(x, y) ((x) <= (y) ? (x) : (y))
#define MAXIMUM(x, y) ((x) >= (y) ? (x) : (y))
#define SIGN(c) ((c) == '-' ? -1.0 : +1.0)

        /* Sign-adjust. */
        im_1 *= SIGN(sg_1);
        im_2 *= SIGN(sg_2);

        /*
         * We have two edges of the rectangle. Now, find the upper-left
         * (i.e. the one with minimum real part and maximum imaginary
         * part) and lower-right (maximum real part, minimum imaginary)
         * corners of the rectangle.
```

```
		 */
		upper_left_re = MINIMUM(re_1, re_2);
		upper_left_im = MAXIMUM(im_1, im_2);
		lower_right_re = MAXIMUM(re_1, re_2);
		lower_right_im = MINIMUM(im_1, im_2);
}

static void
do_center(const char *name, const char *spec)
{
		double width;
		double height;
		double re, im;
		char ii;
		char sg;
		char ch;

		if (sscanf(spec,
			"%lf %c %lf %c %c",
			&re,
			&sg,
			&im,
			&ii,
			&ch) != 4
		   || !PLUS_OR_MINUS(sg)
		   || !IMAGINARY_UNIT(ii)) {
				fprintf(stderr, "invalid center specification.\n");
				exit(1);
		}

		im *= SIGN(sg);
		width = fabs(upper_left_re - lower_right_re);
		height = fabs(upper_left_im - lower_right_im);

		upper_left_re = re - width / 2;
		upper_left_im = im + height / 2;
		lower_right_re = re + width / 2;
		lower_right_im = im - height / 2;
}

static void
do_height(const char *name, const char *spec)
{
		double width;
		double height;
		double re, im;
		char ch;

		if (sscanf(spec,
			"%lf %c",
			&height,
			&ch) != 1
		   || height <= 0.0) {
				fprintf(stderr, "invalid height specification.\n");
				exit(1);
		}

		re = (upper_left_re + lower_right_re) / 2;
		im = (upper_left_im + lower_right_im) / 2;
		width = fabs(upper_left_re - lower_right_re);

		upper_left_re = re - width / 2;
		upper_left_im = im + height / 2;
		lower_right_re = re + width / 2;
		lower_right_im = im - height / 2;
}
```

```
static void
do_width(const char *name, const char *spec)
{
	double width;
	double height;
	double re, im;
	char ch;

	if (sscanf(spec,
		"%lf %c",
		&width,
		&ch) != 1
	   || width <= 0.0) {
			fprintf(stderr, "invalid width specification.\n");
			exit(1);
	}

	re = (upper_left_re + lower_right_re) / 2;
	im = (upper_left_im + lower_right_im) / 2;
	height = fabs(upper_left_im - lower_right_im);

	upper_left_re = re - width / 2;
	upper_left_im = im + height / 2;
	lower_right_re = re + width / 2;
	lower_right_im = im - height / 2;
}

static void
do_method(const char *name, const char *spec)
{
	if (strcmp(spec, "mips32") == 0) {
			plot = &mips32_plot;
			return ;
	}

	plot = &generic;
}

static void
do_output(const char *name, const char *spec)
{
	if (output != NULL) {
			fprintf(stderr, "multiple do output files.");
			exit(1);
	}

	if (strcmp(spec, "-") == 0) {
			output = stdout;
	} else {
			if (!(output = fopen(spec, "w"))) {
					fprintf(stderr, "cannot open output file.\n");
					exit(1);
			}
	}
}

static void
do_plot(void)
{
	param_t parms;

	memset(&parms, 0, sizeof(parms));
	parms.UL_re = upper_left_re;
	parms.UL_im = upper_left_im;
	parms.LR_re = lower_right_re;
```

```
        parms.LR_im = lower_right_im;
    parms.d_re = (lower_right_re - upper_left_re) / x_res;
    parms.d_im = (upper_left_im - lower_right_im) / y_res;
    parms.s_re = seed_re;
        parms.s_im = seed_im;
        parms.x_res = x_res;
    parms.y_res = y_res;
    parms.shades = 256;
    parms.fp = output;

        plot(&parms);
}
```

generic.c

```
#include <debug.h>
#include <stdio.h>
#include <defs.h>
#include <param.h>

void
generic(param_t *parms)
{
        float cr, ci;
        float zr, zi;
        float tr, ti;
        float absz;
        int x, y;
        int c;

        /* Header PGM. */
        fprintf(parms->fp, "P2\n");
        fprintf(parms->fp, "%u\n", (unsigned)parms->x_res);
        fprintf(parms->fp, "%u\n", (unsigned)parms->y_res);
        fprintf(parms->fp, "%u\n", (unsigned)(parms->shades - 1));

        /*
         * Barremos la region rectangular del plano complejo comprendida
         * entre (parms->UL_re, parms->UL_im) y (parms->LR_re, parms->LR_im).
         * El parametro de iteracion es el punto (cr, ci).
         */
        for (y = 0, ci = parms->UL_im;
            y < parms->y_res;
            ++y, ci -= parms->d_im) {
                for (x = 0, cr = parms->UL_re;
                    x < parms->x_res;
                    ++x, cr += parms->d_re) {
                        zr = cr;
                        zi = ci;

                        /*
                         * Determinamos el nivel de brillo asociado al punto
                         * (cr, ci), usando la formula compleja recurrente
                         * f = f^2 + s.
                         */
                        for (c = 0; c < parms->shades; ++c) {
                                if ((absz = zr*zr + zi*zi) >= 4.0f)
                                        break;

                                tr = parms->s_re + zr * zr - zi * zi;
                                ti = parms->s_im + zr * zi * 2.0f;

                                zr = tr;
                                zi = ti;
                        }
```

```
                                if (fprintf(parms->fp, "%u\n", (unsigned)c) < 0) {
                                        fprintf(stderr, "i/o error.\n");
                                        exit(1);
                                }
                        }
                }

                /* Flush any buffered information before quit. */
                if (fflush(parms->fp) != 0) {
                        fprintf(stderr, "cannot flush output file.\n");
                        exit(1);
                }

                fclose(parms->fp);
}
```

mygetopt_long.c

```
/*      $NetBSD: getopt_long.c,v 1.17 2004/06/20 22:20:15 jmc Exp $     */

/*-
 * Copyright (c) 2000 The NetBSD Foundation, Inc.
 * All rights reserved.
 *
 * This code is derived from software contributed to The NetBSD Foundation
 * by Dieter Baron and Thomas Klausner.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *        This product includes software developed by the NetBSD
 *        Foundation, Inc. and its contributors.
 * 4. Neither the name of The NetBSD Foundation nor the names of its
 *    contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

#include <defs.h>
#include <assert.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#ifdef MISSING_GETOPT
#include <mygetopt.h>
```

```
int     opterr = 1;                     /* if error message should be printed */
int     optind = 1;                     /* index into parent argv vector */
int     optopt = '?';                   /* character checked for validity */
int     optreset;               /* reset getopt */
char    *optarg;                /* argument associated with option */

#if !HAVE_GETOPT_LONG
#define IGNORE_FIRST    (*options == '-' || *options == '+')
#define PRINT_ERROR     ((opterr) && ((*options != ':') \
                                      || (IGNORE_FIRST && options[1] != ':')))
#define IS_POSIXLY_CORRECT (getenv("POSIXLY_CORRECT") != NULL)
#define PERMUTE         (!IS_POSIXLY_CORRECT && !IGNORE_FIRST)
/* XXX: GNU ignores PC if *options == '-' */
#define IN_ORDER        (!IS_POSIXLY_CORRECT && *options == '-')

/* return values */
#define BADCH  (int)'?'
#define BADARG                  ((IGNORE_FIRST && options[1] == ':') \
                                 || (*options == ':') ? (int)':' : (int)'?')
#define INORDER (int)1

#define EMSG    ""

#ifndef _DIAGASSERT
#define _DIAGASSERT(x) do {} while (0)
#endif

static int getopt_internal (int, char * const *, const char *);
static int gcd (int, int);
static void permute_args (int, int, int, char * const *);

static char *place = EMSG; /* option letter processing */

/* XXX: set optreset to 1 rather than these two */
static int nonopt_start = -1; /* first non option argument (for permute) */
static int nonopt_end = -1;   /* first option after non options (for permute) */

/* Error messages */
static const char recargchar[] = "option requires an argument -- %c";
static const char recargstring[] = "option requires an argument -- %s";
static const char ambig[] = "ambiguous option -- %.*s";
static const char noarg[] = "option doesn't take an argument -- %.*s";
static const char illoptchar[] = "unknown option -- %c";
static const char illoptstring[] = "unknown option -- %s";


/*
 * Compute the greatest common divisor of a and b.
 */
static int
gcd(a, b)
        int a;
        int b;
{
        int c;

        c = a % b;
        while (c != 0) {
                a = b;
                b = c;
                c = a % b;
        }

        return b;
}
```

```
/*
 * Exchange the block from nonopt_start to nonopt_end with the block
 * from nonopt_end to opt_end (keeping the same order of arguments
 * in each block).
 */
static void
permute_args(panonopt_start, panonopt_end, opt_end, nargv)
        int panonopt_start;
        int panonopt_end;
        int opt_end;
        char * const *nargv;
{
        int cstart, cyclelen, i, j, ncycle, nnonopts, nopts, pos;
        char *swap;

        /*
         * compute lengths of blocks and number and size of cycles
         */
        nnonopts = panonopt_end - panonopt_start;
        nopts = opt_end - panonopt_end;
        ncycle = gcd(nnonopts, nopts);
        cyclelen = (opt_end - panonopt_start) / ncycle;

        for (i = 0; i < ncycle; i++) {
                cstart = panonopt_end+i;
                pos = cstart;
                for (j = 0; j < cyclelen; j++) {
                        if (pos >= panonopt_end)
                                pos -= nnonopts;
                        else
                                pos += nopts;
                        swap = nargv[pos];
                        /* LINTED const cast */
                        ((char **) nargv)[pos] = nargv[cstart];
                        /* LINTED const cast */
                        ((char **)nargv)[cstart] = swap;
                }
        }
}

/*
 * getopt_internal --
 *      Parse argc/argv argument vector.  Called by user level routines.
 *  Returns -2 if -- is found (can be long option or end of options marker).
 */
static int
getopt_internal(nargc, nargv, options)
        int nargc;
        char * const *nargv;
        const char *options;
{
        char *oli;                              /* option letter list index */
        int optchar;

        _DIAGASSERT(nargv != NULL);
        _DIAGASSERT(options != NULL);

        optarg = NULL;

        /*
         * XXX Some programs (like rsyncd) expect to be able to
         * XXX re-initialize optind to 0 and have getopt_long(3)
         * XXX properly function again.  Work around this braindamage.
         */
        if (optind == 0)
                optind = 1;
```

```
        if (optreset)
                nonopt_start = nonopt_end = -1;
start:
        if (optreset || !*place) {              /* update scanning pointer */
                optreset = 0;
                if (optind >= nargc) {          /* end of argument vector */
                        place = EMSG;
                        if (nonopt_end != -1) {
                                /* do permutation, if we have to */
                                permute_args(nonopt_start, nonopt_end,
                                    optind, nargv);
                                optind -= nonopt_end - nonopt_start;
                        }
                        else if (nonopt_start != -1) {
                                /*
                                 * If we skipped non-options, set optind
                                 * to the first of them.
                                 */
                                optind = nonopt_start;
                        }
                        nonopt_start = nonopt_end = -1;
                        return -1;
                }
                if ((*(place = nargv[optind]) != '-')
                    || (place[1] == '\0')) {    /* found non-option */
                        place = EMSG;
                        if (IN_ORDER) {
                                /*
                                 * GNU extension:
                                 * return non-option as argument to option 1
                                 */
                                optarg = nargv[optind++];
                                return INORDER;
                        }
                        if (!PERMUTE) {
                                /*
                                 * if no permutation wanted, stop parsing
                                 * at first non-option
                                 */
                                return -1;
                        }
                        /* do permutation */
                        if (nonopt_start == -1)
                                nonopt_start = optind;
                        else if (nonopt_end != -1) {
                                permute_args(nonopt_start, nonopt_end,
                                    optind, nargv);
                                nonopt_start = optind -
                                    (nonopt_end - nonopt_start);
                                nonopt_end = -1;
                        }
                        optind++;
                        /* process next argument */
                        goto start;
                }
                if (nonopt_start != -1 && nonopt_end == -1)
                        nonopt_end = optind;
                if (place[1] && *++place == '-') {      /* found "--" */
                        place++;
                        return -2;
                }
        }
        if ((optchar = (int)*place++) == (int)':' ||
            (oli = strchr(options + (IGNORE_FIRST ? 1 : 0), optchar)) == NULL) {
                /* option letter unknown or ':' */
```

```
                if (!*place)
                        ++optind;
#if 0
                if (PRINT_ERROR)
                        warnx(illoptchar, optchar);
#endif
                optopt = optchar;
                return BADCH;
        }
        if (optchar == 'W' && oli[1] == ';') {          /* -W long-option */
                /* XXX: what if no long options provided (called by getopt)? */
                if (*place)
                        return -2;

                if (++optind >= nargc) {        /* no arg */
                        place = EMSG;
#if 0
                        if (PRINT_ERROR)
                                warnx(recargchar, optchar);
#endif
                        optopt = optchar;
                        return BADARG;
                } else                                  /* white space */
                        place = nargv[optind];
                /*
                 * Handle -W arg the same as --arg (which causes getopt to
                 * stop parsing).
                 */
                return -2;
        }
        if (*++oli != ':') {                    /* doesn't take argument */
                if (!*place)
                        ++optind;
        } else {                                        /* takes (optional) argument */
                optarg = NULL;
                if (*place)                             /* no white space */
                        optarg = place;
                /* XXX: disable test for :: if PC? (GNU doesn't) */
                else if (oli[1] != ':') {               /* arg not optional */
                        if (++optind >= nargc) {        /* no arg */
                                place = EMSG;
#if 0
                                if (PRINT_ERROR)
                                        warnx(recargchar, optchar);
#endif
                                optopt = optchar;
                                return BADARG;
                        } else
                                optarg = nargv[optind];
                }
                place = EMSG;
                ++optind;
        }
        /* dump back option letter */
        return optchar;
}

#ifdef REPLACE_GETOPT
/*
 * getopt --
 *      Parse argc/argv argument vector.
 *
 * [eventually this will replace the real getopt]
 */
int
getopt(nargc, nargv, options)
```

```
	int nargc;
	char * const *nargv;
	const char *options;
{
	int retval;

	_DIAGASSERT(nargv != NULL);
	_DIAGASSERT(options != NULL);

	if ((retval = getopt_internal(nargc, nargv, options)) == -2) {
		++optind;
		/*
		 * We found an option (--), so if we skipped non-options,
		 * we have to permute.
		 */
		if (nonopt_end != -1) {
			permute_args(nonopt_start, nonopt_end, optind,
				     nargv);
			optind -= nonopt_end - nonopt_start;
		}
		nonopt_start = nonopt_end = -1;
		retval = -1;
	}
	return retval;
}
#endif

/*
 * getopt_long --
 *	Parse argc/argv argument vector.
 */
int
getopt_long(nargc, nargv, options, long_options, idx)
	int nargc;
	char * const *nargv;
	const char *options;
	const struct option *long_options;
	int *idx;
{
	int retval;

	_DIAGASSERT(nargv != NULL);
	_DIAGASSERT(options != NULL);
	_DIAGASSERT(long_options != NULL);
	/* idx may be NULL */

	if ((retval = getopt_internal(nargc, nargv, options)) == -2) {
		char *current_argv, *has_equal;
		size_t current_argv_len;
		int i, match;

		current_argv = place;
		match = -1;

		optind++;
		place = EMSG;

		if (*current_argv == '\0') {		/* found "--" */
			/*
			 * We found an option (--), so if we skipped
			 * non-options, we have to permute.
			 */
			if (nonopt_end != -1) {
				permute_args(nonopt_start, nonopt_end,
				    optind, nargv);
				optind -= nonopt_end - nonopt_start;
```

```
                }
                nonopt_start = nonopt_end = -1;
                return -1;
        }
        if ((has_equal = strchr(current_argv, '=')) != NULL) {
                /* argument found (--option=arg) */
                current_argv_len = has_equal - current_argv;
                has_equal++;
        } else
                current_argv_len = strlen(current_argv);

        for (i = 0; long_options[i].name; i++) {
                /* find matching long option */
                if (strncmp(current_argv, long_options[i].name,
                    current_argv_len))
                        continue;

                if (strlen(long_options[i].name) ==
                    (unsigned)current_argv_len) {
                        /* exact match */
                        match = i;
                        break;
                }
                if (match == -1)                /* partial match */
                        match = i;
                else {
                        /* ambiguous abbreviation */
#if 0
                        if (PRINT_ERROR)
                                warnx(ambig, (int)current_argv_len,
                                    current_argv);
#endif
                        optopt = 0;
                        return BADCH;
                }
        }
        if (match != -1) {                      /* option found */
                if (long_options[match].has_arg == no_argument
                    && has_equal) {
#if 0
                        if (PRINT_ERROR)
                                warnx(noarg, (int)current_argv_len,
                                    current_argv);
#endif
                        /*
                         * XXX: GNU sets optopt to val regardless of
                         * flag
                         */
                        if (long_options[match].flag == NULL)
                                optopt = long_options[match].val;
                        else
                                optopt = 0;
                        return BADARG;
                }
                if (long_options[match].has_arg == required_argument ||
                    long_options[match].has_arg == optional_argument) {
                        if (has_equal)
                                optarg = has_equal;
                        else if (long_options[match].has_arg ==
                            required_argument) {
                                /*
                                 * optional argument doesn't use
                                 * next nargv
                                 */
                                optarg = nargv[optind++];
                        }
```

```
                        }
                        if ((long_options[match].has_arg == required_argument)
                            && (optarg == NULL)) {
                                /*
                                 * Missing argument; leading ':'
                                 * indicates no error should be generated
                                 */
#if 0
                                if (PRINT_ERROR)
                                        warnx(recargstring, current_argv);
#endif
                                /*
                                 * XXX: GNU sets optopt to val regardless
                                 * of flag
                                 */
                                if (long_options[match].flag == NULL)
                                        optopt = long_options[match].val;
                                else
                                        optopt = 0;
                                --optind;
                                return BADARG;
                        }
                } else {                        /* unknown option */
#if 0
                        if (PRINT_ERROR)
                                warnx(illoptstring, current_argv);
#endif
                        optopt = 0;
                        return BADCH;
                }
                if (long_options[match].flag) {
                        *long_options[match].flag = long_options[match].val;
                        retval = 0;
                } else
                        retval = long_options[match].val;
                if (idx)
                        *idx = match;
        }
        return retval;
}
#endif /* !GETOPT_LONG */
#endif /* MISSING_GETOPT */
```

3.2. Código MIPS32

fileFunctions.S

```
#include <mips/regdef.h>
#include <sys/syscall.h>

#include "constants.h"

# Size mensajes
#define BYTES_MENSAJE_ERROR_WRITE   52
#define BYTES_MENSAJE_ERROR_CLOSE_FILE   68
#define BYTES_MENSAJE_ERROR_LOAD_FILE           51

#define FILE_DESCRIPTOR_STDERR                  2

        .globl  quantityCharactersInBuffer
        .globl  quantityCharactersInBuffer
        .section .bss
        .align  2
        .type   quantityCharactersInBuffer, @object
```

```
        .size   quantityCharactersInBuffer, 4
quantityCharactersInBuffer:
        .space  4

#---------------  loadFileDescriptor  ---------------#

        .text
        .align  2
        .globl  loadFileDescriptor
        .ent    loadFileDescriptor
loadFileDescriptor:
        .frame  $fp,48,ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu    sp,sp,48
        .cprestore 16
        sw      ra,40(sp)
        sw      $fp,36(sp)
        sw      gp,32(sp)
        move    $fp,sp
        lw      v0,fileOutput
        bne     v0,zero,$fileno

        # Hubo error al querer escribir en el archivo => Mensaje de error.
    li  a0,FILE_DESCRIPTOR_STDERR # Cargo en a0 FILE_DESCRIPTOR_STDERR.
    la   a1,MENSAJE_ERROR_LOAD_FILE
    li  a2,BYTES_MENSAJE_ERROR_LOAD_FILE # Cargo en a2 la cantidad de bytes a escribir.
    li  v0, SYS_write
    syscall # No controlo error porque sale de por si de la funcion por error.

        # return ERROR_FILE;
    li  v0,ERROR_FILE  # Cargo codigo de error, que sera el resultado de la funcion.
    sw    v0,24($fp)     # Guardo en la direccion 40($fp) el resultado de la funcion.

        b       $returnLoadFile
$fileno:
        lw      v0,fileOutput
        lh      v0,14(v0)
        sw      v0,ofd
        sw      zero,24($fp)
$returnLoadFile:
        lw      v0,24($fp)
        move    sp,$fp
        lw      ra,40(sp)
        lw      $fp,36(sp)
        addu    sp,sp,48
        j       ra
        .end    loadFileDescriptor
        .size   loadFileDescriptor, .-loadFileDescriptor


# ----------------- closeFile ---------
        .text
        .align  2
        .globl  closeFile
        .ent    closeFile
closeFile:
        .frame  $fp,48,ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu    sp,sp,48
        .cprestore 16
        sw      ra,40(sp)
        sw      $fp,36(sp)
```

```
        sw      gp,32(sp)
        move    $fp,sp
        lw      v1,ofd
        li      v0,1                    # 0x1
        bne     v1,v0,$IfFileOutputNull
        sw      zero,fileOutput
        b       $return
$IfFileOutputNull:
        lw      v0,fileOutput
        beq     v0,zero,$return

        # fileOutput is not NULL
        lw      a0,ofd      # a0 = file descriptor
        li      v0, SYS_close
    syscall

        beq     a3,zero,$setOutputNull # Si no hubo error, salto a $setOutputNull.

    # Hubo error al querer escribir en el archivo => Mensaje de error.
    li      a0,FILE_DESCRIPTOR_STDERR # Cargo en a0 FILE_DESCRIPTOR_STDERR.
    la      a1,MENSAJE_ERROR_CLOSE_FILE
    li      a2,BYTES_MENSAJE_ERROR_CLOSE_FILE # Cargo en a2 la cantidad de bytes a escribir.
    li      v0, SYS_write
    syscall # No controlo error porque sale de por si de la funcion por error.
$setOutputNull:
        sw      zero,fileOutput
$return:
        move    sp,$fp
        lw      ra,40(sp)
        lw      $fp,36(sp)
        addu    sp,sp,48
        j       ra
        .end    closeFile
        .size   closeFile, .-closeFile


#---------------   ---------------#
        .globl  writeBufferInOFile
        .ent    writeBufferInOFile
#---------------   ---------------#
writeBufferInOFile:
        .frame  $fp,64,ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu    sp,sp,64
        .cprestore 16
        sw      ra,56(sp)
        sw      $fp,52(sp)
        sw      gp,48(sp)
        move    $fp,sp
        sw      a0,64($fp)
        sw      a1,68($fp)
        lw      v0,fileOutput
        beq     v0,zero,$return_no_data_to_save
        lw      v0,68($fp)
        blez    v0,$return_no_data_to_save
        b       $while_init
$return_no_data_to_save:
        sw      zero,40($fp)
        b       $return_write_buffer_in_OFile
$while_init:
        li      v0,1                    # 0x1
        sw      v0,24($fp)
        sw      zero,28($fp)
        lw      v0,68($fp)
```

```
        sw      v0,32($fp)
$check_delivery:
        lw      v1,24($fp)
        li      v0,1                            # 0x1
        beq     v1,v0,$while
        b       $return_okey
$while:
        lw      v1,64($fp)
        lw      v0,28($fp)
        addu    v0,v1,v0

        lw      a0,ofd          # a0 = output file descriptor
        move    a1,v0       # a1 = v0 = bufferToLoad + bytesWriteAcum
        lw      a2,32($fp)      # a2 = bytesToWrite
        li      v0, SYS_write
    syscall  # Seria int bytesWrite = write(ofd, bufferToLoad + bytesWriteAcum, bytesToWrite);

        beq     a3,zero,$save_bytes_write # Si no hubo error, salto a $save_bytes_write.

        # Hubo error al querer escribir en el archivo => Mensaje de error.
    li      a0,FILE_DESCRIPTOR_STDERR # Cargo en a0 FILE_DESCRIPTOR_STDERR.
    la      a1,MENSAJE_ERROR_WRITE
    li      a2,BYTES_MENSAJE_ERROR_WRITE # Cargo en a2 la cantidad de bytes a escribir.
    li      v0, SYS_write
    syscall # No controlo error porque sale de por si de la funcion por error.

    # return ERROR_WRITE;
    li      v0,ERROR_WRITE  # Cargo codigo de error, que sera el resultado de la funcion.
    sw      v0,40($fp)      # Guardo en la direccion 40($fp) el resultado de la funcion.
    b       $return_write_buffer_in_OFile

$save_bytes_write:
        sw      v0,36($fp)
        lw      v1,28($fp)
        lw      v0,36($fp)
        addu    v0,v1,v0
        sw      v0,28($fp)
        lw      v1,68($fp)
        lw      v0,28($fp)
        subu    v0,v1,v0
        sw      v0,32($fp)
        lw      v0,32($fp)
        bgtz    v0,$check_delivery
        sw      zero,24($fp)
        b       $check_delivery
$return_okey:
        sw      zero,40($fp)
$return_write_buffer_in_OFile:
        lw      v0,40($fp)
        move    sp,$fp
        lw      ra,56(sp)
        lw      $fp,52(sp)
        addu    sp,sp,64
        j       ra
        .end    writeBufferInOFile
        .size   writeBufferInOFile, .-writeBufferInOFile
        .align  2

        .text
        .align  2
        .globl  putch
        .ent    putch
#---------------   ---------------#
putch:
        .frame  $fp,56,ra
        .set    noreorder
```

```
        .cpload t9
        .set    reorder
        subu    sp,sp,56
        .cprestore 16
        sw      ra,48(sp)
        sw      $fp,44(sp)
        sw      gp,40(sp)
        move    $fp,sp
        move    v0,a0
        sb      v0,24($fp)
        lw      v0,quantityCharactersInBuffer
        slt     v0,v0,100
        beq     v0,zero,$writeBuffer
        lb      v0,24($fp)
        move    a0,v0
        la      t9,loadDataInBuffer
        jal     ra,t9
        sw      zero,32($fp)
        b       $returnPutch
$writeBuffer:
        la      a0,buffer
        lw      a1,quantityCharactersInBuffer
        la      t9,writeBufferInOFile
        jal     ra,t9
        sw      v0,28($fp)
        lw      v0,28($fp)
        beq     v0,zero,$exeLoadData
        lw      v0,28($fp)
        sw      v0,32($fp)
        b       $returnPutch
$exeLoadData:
        sw      zero,quantityCharactersInBuffer
        lb      v0,24($fp)
        move    a0,v0
        la      t9,loadDataInBuffer
        jal     ra,t9
        sw      zero,32($fp)
$returnPutch:
        lw      v0,32($fp)
        move    sp,$fp
        lw      ra,48(sp)
        lw      $fp,44(sp)
        addu    sp,sp,56
        j       ra
        .end    putch
        .size   putch, .-putch

#---------------   ---------------#

        .text
        .align  2

        .globl  flush
        .ent    flush
flush:
        .frame  $fp,48,ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu    sp,sp,48
        .cprestore 16
        sw      ra,40(sp)
        sw      $fp,36(sp)
        sw      gp,32(sp)
        move    $fp,sp
        lw      v0,quantityCharactersInBuffer
```

```
        blez    v0,$returnOK
        la      a0,buffer
        lw      a1,quantityCharactersInBuffer
        la      t9,writeBufferInOFile
        jal     ra,t9
        sw      v0,24($fp)
        b       $returnFlush
$returnOK:
        sw      zero,24($fp)
$returnFlush:
        lw      v0,24($fp)
        move    sp,$fp
        lw      ra,40(sp)
        lw      $fp,36(sp)
        addu    sp,sp,48
        j       ra
        .end    flush
        .size   flush, .-flush
        .align  2

#--------------------

        .comm   fileOutput,4

        .comm   ofd,4

        .comm   buffer,100


## Mensajes de error

        .rdata

        .align  2

MENSAJE_ERROR_WRITE:
        .ascii  "[Error] Hubo un error al escribir en el archivo. \n\000"

MENSAJE_ERROR_CLOSE_FILE:
        .ascii  "[Warning] El archivo de output no pudo ser cerrado corre"
        .ascii  "ctamente.\n\000"

MENSAJE_ERROR_LOAD_FILE:
        .ascii  "[Error] No se ha especificado archivo de salida.\n\000"
```

stack frames

int loadFileDescriptor()		
posicion	**contenido**	**grupo**
48	ra	**SRA**
44	fp	
40	ra	
36	fp	
32	gp	
28	-	
24	return writeBufferInOFile | OKEY=0	
20	-	
16	-	**ABA**
12	-	
8	-	
4	-	
0	-	

loadFileDescriptor stack frame

void closeFile()		
posicion	**contenido**	**grupo**
48	-	**LTA**
44	-	
40	ra	**SRA**
36	fp	
32	gp	
28	padding	
24	result	
20	padding	
16	-	**ABA**
12	-	
8	-	
4	-	
0	-	

closeFile stack frame

int writeBufferInOFile(char * bufferToLoad, int quantityCharactersInBufferToLoad)		
posicion	**contenido**	**grupo**
68	quantityCharactersInBufferToLoad	**PARAMS**
64	* bufferToLoad	
60	-	
56	ra	**SRA**
52	fp	
48	gp	
44	-	**LTA**
40	bytesWrite	
36	bytesWrite (return syscall)	
32	bytesWrite nuevo | fileOutput	
28	bytesWriteAcum	
24	completeDelivery	
20	-	
16	-	**ABA**
12	-	
8	-	
4	-	
0	-	

writeBufferInOFile stack frame

int putch(char character)		
posicion	**contenido**	**grupo**
48	ra	**SRA**
44	fp	
40	sp	
36	-	
32	response loadDataInBuffer	
28	-	
24	character	
20	-	
16	-	**ABA**
12	-	
8	-	
4	-	
0	-	

putch stack frame

int flush()		
posicion	**contenido**	**grupo**
48	ra	**SRA**
44	fp	
40	ra	
36	fp	
32	gp	
28	-	
24	return writeBufferInOFile | OKEY=0	
20	-	
16	-	**ABA**
12	-	
8	-	
4	-	
0	-	

flush stack frame

plotFunctions.S

```
#include <mips/regdef.h>
#include <sys/syscall.h>

#include "constants.h"

##-----      mips32_plot -----##

            .text
            .align      2
            .globl      mips32_plot
            .ent        mips32_plot
mips32_plot:
            .frame      $fp,88,ra
            .set        noreorder
            .cpload     t9
            .set        reorder

            # Stack frame creation
            subu        sp,sp,88

            .cprestore 16
            sw          ra,80(sp)
            sw          $fp,76(sp)
            sw          gp,72(sp)
```

```
        # de aqui al fin de la funcion uso $fp en lugar de sp.
        move    $fp,sp

        # Parametro
        sw      a0,88($fp)    # Guardo en la dir 88($fp), la direccion de parms (param_t * parms).

        ########################################################################
        #  Estructura de param_t:                                              #
        #                                                                      #
        #  typedef struct {                                                    #
        #          float UL_re; /* upper left point, real part */              #
        #          float UL_im; /* upper left point, imaginary part */         #
        #          float LR_re; /* lower right point, real part */             #
        #          float LR_im; /* lower right point, imaginary part */        #
        #          float d_re;  /* pixel step, real part */                    #
        #          float d_im;  /* pixel step, imaginary part */               #
        #          float s_re;  /* seed of julia set, real part */             #
        #          float s_im;  /* seed of julia set, imaginary part */        #
        #                                                                      #
        #          size_t x_res;  /* horizontal resolution, e.g. 640 */       #
        #          size_t y_res;  /* vertical resolution, e.g. 480 */         #
        #          size_t shades; /* amount of shades of gray, e.g. 255 */    #
        #                                                                      #
        #          FILE *fp;                                                   #
        #  } param_t;                                                          #
        #                                                                      #
        ########################################################################

        # fileOutput = parms->fp;
        lw      v0,88($fp)    # Cargo en v0, la dir a la que apunta parms.
        lw      v0,44(v0)     # Cargo en v0, la dir a la que apunta parms + 44 (que es FILE *).
        sw      v0,fileOutput # Guardo en la variable global fileOuput, el FILE * guardado en v0.

        # int rdo = loadFileDescriptor();
        la      t9,loadFileDescriptor # Cargo en t9, la direccion de la funcion loadFileDescriptor.
        jal     ra,t9         # Salto a loadFileDescriptor
        sw      v0,24($fp)    # Guardo en la dir 24($fp) el resultado de al funcion, que esta en v0.

        # (rdo != OKEY) ?
        lw      v0,24($fp)    # Cargo en v0 lo que hay en la dir 24($fp), que es rdo.
        beq     v0,OKEY,$writeHeader # If (rdo == OKEY) goto $writeHeader

        # rdo is not OKEY
        b       $returnMips32Plot # Salto incondicional a $returnMips32Plot.
$writeHeader:
        # rdo = writeHeader((unsigned)parms->y_res, (unsigned)parms->x_res, (unsigned)(parms->shades - 1));
        lw      v1,88($fp)    # Cargo en v1, la dir a la que apunta parms.
        lw      a1,88($fp)    # Cargo en a1, la dir a la que apunta parms.
        lw      v0,88($fp)    # Cargo en v0, la dir a la que apunta parms.

        # (parms->shades - 1)
        lw      v0,40(v0)     # Cargo en v0, el contenido apuntado por *parms + 40, que seria shades (parms->shades).
        addu    v0,v0,-1      # A shades le resto 1. Guardo en v0 el resultado de parms->shades - 1.

        # (parms->y_res)
        lw      a0,36(v1)     # Cargo en a0, lo apuntado por *parms + 36, que seria y_res (parms->y_res).

        # (parms->x_res)
        lw      a1,32(a1)     # Cargo en a1, lo apuntado por *parms + 32, que seria y_res (parms->x_res).

        move    a2,v0         # Muevo a a2, el contenido de v0, que es el resultado de parms->shades - 1.

        la      t9,writeHeader # Cargo en t9, la direccion de memoria de la funcion writeHeader.
        jal     ra,t9         # Salto a writeHeader, ejecuto la funcion.
        sw      v0,24($fp)    # Guardo en rdo, dir 24($fp), el resultado de la funcion writeHeader.
```

```
        # (rdo != OKEY) ?
        lw      v0,24($fp)    # Cargo en v0, rdo, guardado en la dir 24($fp).
        beq     v0,OKEY,$loopPixel # If (rdo == OKEY) goto $loopPixel.

        # rdo is not equal to OKEY
        b       $returnMips32Plot # Salto incondicional a $returnMips32Plot.

$loopPixel:

        # for (y = 0, ci = parms->UL_im; y < parms->y_res; ++y, ci -= parms->d_im)

        # y = 0
        sw      zero,60($fp)  # Guardo en la dir 60($fp) el valor 0, que seria y = 0.

        # ci = parms->UL_im
        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        l.s     $f0,4(v0)     # Cargo en f0 lo apuntado por *parms + 4, que es UL_im (es float).
        s.s     $f0,32($fp)   # Guardo en la dir 32($fp) el valor guardado en f0, que es UL_im (float), que seria ci.
$loopInY:

        # (y < parms->y_res) ?
        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        lw      v1,60($fp)    # Cargo en v1 y, guardado en la dir 60($fp).
        lw      v0,36(v0)     # Cargo en v0, lo apuntado por *parms + 36, que seria y_res (parms->y_res).
        sltu    v0,v1,v0      # Guardo en v0 TRUE si y < parms->y_res, sino guardo FALSE.
        bne     v0,FALSE,$loopInX  # Si no es FALSE, es TRUE, continuo el for, salto a $loopInX.

        # y is not less than parms->y_res
        b       $flushRestData # Salto incondicional a $flushRestData.
$loopInX:

        # for (x = 0, cr = parms->UL_re; x < parms->x_res; ++x, cr += parms->d_re)

        # x = 0
        sw      zero,56($fp)  # Guardo en la dir 56($fp) el valor 0, que seria x = 0.

        # cr = parms->UL_re
        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        l.s     $f0,0(v0)     # Cargo en f0 lo apuntado por *parms, que es UL_re (es float).
        s.s     $f0,28($fp)   # Guardo en la dir 28($fp) el valor guardado en f0, que es UL_re (float), que seria cr.
$conditionLoopInX:

        # (x < parms->x_res) ?
        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        lw      v1,56($fp)    # Cargo en v1 lo guardado en la dir 56($fp), que seria x.
        lw      v0,32(v0)     # Cargo en v0 lo guardado en la dir *parms+32, que es x_res.
        sltu    v0,v1,v0      # Guardo en v0 TRUE si x < parms->x_res, sino guardo FALSE.
        bne     v0,FALSE,$insideOfTheLoopInX # Si no es FALSE, es TRUE, continuo dentro del for, salto a
$insideOfTheLoopInX.

        # x is not less than parms->x_res
        b       $putLineBreak # Salto incondicional a $putLineBreak.
$insideOfTheLoopInX:

        # zr = cr;
        l.s     $f0,28($fp)   # Cargo en f0 lo guardado en la dir 28($fp), que es cr (es float).
        s.s     $f0,36($fp)   # Guardo en la dir 36($fp), que representa a zr, lo que tiene f0.

        # zi = ci;
        l.s     $f0,32($fp)   # Cargo en f0 lo guardado en la dir 32($fp), que es ci (es float).
        s.s     $f0,40($fp)   # Guardo en la dir 40($fp), que representa a zi, lo que tiene f0.

        # for (c = 0; c < parms->shades; ++c)

        # c = 0
```

```
        sw      zero,64($fp)   # Guardo en la dir 64($fp) el valor 0, que representa a la variable c.
$loopShades:

        # (c < parms->shades) ?
        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        lw      v1,64($fp)    # Cargo en v1 lo guardado en la dir 64($fp), que seria c.
        lw      v0,40(v0)     # Cargo en v0 lo guardado en la dir *parms+40, que es shades.
        sltu    v0,v1,v0      # Guardo en v0 TRUE si c < parms->shades, sino guardo FALSE.
        bne     v0,FALSE,$insideOfTheLoopShades # Si no es FALSE, es TRUE, continuo dentro
                        # del for, salto a $insideOfTheLoopShades.

        # c is not less than parms->shades
        b       $savePixelBrightness # Salto incondicional a $savePixelBrightness.
$insideOfTheLoopShades:
        # ((absz = zr*zr + zi*zi) >= 4.0f) ?

        # (absz = zr*zr + zi*zi)

        # zr*zr
        l.s     $f2,36($fp)    # Cargo en f2 lo guardado en la dir 36($fp), que es zr (es float).
        l.s     $f0,36($fp)    # Cargo en f0 lo guardado en la dir 36($fp), que es zr (es float).
        mul.s   $f4,$f2,$f0    # Multiplico lo guardado en f0 con lo guardado en f2, y guardo rdo en f4.

        # zi*zi
        l.s     $f2,40($fp)    # Cargo en f2 lo guardado en la dir 40($fp), que es zi (es float).
        l.s     $f0,40($fp)    # Cargo en f0 lo guardado en la dir 40($fp), que es zi (es float).
        mul.s   $f0,$f2,$f0    # Multiplico lo guardado en f0 con lo guardado en f2, y guardo rdo en f0.

        # absz = zr*zr + zi*zi
        add.s   $f0,$f4,$f0    # Sumo lo guardado en f0 (zi*zi), con lo guardado en f4
                # (zr*zr), y guardo rdo en f0.
        mov.s   $f2,$f0        # Muevo el rdo de f0 a f2.
        s.s     $f2,52($fp)    # Guardo el resultado de la suma en la dir 52($fp), que representa a
                # la variable absz.
        l.s     $f0,fourFloat  # Cargo en f0 el valor 4.0f
        c.le.s  $f0,$f2        # Comparo absz con 4.0f. Si f0 (4.0f) <= f2 (absz), coloca el
                # indicador de condicion en TRUE, sino en FALSE.
        bc1t    $savePixelBrightness

        # (absz = zr*zr + zi*zi) is >= 4.0f

        # tr = parms->s_re + zr * zr - zi * zi;

        # zr * zr
        l.s     $f2,36($fp)    # Cargo en f2 lo guardado en la dir 36($fp), que es zr (es float).
        l.s     $f0,36($fp)    # Cargo en f0 lo guardado en la dir 36($fp), que es zr (es float).
        mul.s   $f2,$f2,$f0    # Multiplico lo guardado en f0 con lo guardado en f2, y guardo rdo en f2.

        # parms->s_re
        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        l.s     $f0,24(v0)    # Cargo en f0 lo guardado en la dir *parms+24, que es s_re (es float).

        # parms->s_re + zr * zr
        add.s   $f4,$f2,$f0   # Guardo resultado de la suma en f4.

        # zi * zi
        l.s     $f2,40($fp)    # Cargo en f2 lo guardado en la dir 40($fp), que es zi (es float).
        l.s     $f0,40($fp)    # Cargo en f0 lo guardado en la dir 40($fp), que es zi (es float).
        mul.s   $f0,$f2,$f0    # Multiplico lo guardado en f0 con lo guardado en f2, y guardo rdo en f0.

        # tr = parms->s_re + zr * zr - zi * zi;
        sub.s   $f0,$f4,$f0
        s.s     $f0,44($fp)    # Guardo el resultado de la resta (f0), en la dir 44($fp), representa a tr.

        # ti = parms->s_im + zr * zi * 2.0f;
```

```
        # zr * zi
        l.s     $f2,36($fp)    # Cargo en f2 lo guardado en la dir 36($fp), que es zr (es float).
        l.s     $f0,40($fp)    # Cargo en f0 lo guardado en la dir 40($fp), que es zi (es float).
        mul.s   $f0,$f2,$f0    # Multiplico lo guardado en f0 con lo guardado en f2, y guardo rdo en f0.

        # (zr * zi * 2.0f)  lo que es igual a hacer (zr * zi + zr * zi)
        add.s   $f2,$f0,$f0

        lw      v0,88($fp)    # Cargo en v0 la dir apuntada por parms (* parms), guardada en la dir 88($fp).
        l.s     $f0,28(v0)    # Cargo en f0 lo guardado en la dir *parms+28, que es s_im (es float).

        # ti = parms->s_im + zr * zi * 2.0f;
        add.s   $f0,$f0,$f2
        s.s     $f0,48($fp)   # Guardo el resultado de la suma (f0) en 48($fp), que representa a la variable ti.

        # zr = tr;
        l.s     $f0,44($fp)
        s.s     $f0,36($fp)

        # zi = ti;
        l.s     $f0,48($fp)
        s.s     $f0,40($fp)

        # ++c
        lw      v0,64($fp)
        addu    v0,v0,1
        sw      v0,64($fp)

        b       $loopShades
$savePixelBrightness:

        # rdo = loadPixelBrightness((unsigned)c);
        lw      a0,64($fp)
        la      t9,loadPixelBrightness
        jal     ra,t9
        sw      v0,24($fp)

        # (rdo != OKEY) ?
        lw      v0,24($fp)
        beq     v0,OKEY,$putSpace

        # rdo is not equals OKEY

        # closeFile(parms->fp);
        lw      v0,88($fp)
        lw      a0,44(v0)
        la      t9,closeFile
        jal     ra,t9

        b       $returnMips32Plot
$putSpace:
        # rdo = putch(' ');
        li      a0,32                   # Cargo en a0 el espacio.
        la      t9,putch
        jal     ra,t9
        sw      v0,24($fp)

        # (rdo != OKEY) ?
        lw      v0,24($fp)
        beq     v0,OKEY,$loopInXCompletion

        # rdo is not equals OKEY

        # closeFile();
        la      t9,closeFile
        jal     ra,t9
```

```
        b       $returnMips32Plot
$loopInXCompletion:
        # ++x, cr += parms->d_re

        # ++x
        lw      v0,56($fp)
        addu    v0,v0,1
        sw      v0,56($fp)

        # cr += parms->d_re
        lw      v0,88($fp)
        l.s     $f2,28($fp)
        l.s     $f0,16(v0)
        add.s   $f0,$f2,$f0
        s.s     $f0,28($fp)


        b       $conditionLoopInX
$putLineBreak:
        # rdo = putch('\n');
        li      a0,10                   # 10 = '\n'
        la      t9,putch
        jal     ra,t9
        sw      v0,24($fp)

        # (rdo != OKEY) ?
        lw      v0,24($fp)
        beq     v0,OKEY,$closeLoopInY
        la      t9,closeFile
        jal     ra,t9
        b       $returnMips32Plot
$closeLoopInY:
        # ++y, ci -= parms->d_im

        # ++y
        lw      v0,60($fp)
        addu    v0,v0,1
        sw      v0,60($fp)

        # ci -= parms->d_im
        lw      v0,88($fp)
        l.s     $f2,32($fp)
        l.s     $f0,20(v0)
        sub.s   $f0,$f2,$f0
        s.s     $f0,32($fp)
        b       $loopInY
$flushRestData:
        # flush();
        la      t9,flush
        jal     ra,t9

        # closeFile();
        la      t9,closeFile
        jal     ra,t9
$returnMips32Plot:
        move    sp,$fp
        lw      ra,80(sp)
        lw      $fp,76(sp)

        # destruyo stack frame
        addu    sp,sp,88

        # vuelvo a funcion llamante
        j       ra
```

```
        .end    mips32_plot


##-----  loadPixelBrightness -----##

        .text
        .align  2
        .globl  loadPixelBrightness
        .ent    loadPixelBrightness
loadPixelBrightness:
        .frame  $fp,72,ra
        .set    noreorder
        .cpload t9
        .set    reorder

        # Stack frame creation
        subu    sp,sp,72

        .cprestore 16
        sw      ra,64(sp)
        sw      $fp,60(sp)
        sw      gp,56(sp)

        # de aqui al fin de la funcion uso $fp en lugar de sp.
        move    $fp,sp

        # Parametro

        # int loadPixelBrightness(unsigned int pixelBrightness)
        sw      a0,72($fp)      # Guardo en la dir 72($fp) el parametro recibido en a0,
                # que representa a la variable pixelBrightness.

        # character ch = convertIntToCharacter(pixelBrightness);
        addu    a0,$fp,24
        lw      a1,72($fp)
        la      t9,convertIntToCharacter # El resultado de la funcion se encuentra en v0. TODO
        jal     ra,t9


        # int rdo = OKEY;
        sw      zero,40($fp)

        #    int i;
        sw      zero,44($fp)

        # for (i = 0; i < ch.length; i++)
$loopPutchPixel:
        lw      v0,44($fp)      # Cargo en v0, lo que esta en la dir 44($fp), que es i.
        lw      v1,36($fp)      # Cargo en v1, lo que esta en la dir 36($fp), que es length.
        slt     v0,v0,v1        # Si (i < length), guardo TRUE en v0, sino FALSE.
        bne     v0,FALSE,$insideOfTheLoopPutchPixel # Si no es FALSE, o sea es TRUE, goto
$insideOfTheLoopPutchPixel.

        # No cumple condicion para volver a entrar al loop
        b       $failConditionOfTheLoopPutchPixel # Salto incondicional
$insideOfTheLoopPutchPixel:

        # rdo = putch(ch.data[i]);
        lw      v1,44($fp)      # Cargo en v1 lo de esta guardado en la dir 44($fp), que es i.
        addu    v0,$fp,24       # Busco la direccion en donde esta data.
        addu    v0,v0,v1        # Determino la nueva posicion sobre la variable data: Me muevo data+i.
        lb      v0,0(v0)        # Cargo en v0 el contenido de data+i.
        move    a0,v0           # Muevo el dato guardado en v0 a a0.
        la      t9,putch        # Cargo en t9 la la direccion de la funcion putch.
        jal     ra,t9           # Ejecuto la funcion.
        sw      v0,40($fp)      # Guardo el resultado de la funcion, que esta en v0, en la dir 40($fp).
```

```
	# (rdo != OKEY)
	lw	v0,40($fp)	# Cargo en v0 lo que hay en la dir 40($fp), que es la variable rdo.
	beq	v0,OKEY,$loopPutchPixelCompletion # If (rdo == OKEY) goto $loopPutchPixelCompletion

	# rdo is nor equals OKEY

	# return rdo;
	lw	v0,40($fp)
	sw	v0,48($fp)	# Guardo el nuevo valor de rdo, que esta en el registro v0, en la dir 48($fp).
	b	$returnLoadPixelBrightness
$loopPutchPixelCompletion:

	# i++
	lw	v0,44($fp)
	addu	v0,v0,1
	sw	v0,44($fp)

	b	$loopPutchPixel
$failConditionOfTheLoopPutchPixel:
	# return rdo;
	lw	v0,40($fp)	# Cargo en v0 la variable rdo, que esta en la dir 40($fp).
	sw	v0,48($fp)	# Guardo el nuevo valor de rdo, que esta en el registro v0, en la dir 48($fp).
$returnLoadPixelBrightness:
	lw	v0,48($fp)	# Cargo en v0 la variable rdo, que esta en la dir 48($fp).

	move	sp,$fp
	lw	ra,64(sp)
	lw	$fp,60(sp)

	# destruyo stack frame
	addu	sp,sp,72

	# vuelvo a funcion llamante
	j	ra

	.end	loadPixelBrightness


##-----	loadDataInBuffer -----##

	.text
	.align	2
	.globl	loadDataInBuffer
	.ent	loadDataInBuffer
loadDataInBuffer:
	.frame	$fp,24,ra
	.set	noreorder
	.cpload	t9
	.set	reorder

	# Stack frame creation
	subu	sp,sp,24

	.cprestore 0
	sw	$fp,20(sp)
	sw	gp,16(sp)

	# de aqui al fin de la funcion uso $fp en lugar de sp.
	move	$fp,sp

	# Parametro
	# void loadDataInBuffer(char character)
	move	v0,a0		# Muevo el parametro character, que se encuentra en a0, a v0.
	sb	v0,8($fp)	# Guardo en la dir 8($fp), el contenido de v0, que es character.
```

```
        # buffer[quantityCharactersInBuffer] = character;
        lw      v1,quantityCharactersInBuffer # Cargo en v1 la cantidad de caracteres que me voy a
                                              # mover sobre buffer.
        la      v0,buffer       # Cargo la direccion de inicio de buffer.
        addu    v1,v1,v0        # Guardo en v1 la nueva direccion sobre la variable buffer.
        lbu     v0,8($fp)       # Cargo en v0 el valor de character (un byte), que esta en la dir 8($fp).
        sb      v0,0(v1)        # Guardo en buffer+quantityCharactersInBuffer character

        # quantityCharactersInBuffer++;
        lw      v0,quantityCharactersInBuffer
        addu    v0,v0,1
        sw      v0,quantityCharactersInBuffer

        move    sp,$fp
        lw      $fp,20(sp)

        # destruyo stack frame
        addu    sp,sp,24

        # vuelvo a funcion llamante
        j       ra
        .end    loadDataInBuffer


##-----  writeHeader -----##

        .text
        .align  2
        .globl  writeHeader
        .ent    writeHeader
writeHeader:
        .frame  $fp,120,ra
        .set    noreorder
        .cpload t9
        .set    reorder

        # Stack frame creation
        subu    sp,sp,120

        .cprestore 16
        sw      ra,112(sp)
        sw      $fp,108(sp)
        sw      gp,104(sp)

        # de aqui al fin de la funcion uso $fp en lugar de sp.
        move    $fp,sp

        # Parametro
        # int writeHeader(unsigned int sizeY, unsigned int sizeX, unsigned int shades)
        sw      a0,120($fp)   # Guardo en la dir 120($fp), el primer parametro que viene en a0, y es
                              # sizeY
        sw      a1,124($fp)   # Guardo en la dir 124($fp), el segundo parametro que viene en a1, y es
                              # sizeX
        sw      a2,128($fp)   # Guardo en la dir 128($fp), el tercer parametro que viene en a2, y es
                              # shades

        sw      sp,92($fp)

        # character chY = convertIntToCharacter(sizeY);
        addu    a0,$fp,24
        lw      a1,120($fp)   # Envio sizeY como parametro.
        la      t9,convertIntToCharacter
        jal     ra,t9

        # character chX = convertIntToCharacter(sizeX);
        addu    v0,$fp,40
```

```
        move    a0,v0
        lw      a1,124($fp)
        la      t9,convertIntToCharacter
        jal     ra,t9

        # character chShades = convertIntToCharacter(shades);
        addu    v0,$fp,56
        move    a0,v0
        lw      a1,128($fp)
        la      t9,convertIntToCharacter
        jal     ra,t9

        # int quantityCharactersInBufferToLoad = 6 + chX.length + chY.length + chShades.length;
        lw      v1,52($fp)
        lw      v0,36($fp)
        addu    v0,v1,v0
        lw      v0,68($fp)
        addu    v0,v1,v0
        addu    v0,v0,6
        sw      v0,72($fp)    # Guardo en la dir 72($fp) el resultado de la suma, que seria la
                # variable quantityCharactersInBufferToLoad.

        # char bufferToLoad [quantityCharactersInBufferToLoad];
        lw      v0,72($fp)
        addu    v0,v0,-1
        addu    v0,v0,1
        addu    v0,v0,7
        srl     v0,v0,3
        sll     v0,v0,3
        subu    sp,sp,v0
        addu    v0,sp,16
        sw      v0,96($fp)    # Guardo en la dir 96($fp) bufferToLoad (la direccion al primer
                # elemento del array).

        # bufferToLoad[0] = 'P';
        li      v0,80         # Cargo en v0 el caracter P (80 en ascii).
        lw      v1,96($fp)    # Cargo en v1 bufferToLoad (la direccion al primer
                # elemento del array).
        sb      v0,0(v1)      # Guardo en la primer posicion del array, el caracter P (guardado
                # en v0).

        # bufferToLoad[1] = '2';
        li      v0,50         # Cargo en v0 el caracter 2 (50 en ascii).
        lw      a0,96($fp)
        sb      v0,1(a0)

        # bufferToLoad[2] = '\n';
        li      v0,10         # Cargo en v0 el caracter \n (10 en ascii).
        lw      v1,96($fp)
        sb      v0,2(v1)

        # int idx = 3;
        li      v0,3
        sw      v0,76($fp)    # Guardo en la dir 76($fp) la variable idx inicializada en 3.

        # int i; i = 0;
        sw      zero,80($fp)
$loopWriteHeaderChX:
        # for (i = 0; i < chX.length; i++)
        lw      v0,80($fp)    # Cargo en v0 la variable i, que esta en la dir 80($fp).
        lw      v1,52($fp)    # Cargo en v0 la variable length (de chX), que esta en la dir 52($fp).
        slt     v0,v0,v1
        bne     v0,FALSE,$insideOfTheLoopWriteHeaderChX # Si (i < chX.length) es no es FALSE, o sea
                                                        # es TRUE, goto $insideOfTheLoopWriteHeaderChX.
        b       $loadSpaceInWriteHeader
$insideOfTheLoopWriteHeaderChX:
```

```
        # bufferToLoad[idx] = chX.data[i];

        # bufferToLoad[idx]
        lw      v0,76($fp)
        lw      v1,96($fp)
        addu    a0,v1,v0

        # chX.data[i]
        addu    v1,$fp,40
        lw      v0,80($fp)
        addu    v0,v1,v0
        lbu     v0,0(v0)

        # Hago efectivamente bufferToLoad[idx] = chX.data[i];
        sb      v0,0(a0)

        # idx ++;
        lw      v0,76($fp)
        addu    v0,v0,1
        sw      v0,76($fp)

        # i++
        lw      v0,80($fp)
        addu    v0,v0,1
        sw      v0,80($fp)

        b       $loopWriteHeaderChX
$loadSpaceInWriteHeader:
        # bufferToLoad[idx] = ' ';

        # bufferToLoad[idx]
        lw      v0,76($fp)          # Cargo en v0 idx
        lw      a0,96($fp)
        addu    v1,a0,v0

        # ' ' (32 en ascii)
        li      v0,32

        # Hago efectivamente bufferToLoad[idx] = ' ';
        sb      v0,0(v1)

        # idx ++;
        lw      v0,76($fp)
        addu    v0,v0,1
        sw      v0,76($fp)

        # for (i = 0; i < chY.length; i++)
        # i = 0
        sw      zero,80($fp)
$loopWriteHeaderChY:
        # (i < chY.length) ?
        lw      v0,80($fp)
        lw      v1,36($fp)
        slt     v0,v0,v1
        bne     v0,FALSE,$insideOfTheLoopWriteHeaderChY # Si (i < chX.length) es no es FALSE, o sea
                                                        # es TRUE, goto $insideOfTheLoopWriteHeaderChY.

        b       $loadLineBreakInWriteHeader
$insideOfTheLoopWriteHeaderChY:
        # bufferToLoad[idx] = chY.data[i];

        # bufferToLoad[idx]
        lw      v0,76($fp)
        lw      v1,96($fp)
        addu    a0,v1,v0
```

```
        # chY.data[i]
        lw      v1,80($fp)
        addu    v0,$fp,24
        addu    v0,v0,v1
        lbu     v0,0(v0)

        # Hago efectivamente bufferToLoad[idx] = chY.data[i];
        sb      v0,0(a0)

        # idx ++;
        lw      v0,76($fp)
        addu    v0,v0,1
        sw      v0,76($fp)

        # i++
        lw      v0,80($fp)
        addu    v0,v0,1
        sw      v0,80($fp)

        b       $loopWriteHeaderChY
$loadLineBreakInWriteHeader:
        # bufferToLoad[idx] = '\n';
        lw      v0,76($fp)
        lw      a0,96($fp)
        addu    v1,a0,v0
        li      v0,10           # El salto de linea en ascii es igual a 10 ('\n').
        sb      v0,0(v1)

        # idx ++;
        lw      v0,76($fp)
        addu    v0,v0,1
        sw      v0,76($fp)

        # for (i = 0; i < chShades.length; i++)
        # i = 0
        sw      zero,80($fp)
$loopWriteHeaderChShades:
        # (i < chShades.length) ?
        lw      v0,80($fp)      # Cargo i en v0
        lw      v1,68($fp)      # Cargo chShades.length en v1
        slt     v0,v0,v1
        bne     v0,FALSE,$insideOfTheLoopWriteHeaderChShades # Si (i < chShades.length) es no es FALSE, o sea
                                                             # es TRUE, goto
$insideOfTheLoopWriteHeaderChShades.

        b       $loadLineBreakFinalInWriteHeader

$insideOfTheLoopWriteHeaderChShades:
        # bufferToLoad[idx] = chShades.data[i];

        # bufferToLoad[idx]
        lw      v0,76($fp)
        lw      v1,96($fp)
        addu    a0,v1,v0

        # chShades.data[i]
        addu    v1,$fp,56
        lw      v0,80($fp)
        addu    v0,v1,v0
        lbu     v0,0(v0)

        # Hago efectivamente bufferToLoad[idx] = chShades.data[i];
        sb      v0,0(a0)

        # idx ++;
        lw      v0,76($fp)
```

```
        addu    v0,v0,1
        sw      v0,76($fp)

        # i++
        lw      v0,80($fp)
        addu    v0,v0,1
        sw      v0,80($fp)
        b       $loopWriteHeaderChShades
$loadLineBreakFinalInWriteHeader:

        # bufferToLoad[idx] = '\n';
        lw      v0,76($fp)
        lw      a0,96($fp)
        addu    v1,a0,v0
        li      v0,10       # ascii 10 es el salto de linea
        sb      v0,0(v1)

        # int rdoWrite = writeBufferInOFile(bufferToLoad, quantityCharactersInBufferToLoad);
        lw      a0,96($fp)
        lw      a1,72($fp)
        la      t9,writeBufferInOFile
        jal     ra,t9
        sw      v0,84($fp)    # En la dir 84($fp) guardo el rdo de la funcion, que es la
                              # variable rdoWrite.

        # (rdoWrite != OKEY) ?
        lw      v0,84($fp)
        beq     v0,OKEY,$returnOkeyWriteHeader

        # rdoWrite is not equals OKEY
        # closeFile();
        la      t9,closeFile
        jal     ra,t9
        lw      sp,92($fp)
        li      v0,ERROR_WRITE
        sw      v0,88($fp)
        b       $returnWriteHeader
$returnOkeyWriteHeader:
        lw      sp,92($fp)
        sw      zero,88($fp)  # OKEY = zero
$returnWriteHeader:
        lw      v0,88($fp)
        move    sp,$fp
        lw      ra,112(sp)
        lw      $fp,108(sp)

        # destruyo stack frame
        addu    sp,sp,120

        # vuelvo a funcion llamante
        j       ra
        .end    writeHeader


# ------------------------------------------------------------------------------ #

## Variables auxiliares

        .comm   fileOutput,4

        .comm   ofd,4

        .comm   buffer,100

        .comm   quantityCharactersInBuffer,4
```

```
        .rdata

        .align  2
fourFloat:
        .word   1082130432  # 4.0f
```

stack frames

int writeHeader(unsigned int sizeY, unsigned int sizeX, unsigned int shades)		
posicion	**contenido**	**grupo**
120	character	PARAMS
116	-	
112	ra	SRA
108	fp	
104	gp	
100	-	LTA
96	bufferToLoad	
92	-	
88	-	
84	-	
80	i	
76	idx	
72	quantityCharactersInBufferToLoad	
68	-	
64	-	
60	-	
56	chshades	
52	lengthX	
48	-	
44	-	
40	chx.data	
36	lengthY	
32	-	
28	-	
24	-	
20	-	
16	-	ABA
12	-	
8	-	
4	-	
0	-	

writeHeader stack frame

void loadDataInBuffer(char character)		
posicion	**contenido**	**grupo**
24	ra	LTA
20	fp	
16	gp	
12	-	SRA
8	character	
4	-	
0	-	

loadDataInBuffer stack frame

int loadPixelBrightness(unsigned int pixelBrightness)		
posicion	**contenido**	**grupo**
72	-	**PARAMS**
68	-	
64	ra	**SRA**
60	fp	
56	gp	
52	-	**LTA**
48	rdo	
44	i=0	
40	rdo=0	
36	length	
32	-	
28	-	
24	data(return)	
20	-	
16	-	**ABA**
12	-	
8	-	
4	-	
0	-	

loadPixelBrightness stack frame

void mips32_plot(param_t * parms)		
posicion	**contenido**	**grupo**
88	*parms	**PARAMS**
84	-	**SRA**
80	ra	
76	fp	
72	gp	
68	-	**LTA**
64	c	
60	y	
56	x	
52	absz	
48	ti	
44	tr	
40	zi	
36	zr	
32	ci	
28	cr	
24	rdo	
20	-	
16	-	
12	-	
8	-	
4	-	
0	-	
-4	-	
24	-	
20	-	
16	-	
12	a3	**ABA**
8	a2	
4	a1	
0	a0	

mips32_plot stack frame

utilityFunctions.S

```
#include <mips/regdef.h>
#include <sys/syscall.h>

#include "constants.h"

##-----	convertIntToCharacter -----##

	.text
	.align	2
	.globl	convertIntToCharacter
	.ent	convertIntToCharacter
convertIntToCharacter:
	.frame	$fp,56,ra
	.set	noreorder
	.cpload	t9
	.set	reorder

	# Stack frame creation
	subu	sp,sp,56

	.cprestore 0
	sw	$fp,52(sp)
	sw	gp,48(sp)

	# de aqui al fin de la funcion uso $fp en lugar de sp.
	move	$fp,sp

	# Parametro
	# character convertIntToCharacter(unsigned int number) TODO VER ESTO XQ NO ENTIENDO X Q TENGO 8
B Y NO 4 SOLAMENTE
	sw	a0,56($fp)
	sw	a1,60($fp)	# Guardo en la direccion 60($fp) el parametro number

	sw	zero,20($fp)	# Guardo en la direccion 20($fp) el valor 0 (oneCharacter.length = 0;).
	sw	zero,24($fp)	# Guardo en la direccion 24($fp) el valor 0 (int i = 0;).
	sw	zero,28($fp)	# Guardo en la direccion 28($fp) el valor 0 (int rest = 0;).
$whileConvert:
	lw	v0,24($fp)	# Cargo en el registro v0 lo que hay guardado en la dir 24($fp), que es i.

	# (i < MAX_LENGTH_CHARACTER) ?
	slt	v0,v0,MAX_LENGTH_CHARACTER # Guardo en v0 TRUE si i es mas chico que
MAX_LENGTH_CHARACTER, sino guardo FALSE.
	beq	v0,FALSE,$loadLengthOfCharacter # If (i >= MAX_LENGTH_CHARACTER) goto
$loadLengthOfCharacter

	# i is less then MAX_LENGTH_CHARACTER

	# (number != 0) ?
	lw	v0,60($fp)	# Cargo en v0, la variable number, guardada en la dir 60($fp).
	bne	v0,zero,$internalLoopConvert # If (number != 0) goto $internalLoopConvert
	b	$loadLengthOfCharacter # Sale del while porque no cumple condiciones.
$internalLoopConvert:
	# Busco el resto de la division
	# rest = number % 10;
	lw	a0,60($fp)	# Cargo en a0 number
	li	v0,-859045888 # 0xffffffffcccc0000
	ori	v0,v0,0xcccd	# OR con un inmediato
	multu	a0,v0
	mfhi	v0		# Muevo hi
	srl	v1,v0,3		# Shift logico a derecha
	move	v0,v1
	sll	v0,v0,2		# Shift logico a izquierda
	addu	v0,v0,v1
```

```
        sll     v0,v0,1
        subu    v0,a0,v0    # En v0 queda el resto de la division.
        sw      v0,28($fp)  # Guardo el resultado de la division (rest) en la dir 28($fp).

        # oneCharacter.data[i] = rest + 48;
        lw      v1,24($fp)  # Cargo en v1, lo que hay en dir 24($fp), que es i.
        addu    v0,$fp,8    # Es para moverme 8 posiciones desde donde inicia el stack frame.
        addu    v1,v0,v1    # Me muevo i posiciones sobre el valor obtenido anteriormente.
                # Cargo en v1 la direccion obtenida.
        lbu     v0,28($fp)  # Cargo en v0, rest, guardado en la dir 28($fp).
        addu    v0,v0,48    # A rest le sumo 48 y guardo el resultado en v0.
        sb      v0,0(v1)    # Guardo el byte, resultado de la suma anterior, esta en v0, en la dir apuntada por v1.

        # number /= 10;
        lw      v1,60($fp)  # Cargo en v1 number.
        li      v0,-859045888 # 0xffffffffcccc0000
        ori     v0,v0,0xcccd  # OR con un inmediato
        multu   v1,v0
        mfhi    v0          # Muevo hi
        srl     v0,v0,3     # Shift logico a derecha. Tengo en v0, el cociente de la division.
        sw      v0,60($fp)  # Guardo en la dir 60($fp), en number, el resultado, entero, de la division.

        # i ++;
        lw      v0,24($fp)  # Cargo en v0, el valor de i, guardado en la dir 24($fp).
        addu    v0,v0,1     # Incremento en 1 a i. Cargo resultado en v0.
        sw      v0,24($fp)  # Guardo en la dir 24($fp), el nuevo valor de i, que esta en v0.

        b       $whileConvert # Salto incondicional al comienzo del while.

$loadLengthOfCharacter:
        # oneCharacter.length = i;
        lw      v0,24($fp)  # Cargo en v0, el valor de i, guardado en la dir 24($fp).
        sw      v0,20($fp)  # Guardo en la dir 20($fp), que es length, lo que tiene v0, que es i.

        # (oneCharacter.length == 1) ?
        lw      v1,20($fp)  # Cargo en v1 lo que hay en la dir 20($fp), que es length.
        li      v0,1        # Cargo en v0 el inmediato 1, para hacer la comparacion.
        bne     v1,v0,$investNumber # If (oneCharacter.length != 1) goto $investNumber.

        # oneCharacter.length is equal to 1.

        # Preparo los datos para hacer el return.
        lw      v0,8($fp)   # Cargo en v0, oneCharacter.data
        lw      v1,56($fp)  #
        sw      v0,0(v1)
        lw      v0,12($fp)
        lw      v1,56($fp)
        sw      v0,4(v1)
        lw      v0,16($fp)
        lw      v1,56($fp)
        sw      v0,8(v1)
        lw      v0,20($fp)
        lw      v1,56($fp)
        sw      v0,12(v1)
        b       $L23
$investNumber:
        l.s     $f0,20($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,$LC2
        div.d   $f0,$f2,$f0
        s.d     $f0,32($fp)
        sw      zero,24($fp)
        lw      v0,20($fp)
        addu    v0,v0,-1
        sw      v0,40($fp)
$L29:
```

```
        l.s     $f0,24($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,32($fp)
        c.lt.d  $f2,$f0
        bc1t    $L33
        b       $L30
$L33:
        l.s     $f0,40($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,32($fp)
        c.le.d  $f0,$f2
        bc1t    $L31
        b       $L30
$L31:
        lw      v1,24($fp)
        addu    v0,$fp,8
        addu    v0,v0,v1
        lbu     v0,0(v0)
        sb      v0,44($fp)
        lw      v1,40($fp)
        addu    v0,$fp,8
        addu    v0,v0,v1
        lbu     v0,0(v0)
        sb      v0,45($fp)
        lw      v1,24($fp)
        addu    v0,$fp,8
        addu    v1,v0,v1
        lbu     v0,45($fp)
        sb      v0,0(v1)
        lw      v1,40($fp)
        addu    v0,$fp,8
        addu    v1,v0,v1
        lbu     v0,44($fp)
        sb      v0,0(v1)
        lw      v0,24($fp)
        addu    v0,v0,1
        sw      v0,24($fp)
        lw      v0,40($fp)
        addu    v0,v0,-1
        sw      v0,40($fp)
        b       $L29
$L30:
        lw      v0,8($fp)
        lw      v1,56($fp)
        sw      v0,0(v1)
        lw      v0,12($fp)
        lw      v1,56($fp)
        sw      v0,4(v1)
        lw      v0,16($fp)
        lw      v1,56($fp)
        sw      v0,8(v1)
        lw      v0,20($fp)
        lw      v1,56($fp)
        sw      v0,12(v1)
$L23:
        lw      v0,56($fp)
        move    sp,$fp
        lw      $fp,52(sp)
        addu    sp,sp,56
        j       ra
        .end    convertIntToCharacter
        .size   convertIntToCharacter, .-convertIntToCharacter

        .rdata
        .align  3
$LC2:
```

```
.word   0
.word   1073741824
```

stack frames

character convertIntToCharacter(unsigned int value)		
posicion	**contenido**	**grupo**
56	ra	**SRA**
52	fp	
48	gp	
44		**LTA**
40	i	
36		
32		
28		
24	finish(false)	
20		
16		**ABA**
12		
8		
4		
0		

convertIntToCharacter stack frame

4. Ejecución

Para la ejecución utilizamos un makefile. Desde netBSD ejecutar:

- Para ***compilar*** el código:

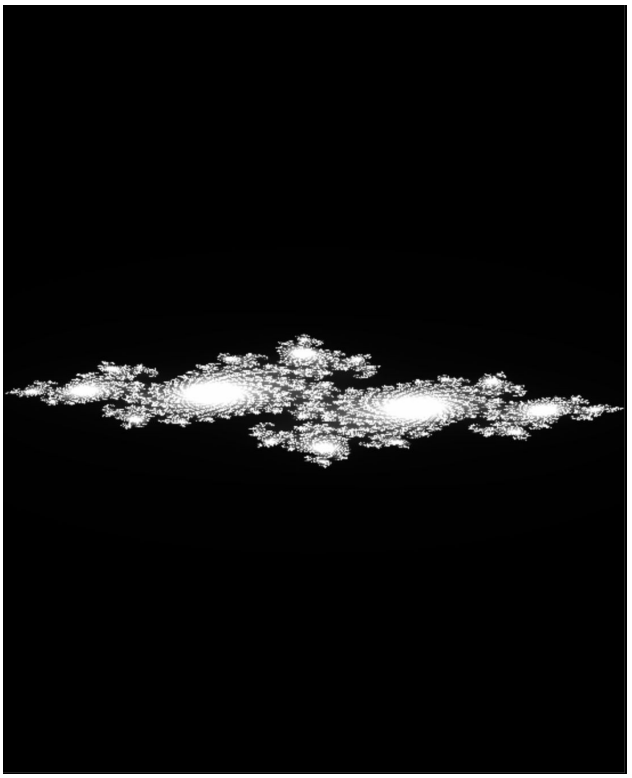
  ```
  $make
  ```

4.1. Casos de pruebas ejecutados

Para poder desarrollar la funcionalidad requerida, dibujo del conjunto de Julia y sus vecindades, se especifican los parámetros permitidos, junto a sus limitaciones. Algunos ejemplos utilizados:

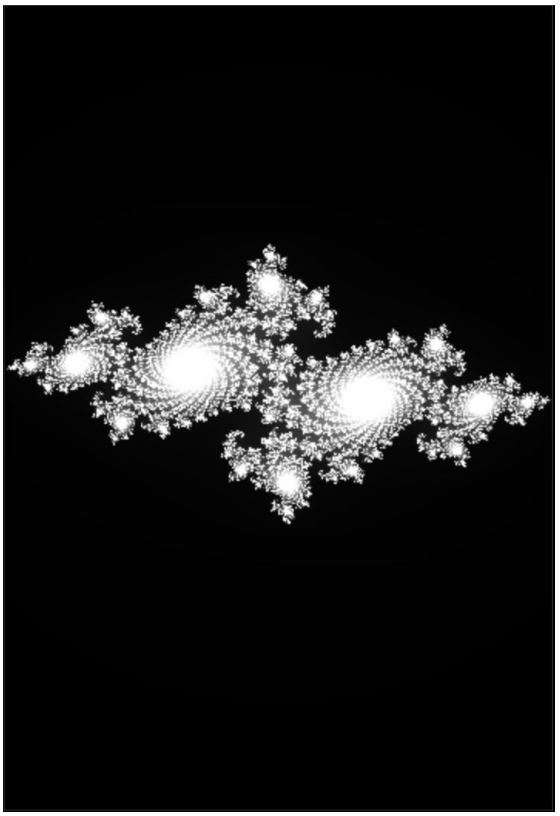
```
$ ./tp1 --resolution 546x674 -c -0.956+0.15236i --width 3 --height 10 -o prueba7.pgm
```

Resultado obtenido:



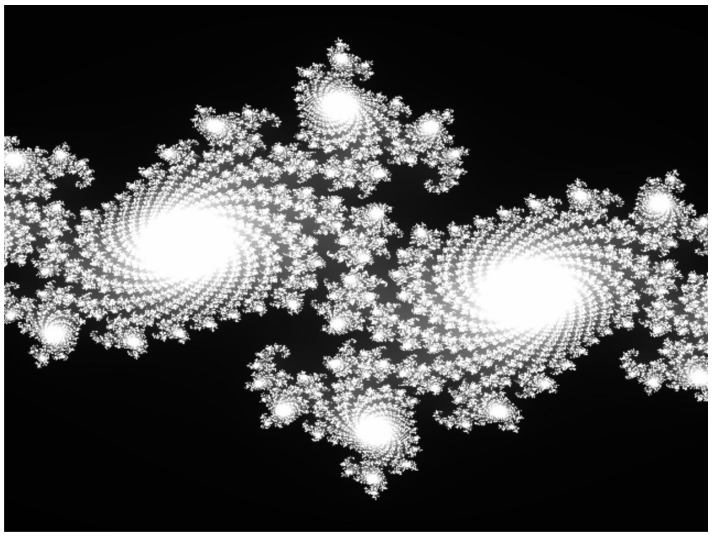
```
$ ./tp1 --resolution 356X523 -c -0.5201-0.15236i --width 3 --height 5 -o -
```

Resultado obtenido:



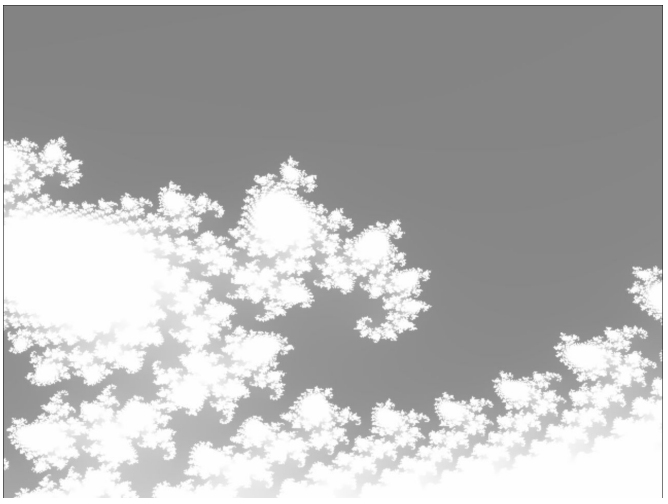
```
$ ./tp1 -o uno.pgm
```

Resultado obtenido:



```
$ ./tp1 -c 0.282-0.007i -w 0.005 -H 0.005 -o dos.pgm
```

Resultado obtenido:



Aclaración: El tamaño de las imágenes es meramente ilustrativo; en la entrega digital, se han incluido las originales.

4.2. Casos de prueba automáticos

4.2.1. Código fuente:

```
#!/bin/bash

echo "#########################################################################################"
echo "############################ Tests automaticos  ###################################"
echo "#########################################################################################"


#DIRECTORIO=/root1
#echo "El directorio ${DIRECTORIO} existe"

echo "Se guardaran los archivos resultantes de los tests en el directorio ./outputs-automatic-tests"

if [ -d "./outputs-automatic-tests" ]
then
```

```
  echo "El directorio ./outputs-automatic-tests existe, por lo tanto se elimina su contenido."
  rm -r outputs-automatic-tests/*
else
  echo "El directorio ./outputs-automatic-tests no existe, por lo tanto se creara."
  mkdir outputs-automatic-tests
fi


echo "###----------###    COMIENZA la generacion de imagenes automaticamente.    ###----------###"

./tp1 -r 50x50 --center 0.5+0.5i -w 25 --height 10 --output ./outputs-automatic-tests/prueba1.pgm
echo "Se genero el archivo prueba1.pgm, para lo cual se especifico center, w, height, s y output."

./tp1 --resolution 70x70 -c 0.5+0.5i --width 10 -H 30 -o ./outputs-automatic-tests/prueba2.pgm
echo "Se genero el archivo prueba2.pgm, para lo cual se especifico resolution, c, width, H y o."

./tp1 -c 0.5+0.5025863i --width 5 -H 8 -o ./outputs-automatic-tests/prueba3.pgm
echo "Se genero el archivo prueba3.pgm, para lo cual se especifico c, width, H y o."

./tp1 --resolution 100x300 --width 10 -H 5 -o ./outputs-automatic-tests/prueba4.pgm
echo "Se genero el archivo prueba4.pgm, para lo cual se especifico resolution, width, H y o."

./tp1 --resolution 50x30 -c 0.7+0.9i -H 5 -o ./outputs-automatic-tests/prueba5.pgm
echo "Se genero el archivo prueba5.pgm, para lo cual se especifico resolution, c, H y o."

./tp1 --resolution 69x35 -c -0.956+0.15236i --width 3 -o ./outputs-automatic-tests/prueba6.pgm
echo "Se genero el archivo prueba6.pgm, para lo cual se especifico resolution, c, width y o."

./tp1 --resolution 54x67 -c -0.956+0.15236i --width 3 --height 10 -o ./outputs-automatic-tests/prueba7.pgm
echo "Se genero el archivo prueba6.pgm, para lo cual se especifico resolution, c, width, height y o."

./tp1 --resolution 35X52 -c -0.5201-0.15236i --width 3 --height 5 -o - > ./outputs-automatic-tests/outputStdout.pgm
echo "Se ejecuto con salida estandar, que se redirecciono al archivo outputStdout.pgm, para lo cual se especifico resolution,
```

```
c, width, height y o (stdout, como - )."

./tp1 -o ./outputs-automatic-tests/uno.pgm
echo "Se genero el archivo uno.pgm, para lo cual se especifico o."

./tp1 -c 0.282-0.007i -w 0.005 -H 0.005 -o ./outputs-automatic-tests/dos.pgm
echo "Se genero el archivo dos.pgm, para lo cual se especifico c, w, H y o."

echo "###----------###    FIN de la generacion de imagenes automaticamente.    ###----------###"

echo "###-------------------------------------------------------------------------###"
echo "###-------------------------------------------------------------------------###"
echo "###-------------------------------------------------------------------------###"

echo "###----------###    COMIENZA la validacion de los parametros.    ###----------###"

varError=$(./tp1 -r 54x45d 2>&1)
if [ "$varError" = "[Error] La resolucion especificada es incorrecta." ]
 then
  echo "./tp1 -r 54x45d corrio ..... [OK]";
else
 echo "./tp1 -r 54x45d corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -r 54A45 2>&1)
if [ "$varError" = "[Error] La resolucion especificada es incorrecta." ];
 then echo "./tp1 -r 54A45 corrio ..... [OK]";
else
 echo "./tp1 -r 54A45 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi
```

```
varError=$(./tp1 -r -133x23 2>&1)
if [ "$varError" = "[Error] La resolucion especificada es incorrecta." ];
 then echo "./tp1 -r -133x23 corrio ..... [OK]";
else
 echo "./tp1 -r -133x23 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -c 896-233 2>&1)
if [ "$varError" = "[Error] Formato incorrecto del numero complejo para el centro." ];
 then echo "./tp1 -c 896-233 corrio ..... [OK]";
else
 echo "./tp1 -c 896-233 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -c i33-33 2>&1)
if [ "$varError" = "[Error] Formato incorrecto del numero complejo para el centro." ];
 then echo "./tp1 -c i33-33 corrio ..... [OK]";
else
 echo "./tp1 -c i33-33 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -c 3d4+56i 2>&1)
if [ "$varError" = "[Error] Formato incorrecto del numero complejo para el centro." ];
 then echo "./tp1 -c 3d4+56i corrio ..... [OK]";
else
 echo "./tp1 -c 3d4+56i corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -w -31 2>&1)
```

```
if [ "$varError" = "[Error] Formato incorrecto para el ancho." ];
 then echo "./tp1 -w -31 corrio ..... [OK]";
else
 echo "./tp1 -w -31 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -w 3e 2>&1)
if [ "$varError" = "[Error] Formato incorrecto para el ancho." ];
 then echo "./tp1 -w 3e corrio ..... [OK]";
else
 echo "./tp1 -w 3e corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -H -5.25 2>&1)
if [ "$varError" = "[Error] Formato incorrecto para el alto." ];
 then echo "./tp1 -H -5.25 corrio ..... [OK]";
else
 echo "./tp1 -H -5.25 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -H 3e 2>&1)
if [ "$varError" = "[Error] Formato incorrecto para el alto." ];
 then echo "./tp1 -H 3e corrio ..... [OK]";
else
 echo "./tp1 -H 3e corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -o uno. 2>&1)
if [ "$varError" = "[Error] Extension incorrecta para el archivo de salida." ];
```

```
 then echo "./tp1 -o uno. corrio ..... [OK]";
else
 echo "./tp1 -o uno. corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -o uno.txt 2>&1)
if [ "$varError" = "[Error] Extension incorrecta para el archivo de salida." ];
 then echo "./tp1 -o uno.txt corrio ..... [OK]";
else
 echo "./tp1 -o uno.txt corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi

varError=$(./tp1 -o uno.pgm23 2>&1)
if [ "$varError" = "[Error] Extension incorrecta para el archivo de salida." ];
 then echo "./tp1 -o uno.pgm23 corrio ... [OK]";
else
 echo "./tp1 -o uno.pgm23 corrio con ERROR!!! - Resultado obtenido: ";
 echo $varError
fi


echo "###----------###    FIN de la validacion de los parametros.    ###----------###"

echo "################################################################################"
echo "############################ FIN Tests automaticos  ##############################"
echo "################################################################################"
```

4.2.2. Ejecución de bash de prueba

```
$bash automatic-tests.sh
```

4.3. Casos de prueba automáticos de imágenes

Se genero una imagen pequeña (5x5) a mano usando el algoritmo base para generar cada pixel, provisto en el enunciado.
Se comparó obteniéndose los siguientes resultados:

```
$./tp1 -r 5x5 -m mips32
```

5x5manual.pgm

```
P2
5 5
255
2 2 3 2 2
3 4 10 14 3
117 201 103 14 255
27 254 14 104 201
2 4 9 11 4
```

5x5auto.pgm

```
P2
5 5
255
2 2 3 2 2
3 4 10 14 3
117 201 103 14 255
27 254 14 104 201
2 4 9 11 4
```

5x5manual.pgm

5x5auto.pgm

Como puede verse la imagen patrón utilizada y la imagen autogenerada sin idénticas; y aunque este es un caso de testing con una imagen de un tamaño bastante pequeño permite verificar de forma automática que el generador de imagen de Julia funciona correctamente.

5. Conclusión

A través del presente trabajo se logró realizar una implementación pequeña de un programa C y MIPS32 verificando las diferentes figuras de Julia. Se intentó implementar un método de comparación a partir de una imagen patrón con las imágenes generadas por el programa. A partir de una imagen muy pequeña que permitió verificar el correcto funcionamiento de la aplicación de forma automática.

Al mismo tiempo se pudo verificar que teniendo una implementación en C y otra en código MIPS32, la implementación MIPS32 es más rápida en al generación de la imagen que la implementación en C puro.