Trabajo Práctico Nº 0: Infraestructura Básica

Sebastian Ripari, *Padrón Nro. 96453*
sebastiandanielripari@hotmail.com

César Emanuel Lencina, *Padrón Nro. 96078*
cel_1990@live.com

Pablo Daniel Sivori, *Padrón Nro. 84026*
sivori.daniel@gmail.com

Entrega: 07/09/2017

2do. Cuatrimestre de 2017
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

En el presente trabajo práctico se describirán todos los pasos y conclusiones relacionadas al desarrollo e implementación de una versión en lenguaje C, de la Criba de Eratóstenes. Y su posterior compilacion y ejecucion bajo el procesador MIPS.

Índice

1. Introducción

El objetivo del presente trabajo práctico es familiarizarse con el emulador gxemul mediante la implementacion de un programa que nos devuelva por stdout o en un archivo de salida, los números primos menores a un número natural N el cuál es ingresado por parámetro.

2. Entorno de trabajo

Mediante el emulador Gxemule pudimos simular una DEC Station 5000/200 con un microprocesador MIPS de 32 Bits, corriendo un sistema operativo NetBSD/pmax.

2.1. Lenguaje

Como lenguaje de implementación se eligio C ya que el mismo permite una alta portabilidad entre diferentes plataformas. El desarrollo del programa se realizó usando editores de texto (gedit,vim, kwrite y sublime) y compilando los archivos fuente con GCC que viene en linux. Ya que este compilador es compatible con el sistema operativo NetBSD y con la arquitecura MIPS. Para compilar, ejecutar el siguiente comando:

$ make

2.2. Descripción del programa

Por empezar tenemos un archivo llamado `erat.c` que contiene la función `main`, aquí lo primero que hacemos es la validacion de los argumentos, mediante una función llamada `validarArgumentos`. Caso de ingresarse algo inválido el programa no continuará, y retornara un valor distinto de 0, osea un código de error. Caso contrario, se llama a la función `realizarAccion` y aquí comienza el procesamiento de calcular la cantidad de números primos, entre 2 y el número `tope` ingresado. Básicamente lo que hacemos es crear un array que contiene todos los números entre 2 y el número ingresado, el `tope`. y luego le aplicamos a este array el algoritmo de la Criba de Erastostenes, mediante la función `encontrarNumerosPrimos`, que en las posiciones del array donde no hay un numero primo setea un cero. Para luego tomar este array desde una función que se llama `imprimirPorPantalla`, y aquí imprimir todos los números distintos de cero, que son los primos.

2.2.1. Errores posibles

1. Comando invalido: Cuando se invoca al programa con flags inexistentes o escritos de forma desordenada. El programa retorna 2.

2. Fuera de rango: Cuando se ingresa un numero menor o igual que 1. El programa retorna 1.

2.2.2. Corridas de prueba

1. Caso N = -5 El programa no imprime nada y retorna el código de error 1.
2. Caso N = 1 El programa no imprime nada y retorna el código de error 1.
3. Caso N = 10 El programa imprime: 2 3 5 7 y retorna el código de éxito 0.
4. Caso N = 50 El programa imprime 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 y retorna el código de éxito 0.
5. Caso N = 100 El programa imprime 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 y retorna el código de exito 0.

3. Conclusiones

1. Si bien lo solicitado por el programa no era excesivamente difícil, la realización completa del TP llevó cierta dificultad al tener que realizarlo en el contexto solicitado: alta portabilidad, desarrollo en C, e informe hecho en LaTeX.

2. En el primer caso la dificultad radicaba en tener configurado y funcionando el GXEmul dentro de un Linux, y lograr que en ambos casos el programa compile y corra sin problemas.

3. Debido a nuestro desconocimiento con LaTeX, tuvimos que invertir tiempo en encontrar forma de realizar el presente documento de la manera más correcta posible

4. En cuanto al trabajo grupal en si mismo, no hubo inconvenientes de ningún tipo ya que al ser el grupo relativamente chico y tener conocimiento del manejo del versionado de un proyecto ante cambios ingresado por los integrantes (por medio del GIT), la introducción de modificaciones y correcciones fué fluida.

4. Código fuente

```
#include <stdio.h>
#include "eratfunc.h"


int main(int argc, char **argv){
        int alerta = validarArgumentos(argc, argv);
        if (alerta == TODO_OK){
                alerta = realizarAccion(argc, argv);
        }
        return alerta;
}


#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "eratfunc.h"

void inicializarNumerosPrimos(int tope, int arreglo[]){

        int valor = INICIO_VALOR;
        int posicion;
        for (posicion = INICIO_ARRAY; posicion <= tope; posicion++){
                valor++;
                arreglo[posicion] = valor;
        }
}

void encontrarNumerosPrimos(int tope, int arreglo[]){

        inicializarNumerosPrimos(tope, arreglo);
        int posicionPivote;
        for (posicionPivote = INICIO_ARRAY; (posicionPivote * posicionPivote) <= tope; posicionPivote++) {
       if (arreglo[posicionPivote] != CAMPO_VACIO) {
            int posicionMovil;
            for (posicionMovil = INICIO_ARRAY; (posicionPivote * posicionMovil) <= tope; posicionMovil++) {
                                arreglo[posicionPivote*posicionMovil] = CAMPO_VACIO;
                        }
                }
        }
}

void imprimirPorPantalla(int tope, int arreglo[]){
        int i;
        for (i = INICIO_ARRAY; i <= tope; i++){
                if (arreglo[i] != CAMPO_VACIO){
                        printf("%d  ", arreglo[i]);
                }
        }
        printf("\n");
}

int generarArchivo(int tope, int arreglo[], const char* name){
        FILE *archivo;
        archivo = fopen(name, "w");
        if (archivo != ALGUN_PROBLEMA){
                int i;
                for (i = INICIO_ARRAY; i <= tope; i++) {
                        if (arreglo[i] != CAMPO_VACIO) {
                                fprintf(archivo, "%d\n", arreglo[i]);
                        }
                }
                fclose(archivo);
           return TODO_OK;
        } else{
            return ERROR_NOMBRE_ARCH_INVALIDO;
        }
}

void mostrarVersion(){
        printf("ERAT version 1.1 \n");
}

void mostrarAyuda(){
        printf("Usage: \n");
        printf("        erat -h\n");
        printf("        erat -V\n");
        printf("        erat [options] N\n");
        printf("Options:\n");
        printf("        -h, --help          Prints usage information.\n");
        printf("        -V, --version       Prints version information.\n");
        printf("        -o, --output        Path to output file.;\n");
}

int validarArgumentos(int argc, char **argv){
        if ((argc > TODOS_LOS_ARGUMENTOS) || (argc < DOS_ARGUMENTOS)) {
                return ERROR_COMANDO_INVALIDO;
        }

        if (strcmp(argv[OPCION], "-h") != CORRECTO){
                if (strcmp(argv[OPCION], "-V") != CORRECTO){
                        if ((strcmp(argv[OPCION], "-o") != CORRECTO) || ((strcmp(argv[OPCION], "-o") == CORRECTO) &&
                                return ERROR_COMANDO_INVALIDO;
                        }
                }
```

```
                else {
                if (argc > DOS_ARGUMENTOS) {
                        return ERROR_COMANDO_INVALIDO;
                }
        }
        }
        else {
                if (argc > DOS_ARGUMENTOS) {
                        return ERROR_COMANDO_INVALIDO;
                }
        }

        if (argc == TODOS_LOS_ARGUMENTOS) {
                if ((atoi(argv[ENTERO_POS_3]) > MAXIMO) || (atoi(argv[ENTERO_POS_3]) < MINIMO)){
                        return ERROR_FUERA_DE_RANGO;
                }
        }

        if (argc == TRES_ARGUMENTOS) {
                if ((atoi(argv[ENTERO_POS_2]) > MAXIMO) || (atoi(argv[ENTERO_POS_2]) < MINIMO)){
                        return ERROR_FUERA_DE_RANGO;
                }
        }
        return TODO_OK;
}


int realizarAccion(int argc, char **argv) {
        int mensajeDeError = TODO_OK;
        if (strcmp(argv[OPCION], "-h") == CORRECTO){
                mostrarAyuda();
        }
        else {
                if (strcmp(argv[OPCION], "-V") == CORRECTO){
                        mostrarVersion();
                }
                else {
                        int tope;
                        if (argc == TRES_ARGUMENTOS){
                                tope = atoi(argv[ENTERO_POS_2]);
                        }
                        else {
                                tope = atoi(argv[ENTERO_POS_3]);
                        }
                        int arreglo[tope];
                        encontrarNumerosPrimos(tope,arreglo);
                        if ((strcmp(argv[OPCION], "-o") == CORRECTO) && (strcmp(argv[OPCION_1], "-") == CORRECTO)){
                                imprimirPorPantalla(tope,arreglo);
                        }
                        else {
                                if(strcmp(argv[OPCION],"-o") == CORRECTO){
                                        mensajeDeError = generarArchivo(tope,arreglo,argv[OPCION_1]);
                                }
                        }
                }
        }
        return mensajeDeError;
}


#define MAXIMO 4096
#define MINIMO 2

#define CAMPO_VACIO 0
#define INICIO_VALOR 1
#define INICIO_ARRAY 2

#define TODO_OK 0
#define ERROR_FUERA_DE_RANGO 1
#define ERROR_COMANDO_INVALIDO 2
#define ERROR_NOMBRE_ARCH_INVALIDO 4

#define CORRECTO 0
#define OPCION 1
#define OPCION_1 2
#define ENTERO_POS_2 2
#define ENTERO_POS_3 3
#define DOS_ARGUMENTOS 2
#define TRES_ARGUMENTOS 3
#define TODOS_LOS_ARGUMENTOS 4

#define ALGUN_PROBLEMA 0

//Pre: Se ingresa el tope hasta el cual se calcularan los numeros primos y un arreglo de enteros no inicializado.
//Pos: El arreglo se encuentra cargado con todos los numeros correspondientes hasta el valor tope.
void inicializarNumerosPrimos(int tope, int arreglo[]);

//Pre: Se ingresa el arreglo y su cantidad maxima de elementos.
//Pos: Los valores distintos a cero en el rreglo seran los numeros primos buscados.
void encontrarNumerosPrimos(int tope, int arreglo[]);


//Pre: Se ingresa el arreglo y su cantidad maxima de elementos.
//Pos: Se imprimen por pantalla los numeros primos.
void imprimirPorPantalla(int tope, int arreglo[]);

//Pre: Se ingresa el arreglo y su cantidad maxima de elementos.
//Pos: Se crea un archivo de texto en el cual se guardaran los numeros primos.
int generarArchivo(int tope, int arreglo[], const char* name);
```

```
//Imprime por pantalla las instrucciones de uso de erat
void mostrarAyuda();

//Imprime por pantalla la version de erat
void mostrarVersion();

//Verifica que los argumentos sean -h, -V, -o N, - N; donde N es un entero mayor que 2 y menor que el maximo (yo pus
//Devuelve: 1 si el entero se escapa del rango predeterminado.
//                      2 si no se ingresaron comando validos.
//                      0 si todos los argumentos son validos y se puede proceder.
int validarArgumentos(int argc, char **argv);

//Realizara alguna de las acciones:
//      -h muestra el menu de ayuda.
//      -V muestra la version.
//      -o muestra por pantalla y crea un archivo.
//      -  muestra por pantalla.
int realizarAccion(int argc, char **argv);
```

5. Código assembly generado en MIPS32

```
        .file   1 "erat.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .text
        .align  2
        .globl  main
        .ent    main
main:
        .frame  $fp,48,$31              # vars= 8, regs= 3/0, args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $31,40($sp)
        sw      $fp,36($sp)
        sw      $28,32($sp)
        move    $fp,$sp
        sw      $4,48($fp)
        sw      $5,52($fp)
        lw      $4,48($fp)
        lw      $5,52($fp)
        la      $25,validarArgumentos
        jal     $31,$25
        sw      $2,24($fp)
        lw      $2,24($fp)
        bne     $2,$0,$L6
        lw      $4,48($fp)
        lw      $5,52($fp)
        la      $25,realizarAccion
        jal     $31,$25
        sw      $2,24($fp)
$L6:
        lw      $2,24($fp)
        move    $sp,$fp
        lw      $31,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $31
        .end    main
        .size   main, .-main
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

        .file   1 "eratfunc.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .text
        .align  2
```

```
        .globl  inicializarNumerosPrimos
        .ent    inicializarNumerosPrimos
inicializarNumerosPrimos:
        .frame  $fp,24,$31              # vars= 8, regs= 2/0, args= 0, extra= 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $25
        .set    reorder
        subu    $sp,$sp,24
        .cprestore 0
        sw      $fp,20($sp)
        sw      $28,16($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        li      $2,1                    # 0x1
        sw      $2,8($fp)
        li      $2,2                    # 0x2
        sw      $2,12($fp)
$L18:
        lw      $2,12($fp)
        lw      $3,24($fp)
        slt     $2,$3,$2
        beq     $2,$0,$L21
        b       $L17
$L21:
        lw      $2,8($fp)
        addu    $2,$2,1
        sw      $2,8($fp)
        lw      $2,12($fp)
        sll     $3,$2,2
        lw      $2,28($fp)
        addu    $3,$3,$2
        lw      $2,8($fp)
        sw      $2,0($3)
        lw      $2,12($fp)
        addu    $2,$2,1
        sw      $2,12($fp)
```

6. Enunciado del trabajo practico

66:20 Organización de Computadoras
Trabajo práctico 0: Infraestructura básica

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa (y su correspondiente documentación) que resuelva el problema piloto que presentaremos más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 7), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión relativamente reciente del sistema operativo NetBSD [2].

5. Programa

Se trata de una versión en lenguaje C de la Criba de Eratóstenes [3]. El programa recibirá por `stdin` un número natural N, y dará por `stdout` (o

escribirá en un archivo) una lista de todos los números primos menores que N. De haber errores, los mensajes de error deberán salir exclusivamente por `stderr`.

5.1. Comportamiento deseado

Primero, usamos la opción `-h` para ver el mensaje de ayuda:

```
$ erat -h
Usage:
  erat -h
  erat -V
  erat [options] N
Options:
  -h, --help     Prints usage information.
  -V, --version  Prints version information.
  -o, --output   Path to output file.
Examples:
  erat -o - 10
```

Ahora usaremos el programa para obtener los números primos menores que 10. Usamos "-" como argumento de `-o` para indicarle al programa que imprima el tablero por `stdout`:

```
$ (echo 10) | erat
2
3
5
7
```

El programa deberá retornar un error si su argumento está fuera del rango $[2, \text{MAXINT}]$.

6. Implementación

El programa a implementar deberá satisfacer algunos requerimientos mínimos, que detallamos a continuación.

El propósito de `erat()` es simple: eliminar, de un arreglo de números naturales consecutivos que comienza por 2, todos los números que no sean primos.

```
void erat(unsigned int *p, int n);
```

El programa deberá procesar los argumentos de entrada, crear el arreglo de números naturales consecutivos, de los cuales `erat()` debe poner en 0 los que no sean primos, y escribir en `stdout` o un archivo aquellos que hayan resultado ser primos.

6.1. Algoritmo

El algoritmo a implementar es la Criba de Eratóstenes [3], explicado en clase.

7. Informe

El informe deberá incluir:

- Este enunciado;
- Documentación relevante al diseño e implementación del programa;
- Corridas de prueba para N= -5, 1, 10, 50 y 100, con los comentarios pertinentes;
- El código fuente completo, en dos formatos: digitalizado e impreso en papel.

8. Fecha de entrega

La última fecha de entrega y presentación es el jueves 7 de Septiembre de 2017.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project, `http://www.netbsd.org/`.

[3] Criba de Eratóstenes, `http://http://es.wikipedia.org/wiki/Criba_de_Erat%C3%B3stenes`.