Trabajo Práctico 0: Infraestructura básica

[66.20] Organización de Computadoras
2do cuatrimestre de 2019

Alumno:	Bobadilla Catalan, German
Número de padrón:	901232
Email:	bobadillagerman@gmail.com
Alumno:	Leloutre, Daniela
Número de padrón:	96783
Email:	daniela.leloutre@gmail.com
Alumno:	Soro, Lucas
Número de padrón:	95665
Email:	lugusor@gmail.com

Índice

1. Resumen

Con el objetivo de familiarizarnos con las herramientas a utilizar en los siguientes trabajos prácticos, desarrollamos un programa en lenguaje C capaz de multiplicar matrices cuadradas de números reales. Utilizando el programa GXemul y una máquina MIPS generamos el código de compilación del programa en código MIPS32 asegurando la portabilidad del mismo.

2. Diseño e implementación

Dado que las dos matrices a multiplicar están representadas en una sola linea con el formato row major order, y dado que con el primer elemento de la línea podemos saber el tamaño (ambas son cuadradas de la misma dimensión). El algoritmo propuesto para leer las matrices consiste en guardar el primer elemento de cada línea en una variable que representa la dimensión N de la matriz, luego se toman los N siguientes números para crear la primera matriz y finalmente los N últimos números de la línea para crear la segunda matriz. Este algoritmos se repite hasta llegar al End Of File (EOF).

Una vez creadas las matrices se efectúa el producto entre ellas. Para esto se cicla por fila y columna sumando el producto de los índices que correspondan y se asigna el resultado a una nueva matriz creada previamente.

3. Comandos para compilar y ejecutar el programa

Se puede compilar el programa con el siguiente comando:

```
$ gcc -o tp0 tp0.c
```

Y luego ejecutarlo con el comando:

```
$ ./tp0 < in.txt > out.txt
```

Siendo in.txt el archivo con las matrices a multiplicar, y out.txt el archivo con el resultado.

Alternativamente si queremos que muestre la salida por pantalla se puede utilizar el siguiente comando:

```
$ cat in.txt | ./tp0
```

Para visualizar los mensajes de ayuda y versión del programa los comandos son:

```
$ ./tp0 -h
$ ./tp0 -V
```

4. Corridas de prueba

A continuación se muestran unos casos de prueba corridos desde la consola del GXEmul.

- Prueba de multiplicar dos matrices de un mínimo tamaño con enteros:

  ```
  root@:/home/root/TP0# cat doc1
  2 1 2 3 4 5 6 7 8
  root@:/home/root/TP0# ./tp0 < doc1
  2 19 22 43 50
  ```

- Prueba con dos lineas, mas de un espacio en blanco entre números y algún valor con decimales.

```
root@:/home/root/TP0# cat doc2
2 1      2 3 4 1 2 3 4
3 1 2 3 4 5 6.1 3 2 1 1 0 0 0 1 0 0 0 1
root@:/home/root/TP0# ./tp0 < doc2
2 7 10 15 22
3 1 2 3 4 5 6.1 3 2 1
```

- Prueba con tres lineas y algún valor con notación exponencial.

```
root@:/home/root/TP0# cat doc3
2 1 2 3 4 5 6 7 8
3 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
4 1 2 3 4 5 6 2.345670e+002 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
root@:/home/root/TP0# ./tp0 < doc3
2 19 22 43 50
3 84 90 96 201 216 231 318 342 366
4 250 260 270 280 6307.17 6560.74 6814.31 7067.88 986 1028 1070 1112 1354 1412 1470 1528
```

- Prueba de informar por stderr cuando hay un valor erróneo en una matriz como por ejemplo una letra.

```
root@:/home/root/TP0# cat doc4
2 1 2 a 4 5 6 7 8
root@:/home/root/TP0# ./tp0 < doc4
Error fscanf: Valor erroneo en una matriz
```

- Prueba de informar por stderr cuando hay un valor erróneo en la dimensión de la matriz como por ejemplo una letra.

```
root@:/home/root/TP0# cat doc5
b 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
root@:/home/root/TP0# ./tp0 < doc5
Error fscanf: Dimension erronea en una matriz
```

- Prueba de informar por stderr cuando hay el numero de elementos de la matriz no coincido con la dimensión.

```
root@:/home/root/TP0# cat doc6
3 1 2 3 4 5 6
root@:/home/root/TP0# ./tp0 < doc6
Error fscanf: Valor erroneo en una matriz
```

5. Código fuente en lenguaje C

```
#include <stdio.h>
#include <string.h>
#include <getopt.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

#define ERROR -1
#define SALIDA_EXITOSA 0
```

```
typedef struct matrix {
    size_t rows;
    size_t cols;
    double* array;
} matrix_t;

// Constructor de matrix_t
matrix_t* create_matrix(size_t rows, size_t cols){
matrix_t* matriz = malloc(sizeof(matrix_t));
matriz->rows = rows;
matriz->cols = cols;
return matriz;
}

// Destructor de matrix_t
void destroy_matrix(matrix_t* m){
    free(m->array);
free(m);
}

// Imprime matrix_t sobre el file pointer fp en el formato solicitado
// por el enunciado
int print_matrix(FILE* fp, matrix_t* m){
    if (fprintf(fp, "%zd ", m->cols) < 0) return ERROR;
    int i = 0;
    while (i < (int)(m->cols*m->rows)){
        if (fprintf(fp, "%g ", m->array[i]) < 0) return ERROR;
        i++;
    }
    fprintf(fp, "\n");

    return SALIDA_EXITOSA;
}

// Multiplica las matrices en m1 y m2
matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2){
matrix_t* resultado;
size_t dimension;
dimension = m2->cols;
double* array = (double*) malloc(dimension*dimension*sizeof(double));
resultado = create_matrix(dimension,dimension);
    int i;
for (i = 0; i < dimension; i++) {
        int j;
        for (j = 0; j < dimension; j++) {
            float sum = 0.0;
            int k;
            for (k = 0; k < dimension; k++)
                sum = sum + m1->array[i * dimension + k] * m2->array[k * dimension + j];
            array[i * dimension + j] = sum;
        }
}
resultado->array = array;
return resultado;
```

```
}


int main(int argc, char *argv[]) {
    int option = 0;
    const char *short_opt = "hV";
    struct option long_opt[] = {
            {"version", no_argument,       NULL, 'V'},
            {"help",    no_argument,       NULL, 'h'},
            {NULL, 0,                      NULL, 0}
    };
    FILE *inputFile = NULL;
    FILE *outputFile = NULL;
    size_t dimension;

    while ((option = getopt_long(argc, argv, short_opt, long_opt, NULL)) != -1) {
        switch (option) {
            case 'V':
                printf("TP #0 de la materia Organizacion de Computadoras \n");
                printf("Alumnos: \n");
                printf(" Bobadilla Catalan, German\n Leloutre, Daniela \n Soro, Lucas \n");
                return SALIDA_EXITOSA;
            case 'h':
                printf("Usage: \n");
                printf(" %s -h \n", argv[0]);
                printf(" %s -V \n", argv[0]);
                printf(" %s < in_file > out_file \n", argv[0]);
                printf("Options: \n");
                printf(" -V, --version  Print version and quit. \n");
                printf(" -h, --help     Print this information. \n");
printf("Examples: \n");
printf(" tp0 < in.txt > out.txt \n");
                printf(" cat in.txt | tp0 > out.txt \n");
                return SALIDA_EXITOSA;
            default:
                // así está en el manual de getopt
                abort();
        }
    }

    inputFile = stdin;
    outputFile = stdout;

    matrix_t* matriz1;
    matrix_t* matriz2;

    int k;
    while ((k=fscanf(inputFile,"%zd",&dimension)) != EOF) {
if (k==0){
fprintf(stderr, "Error fscanf: Dimension erronea en una matriz \n");
return ERROR;
}
matriz1 = create_matrix(dimension,dimension);
matriz2 = create_matrix(dimension,dimension);
```

```
float dato;

double* arreglo1 = (double*) malloc(dimension*dimension*sizeof(double));
double* arreglo2 = (double*) malloc(dimension*dimension*sizeof(double));

        int i;
for(i=0;i<(int)(dimension*dimension);i++){
if (fscanf(inputFile,"%g",&dato)==1)
arreglo1[i] = dato;
else{
fprintf(stderr, "Error fscanf: Valor erroneo en una matriz \n");
             return ERROR;
}
}

        int j;
for(j=0;j<(int)(dimension*dimension);j++){
if (fscanf(inputFile,"%g",&dato)==1)
arreglo2[j] = dato;
else{
fprintf(stderr, "Error fscanf: Valor erroneo en una matriz \n");
             return ERROR;
}
}

matriz1->array = arreglo1;
matriz2->array = arreglo2;

matrix_t* resultado;
resultado = matrix_multiply(matriz1, matriz2);

if(print_matrix(outputFile, resultado) == ERROR) {
fprintf(stderr, "Error print_matrix: No se pudo escribir la matriz resultado \n");
return ERROR;
}
destroy_matrix(resultado);

        destroy_matrix(matriz1);
        destroy_matrix(matriz2);
    }

    if(fclose(inputFile)==EOF){
        fprintf(stderr, "Error fclose: %s\n", strerror( errno ));
        return ERROR;
    }

    if(fclose(outputFile)==EOF){
        fprintf(stderr, "Error fclose: %s\n", strerror( errno ));
        return ERROR;

    }

    return SALIDA_EXITOSA;
}
```

6. Código MIPS32 generado por el compilador

```
    .file 1 "tp0.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align 2
.globl create_matrix
.ent create_matrix
create_matrix:
.frame $fp,48,$ra # vars= 8, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.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
sw $a0,48($fp)
sw $a1,52($fp)
li $a0,12 # 0xc
la $t9,malloc
jal $ra,$t9
sw $v0,24($fp)
lw $v1,24($fp)
lw $v0,48($fp)
sw $v0,0($v1)
lw $v1,24($fp)
lw $v0,52($fp)
sw $v0,4($v1)
lw $v0,24($fp)
move $sp,$fp
lw $ra,40($sp)
lw $fp,36($sp)
addu $sp,$sp,48
j $ra
.end create_matrix
.size create_matrix, .-create_matrix
.align 2
.globl destroy_matrix
.ent destroy_matrix
destroy_matrix:
.frame $fp,40,$ra # vars= 0, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,40
```

```
.cprestore 16
sw $ra,32($sp)
sw $fp,28($sp)
sw $gp,24($sp)
move $fp,$sp
sw $a0,40($fp)
lw $v0,40($fp)
lw $a0,8($v0)
la $t9,free
jal $ra,$t9
lw $a0,40($fp)
la $t9,free
jal $ra,$t9
move $sp,$fp
lw $ra,32($sp)
lw $fp,28($sp)
addu $sp,$sp,40
j $ra
.end destroy_matrix
.size destroy_matrix, .-destroy_matrix
.rdata
.align 2
$LC0:
.ascii "%zd \000"
.align 2
$LC1:
.ascii "%g \000"
.align 2
$LC2:
.ascii "\n\000"
.text
.align 2
.globl print_matrix
.ent print_matrix
```

7. Conclusiones

El trabajo práctico nos resultó interesante, principalmente por lo que representó trabajar con el emulador GXEmul, emular la arquitectura MIPS, crear el túnel de comunicación entre el host OS (Linux, distribución Ubuntu) y el guest OS (NetBSD). Aprendimos como transferir archivos entre ambos sistemas y también ciertos aspectos del lenguaje C con el cual no estábamos totalmente familiarizados.