Trabajo Practico N^o 0 "Infraestructura Básica"

Samanta Loiza, *Padrón Nro. 91.935*
samiloiza@gmail.com
Victoria Pérez Bustos, *Padrón Nro. 92.060*
vickyperezbustos@gmail.com
Anton Fossum Hylin, *Intercambio*
anton.hylin@gmail.com

2do. Cuatrimestre de 2015
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Índice

Resumen

El objetivo de este trabajo se basa en familiarizarse con las herramientas de software que se utilizará en los trabajos futuros. Para simular el entorno de desarrollo que se utilizará en los trabajos, se usó el programa GXemul, el cual nos permite trabajar sobre una máquina MIPS que corre el sistema operativo NetBSD. Para este trabajo se creó un programa el cual lee el contenido de dos matrices y las multiplica. Por cada multiplicación, imprime la matriz en forma de línea.

1. Introducción

2. Compilación y Ejecución del programa

Para compilar:

- Parados en la carpeta donde tenemos el fuente ejecutamos.

```
make all
```

Para ejecutar:

- Parados en la carpeta donde tenemos el ejecutable generado por la compilación, el programa se invoca a través de línea de comandos de la siguiente manera:

```
 cat fileTest/test1.txt | ./tp0
cat fileTest/test1.txt | ./tp0 >out1.txt
./tp0 <test1.txt
./tp0 <test1.txt >out1.txt
```

El menú de ayuda especifica las opciones disponibles al momento de invocar el programa:

```
Usage:
./tp0 -h
./tp0 -V
OPTIONS:
   -h, --help Print this information and quit.
   -V, --version Print version and quit
```

Ejemplo:

```
   ./tp0 <in.txt >out.txt
    cat in.txt | ./tp0 >out.txt
```

3. Implementación

3.1. Desarrollo

Para el desarrollo del trabajo se realizó un programa escrito en lenguaje C. El usuario tendrá la posibilidad de ver un mensaje de ayuda y la versión del mismo.

En base a un stream de entrada armamos matrices para ser multiplicadas de a pares para luego mostrar su resultado en un stream de salida.

Finalmente el programa fue compilado en la máquina virtual proporcionada por el GXemul, para obtener el código MIPS32 generado por el compilador de dicha máquina.

3.2. Implementación en C

Implementamos diversas funciones para este trabajo, las cuales se encargan de:

- Checkear los argumentos recibidos:

  int checkArguments(int,char[])*

- Validar la línea leída, que contenga caracteres válidos:

  *int validoLinea(char *linea)*

4. Pruebas

Se utilizaron los siguientes archivos de prueba

- *badDim.txt*:

```
4x2 3 2 1 4 4 1 2 4
3x1 4 3 2
```

- *charTest.txt*

```
2x3 1 a 3 4 5 6.1
3x2 1 0 0 0 0 1
3x3 1 2 3 4 a 6.1 3 2 1
3x1 1 1 0
```

- *test2.txt*

```
2x2 2.0 1.0 3.0 4.0
2x3 1.0 4.0 6.0 2.0 5.0 3.0
2x3 1.0 2.0 3.0 4.0 5.0 6.0
3x1 1.0 0.0 0.0
```

4.1. 1^o Prueba

En esta prueba probamos que la matriz sea válida.

```
cat fileTest/charTest.txt | ./tp0
      Matriz incorrecta. 2x3 1 a 3 4 5 6.1
```

4.2. 2^o Prueba

```
    cat fileTest/emptyfile.txt | ./tp0
      No stdin
```

4.3. 3^o Prueba

```
   cat fileTest/badDim.txt | ./tp0
      Matrices incorrectas para la multiplicacion.
```

4.4. 4^o Prueba

```
   ./tp0 <fileTest/test2.txt
      2x3 4.000000 13.000000 15.000000 11.000000 32.000000 30.000000
      2x1 1.000000 4.000000
```

Con esta prueba multiplicamos dos pares de matrices y mostramos por pantalla el resultado obtenido. Si queremos redireccionar el resultado de salida `./tp0 <fileTest/test2.txt >fileTest/out2.txt`

- *out2.txt*:
  ```
  2x3 4.000000 13.000000 15.000000 11.000000 32.000000 30.000000
  2x1 1.000000 4.000000
  ```

5. Conclusiones

El desarrollo de este trabajo nos permitió familiarizarnos con el uso del GXemul para lograr levantar una máquina virtual de MIPS y poder trabajar en ella. Aprendimos como crear un túnel entre la máquina virtual del GXemul y una pc mediante el loopback, con el cual pudimos realizar traspasos de archivos de uno a otro. Con esto logramos codificar el programa en la computadora, pasarlo a la maquina virtual, compilarlo en ella y generar asi el código MIPS generado en dicha máquina.

6. Codigo C

6.1. tp0.h

```
#ifndef TP0_H_
#define TP0_H_
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <error.h>
#include <errno.h>
#include <stdbool.h>


#define SIZE_MAT 100


void printHelp();

int  checkArguments(int,char*[]);

int validoLinea(char *linea);


#endif /* TP0_H_ */
```

6.2. tp0.c

```
#include "tp0.h"

extern int errno ;


void printHelp() {
    char *help = "Usage:"
            "\t ./tp0 -h \n"
            "\t ./tp0 -V \n"
            "\t ./tp0 < in_file > out_file \n"
            "Options: \n"
            " -V, --version \n\t Print version and quit. \n"
            " -h, --help \n\t Print this information and quit
                .\n"
            " Examples: \n"
            " ./tp0 < in.txt > out.txt \n"
            " cat in.txt | ./tp0 > out.txt \n";

    printf("%s", help);
}
int checkArguments(int cantidadArgumentos, char* argumentos[])
    {
    int retorno = 1;
    if ((cantidadArgumentos == 2)
```

```
            && ((strcmp(argumentos[1], "-h") == 0)
                    || (strcmp(argumentos[1], "--help") == 0))
                        ) {
        printHelp();
        retorno = 0;
    } else if ((cantidadArgumentos == 2)
            && (((strcmp(argumentos[1], "-V") == 0))
                    || (strcmp(argumentos[1], "--version") ==
                        0) || (strcmp(argumentos[1], "-v") ==
                        0))) {
        printf("Version 1.0\n");
        retorno = 0;
    }

    return retorno;
}

int validoLinea(char *linea){

    int desp =0;
    char fil = *(linea+desp);
    while( fil >='0' && fil <= '9'){
        desp++;
        fil = *(linea+desp);
    }
    if( fil != 'x'){
        fprintf (stderr,"Fila incorrecta. \n");
        exit(1);
    }
    char x = *(linea+desp);
    if(!((x=='x') || (x=='X'))){
        fprintf(stderr,"Formato FxC incorrecto.  \n");
        exit(1);
    }
    desp++;
    char col = *(linea+desp);
    while( col >='0' && col <= '9'){
        desp++;
        col = *(linea+desp);
    }

    int cantNums = 0;
    char car =*(linea+desp);
    if(car == ' '){
        desp++;
        car = *(linea+desp);
    }else{
        fprintf(stderr, "Bad space \n" );
    }

    while(car!= 0 && car != EOF){
        bool point = false;
        while((car != ' ' && car >= '0' && car <= '9') || (car
            == '.')){
```

```
            if(point && car == '.'){
                fprintf(stderr,"Valores incorrectos. \n");
                exit(1);
            }else{
                if(car == '.'){
                    point = true;
                }
                desp++;
                car = *(linea+desp);
            }

        }
        desp++;
        car = *(linea+desp);
        cantNums++;

    }
    return cantNums;

}


 int main(int argc, char *argv[])
{
    if (!checkArguments(argc,argv)){
        return 1;
    }else{

        char *buffer= NULL;

        int read = 0;
        size_t len =0;
        char s;
        int fil=0 ,col=0, i, j,k, fil2=0, col2=0, bytes_now=0;
        int bytes_consumed = 0;
        read = getline(&buffer, &len, stdin);

        if(read == -1){
            //No recibio stream de entrada
            fprintf(stderr,"No stdin \n");
            exit(1);
        }
        while(-1 != read){
            int cantNumsMatriz = validoLinea(buffer);
            bytes_consumed = 0;
            sscanf(buffer+ bytes_consumed, "%d%c%d%n", &fil,&s
                ,&col, & bytes_now);
            bytes_consumed += bytes_now;
            if(cantNumsMatriz!= (fil*col)){

                fprintf(stderr,"Matriz incorrecta. %s \n",
                    buffer);
                exit(1);
            }
```

```
double matriz1[fil][col];
for(i=0; i<fil;i++){
    for(j=0;j<col;j++){
    sscanf(buffer+ bytes_consumed, "%lf%n", &
        matriz1[i][j], & bytes_now);
    bytes_consumed += bytes_now;
    }
}

memset (buffer,0,SIZE_MAT);
read = getline(&buffer, &len, stdin);
cantNumsMatriz = validoLinea(buffer);
bytes_consumed = 0;
sscanf(buffer+ bytes_consumed, "%d%c%d%n", &fil2,&
    s,&col2, & bytes_now);
bytes_consumed += bytes_now;
if(cantNumsMatriz!= (fil2*col2)){
    fprintf(stderr,"Matriz incorrecta. %s", buffer
        );
    exit(1);
}
double matriz2[fil2][col2];
for(i=0; i<fil2;i++){
    for(j=0;j<col2;j++){
        sscanf(buffer+ bytes_consumed, "%lf%n", &
            matriz2[i][j], & bytes_now);
        bytes_consumed += bytes_now;
    }
}
if (col== fil2){

    double sum;
    double out[fil][col2];
    for(i=0;i<fil;i++){ // fila de matriz 1
        for(j=0;j<col2;j++){  // columna de
            segunda matriz
            sum=0;
            for(k=0;k<col;k++)
                sum=sum+matriz1[i][k]*matriz2[k][j
                    ];
                out[i][j]=sum;
        }
    }
    bytes_consumed = 0;

    memset (buffer,0,SIZE_MAT);
    sprintf(buffer+ bytes_consumed,"%dx%d%n\n",
        fil,col2, & bytes_now);
    bytes_consumed += bytes_now;
    for(i=0;i<fil;i++){
        for(j=0;j<col2;j++){
            sprintf(buffer+ bytes_consumed," %lf%n
                ",out[i][j], & bytes_now);
            bytes_consumed += bytes_now;
```

```
                    }
                }
                puts(buffer);
            }else{
                fprintf(stderr,"Matrices incorrectas para la
                    multiplicacion. \n");
                exit(1);
            }
            memset (buffer,0,SIZE_MAT);
            read = getline(&buffer, &len, stdin);

        }

        free(buffer);
        return 0;
    }
}
```

7. Código MIPS32

7.1. tp0.s

```
    .file   1 "tp0.c"
    .section .mdebug.abi32
    .previous
    .abicalls
    .rdata
    .align  2
$LC0:
    .ascii  "Usage:\t ./tp0 -h \n"
    .ascii  "\t ./tp0 -V \n"
    .ascii  "\t ./tp0 < in_file > out_file \n"
    .ascii  "Options: \n"
    .ascii  " -V, --version \n"
    .ascii  "\t Print version and quit. \n"
    .ascii  " -h, --help \n"
    .ascii  "\t Print this information and quit.\n"
    .ascii  " Examples: \n"
    .ascii  "  ./tp0 < in.txt > out.txt \n"
    .ascii  "  cat in.txt | ./tp0 > out.txt \n\000"
    .align  2
$LC1:
    .ascii  "%s\000"
    .text
    .align  2
    .globl  printHelp
    .ent    printHelp
printHelp:
    .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
    la  $v0,$LC0
    sw  $v0,24($fp)
    la  $a0,$LC1
    lw  $a1,24($fp)
    la  $t9,printf
    jal $ra,$t9
    move    $sp,$fp
    lw  $ra,40($sp)
    lw  $fp,36($sp)
    addu    $sp,$sp,48
    j   $ra
```

```
    .end    printHelp
    .size   printHelp, .-printHelp
    .rdata
    .align  2
$LC2:
    .ascii  "-h\000"
    .align  2
$LC3:
    .ascii  "--help\000"
    .align  2
$LC4:
    .ascii  "-V\000"
    .align  2
$LC5:
    .ascii  "--version\000"
    .align  2
$LC6:
    .ascii  "-v\000"
    .align  2
$LC7:
    .ascii  "Version 1.0\n\000"
    .text
    .align  2
    .globl  checkArguments
    .ent    checkArguments
```

Referencias

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

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