

# Organización de Computadoras 66.20

## Trabajo Práctico 1

---

Autor	Padron	Correo electrónico
Quino López, Julián	94224	julian.quino2@gmail.com
Del Carril, Manuel	100772	manueldelcarril@gmail.com
Bobadilla Catalan, German	90123	bobadillagerman@gmail.com



### Facultad de Ingeniería

Universidad de Buenos Aires

Av. Paseo Colón 850 - C1063ACV

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel: +54 (11) 4343-0893 / 4343-0092

<http://www.fi.uba.ar>

# Historial de Revisiones

Fecha	Revisor	Detalle
07/05/2019	-	Entrega del TP

## Resumen

El siguiente trabajo práctico tiene como objetivo familiarizarse con el conjunto de instrucciones MIPS y el concepto de ABI. Para lograr tal propósito se escribe en lenguaje assembly MIPS dos programas que permitan convertir archivos de texto desde Windows hacia UNIX, y viceversa.

## 1 Introducción

Los archivos de texto requieren de un carácter especial (o secuencia de caracteres) para indicar el fin de una línea. La codificación de este varía según el sistema operativo, lo que lleva a la incorrecta visualización de un archivo en un sistema operativo que fue creado en otro. Linux utiliza el salto de línea (`\n`) mientras que Windows utiliza el retorno del carro seguido del salto de línea (`\r\n`).

## 2 Desarrollo

El algoritmo propuesto por el grupo consiste en recorrer caracter por caracter hasta encontrar un `\r`, si el programa usado es `dos2unix` y el siguiente caracter es un `\n` se eliminará el `\r`, si en cambio se usa el programa `unix2dos` se agregará `\r` antes de un `\n`, esto se realiza ya sea desde el archivo o utilizando el stream leído por entrada standard.

Una aclaración importante, si un archivo windows se le ingresa por el programa `unix2dos` no le agregará un `\r` antes de un `\n` por que ya es un archivo de windows.

### 2.1 Implementación

En ambos programas la implementación es similar, simplemente difieren en los caracteres que detectan y los que reemplazan. Por ende, la siguiente descripción servirá para ambos casos. Tenemos cuatro funciones globales: `myRead`, `myWrite`, `processInput` y `main`, cada cual con tareas específicas.

`myRead` se encarga de leer los caracteres del documento y los devuelve, verificando previamente de que no haya errores. Por otro lado, `myWrite` funciona como `myRead`, solo que como su nombre lo indica, escribe un caracter en el documento.

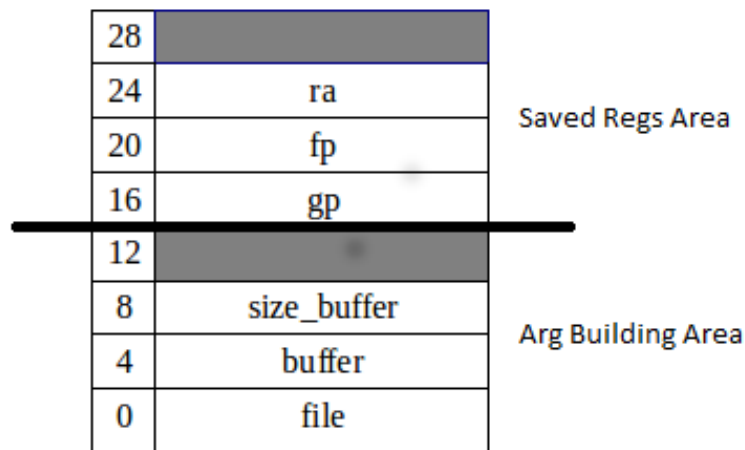


Figure 1: Stack frame de las funciones `myRead` y `myWrite`.

`processInput` utiliza estas dos funciones para procesar el archivo caracter a caracter y escribir el resultado.

44	file_output	Stack del main
40	file_input	
<hr/>		
36		Saved Regs Area
32	ra	
28	fp	
24	gp	
<hr/>		
20		Local Area
16	numero_bytes	
<hr/>		
12		Arg Building Area
8	size_buffer	
4	buffer	
0	archivo	

Figure 2: Stack frame de la funcion processInput.

main es como su nombre indica, la función que inicia el programa y se encarga de recibir los comandos pasados por stdin y actuar en consecuencia, donde en caso de ser necesario llamará a processInput.

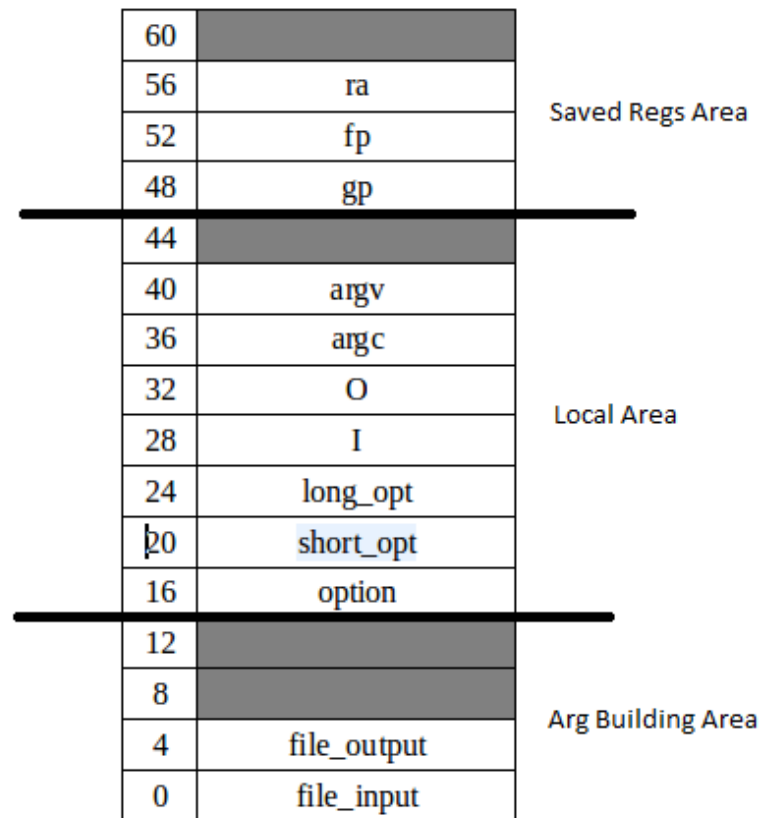


Figure 3: Stack frame de la funcion main.

## 2.2 Comandos para compilar y ejecutar el programa

Se puede compilar los programas con los siguientes comandos:

```
$ gcc unix2dos.S -o unix2dos
$ gcc dos2unix.S -o dos2unix
```

Y luego ejecutarlos con los comandos:

```
$ ./unix2dos -i input.txt -o output.txt
$ ./dos2unix -i input.txt -o output.txt
```

En caso de sólo querer especificar el archivo de entrada, debe ejecutarse, por ejemplo, de la siguiente manera:

```
$ ./unix2dos -i input.txt -o -
$ ./dos2unix -i input.txt -o -
```

Análogamente si se quiere ingresar un archivo de salida:

```
$ ./unix2dos -i - -o output.txt
$ ./dos2unix -i - -o output.txt
```

Es decir que con un gui3n medio indicamos que no se proporcionar3 un archivo para entrada/salida, acorde a lo que indica el enunciado.

## 2.3 Otros comandos

Pueden utilizarse comandos tales como help y version, de la siguiente forma:

```
$ ./unix2dos -h
$ ./dos2unix -h

$ ./unix2dos -V
$ ./dos2unix -V
```

## 2.4 C3digo fuente de unix2dos.S

```
#include <mips/regdef.h>
#include <sys/syscall.h>
#define SALIDA_EXITOSA 0
#define ERROR -1
#define ARCHIVO_NULO -1
#define ENTRADA_ESTANDAR 0
#define SALIDA_ESTANDAR 1

        #int myRead(file,buffer,cantidad)
        .text
        .align 2
        .globl myRead
        .ent myRead
myRead:
        .frame $fp,32,ra
        .set noreorder
        .cpload t9
        .set reorder
        subu sp,sp,32
        .cpstore 16
        sw $fp,20(sp)
        move $fp,sp
        sw ra,24($fp)
        sw a0,0($fp)
        sw a1,4($fp)
        sw a2,8($fp)
        lw a0,0($fp)
        lw a1,4($fp)
        lw a2,8($fp)
        li v0,SYS_read
        syscall
        #vemos si hay errores primero
        bne a3,zero,ErrorEnRead
        bgt zero,v0,ErrorEnRead
        b salidaDeMyRead
ErrorEnRead:
        li v0,SYS_exit
        li a0,ERROR
        syscall
```

```

salidaDeMyRead:
    lw          ra,24($fp)
    lw          $fp,20(sp)
    addu       sp, sp,32
    j          ra
    .end       myRead

#int myWrite(file, buffer,cantidad)
    .text
    .align     2
    .globl     myWrite
    .ent       myWrite
myWrite:
    .frame     $fp,32,ra
    .set       noreorder
    .cpload    t9
    .set       reorder
    subu       sp,sp,32
    .cpstore   16
    sw         $fp,20(sp)
    move       $fp,sp
    sw         ra,24($fp)
    sw         a0,0($fp)
    sw         a1,4($fp)
    sw         a2,8($fp)
    lw         a0,0($fp)
    lw         a1,4($fp)
    lw         a2,8($fp)
    li         v0,SYS_write
    syscall
    #vemos si hay errores primero
    bne        a3, zero, ErrorEnWrite
    bgt        zero,v0, ErrorEnWrite
    b          salidaDeMyWrite
ErrorEnWrite:
    li         v0, SYS_exit
    li         a0, ERROR
    syscall
salidaDeMyWrite:
    lw          ra,24($fp)
    lw          $fp, 20(sp)
    addu       sp, sp, 32
    j          ra
    .end       myWrite

#int processInput(file input, file output)
    .text
    .align     2
    .globl     processInput
    .ent       processInput
processInput:
    .frame     $fp,40,ra
    .set       noreorder
    .cpload    t9

```

```

.set      reorder
subu      sp,sp,40
.cprestore 24
sw         $fp,28(sp)
move      $fp,sp
sw         ra,32($fp)
sw         zero,4($fp)
#int  tamaño=myRead(file,&buffer,cantidad);
lw         a0,40($fp)
sw         a0,0($fp)
addu      a1,$fp,4
li         a2,1
jal        myRead
sw         v0,16($fp)
while:
lw         v0,16($fp)
blez       v0,salidaDeProcessInput
# -if(buffer=='\r')
lw         a0,4($fp)
sll        a0,a0,24
sra        a0,a0,24
li         t0,13
bne        a0,t0,verSiHaySaltoDeLinea
#int  tamaño=myRead(file,&buffer,cantidad);
lw         a0,40($fp)
sw         a0,0($fp)
addu      a1,$fp,4
li         a2,1
jal        myRead
sw         v0,16($fp)
#-if(tamaño == 0){
bgtz       v0,fijarseSaltoLinea
#fprintf(outputFile,"\r")
lw         a0,44($fp)
sw         a0,0($fp)
addu      a1,$fp,4
li         a2,1
jal        myWrite
b          salidaDeProcessInput
fijarseSaltoLinea:
#-if(buffer=='\n')
lw         a0,4($fp)
sll        a0,a0,24
sra        a0,a0,24
li         t0,10
bne        a0,t0,escribirRetornoDeCarro
escribirSaltoYcarro:
#myWrite(file,"\r\n",cantidad);
lw         a0,44($fp)
sw         a0,0($fp)
la         a1,saltoCarro
li         a2,1
jal        myWrite
lw         a0,44($fp)

```



```

        sw            a0,0($fp)
        la            a1,saltoLinea
        li            a2,1
        jal           myWrite
        b             obtenerCaracter
verSiHaySaltoDeLinea:
        #-else if(buffer=='\n')
        lw            a0,4($fp)
        sll           a0,a0,24
        sra           a0,a0,24
        li            t0,10
        bne           a0,t0,escribirElCaracter
        b             escribirSaltoYcarro
escribirRetornoDeCarro:
        #myWrite(file,"\r",cantidad);
        lw            a0,44($fp)
        sw            a0,0($fp)
        la            a1,saltoCarro
        li            a2,1
        jal           myWrite
escribirElCaracter:
        #myWrite(file,&buffer,cantidad);
        lw            a0,44($fp)
        sw            a0,0($fp)
        addu          a1,$fp,4
        li            a2,1
        jal           myWrite
obtenerCaracter:
        #tamano=myRead(&buffer,cantidad,file);
        lw            a0,40($fp)
        sw            a0,0($fp)
        addu          a1,$fp,4
        li            a2,1
        jal           myRead
        sw            v0,16($fp)
        b             while
salidaDeProcessInput:
        li            v0,SALIDA_EXITOSA
        lw            ra,32($fp)
        lw            $fp, 28(sp)
        addu          sp, sp, 40
        j             ra
        .end          processInput

#int main(int argc, char *argv[])
        .text
        .align 2
        .globl main
        .ent main
main:
        .frame $fp,64,ra
        .set noreorder
        .cpload t9
        .set reorder

```

```

subu    sp,sp,64
.cprestore 48
sw      $fp,52(sp)
move    $fp,sp
sw      ra,56($fp)
sw      a0,36($fp)
sw      a1,40($fp)
#int option = 0;
sw      zero,16($fp)
#const char *short_opt = "i:o:hV";
la      t0,short_opt
sw      t0,20($fp)
#struct option long_opt[] =
la      t0,long_opt
sw      t0,24($fp)
#FILE *inputFile = NULL;
#FILE *outputFile = NULL;
li      t0,ARCHIVO_NULO
sw      t0,28($fp)
sw      t0,32($fp)
#while ((option = getopt_long(argc, argv, short_opt,
long_opt, NULL)) != -1)
while_option:
lw      a0,36($fp)
lw      a1,40($fp)
lw      a2,20($fp)
lw      a3,24($fp)
jal     getopt_long
li      t0,-1
beq     t0,v0,salirDeWhile
#case 'V':
case_V:
li      t0,86
bne     t0,v0,case_h
la      a0,imprimir_V
li      a1,132
jal     myWrite
li      v0,SALIDA_EXITOSA
b       salir
case_h:
li      t0,104
bne     t0,v0,case_i
la      a0,imprimir_h
li      a1,247
jal     myWrite
li      v0,SALIDA_EXITOSA
b       salir
case_i:
li      t0,105
bne     t0,v0,case_o
#-if(strcmp(optarg, "-") != 0)
lw      a0,optarg
la      a1,guion
jal     strcmp

```

```

        beq      v0,zero,while_option
        #inputFile = fopen(optarg, "r");
        lw      a0,optarg
        li      a1,0
        li      a2,0
        li      v0,SYS_open
        syscall
        #-if(inputFile == NULL)
        bltz    v0,errorEnArchivoInput
        bnez    a3,errorEnArchivoInput
        sw      v0,28($fp)
        b       while_option
errorEnArchivoInput:
        li      v0,SYS_exit
        li      a0,ERROR
        syscall
case_o:
        li      t0,111
        bne     t0,v0,case_default
        #lo de o
        #-if(strcmp(optarg, "-") != 0)
        lw      a0,optarg
        la      a1,guion
        jal     strcmp
        beq     v0,zero,while_option
        #inputFile = fopen(optarg, "r");
        lw      a0,optarg
        li      a1,1
        li      a2,0
        li      v0,SYS_open
        syscall
        #-if(inputFile == NULL)
        bltz    v0,errorEnArchivoOutput
        bnez    a3,errorEnArchivoOutput
        sw      v0,32($fp)
        b       while_option
errorEnArchivoOutput:
        li      v0,SYS_exit
        li      a0,ERROR
        syscall
case_default:
        li      v0,-1
        lw      ra,12($fp)
        b       salir
salirDeWhile:
        lw      t0,28($fp)
        sw      t0,0($fp)
        lw      t0,32($fp)
        sw      t0,4($fp)
        #-if(inputFile == NULL){
        li      t0,ARCHIVO_NULO
        lw      a0,0($fp)
        bne     t0,a0,procesarOutput
        li      t0,0

```

```

        sw            t0,0($fp)
        move         a0,t0
procesarOutput:
        #-if(inputFile == NULL)
        li           t0,ARCHIVO_NULO
        lw           a1,4($fp)
        bne          t0,a1,procesarArchivo
        li           t0,1
        sw           t0,4($fp)
        move         a1,t0
procesarArchivo:
        #processInput(file input)
        jal          processInput
        #close(input);
        li           t0,ENTRADA_ESTANDAR
        lw           a0,0($fp)
        beq          t0,a0,closeOutput
        li           v0,SYS_close
        syscall
closeOutput:
        #close(output);
        li           t0,SALIDA_ESTANDAR
        lw           a0,4($fp)
        beq          t0,a0,salir
        li           v0,SYS_close
        syscall
salir:
        li           v0,SALIDA_EXITOSA
        lw           ra,56($fp)
        lw           $fp, 52(sp)
        addu         sp, sp, 64
        j            ra
        .end         main

.align 2
version:          .asciz  "version"
.align 2
help:             .asciz  "help"
.align 2
input:            .asciz  "input"
.align 2
output:           .asciz  "output"
        .data
        .align 2
long_opt:
        .word        version
        .word        0
        .word        0
        .word        86
        .word        help
        .word        0
        .word        0
        .word        104
        .word        input

```

```

.word    1
.word    0
.word    105
.word    output
.word    1
.word    0
.word    111
.word    0
.word    0
.word    0
.word    0
.word    0
.align 2
short_opt:      .asciz  "i:o:hV"
saltoLinea:     .asciz  "\n"
saltoCarro:     .asciz  "\r"
imprimir_V:     .asciz  "TP #0 de la materia Organizaci n
                  de Computadoras \n"
                  .asciz  "Alumnos: \n"
                  .asciz  "          Bobadilla Catalan
                  German\n"
                  .asciz  "          Del Carril Manuel \
                  n"
                  .asciz  "          Quino Lopez Julian
                  \n"

.align 2
imprimir_h:     .asciz  "Usage: \n"
                  .asciz  "      ./unix2dos -h\n"
                  .asciz  "      ./unix2dos -V\n"
                  .asciz  "      ./unix2dos [options]\n"
                  .asciz  "Options: \n"
                  .asciz  "      -V, --version  Print version and
                  quit.\n"
                  .asciz  "      -h, --help    Print this
                  information.\n"
                  .asciz  "      -o, --output  Location of the
                  output file.\n"
                  .asciz  "      -i, --input  Location of the
                  input file.\n"

.align 2
guion:          .asciz  "--"

```

## 2.5 Código fuente de dos2unix.S

```

#include <mips/regdef.h>
#include <sys/syscall.h>
#define SALIDA_EXITOSA 0
#define ERROR -1
#define ARCHIVO_NULO -1
#define ENTRADA_ESTANDAR 0
#define SALIDA_ESTANDAR 1

.int myRead(file,buffer,cantidad)
.text
.align 2

```

```

        .globl  myRead
        .ent    myRead
myRead:
        .frame  $fp,32,ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu    sp,sp,32
        .cprestore 16
        sw      $fp,20(sp)
        move    $fp,sp
        sw      ra,24($fp)
        sw      a0,0($fp)
        sw      a1,4($fp)
        sw      a2,8($fp)
        lw      a0,0($fp)
        lw      a1,4($fp)
        lw      a2,8($fp)
        li      v0,SYS_read
        syscall
        #vemos si hay errores primero
        bne     a3, zero, ErrorEnRead
        bgt     zero,v0, ErrorEnRead
        b       salidaDeMyRead
ErrorEnRead:
        li      v0, SYS_exit
        li      a0, ERROR
        syscall
salidaDeMyRead:
        lw      ra,24($fp)
        lw      $fp,20(sp)
        addu    sp, sp,32
        j       ra
        .end    myRead

#int myWrite(file, buffer,cantidad)
        .text
        .align  2
        .globl  myWrite
        .ent    myWrite
myWrite:
        .frame  $fp,32,ra
        .set    noreorder
        .cpload t9
        .set    reorder
        subu    sp,sp,32
        .cprestore 16
        sw      $fp,20(sp)
        move    $fp,sp
        sw      ra,24($fp)
        sw      a0,0($fp)
        sw      a1,4($fp)
        sw      a2,8($fp)
        lw      a0,0($fp)

```

```

        lw            a1,4($fp)
        lw            a2,8($fp)
        li            v0,SYS_write
        syscall
        #vemos si hay errores primero
        bne           a3, zero, ErrorEnWrite
        bgt           zero,v0, ErrorEnWrite
        b             salidaDeMyWrite
ErrorEnWrite:
        li            v0, SYS_exit
        li            a0, ERROR
        syscall
salidaDeMyWrite:
        lw            ra,24($fp)
        lw            $fp, 20(sp)
        addu          sp, sp, 32
        j             ra
        .end          myWrite

#int processInput(input,output)
.text
        .align        2
        .globl        processInput
        .ent           processInput
processInput:
        .frame        $fp,40,ra
        .set           noreorder
        .cpload        t9
        .set           reorder
        subu           sp,sp,40
        .cpstore        24
        sw             $fp,28(sp)
        move           $fp,sp
        sw             ra,32($fp)
        sw             zero,4($fp)
        #int tamano=myRead(file,&buffer,cantidad);
        lw            a0,40($fp)
        sw            a0,0($fp)
        addu           a1,$fp,4
        li            a2,1
        jal            myRead
        sw            v0,16($fp)
while:
        lw            v0,16($fp)
        blez           v0,salidaDeProcessInput
        # -if(buffer=='\r')
        lw            a0,4($fp)
        sll            a0,a0,24
        sra            a0,a0,24
        li            t0,13
        bne           a0,      t0,escribirElCaracter
        #int tamano=myRead(file,&buffer,cantidad);
        lw            a0,40($fp)
        sw            a0,0($fp)

```

```

        addu    a1,$fp,4
        li      a2,1
        jal     myRead
        sw      v0,16($fp)
        #-if(tamano == 0){
        bgtz    v0,fijarseSaltoLinea
#myWrite(file,&buffer,cantidad);
        lw      a0,44($fp)
        sw      a0,0($fp)
        addu    a1,$fp,4
        li      a2,1
        jal     myWrite
        b       salidaDeProcessInput
fijarseSaltoLinea:
        #-if(buffer=='\n')
        lw      a0,4($fp)
        sll     a0,a0,24
        sra     a0,a0,24
        li      t0,10
        bne     a0,t0,escribirRetornoDeCarro
#myWrite(file,"\n",cantidad);
        lw      a0,44($fp)
        sw      a0,0($fp)
        la      a1,saltoLinea
        li      a2,1
        jal     myWrite
        b       obtenerCaracter
escribirRetornoDeCarro:
        #myWrite(file,"\r",cantidad);
        lw      a0,44($fp)
        sw      a0,0($fp)
        la      a1,saltoCarro
        li      a2,1
        jal     myWrite
escribirElCaracter:
        #myWrite(file,&buffer,cantidad);
        lw      a0,44($fp)
        sw      a0,0($fp)
        addu    a1,$fp,4
        li      a2,1
        jal     myWrite
obtenerCaracter:
        #int tamano=myRead(file,&buffer,cantidad);
        lw      a0,40($fp)
        sw      a0,0($fp)
        addu    a1,$fp,4
        li      a2,1
        jal     myRead
        sw      v0,16($fp)
        b       while
salidaDeProcessInput:
        li      v0,SALIDA_EXITOSA
        lw      ra,32($fp)
        lw      $fp,28(sp)

```



```

        addu    sp, sp, 40
        j       ra
    .end       processInput

#int main(int argc, char *argv[])
    .text
    .align 2
    .globl main
    .ent     main
main:
    .frame    $fp,64,ra
    .set      noreorder
    .cpld    t9
    .set      reorder
    subu     sp,sp,64
    .cprestore 48
    sw       $fp,52(sp)
    move     $fp,sp
    sw       ra,56($fp)
    sw       a0,36($fp)
    sw       a1,40($fp)
    #int option = 0;
    sw       zero,16($fp)
    #const char *short_opt = "i:o:hV";
    la       t0,short_opt
    sw       t0,20($fp)
    #struct option long_opt[] =
    la       t0,long_opt
    sw       t0,24($fp)
    #FILE *inputFile = NULL;
    #FILE *outputFile = NULL;
    li       t0,ARCHIVO_NULO
    sw       t0,28($fp)
    sw       t0,32($fp)
    #while ((option = getopt_long(argc, argv, short_opt,
        long_opt, NULL)) != -1)
while_option:
    lw       a0,36($fp)
    lw       a1,40($fp)
    lw       a2,20($fp)
    lw       a3,24($fp)
    jal      getopt_long
    li       t0,-1
    beq      t0,v0,salirDeWhile
case_V:
    li       t0,86
    bne     t0,v0,case_h
    la       a0,imprimir_V
    li       a1,132
    jal      myWrite
    li       v0,SALIDA_EXITOSA
    b       salir
case_h:
    li       t0,104

```

```

        bne            t0,v0,case_i
        la            a0,imprimir_h
        li            a1,247
        jal          myWrite
        li            v0,SALIDA_EXITOSA
        b            salir
case_i:
        li            t0,105
        bne            t0,v0,case_o
        #-if(strcmp(optarg, "-") != 0)
        lw            a0,optarg
        la            a1,guion
        jal          strcmp
        beq            v0,zero,while_option
        #inputFile = fopen(optarg, "r");
        lw            a0,optarg
        li            a1,0
        li            a2,0
        li            v0,SYS_open
        syscall
        #-if(inputFile == NULL)
        bltz          v0,errorEnArchivoInput
        bnez          a3,errorEnArchivoInput
        sw            v0,28($fp)
        b            while_option
errorEnArchivoInput:
        li            v0,SYS_exit
        li            a0,ERROR
        syscall
case_o:
        li            t0,111
        bne            t0,v0,case_default
        #-if(strcmp(optarg, "-") != 0)
        lw            a0,optarg
        la            a1,guion
        jal          strcmp
        beq            v0,zero,while_option
        #inputFile = fopen(optarg, "r");
        lw            a0,optarg
        li            a1,1
        li            a2,0
        li            v0,SYS_open
        syscall
        #-if(inputFile == NULL)
        bltz          v0,errorEnArchivoOutput
        bnez          a3,errorEnArchivoOutput
        sw            v0,32($fp)
        b            while_option
errorEnArchivoOutput:
        li            v0,SYS_exit
        li            a0,ERROR
        syscall
case_default:
        li            v0,-1

```

```

        lw          ra,12($fp)
        b           salir
salirDeWhile:
        lw          t0,28($fp)
        sw          t0,0($fp)
        lw          t0,32($fp)
        sw          t0,4($fp)
        #-if(inputFile == NULL){
        li          t0,ARCHIVO_NULO
        lw          a0,0($fp)
        bne         t0,a0,procesarOutput
        li          t0,0
        sw          t0,0($fp)
        move        a0,t0
procesarOutput:
        #-if(inputFile == NULL)
        li          t0,ARCHIVO_NULO
        lw          a1,4($fp)
        bne         t0,a1,procesarArchivo
        li          t0,1
        sw          t0,4($fp)
        move        a1,t0
procesarArchivo:
        #processInput(file input)
        jal         processInput
        #close(input);
        li          t0,ENTRADA_ESTANDAR
        lw          a0,0($fp)
        beq         t0,a0,closeOutput
        li          v0,SYS_close
        syscall
closeOutput:
        #close(output);
        li          t0,SALIDA_ESTANDAR
        lw          a0,4($fp)
        beq         t0,a0,salir
        li          v0,SYS_close
        syscall
salir:
        li          v0,SALIDA_EXITOSA
        lw          ra,56($fp)
        lw          $fp, 52(sp)
        addu        sp, sp, 64
        j           ra
        .end        main

.align 2
version:          .asciz  "version"
.align 2
help:             .asciz  "help"
.align 2
input:            .asciz  "input"
.align 2
output:           .asciz  "output"

```

```

        .data
        .align 2
long_opt:
        .word    version
        .word    0
        .word    0
        .word    86
        .word    help
        .word    0
        .word    0
        .word    104
        .word    input
        .word    1
        .word    0
        .word    105
        .word    output
        .word    1
        .word    0
        .word    111
        .word    0
        .word    0
        .word    0
        .word    0
        .align 2
short_opt:
        .asciz    "i:o:hV"
saltoLinea:    .asciz    "\n"
saltoCarro:    .asciz    "\r"
imprimir_V:    .asciz    "TP #0 de la materia Organizaci n
                de Computadoras \n"
                .asciz    "Alumnos: \n"
                .asciz    "                Bobadilla Catalan
                German\n"
                .asciz    "                Del Carril Manuel \
                n"
                .asciz    "                Quino Lopez Julian
                \n"
        .align 2
imprimir_h:    .asciz    "Usage: \n"
                .asciz    "        ./dos2unix -h\n"
                .asciz    "        ./dos2unix -V\n"
                .asciz    "        ./dos2unix [options]\n"
                .asciz    "Options: \n"
                .asciz    "        -V, --version    Print version and
                quit.\n"
                .asciz    "        -h, --help        Print this
                information.\n"
                .asciz    "        -o, --output    Location of the
                output file.\n"
                .asciz    "        -i, --input    Location of the
                input file.\n"
        .align 2
guion:    .asciz    "-"

```

### 3 Casos de prueba

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

```
julian@julian-System: ~/Escritorio/Carpeta sin titulo/gxemul-6620-20070927
root@:/66.20/TP0# od -t c casos-de-prueba/entrada/docLinux
0000000 S o l o d e s p u e s d e l
0000020 d e s a s t r e p o d e m o
0000040 s r e s u c i t a r \n S o l o
0000060 d e s p u e s d e h a b e
0000100 r l o p e r d i d o t o d o
0000120 \n d i c e T y l e r \n e r e s
0000140 l i b r e p a r a h a c e
0000160 r c u a l q u i e r c o s a
0000200 \n
0000201
root@:/66.20/TP0# ./unix2dos -i casos-de-prueba/entrada/docLinux -o casos-de-prue
ba/salida/docLinux-aplicarle-unix2dos
root@:/66.20/TP0# od -t c casos-de-prueba/salida/docLinux-aplicarle-unix2dos
0000000 S o l o d e s p u e s d e l
0000020 d e s a s t r e p o d e m o
0000040 s r e s u c i t a r \r \n S o l
0000060 o d e s p u e s d e h a b
0000100 e r l o p e r d i d o t o d
0000120 o \r \n d i c e T y l e r \r \n e
0000140 r e s l i b r e p a r a h
0000160 a c e r c u a l q u i e r c
0000200 o s a \r \n
0000205
root@:/66.20/TP0#
```

Figure 4: Prueba de transformar un archivo UNIX a Windows, utilizando archivo de entrada y salida.

```
julian@julian-System: ~/Escritorio/Carpeta sin titulo/gxemul-6620-20070927
root@:/66.20/TP0# od -t c casos-de-prueba/entrada/docLinux
0000000 S o l o d e s p u e s d e l
0000020 d e s a s t r e p o d e m o
0000040 s r e s u c i t a r \n S o l o
0000060 d e s p u e s d e h a b e
0000100 r l o p e r d i d o t o d o
0000120 \n d i c e T y l e r \n e r e s
0000140 l i b r e p a r a h a c e
0000160 r c u a l q u i e r c o s a
0000200 \n
0000201
root@:/66.20/TP0# ./unix2dos -i casos-de-prueba/entrada/docLinux -o - | od -t c
0000000 S o l o d e s p u e s d e l
0000020 d e s a s t r e p o d e m o
0000040 s r e s u c i t a r \r \n S o l
0000060 o d e s p u e s d e h a b
0000100 e r l o p e r d i d o t o d
0000120 o \r \n d i c e T y l e r \r \n e
0000140 r e s l i b r e p a r a h
0000160 a c e r c u a l q u i e r c
0000200 o s a \r \n
0000205
root@:/66.20/TP0#
```

Figure 5: Prueba de transformar un arhivo UNIX a Windows, utilizando solamente archivo de entrada.

```

julian@julian-System: ~/Escritorio/Carpeta sin título/gxemul-6620-20070927
root@:/66.20/TP0# od -t c casos-de-prueba/entrada/docWin.txt
0000000 S e r B a r r i s t a n a m
0000020 a s u h o n o r \r \n e l g
0000040 r a n m a e s t r e P y c e
0000060 l l e a m a s u c a r g o
0000100 \r \n y M e 361 i q u e a m a
0000120 a M e 361 i q u e
0000131
root@:/66.20/TP0# ./dos2unix -i casos-de-prueba/entrada/docWin.txt -o - | od -t c
0000000 S e r B a r r i s t a n a m
0000020 a s u h o n o r \n e l g r
0000040 a n m a e s t r e P y c e l
0000060 l e a m a s u c a r g o \n
0000100 y M e 361 i q u e a m a a
0000120 M e 361 i q u e
0000127
root@:/66.20/TP0# █

```

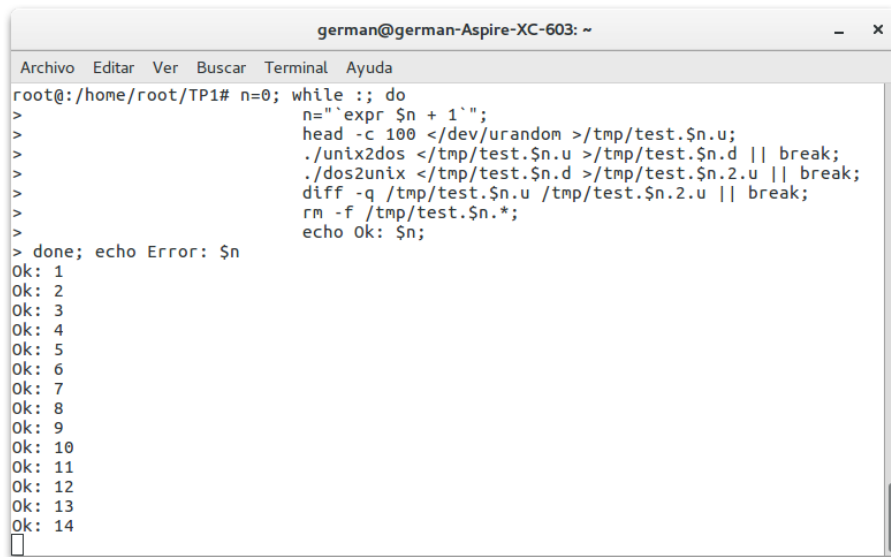
Figure 6: Prueba de transformar un archivo Windows a UNIX, utilizando solo archivo de entrada.

```

julian@julian-System: ~/Escritorio/Carpeta sin título/gxemul-6620-20070927
root@:/66.20/TP0# od -t c casos-de-prueba/entrada/docWin.txt
0000000 S e r B a r r i s t a n a m
0000020 a s u h o n o r \r \n e l g
0000040 r a n m a e s t r e P y c e
0000060 l l e a m a s u c a r g o
0000100 \r \n y M e 361 i q u e a m a
0000120 a M e 361 i q u e
0000131
root@:/66.20/TP0# ./unix2dos -i casos-de-prueba/entrada/docWin.txt -o - | od -t c
0000000 S e r B a r r i s t a n a m
0000020 a s u h o n o r \r \n e l g
0000040 r a n m a e s t r e P y c e
0000060 l l e a m a s u c a r g o
0000100 \r \n y M e 361 i q u e a m a
0000120 a M e 361 i q u e
0000131
root@:/66.20/TP0# █

```

Figure 7: Prueba de transformar un archivo Windows a un archivo Windows, la salida es la misma.



```
german@german-Aspire-XC-603: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@:/home/root/TP1# n=0; while :; do
>     n=`expr $n + 1`;
>     head -c 100 </dev/urandom >/tmp/test.$n.u;
>     ./unix2dos </tmp/test.$n.u >/tmp/test.$n.d || break;
>     ./dos2unix </tmp/test.$n.d >/tmp/test.$n.2.u || break;
>     diff -q /tmp/test.$n.u /tmp/test.$n.2.u || break;
>     rm -f /tmp/test.$n.*;
>     echo Ok: $n;
> done; echo Error: $n
Ok: 1
Ok: 2
Ok: 3
Ok: 4
Ok: 5
Ok: 6
Ok: 7
Ok: 8
Ok: 9
Ok: 10
Ok: 11
Ok: 12
Ok: 13
Ok: 14
```

Figure 8: Prueba del programa que genera secuencias de datos aleatorias.

## 4 Conclusiones

Logramos el objetivo de implementar un conversor de texto capaz de traducir documentos del sistema Unix al de Microsoft y viceversa, que logra pasar todas las pruebas impuestas por la cátedra y las nuestras mediante una implementación que, a nuestros ojos, es limpia y prolija en código Assembly para el sistema MIPS.

Aplicamos los conceptos dados en clase de forma exitosa, tales como: el uso de funciones, branches y otros más complejos como la correcta segmentación de el stack frame (ABA, LTA y SRA) y el llamado a código propio del sistema operativo mediante la operación de syscall.

A grandes rasgos, estamos satisfechos con lo obtenido en el presente trabajo, tanto a nivel proyecto como aprendizaje.

## References

- [1] GetOpt library, [https://www.gnu.org/software/libc/manual/html\\_node/Example-of-Getopt.html](https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html).
- [2] StackOverflow, <https://www.stackoverflow.com>.