

FACULTAD DE INGENIERÍA - U.B.A.

66.20 ORGANIZACIÓN DE COMPUTADORAS - PRÁCTICA MARTES
2DO. CUATRIMESTRE DE 2017

Trabajo práctico N° 0

Infraestructura básica

MATIAS FELD, PADRÓN: 99170
feldmatias@gmail.com

FEDERICO FUNES, PADRÓN: 98472
fedefunes96@gmail.com

AGUSTÍN ZORZANO, PADRÓN: 99224
aguszorza@gmail.com

1. Documentación e implementación

El objetivo del trabajo es realizar un programa en lenguaje C que lea palabras de un archivo (o de entrada estándar) y guarde en otro archivo (mostrar por salida estándar) únicamente aquellas palabras que sean palíndromos. Para ello, dividimos el programa en las siguientes tres funciones:

1. La función principal, `main`, que se encargara de la lógica de leer los parámetros de entrada, el manejo de los archivos, y del bucle principal, que consiste en leer una palabra del archivo de entrada, comprobar si es palíndromo y escribirla en el archivo de salida si corresponde. Si algún archivo no se puede abrir, o no se pasaron correctamente los parámetros, el programa mostrará un mensaje de error en el archivo `stderr` y finalizará con un código de error.
2. La función `leer palabra`, que se encarga de leer una palabra del archivo. Debido a las limitaciones de lo que se considera palabra, y a que no hay limitación con respecto a cantidad de letras de una palabra, lo que hacemos es leer carácter por carácter, guardándolos en un vector alojado en memoria dinámica que se irá redimensionando a medida que sea necesario.
3. Por último, la función `es capicúa`, que se encarga de comprobar si la palabra es o no un palíndromo, y devuelve un valor booleano según corresponda.

2. Comandos para compilación

Para compilar el programa, tanto en Linux como en NetBSD utilizamos el siguiente comando:

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

Para obtener el código MIPS32 generado por el compilador utilizamos el siguiente comando en el sistema operativo NetBSD:

```
$ gcc -Wall -O0 -S -mrnames tp0.c
```

3. Pruebas

Para probar el programa utilizamos un archivo de texto `"entrada.txt"` que contiene un conjunto de palabras con combinaciones de letras, números y guiones y mezclando mayúsculas y minúsculas. Luego tenemos otro archivo, `"resultado.txt"` que es lo que se espera que devuelva el programa al ejecutarse con ese archivo de entrada. Para comprobar el resultado, utilizamos el siguiente comando:

```
$ diff salida.txt resultado.txt
```

Donde si no muestra nada significa que ambos archivos son iguales, y que por lo tanto el programa funciona correctamente.

También realizamos pruebas utilizando salida estándar y entrada estándar, los cuales funcionaron correctamente. Cuando se trabaja con entrada estándar y se desea finalizar se debe ingresar `"ctrl D"`, ya que utilizando `"ctrl C"` finaliza abruptamente y no se guarda correctamente el resultado.

3.1. Archivo 'entrada.txt'

Pruebas varias:

aaa pelota hola como estas

pepep aaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaa
aa

aAAa

-a-a-

-a-a

Neuquen

-Neuquen- neu %q %uen

1234321 ?123?123abc4cba321

Prueba del enunciado:

Somos los primeros en completar el TP 0.

Ojo que La fecha de entrega del TP0 es el martes 12 de septiembre.

Palabras largas mezcladas:

abcdefghijklmnopqrstuvwxyz0123456789_—_9876543210zyxwvutsrqponmlkjihgfedcba??=
 ABCDEFGHIJKLMnopqrstuvwxyz0123456789_—_9876543210zyxwvutsrqponmlkjihgfedcba??=
 Esto es Un Palindromo OM Ordnilap NUSEOTse EStono

Pruebas de guiones guiones bajos:

__—__?????# # # # # \$ \$ \$ \$ _ _ _ @ @ @ @ _ _ _ _ _ ! — — —

Pruebas de palabras de una letra:

a % % % 1 2 ^ 4 ^ — _ C D

b ! @ # \$ % ^ & * () = + \

c

d

Pruebas solo mayusculas:

AAA ABCDEDCBA ABC123—321CBA WXXW

PALINDROMO —ABCB—

3.2. Archivo 'resultado.txt'

aaa

pepep

aaaaaaaaaaaaaaaa

aaaaaaaaaaaaaaaa

aa

aAAa

-a-a-

Neuquen

—Neuquen—

q

1234321

123abc4cba321

Somos

0

Ojo

abcdefghijklmnopqrstuvwxyz0123456789_—_9876543210zyxwvutsrqponmlkjihgfedcba

ABCDEFGHIJKLMnopqrstuvwxyz0123456789_—_9876543210zyxwvutsrqponmlkjihgfedcba

Esto es Un Palindromo OM Ordnilap NUSE OT se

--_--

--_--

--_--_--

a

1

2

4

—

—

C

D

b

c

d

AAA

ABCDEDCBA

ABC123—321CBA

WXXW

4. Código fuente

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <stdbool.h>
4 #include <stdlib.h>
5 #include <ctype.h>
6 #define TAM 10
7
8
9 char * leer_palabra(FILE* archivo, int* longitud){
10     char* palabra = realloc(NULL,TAM);
11     int len = 0;
12     while(true){
13         int c = fgetc(archivo);
14         if((c>=48 && c<=57) || (c>=65 && c<=90) || (c>=97 && c<=122)
            || (c == 95) || (c == 45)){
15             palabra[len] = c;
16             len ++;
17             if (len %TAM == 0){
18                 palabra = realloc(palabra, TAM + len);
19             }
20         }
21     }

```

```

22         palabra[len] = '\0';
23         *longitud = len;
24         return palabra;
25     }
26 }
27
28
29 bool es_capicua(char* palabra, int len){
30     if (len == 0){
31         return false;
32     }
33     int inicio = 0;
34     int final = len - 1;
35     while(inicio < final){
36         if (tolower((unsigned char)palabra[inicio]) != tolower((
            unsigned char)palabra[final])){
37             return false;
38         }
39         inicio++;
40         final--;
41     }
42     return true;
43 }
44
45 int main(int argc, char* argv[]){
46     FILE* entrada = stdin;
47     FILE* salida = stdout;
48     char* parametro;
49
50     int i;
51     for (i = 1; i < argc; i += 2){
52         if (strcmp(argv[i], "-i") == 0){
53             if (i + 1 >= argc){
54                 fputs("Debe_indicar_un_archivo_de_entrada_
                    luego_de_-i\n", stderr);
55                 return 2;
56             }
57             parametro = argv[i + 1];
58             if (strcmp(parametro, "-") != 0){
59                 entrada = fopen(argv[i + 1], "r");
60                 if (!entrada){
61                     fputs("El_archivo_de_entrada_no_
                        pudo_abrirse\n", stderr);
62                     return 1;
63                 }
64             }
65         }
66         else if (strcmp(argv[i], "-o") == 0){
67             if (i + 1 >= argc){
68                 fputs("Debe_indicar_un_archivo_de_salida_
                    luego_de_-o\n", stderr);
69                 return 2;
70             }

```

```

71         parametro = argv[i + 1];
72         if (strcmp(parametro, "-") != 0){
73             salida = fopen(argv[i + 1], "w");
74             if (!salida){
75                 fputs("El_archivo_de_salida_no_pudo
76                     _abrirse\n", stderr);
77                 return 1;
78             }
79         }
80         else if (strcmp(argv[i], "-V") == 0){
81             fprintf(stdout, "TP0_version_1.0001\n");
82             return 0;
83         }
84         else if (strcmp(argv[i], "-h") == 0){
85             fprintf(stdout, "Usage:\n\ntp0_-h\ntp0_-V\ntp0_[
            options]\n\nOptions:\n-V, _--version__Print_
            version_and_quit.\n-h, _--help___Print_this_
            information.\n-i, _--input___Location_of_the_
            input_file.\n-o, _--output___Location_of_the_
            output_file.\n\nExample:\ntp0_-i_~/input_-o_~/
            output\n");
86             return 0;
87         }
88     }
89
90     char* palabra;
91     int len;
92     while(!feof(entrada)){
93         palabra = leer_palabra(entrada, &len);
94         if (es_capicua(palabra, len)){
95             fprintf(salida, "%s\n", palabra);
96         }
97         free (palabra);
98     }
99
100     fclose(entrada);
101     fclose(salida);
102
103     return 0;
104 }

```

5. Codigo MIPS32

```

.file    1 "tp0.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align   2
.globl   leer_palabra
.ent     leer_palabra
leer_palabra:

```

```

        .frame    $fp,56,$ra                # vars= 16, regs= 3/0, args= 16,
            extra= 8
        .mask     0xd0000000,-8
        .fmask    0x00000000,0
        .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
        sw        $a0,56($fp)
        sw        $a1,60($fp)
        move      $a0,$zero
        li        $a1,10                    # 0xa
        la        $t9,realloc
        jal       $ra,$t9
        sw        $v0,24($fp)
        sw        $zero,28($fp)

$L18:
        lw        $a0,56($fp)
        la        $t9,fgetc
        jal       $ra,$t9
        sw        $v0,32($fp)
        lw        $v0,32($fp)
        slt       $v0,$v0,48
        bne       $v0,$zero,$L23
        lw        $v0,32($fp)
        slt       $v0,$v0,58
        bne       $v0,$zero,$L22

$L23:
        lw        $v0,32($fp)
        slt       $v0,$v0,65
        bne       $v0,$zero,$L24
        lw        $v0,32($fp)
        slt       $v0,$v0,91
        bne       $v0,$zero,$L22

$L24:
        lw        $v0,32($fp)
        slt       $v0,$v0,97
        bne       $v0,$zero,$L25
        lw        $v0,32($fp)
        slt       $v0,$v0,123
        bne       $v0,$zero,$L22

$L25:
        lw        $v1,32($fp)
        li        $v0,95                    # 0x5f
        beq       $v1,$v0,$L22
        lw        $v1,32($fp)
        li        $v0,45                    # 0x2d
        beq       $v1,$v0,$L22

```

```

    b        $L21
$L22:
    lw       $v1,24($fp)
    lw       $v0,28($fp)
    addu     $v1,$v1,$v0
    lbu      $v0,32($fp)
    sb       $v0,0($v1)
    lw       $v0,28($fp)
    addu     $v0,$v0,1
    sw       $v0,28($fp)
    lw       $a0,28($fp)
    li       $v0,1717960704          # 0x66660000
    ori      $v0,$v0,0x6667
    mult     $a0,$v0
    mfhi     $v0
    sra      $v1,$v0,2
    sra      $v0,$a0,31
    subu     $v1,$v1,$v0
    move     $v0,$v1
    sll      $v0,$v0,2
    addu     $v0,$v0,$v1
    sll      $v0,$v0,1
    subu     $v0,$a0,$v0
    bne      $v0,$zero,$L18
    lw       $v0,28($fp)
    addu     $v0,$v0,10
    lw       $a0,24($fp)
    move     $a1,$v0
    la       $t9,realloc
    jal      $ra,$t9
    sw       $v0,24($fp)
    b        $L18
$L21:
    lw       $v1,24($fp)
    lw       $v0,28($fp)
    addu     $v0,$v1,$v0
    sb       $zero,0($v0)
    lw       $v1,60($fp)
    lw       $v0,28($fp)
    sw       $v0,0($v1)
    lw       $v0,24($fp)
    move     $sp,$fp
    lw       $ra,48($sp)
    lw       $fp,44($sp)
    addu     $sp,$sp,56
    j        $ra
    .end     leer_palabra
    .size    leer_palabra,.-leer_palabra
    .align   2
    .globl   es_capicua
    .ent     es_capicua
es_capicua:
    .frame   $fp,32,$ra          # vars= 16, regs= 2/0, args= 0,

```



```

        extra= 8
        .mask    0x50000000,-4
        .fmask   0x00000000,0
        .set     noreorder
        .cpload  $t9
        .set     reorder
        subu     $sp,$sp,32
        .cprestore 0
        sw       $fp,28($sp)
        sw       $gp,24($sp)
        move     $fp,$sp
        sw       $a0,32($fp)
        sw       $a1,36($fp)
        lw       $v0,36($fp)
        bne      $v0,$zero,$L29
        sw       $zero,16($fp)
        b        $L28
$L29:
        sw       $zero,8($fp)
        lw       $v0,36($fp)
        addu     $v0,$v0,-1
        sw       $v0,12($fp)
$L30:
        lw       $v0,8($fp)
        lw       $v1,12($fp)
        slt      $v0,$v0,$v1
        bne      $v0,$zero,$L32
        b        $L31
$L32:
        lw       $v1,32($fp)
        lw       $v0,8($fp)
        addu     $v0,$v1,$v0
        lbu      $v0,0($v0)
        sll      $v1,$v0,1
        lw       $v0,_tolower_tab_
        addu     $v0,$v1,$v0
        addu     $a0,$v0,2
        lw       $v1,32($fp)
        lw       $v0,12($fp)
        addu     $v0,$v1,$v0
        lbu      $v0,0($v0)
        sll      $v1,$v0,1
        lw       $v0,_tolower_tab_
        addu     $v0,$v1,$v0
        addu     $v0,$v0,2
        lh       $v1,0($a0)
        lh       $v0,0($v0)
        beq      $v1,$v0,$L33
        sw       $zero,16($fp)
        b        $L28
$L33:
        lw       $v0,8($fp)
        addu     $v0,$v0,1

```

```

        sw      $v0,8($fp)
        lw      $v0,12($fp)
        addu    $v0,$v0,-1
        sw      $v0,12($fp)
        b       $L30
$L31:
        li      $v0,1                      # 0x1
        sw      $v0,16($fp)
$L28:
        lw      $v0,16($fp)
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    es_capicua
        .size   es_capicua, .-es_capicua
        .rdata
        .align  2
$LC0:
        .ascii  "-i\000"
        .align  2
$LC1:
        .ascii  "Debe indicar un archivo de entrada luego de -i\n\000"
        .align  2
$LC2:
        .ascii  "-\000"
        .align  2
$LC3:
        .ascii  "r\000"
        .align  2
$LC4:
        .ascii  "El archivo de entrada no pudo abrirse\n\000"
        .align  2
$LC5:
        .ascii  "-o\000"
        .align  2
$LC6:
        .ascii  "Debe indicar un archivo de salida luego de -o\n\000"
        .align  2
$LC7:
        .ascii  "w\000"
        .align  2
$LC8:
        .ascii  "El archivo de salida no pudo abrirse\n\000"
        .align  2
$LC9:
        .ascii  "-V\000"
        .align  2
$LC10:
        .ascii  "TP0 version 1.0001\n\000"
        .align  2
$LC11:
        .ascii  "-h\000"

```

```

        .align    2
$LC12:
        .ascii    "Usage:\n\n"
        .ascii    "tp0 -h\n"
        .ascii    "tp0 -V\n"
        .ascii    "tp0 [options]\n\n"
        .ascii    "Options:\n"
        .ascii    "-V, --version  Print version and quit.\n"
        .ascii    "-h, --help    Print this information.\n"
        .ascii    "-i, --input   Location of the input file.\n"
        .ascii    "-o, --output   Location of the output file.\n\n"
        .ascii    "Example:\n"
        .ascii    "tp0 -i ~/input -o ~/output\n\000"
        .align    2
$LC13:
        .ascii    "%s\n\000"
        .text
        .align    2
        .globl    main
        .ent      main
main:
        .frame    $fp,72,$ra                # vars= 32, regs= 3/0, args= 16,
            extra= 8
        .mask     0xd0000000,-8
        .fmask     0x00000000,0
        .set       noreorder
        .cpload    $t9
        .set       reorder
        subu       $sp,$sp,72
        .cprestore 16
        sw         $ra,64($sp)
        sw         $fp,60($sp)
        sw         $gp,56($sp)
        move       $fp,$sp
        sw         $a0,72($fp)
        sw         $a1,76($fp)
        la         $v0, __sF
        sw         $v0,24($fp)
        la         $v0, __sF+88
        sw         $v0,28($fp)
        li         $v0,1                    # 0x1
        sw         $v0,36($fp)
$L35:
        lw         $v0,36($fp)
        lw         $v1,72($fp)
        slt        $v0,$v0,$v1
        bne        $v0,$zero,$L38
        b          $L36
$L38:
        lw         $v0,36($fp)
        sll        $v1,$v0,2
        lw         $v0,76($fp)
        addu       $v0,$v1,$v0

```

```

        lw      $a0,0($v0)
        la      $a1,$LC0
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L39
        lw      $v0,36($fp)
        addu    $v1,$v0,1
        lw      $v0,72($fp)
        slt     $v0,$v1,$v0
        bne     $v0,$zero,$L40
        la      $a0,$LC1
        la      $a1, __sF+176
        la      $t9,fputs
        jal     $ra,$t9
        li      $v0,2                      # 0x2
        sw      $v0,48($fp)
        b       $L34
$L40:
        lw      $v0,36($fp)
        sll     $v1,$v0,2
        lw      $v0,76($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        sw      $v0,32($fp)
        lw      $a0,32($fp)
        la      $a1,$LC2
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L37
        lw      $v0,36($fp)
        sll     $v1,$v0,2
        lw      $v0,76($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC3
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        bne     $v0,$zero,$L37
        la      $a0,$LC4
        la      $a1, __sF+176
        la      $t9,fputs
        jal     $ra,$t9
        li      $v0,1                      # 0x1
        sw      $v0,48($fp)
        b       $L34
$L39:
        lw      $v0,36($fp)
        sll     $v1,$v0,2
        lw      $v0,76($fp)

```

```

    addu    $v0,$v1,$v0
    lw      $a0,0($v0)
    la      $a1,$LC5
    la      $t9,strcmp
    jal     $ra,$t9
    bne     $v0,$zero,$L44
    lw      $v0,36($fp)
    addu    $v1,$v0,1
    lw      $v0,72($fp)
    slt     $v0,$v1,$v0
    bne     $v0,$zero,$L45
    la      $a0,$LC6
    la      $a1,__$SF+176
    la      $t9,fputs
    jal     $ra,$t9
    li      $v0,2                                # 0x2
    sw      $v0,48($fp)
    b       $L34
$L45:
    lw      $v0,36($fp)
    sll     $v1,$v0,2
    lw      $v0,76($fp)
    addu    $v0,$v1,$v0
    addu    $v0,$v0,4
    lw      $v0,0($v0)
    sw      $v0,32($fp)
    lw      $a0,32($fp)
    la      $a1,$LC2
    la      $t9,strcmp
    jal     $ra,$t9
    beq     $v0,$zero,$L37
    lw      $v0,36($fp)
    sll     $v1,$v0,2
    lw      $v0,76($fp)
    addu    $v0,$v1,$v0
    addu    $v0,$v0,4
    lw      $a0,0($v0)
    la      $a1,$LC7
    la      $t9,fopen
    jal     $ra,$t9
    sw      $v0,28($fp)
    lw      $v0,28($fp)
    bne     $v0,$zero,$L37
    la      $a0,$LC8
    la      $a1,__$SF+176
    la      $t9,fputs
    jal     $ra,$t9
    li      $v0,1                                # 0x1
    sw      $v0,48($fp)
    b       $L34
$L44:
    lw      $v0,36($fp)
    sll     $v1,$v0,2

```

```

        lw      $v0,76($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC9
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L49
        la      $a0,__$sF+88
        la      $a1,$LC10
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $zero,48($fp)
        b       $L34
$L49:
        lw      $v0,36($fp)
        sll     $v1,$v0,2
        lw      $v0,76($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC11
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L37
        la      $a0,__$sF+88
        la      $a1,$LC12
        la      $t9,fprintf
        jal     $ra,$t9
        sw      $zero,48($fp)
        b       $L34
$L37:
        lw      $v0,36($fp)
        addu    $v0,$v0,2
        sw      $v0,36($fp)
        b       $L35
$L36:
        .set    noreorder
        nop
        .set    reorder
$L52:
        lw      $v0,24($fp)
        lhu     $v0,12($v0)
        srl     $v0,$v0,5
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L54
        b       $L53
$L54:
        addu    $v0,$fp,44
        lw      $a0,24($fp)
        move    $a1,$v0
        la      $t9,leer_palabra
        jal     $ra,$t9
        sw      $v0,40($fp)
        lw      $a0,40($fp)

```

```

        lw      $a1,44($fp)
        la      $t9,es_capicua
        jal     $ra,$t9
        beq     $v0,$zero,$L55
        lw      $a0,28($fp)
        la      $a1,$LC13
        lw      $a2,40($fp)
        la      $t9,fprintf
        jal     $ra,$t9
$L55:
        lw      $a0,40($fp)
        la      $t9,free
        jal     $ra,$t9
        b       $L52
$L53:
        lw      $a0,24($fp)
        la      $t9,fclose
        jal     $ra,$t9
        lw      $a0,28($fp)
        la      $t9,fclose
        jal     $ra,$t9
        sw      $zero,48($fp)
$L34:
        lw      $v0,48($fp)
        move     $sp,$fp
        lw      $ra,64($sp)
        lw      $fp,60($sp)
        addu     $sp,$sp,72
        j       $ra
        .end     main
        .size    main,.-main
        .ident    "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```

66:20 Organización de Computadoras
Trabajo práctico #0: Infraestructura básica
1^{er} cuatrimestre de 2017

\$Date: 2017/08/22 09:15:02 \$

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 6), 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 grupo¹, 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 reciente del sistema operativo NetBSD [2].

En la clase del 15/8 hemos repasado los pasos necesarios para la instalación y configuración del entorno de desarrollo.

¹<http://groups.yahoo.com/group/orga-comp>

5. Programa

Se trata de escribir, en lenguaje C, un programa para procesar archivos de texto por línea de comando: el programa recibirá los archivos o *streams* de entrada y salida, y deberá imprimir aquellas palabras del archivo de entrada (componentes léxicos) que sean palíndromos.

A fin de facilitar el proceso de desarrollo y corrección del TP, definiremos como *palabra* a aquellos componentes léxicos del *stream* de entrada computados exclusivamente por combinaciones de caracteres **a-z**, **0-9**, “-” (signo menos) y “_” (guión bajo). El juego de caracteres utilizado en un stream de entrada válido es ASCII.

A los efectos de la salida, el comportamiento del programa deberá ser *case insensitive*; es decir, la salida permanece alterada ante permutaciones de mayúsculas y minúsculas.

De no recibir los nombres de los archivos (o en caso de recibir - como nombre de archivo) usaremos los *streams* estándar, **stdin** y **stdout**, según corresponda. A continuación, el programa deberá ir leyendo los datos de la entrada, generando la salida correspondiente. De ocurrir errores usaremos **stderr**. Una vez agotados los datos de entrada, el programa debe finalizar adecuadamente, retornando al sistema operativo con un código de finalización adecuado (de tal forma de retornar 0 siempre y cuando el programa finalice normalmente y no hayan ocurrido errores).

5.1. Ejemplos

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

```
$ tp0 -h
Usage:
  tp0 -h
  tp0 -V
  tp0 [options]
Options:
  -V, --version      Print version and quit.
  -h, --help         Print this information.
  -i, --input        Location of the input file.
  -o, --output       Location of the output file.
Examples:
  tp0 -i ~/input -o ~/output
```

Codificamos un archivo vacío (cantidad de bytes nula):

```
$ touch /tmp/zero.txt
$ tp0 -i /tmp/zero.txt -o /tmp/out.txt
$ ls -l /tmp/out.txt
-rw-r--r-- 1 user group 0 2017-03-19 15:14 /tmp/out.txt
```

Leemos un *stream* cuyo único contenido es el carácter ASCII **M**,

```
$ echo Hola M | tp0
M
```

Observar que la salida del programa contiene aquellas palabras de la entrada que sean palíndromos (M en este caso).

Veamos que sucede al procesar archivo de mayor complejidad:

```
$ cat entrada.txt
```

Somos los primeros en completar el TP 0.

Ojo que La fecha de entrega del TP0 es el martes 12 de septiembre.

```
$ tp0 -i entrada.txt -o -
```

Somos

Ojo

6. Informe

El informe deberá incluir al menos las siguientes secciones:

- Documentación relevante al diseño e implementación del programa;
- Comando(s) para compilar el programa;
- Las corridas de prueba, con los comentarios pertinentes;
- El código fuente, en lenguaje C, el cual también deberá entregarse en formato digital compilable (incluyendo archivos de entrada y salida de pruebas);
- El código MIPS32 generado por el compilador;
- Este enunciado.

El informe deberá entregarse en formato impreso y digital.

7. Fechas

- Entrega: 29/8/2017.
- Vencimiento: 12/9/2017.

Referencias

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

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