




## Infraestructura básica

### Trabajo Práctico 0

Apellido y Nombre	Padrón	Correo electrónico
Llauró, Manuel Luis	95736	llauromanuel@gmail.com
Pinto, Santiago	96850	pinto.santiago.augusto@gmail.com
Reimondo, Matias	95899	matiasreimondo@gmail.com

GitHub : <https://github.com/MatiasReimondo/Orga6620TP0>

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Diseño e implementación</b>	<b>2</b>
<b>3. Modo de uso</b>	<b>2</b>
<b>4. Herramientas utilizadas y testing</b>	<b>3</b>
<b>5. Problemas encontrados</b>	<b>3</b>
<b>6. Casos de prueba</b>	<b>3</b>
<b>7. Conclusión y análisis</b>	<b>7</b>
<b>8. Anexo A: Código C</b>	<b>8</b>
8.1. main . . . . .	8
8.2. functions . . . . .	8
<b>9. Anexo B: Código Assembly MIPS</b>	<b>12</b>
9.1. main . . . . .	12
9.2. functions . . . . .	13

## 1. Introducción

El objetivo de este trabajo práctico familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa que tenga la funcionalidad del comando de Unix **"wc"**, que toma como entrada un archivo o stdin, y cuenta las palabras, las líneas y la cantidad de caracteres que contiene.

## 2. Diseño e implementación

Para lograr armar el programa se decidió crear una pequeña librería de funciones, en la cual se escribió la mayor parte de la funcionalidad del programa.

El archivo **main.c** llama a la función **parse\_arguments**, la cual tiene la lógica interna para poder leer los flags pasados al programa y a partir de estos poder ejecutar correctamente la opción pedida, utilizando las funciones creadas, correspondientes.

Una precondition importante a la implementación del programa es que no se podrán procesar archivos que tengan mas caracteres que la capacidad máxima de un **unsigned long long**.

El programa fue desarrollado completamente en entornos GNU/Linux, bajo arquitecturas x86-64, teniendo en cuenta que el programa también debería poder ejecutarse en arquitecturas MIPS. Afortunadamente la simplicidad del programa, y de las herramientas de desarrollo del lenguaje de programación C, permitieron la transición entre arquitecturas sin problemas: el programa se compila y se comporta de la misma manera en ambos casos.

Con la suma de todo esto se logró armar el programa pedido.

## 3. Modo de uso

El ejecutable compilado no tiene dependencias con otros archivos o librerías, y puede moverse y ejecutarse desde cualquier directorio. Al ejecutarse desde una terminal sin argumentos adicionales, leerá de la entrada estándar palabras (es decir, componentes léxicos con caracteres alfanuméricos, y dígitos del 0 al 9), e imprimirá por la salida estándar aquellos que sean palíndromos. Al leer un carácter del final de archivo (EOF), finalizará su ejecución. El programa, adicionalmente, acepta varios parámetros adicionales (todos opcionales):

- **-h**: Muestra en pantalla los parámetros aceptados y finaliza su ejecución
- **-V**: Muestra la versión del programa compilado y finaliza su ejecución
- **-i <archivo>**: Lee la entrada del programa desde el archivo especificado
- **-l**: Imprime la cantidad de lineas del archivo pasado por la entrada del programa
- **-w**: Imprime la cantidad de palabras del archivo pasado por la entrada del programa
- **-c**: Imprime la cantidad de caracteres del archivo pasado por la entrada del programa

El programa puede retornar los siguientes códigos de salida al finalizar la ejecución: -1 cuando hay problemas con la apertura del archivo, -4 cuando se ingresaron argumentos inválidos, -3 cuando el archivo tiene mas caracteres que un **unsigned long long** o 0 si la ejecución fue exitosa.

## 4. Herramientas utilizadas y testing

El funcionamiento correcto del proyecto se sometió a prueba haciendo uso de varias herramientas propias de los entornos Unix-like, principalmente de *bash*, y de las *coreutils* de GNU, para armar un simple script que busque archivos de entrada en un directorio, y compare los resultados con archivos de salida. También se usa como compilador el designado por la cátedra, *gcc*.

## 5. Problemas encontrados

El desarrollo del programa no tuvo mayores inconvenientes, teniendo todos los integrantes experiencia programando en C. Como en todo proyecto, se debe explorar algunas tecnologías en las que uno mismo no está familiarizado, y se tuvo que dedicar un tiempo sustancial al estudio de *bash*, y  $\text{\LaTeX}$ .

## 6. Casos de prueba

Se realizaron para el trabajo práctico 14 casos de pruebas distintos para verificar el correcto funcionamiento del código.

Para correr las pruebas se creó un archivo **run.sh**, el cual corre todas las pruebas que se encuentran en la carpeta de Test.

A continuación se presentan las pruebas que se corren en **run.sh**. Por cuestiones de espacio, no se mostraran los archivos de texto en este informe, pero se pueden ver en el **github** aclarado en la carátula de este informe:

---

```
#!/bin/bash
gcc -Wall functions.c main.c -std=c99 -o tp0
## Algunos codigos a tener en cuenta: 139(Segmentation fault) , 251(
#####TEST 1#####
./tp0 -i wrongfile.txt > dest_test1.txt
if [ "$?" != 255 ]; then ##Pregunto por la salida del programa, el
    codigo 255 implica que el archivo que se intento abrir no existe
    echo "Test archivo inexistente: ERROR";
else
    echo "Test archivo inexistente: OK";
fi
####FIN TEST 1#####

#####TEST 2#####
./tp0 -w -c -l < input_test_2.txt > dest_test2.txt #Redirecciono el
    contenido de input_test_2.txt a la entrada standard
DIFF=$(diff dest_test_2.txt expected_result_test_2.txt)
if [ "$DIFF" != "" ]; then
    echo "Test stdin corto: ERROR";
```

```

else
    echo "Test stdin corto: OK";
fi
####FIN TEST 2#####

#####TEST 3#####
./tp0 -h > dest_test_3.txt
DIFF=$(diff dest_test_3.txt expected_result_test_3.txt)
if [ "$DIFF" != "" ]; then
    echo "Test opcion --help y -h: ERROR";
else
    ./tp0 -help > dest_test_3.txt #Corro el mismo comando pero con el
    otro formato de help
    DIFF=$(diff dest_test_3.txt expected_result_test_3.txt)
    if [ "$DIFF" != "" ]; then
        echo "Test opcion -help y -h: ERROR";
    else
        echo "Test opcion -help y -h : OK";
    fi
fi
####FIN TEST 3#####

#####TEST 4#####
./tp0 -V > dest_test_4.txt
DIFF=$(diff dest_test_4.txt expected_result_test_4.txt)
if [ "$DIFF" != "" ]; then
    echo "Test opcion -version y -V: ERROR";
else
    ./tp0 -Version > dest_test_4.txt #Corro el mismo comando pero con el
    otro formato de version
    DIFF=$(diff dest_test_4.txt expected_result_test_4.txt)
    if [ "$DIFF" != "" ]; then
        echo "Test opcion -version y -V: ERROR";
    else
        echo "Test opcion -version y -V : OK";
    fi
fi
####FIN TEST 4#####

#####TEST 5#####
./tp0 -c -w -l -i input_test_5.txt > dest_test_5.txt
DIFF=$(diff dest_test_5.txt expected_result_test_5.txt)
if [ "$DIFF" != "" ]; then
    echo "Test archivo con un solo caracter: ERROR";
else
    echo "Test archivo con un solo caracter: OK";
fi

```

```

####FIN TEST 5#####

#####TEST 6#####
./tp0 -c -w -l -i input_test_6.txt > dest_test_6.txt
DIFF=$(diff dest_test_6.txt expected_result_test_6.txt)
if [ "$DIFF" != "" ]; then
    echo "Test archivo con delimitadores distintos: ERROR";
else
    echo "Test archivo con delimitadores distintos: OK";
fi
####FIN TEST 6#####

#####TEST 7#####
./tp0 -c -w -l -i input_test_7.txt > dest_test_7.txt
DIFF=$(diff dest_test_7.txt expected_result_test_7.txt)
if [ "$DIFF" != "" ]; then
    echo "Test archivo corto: ERROR";
else
    echo "Test archivo corto: OK";
fi
####FIN TEST 7#####

#####TEST 8#####
./tp0 -c -w -l -i input_test_8.txt > dest_test_8.txt
DIFF=$(diff dest_test_8.txt expected_result_test_8.txt)
if [ "$DIFF" != "" ]; then
    echo "Test archivo vacio: ERROR";
else
    echo "Test archivo vacio: OK";
fi
####FIN TEST 8#####

#####TEST 9#####
./tp0 -c -w -l -i input_test_9.txt > dest_test_9.txt
DIFF=$(diff dest_test_9.txt expected_result_test_9.txt)
if [ "$DIFF" != "" ]; then
    echo "Test archivo mediano: ERROR";
else
    echo "Test archivo mediano: OK";
fi
####FIN TEST 9#####

#####TEST 10#####
./tp0 -c -w -l -i alice.txt > dest_test_10.txt
DIFF=$(diff dest_test_10.txt expected_result_test_10.txt)
if [ "$DIFF" != "" ]; then
    echo "Test alice: ERROR";
else
    echo "Test alice: OK";

```

```

fi
####FIN TEST 10#####

#####TEST 11#####
./tp0 -c -w -l -i beowulf.txt > dest_test_11.txt
DIFF=$(diff dest_test_11.txt expected_result_test_11.txt)
if [ "$DIFF" != "" ]; then
    echo "Test beowulf: ERROR";
else
    echo "Test beowulf: OK";
fi
####FIN TEST 11#####

#####TEST 12#####
./tp0 -c -w -l -i cyclopedia.txt > dest_test_12.txt
DIFF=$(diff dest_test_12.txt expected_result_test_12.txt)
if [ "$DIFF" != "" ]; then
    echo "Test archivo cyclopedia: ERROR";
else
    echo "Test archivo cyclopedia: OK";
fi
####FIN TEST 12#####

#####TEST 13#####
./tp0 -c -w -l -i elquijote.txt > dest_test_13.txt
DIFF=$(diff dest_test_13.txt expected_result_test_13.txt)
if [ "$DIFF" != "" ]; then
    echo "Test El Quijote: ERROR";
else
    echo "Test El Quijote: OK";
fi
####FIN TEST 13#####

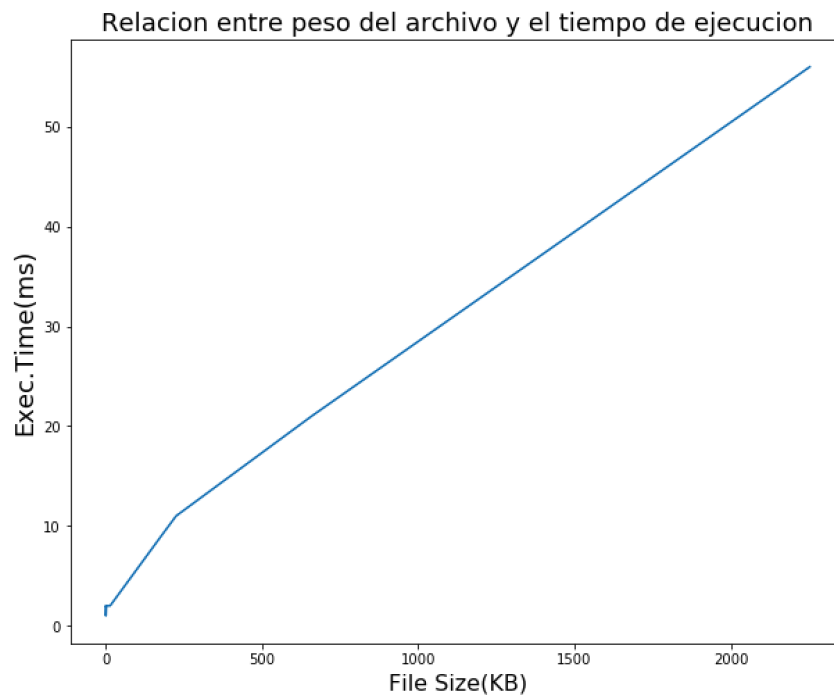
#####TEST 14#####
./tp0 -c -w -l < elquijote.txt > dest_test_14.txt #Cargo todo el quijote
por stdin con '<'
DIFF=$(diff dest_test_14.txt expected_result_test_14.txt)
if [ "$DIFF" != "" ]; then
    echo "Test stdin largo: ERROR";
else
    echo "Test stdin largo: OK";
fi
####FIN TEST 14#####

```

---

## 7. Conclusión y análisis

Finalizada la realización del programa, se procedió con los test y se estudio cuanto crece el tiempo de ejecución en función del tamaño del archivo a procesar. Con este fin, se creo el siguiente gráfico con los resultados obtenidos.



Como era de esperarse, a medida que crece el tamaño del archivo, prácticamente el tiempo de ejecución crece de manera lineal. Esta es una clara señal que para archivos excesivamente grandes, es posible que el algoritmo utilizado no sea el mas eficiente.



## 8. Anexo A: Código C

### 8.1. main

---

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

#define TWO_ARGS 5 //numero de argumentos -i -o
#define ONE_ARG 3 // numero de argumentos -i/-o
#define HELP_VERSION 2
#define NO_ARGS 1 // Sin argumentos

//Bloque main, lo unico qu hace es redirigir al respectivo comportamiento
int main(int argc, char *argv[]) {
    int exe_code = parse_arguments(argc,argv);
    return exe_code;
}
```

---

### 8.2. functions

---

```
#ifndef TPOP_FUNCTIONS_H
#define TPOP_FUNCTIONS_H

#include <stdio.h>
#define ERROR_NO_FILE -1
#define FALSE_ARGS -2
#define COUNTER_OUT_OF_RANGE -3
#define NO_VALID_ARGS -4
#define NO_FILE_OPTION -5

//lista de opciones del programa
static const char OPTIONS[] = "Vhllwci:";

//Parsea los argumentos, abre el archivo y pasa los flags y el file
//pointer correspondiente
int parse_arguments(int argc, char *argv[]);

// Lee el texto y de acuerdo a la flag indicada imprime cantidad de
// lineas, palabras y caracteres
int read_text(int lineFlag, int wordFlag, int charFlag , FILE *fr);

//Muestra la descripcion de la version
int versionDisplay();

//Muestra el menu de ayuda
int helpDisplay();

#endif //TPOP_FUNCTIONS_H
```

---

---

```

#include "functions.h"
#include <ctype.h>
#include <getopt.h>
#include <errno.h>

int parse_arguments(int argc, char *argv[]){
    int option, linesFlag = 0, wordsFlag = 0, charFlag = 0, file_flag = 0;
    FILE *fr;
    int exe_code = 0;
    option = getopt(argc,argv,OPTIONS);
    while(option != -1){
        switch(option){
            case 'V':
                versionDisplay();
                exe_code = 0;
                return exe_code;
            case 'h':
                helpDisplay();
                exe_code = 0;
                return exe_code;
            case 'l':
                linesFlag = 1;
                break;
            case 'w':
                wordsFlag = 1;
                break;
            case 'c':
                charFlag = 1;
                break;
            case 'i':
                file_flag = 1;
                errno = 0;
                fr = fopen(optarg,"r");
                if(errno != 0){
                    printf("No se pudo abrir el archivo \n");
                    exe_code = ERROR_NO_FILE;
                    return exe_code;
                }
                break;
            default:
                printf("No existe el comando \n");
                exe_code = FALSE_ARGS;
                break;
        }
        option = getopt(argc,argv,OPTIONS);
    }
    if(!linesFlag && !charFlag && !wordsFlag){
        printf("Debe ingresar alguna opcion valida. Consultar con
        -h\n");
        exe_code = NO_VALID_ARGS;
        if(file_flag) {
            fclose(fr);
        }
    }
}

```

```

        return exe_code;
    }
    if((exe_code != FALSE_ARGS) && file_flag){
        exe_code = read_text(linesFlag,wordsFlag,charFlag,fr);
    }
    if(!file_flag){
        exe_code = read_text(linesFlag,wordsFlag,charFlag,stdin);
        return exe_code;
    }
    fclose(fr);
    return exe_code;
}

// Explico paso a paso lo que hice asi mas o menos se entiende
int read_text(int lineFlag, int wordFlag, int charFlag, FILE *fr){
    unsigned long long max_range = 18446744073709551600ULL;
    unsigned char input_char;
    //Caracter que se leera del archivo
    unsigned char last_read = ' ';
    //Ultimo caracter leido
    unsigned long long lines_read = 0, words_read = 0, chars_read = 0;
    //Contadores unsigned long long range 0 a
    +18,446,744,073,709,551,615
    int flagfile;
    int in_space = 1;
    while((flagfile = getc(fr)) != EOF){
        input_char = flagfile;
        last_read = input_char;
        chars_read++;
        // Cada vez
        // que lee un caracter aumenta la cantidad de caracteres
        if (isspace(input_char)) {
            // Si el
            // caracter leido es un espacio
            in_space = 1;
            if (input_char == '\n') {
                lines_read++;
            }
        }
        else {
            words_read += in_space;
            // Sino es un
            // espacio aumenta en 1 las palabrras y setea in_sppace en 0
            in_space = 0;
        }

        if(chars_read > max_range){
            // Si se
            // sobrepasa el limite del contador se cierra el programa
            printf("Se ha excedido el rango del contador \n");
            return COUNTER_OUT_OF_RANGE;
        }
    }

    if ((last_read != '\n' )&& (chars_read!= 0)) {
        lines_read++;
    }
}

```

```

    if(lineFlag){
        printf("Lines: %llu \n",lines_read);
    }
    if(wordFlag){
        printf("Words: %llu \n",words_read);
    }
    if(charFlag){
        printf("Characters: %llu \n",chars_read);
    }
    printf("\n");
    return 0;
}

// Completen con sus nombres
int versionDisplay(){
    printf("TP0 - Version 1.0 FIUBA 2018\n");
    printf("Alumnos:\n");
    printf("Pinto, Santiago 96850\n");
    printf("Llauro, Manuel 95736\n");
    printf("Reimondo, Matias 95899\n");
    printf("\n");
    return 0;
}

int helpDisplay(){
    printf("Usage:\n"
           "tp0 -h\n"
           "tp0 -V\n"
           "tp0 [options]\n"
           "Options:\n"
           "-V, --version Print version and quit.\n"
           "-h, --help  Print this information.\n"
           "-l, --lines Print number of lines in file.\n"
           "-w, --words Print number of words in file.\n"
           "-c, --chars Print number of characters in file.\n"
           "-i, --input Path to the input file.\n"
           "Examples:\n"
           "tp0 -w -i input.txt\n");
    printf("\n");
    return 0;
}

```

---

## 9. Anexo B: Código Assembly MIPS

Este código fue generado compilando el programa en la plataforma MIPS con los siguientes parámetros:

```
gcc -Wall
```

### 9.1. main

---

```
.file 1 "main.c"
.section .mdebug.abi32
.previous
.abicalls
.rdata
.align 2
.type OPTIONS, @object
.size OPTIONS, 8
OPTIONS:
.ascii "Vhlwci:\000"
.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)
sw $0,24($fp)
lw $4,48($fp)
lw $5,52($fp)
la $25,parse_arguments
jal $31,$25
sw $2,24($fp)
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)"
```

---

## 9.2. functions

---

```
.file 1 "functions.c"
.section .mdebug.abi32
.previous
.abicalls
.rdata
.align 2
.type OPTIONS, @object
.size OPTIONS, 8
OPTIONS:
.ascii "Vhlwci:\000"
.align 2
$LC0:
.ascii "r\000"
.align 2
$LC1:
.ascii "No se pudo abrir el archivo \n\000"
.align 2
$LC2:
.ascii "No existe el comando \n\000"
.align 2
$LC3:
.ascii "Debe ingresar un archivo con la opcion -i \n\000"
.align 2
$LC4:
.ascii "Debe ingresar alguna opcion validad -l -w -c \n\000"
.text
.align 2
.globl parse_arguments
.ent parse_arguments
parse_arguments:
.frame $fp,80,$31 # vars= 40, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $25
.set reorder
subu $sp,$sp,80
.cprestore 16
sw $31,72($sp)
sw $fp,68($sp)
sw $28,64($sp)
move $fp,$sp
sw $4,80($fp)
sw $5,84($fp)
sw $0,28($fp)
sw $0,32($fp)
sw $0,36($fp)
sw $0,40($fp)
sw $0,48($fp)
lw $4,80($fp)
lw $5,84($fp)
```

```

        la $6,OPTIONS
        la $25,getopt
        jal $31,$25
        sw $2,24($fp)
$L18:
        lw $3,24($fp)
        li $2,-1      # 0xffffffffffffffff
        bne $3,$2,$L20
        b $L19
$L20:
        lw $2,24($fp)
        addu $2,$2,-86
        sw $2,56($fp)
        lw $3,56($fp)
        sltu $2,$3,34
        beq $2,$0,$L29
        lw $2,56($fp)
        sll $3,$2,2
        la $2,$L30
        addu $2,$3,$2
        lw $2,0($2)
        .cpadd $2
        j $2
        .rdata
        .align 2
$L30:
        .gpword $L22
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L26
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L23
        .gpword $L27
        .gpword $L29
        .gpword $L29
        .gpword $L24
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29

```

```

        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L29
        .gpword $L25
        .text
$L22:
        la $25,versionDisplay
        jal $31,$25
        sw $0,48($fp)
        lw $2,48($fp)
        sw $2,52($fp)
        b $L17
$L23:
        la $25,helpDisplay
        jal $31,$25
        sw $0,48($fp)
        lw $2,48($fp)
        sw $2,52($fp)
        b $L17
$L24:
        li $2,1          # 0x1
        sw $2,28($fp)
        b $L21
$L25:
        li $2,1          # 0x1
        sw $2,32($fp)
        b $L21
$L26:
        li $2,1          # 0x1
        sw $2,36($fp)
        b $L21
$L27:
        li $2,1          # 0x1
        sw $2,40($fp)
        la $25,__errno
        jal $31,$25
        sw $0,0($2)
        lw $4,optarg
        la $5,$LC0
        la $25,fopen
        jal $31,$25
        sw $2,44($fp)
        la $25,__errno
        jal $31,$25
        lw $2,0($2)
        beq $2,$0,$L21
        la $4,$LC1
        la $25,printf
        jal $31,$25
        li $2,-1         # 0xffffffffffffffff
        sw $2,48($fp)
        lw $2,48($fp)

```



```

        sw $2,52($fp)
        b $L17
$L29:
        la $4,$LC2
        la $25,printf
        jal $31,$25
        li $2,-2      # 0xfffffffffffffffe
        sw $2,48($fp)
$L21:
        lw $4,80($fp)
        lw $5,84($fp)
        la $6,OPTIONS
        la $25,getopt
        jal $31,$25
        sw $2,24($fp)
        b $L18
$L19:
        lw $2,40($fp)
        bne $2,$0,$L31
        la $4,$LC3
        la $25,printf
        jal $31,$25
        li $2,-5      # 0xfffffffffffffffb
        sw $2,48($fp)
        lw $2,48($fp)
        sw $2,52($fp)
        b $L17
$L31:
        lw $2,28($fp)
        bne $2,$0,$L32
        lw $2,36($fp)
        bne $2,$0,$L32
        lw $2,32($fp)
        bne $2,$0,$L32
        la $4,$LC4
        la $25,printf
        jal $31,$25
        li $2,-4      # 0xfffffffffffffffc
        sw $2,48($fp)
        lw $2,48($fp)
        sw $2,52($fp)
        b $L17
$L32:
        lw $3,48($fp)
        li $2,-2      # 0xfffffffffffffffe
        beq $3,$2,$L33
        lw $4,28($fp)
        lw $5,32($fp)
        lw $6,36($fp)
        lw $7,44($fp)
        la $25,read_text
        jal $31,$25
        sw $2,48($fp)
$L33:

```

```

        lw $4,44($fp)
        la $25,fclose
        jal $31,$25
        lw $2,48($fp)
        sw $2,52($fp)
$L17:
        lw $2,52($fp)
        move $sp,$fp
        lw $31,72($sp)
        lw $fp,68($sp)
        addu $sp,$sp,80
        j $31
        .end parse_arguments
        .size parse_arguments, .-parse_arguments
        .rdata
        .align 2
$L17:
        .ascii "Se ha excedido el rango del contador \n\000"
        .align 2
$L17:
        .ascii "Lines: %llu \n\000"
        .align 2
$L17:
        .ascii "Words: %llu \n\000"
        .align 2
$L17:
        .ascii "Characters: %llu \n\000"
        .align 2
$L17:
        .ascii "\n\000"
        .text
        .align 2
        .globl read_text
        .ent read_text
read_text:
        .frame $fp,96,$31 # vars= 56, regs= 3/0, args= 16, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set noreorder
        .cpld $25
        .set reorder
        subu $sp,$sp,96
        .cprestore 16
        sw $31,88($sp)
        sw $fp,84($sp)
        sw $28,80($sp)
        move $fp,$sp
        sw $4,96($fp)
        sw $5,100($fp)
        sw $6,104($fp)
        sw $7,108($fp)
        li $3,-1
        li $2,-16
        sw $2,24($fp)

```

```

    sw $3,28($fp)
    li $2,32      # 0x20
    sb $2,33($fp)
    move $2,$0
    move $3,$0
    sw $2,40($fp)
    sw $3,44($fp)
    move $8,$0
    move $9,$0
    sw $8,48($fp)
    sw $9,52($fp)
    move $2,$0
    move $3,$0
    sw $2,56($fp)
    sw $3,60($fp)
    li $2,1      # 0x1
    sw $2,68($fp)
$L35:
    lw $3,108($fp)
    lw $2,108($fp)
    lw $2,4($2)
    addu $2,$2,-1
    sw $2,4($3)
    bgez $2,$L38
    lw $4,108($fp)
    la $25,___srget
    jal $31,$25
    sw $2,76($fp)
    b $L39
$L38:
    lw $2,108($fp)
    lw $3,0($2)
    move $4,$3
    lbu $4,0($4)
    sw $4,76($fp)
    addu $3,$3,1
    sw $3,0($2)
$L39:
    lw $3,76($fp)
    sw $3,64($fp)
    li $2,-1     # 0xffffffffffffffff
    bne $3,$2,$L37
    b $L36
$L37:
    lbu $2,64($fp)
    sb $2,32($fp)
    lbu $2,32($fp)
    sb $2,33($fp)
    lw $2,56($fp)
    lw $3,60($fp)
    addu $2,$2,1
    sltu $4,$2,1
    addu $3,$3,$4
    sw $2,56($fp)

```

```

sw $3,60($fp)
lbu $3,32($fp)
lw $2,_ctype_
addu $2,$3,$2
addu $2,$2,1
lbu $2,0($2)
srl $2,$2,3
andi $2,$2,0x1
beq $2,$0,$L40
li $2,1      # 0x1
sw $2,68($fp)
lbu $3,32($fp)
li $2,10     # 0xa
bne $3,$2,$L42
lw $2,40($fp)
lw $3,44($fp)
addu $2,$2,1
sltu $4,$2,1
addu $3,$3,$4
sw $2,40($fp)
sw $3,44($fp)
b $L42
$L40:
lw $2,68($fp)
sra $4,$2,31
lw $2,68($fp)
move $3,$4
lw $4,48($fp)
lw $5,52($fp)
addu $8,$4,$2
sltu $6,$8,$2
addu $9,$5,$3
addu $9,$9,$6
move $2,$8
move $3,$9
sw $2,48($fp)
sw $3,52($fp)
sw $0,68($fp)
$L42:
lw $2,60($fp)
lw $3,28($fp)
sltu $2,$3,$2
bne $2,$0,$L44
lw $3,60($fp)
lw $2,28($fp)
bne $3,$2,$L35
lw $2,56($fp)
lw $3,24($fp)
sltu $2,$3,$2
bne $2,$0,$L44
b $L35
$L44:
la $4,$LC5
la $25,printf

```

```

        jal    $31,$25
        li     $9,-3      # 0xfffffffffffffffd
        sw     $9,72($fp)
        b      $L34
$L36:
        lbu    $3,33($fp)
        li     $2,10      # 0xa
        beq    $3,$2,$L45
        lw     $2,56($fp)
        lw     $3,60($fp)
        or     $2,$2,$3
        beq    $2,$0,$L45
        lw     $2,40($fp)
        lw     $3,44($fp)
        addu   $2,$2,1
        sltu   $4,$2,1
        addu   $3,$3,$4
        sw     $2,40($fp)
        sw     $3,44($fp)
$L45:
        lw     $2,96($fp)
        beq    $2,$0,$L46
        la     $4,$LC6
        lw     $6,40($fp)
        lw     $7,44($fp)
        la     $25,printf
        jal    $31,$25
$L46:
        lw     $2,100($fp)
        beq    $2,$0,$L47
        la     $4,$LC7
        lw     $6,48($fp)
        lw     $7,52($fp)
        la     $25,printf
        jal    $31,$25
$L47:
        lw     $2,104($fp)
        beq    $2,$0,$L48
        la     $4,$LC8
        lw     $6,56($fp)
        lw     $7,60($fp)
        la     $25,printf
        jal    $31,$25
$L48:
        la     $4,$LC9
        la     $25,printf
        jal    $31,$25
        sw     $0,72($fp)
$L34:
        lw     $2,72($fp)
        move   $sp,$fp
        lw     $31,88($sp)
        lw     $fp,84($sp)
        addu   $sp,$sp,96

```

```

j $31
.end read_text
.size read_text, .-read_text
.rdata
.align 2
$LC10:
.ascii "TPO - Version 1.0 FIUBA 2018\n\000"
.align 2
$LC11:
.ascii "Alumnos:\n\000"
.align 2
$LC12:
.ascii "Pinto, Santiago 96850\n\000"
.align 2
$LC13:
.ascii "Llauro, Manuel 95736\n\000"
.align 2
$LC14:
.ascii "Reimondo, Matias 95899\n\000"
.text
.align 2
.globl versionDisplay
.ent versionDisplay
versionDisplay:
.frame $fp,40,$31 # vars= 0, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $25
.set reorder
subu $sp,$sp,40
.cprestore 16
sw $31,32($sp)
sw $fp,28($sp)
sw $28,24($sp)
move $fp,$sp
la $4,$LC10
la $25,printf
jal $31,$25
la $4,$LC11
la $25,printf
jal $31,$25
la $4,$LC12
la $25,printf
jal $31,$25
la $4,$LC13
la $25,printf
jal $31,$25
la $4,$LC14
la $25,printf
jal $31,$25
la $4,$LC9
la $25,printf
jal $31,$25

```

```

move $2,$0
move $sp,$fp
lw $31,32($sp)
lw $fp,28($sp)
addu $sp,$sp,40
j $31
.end versionDisplay
.size versionDisplay, .-versionDisplay
.rdata
.align 2
$LC15:
.ascii "Usage:\n"
.ascii "tp0 -h\n"
.ascii "tp0 -V\n"
.ascii "tp0 [options]\n"
.ascii "Options:\n"
.ascii "-V, --version Print version and quit.\n"
.ascii "-h, --help Print this information.\n"
.ascii "-l, --lines Print number of lines in file.\n"
.ascii "-w, --words Print number of words in file.\n"
.ascii "-c, --chars Print number of characters in file.\n"
.ascii "-i, --input Path to the input file.\n"
.ascii "Examples:\n"
.ascii "tp0 -w -i input.txt\n\000"
.text
.align 2
.globl helpDisplay
.ent helpDisplay
helpDisplay:
.frame $fp,40,$31 # vars= 0, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $25
.set reorder
subu $sp,$sp,40
.cprestore 16
sw $31,32($sp)
sw $fp,28($sp)
sw $28,24($sp)
move $fp,$sp
la $4,$LC15
la $25,printf
jal $31,$25
la $4,$LC9
la $25,printf
jal $31,$25
move $2,$0
move $sp,$fp
lw $31,32($sp)
lw $fp,28($sp)
addu $sp,$sp,40
j $31
.end helpDisplay

```

```
.size helpDisplay, .-helpDisplay  
.ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

---