




Infraestructura básica

Trabajo Práctico 0

Apellido y Nombre	Padrón	Correo electrónico
Blanco, Sebastian	98539	sebastian.e.blanco@gmail.com
Lavandeira, Lucas	98042	lucaslavandeira@gmail.com
Llauró, Manuel Luis	95736	llauromanuel@gmail.com

GitHub : <https://github.com/lucaslavandeira/palindrome>

Índice

1. Introducción	2
2. Diseño e implementación	2
3. Modo de uso	2
4. Herramientas utilizadas y testing	3
5. Problemas encontrados	3
6. Anexo A: Código C	4
7. Anexo B: Código Assembly MIPS	8

1. Introducción

El objetivo de este trabajo práctico familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando 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.

2. Diseño e implementación

Para lograr armar el programa se decidió utilizar el método en programación conocido como "divide y conquistarás", dividiendo el problema en una suma de problemas mas chicos que son resueltos por las distintas funciones creadas.

Primero se guardó en un array de char los caracteres que fueron pedidos en el enunciado que formen las palabras, considerando el resto de los caracteres separadores entre palabras. Y junto con este array se creó una función correspondiente la cual contesta si un caracter dado pertenece a una palabra o si es un separador de palabras.

Por otro lado se creó una función para saber si una palabra dada es o no un palíndromo, junto a otra función encargada de leer la palabra del archivo de entrada. Lo siguiente fue tratar el comportamiento esperado para el modo de uso del programa, como el modo de compilación y las entradas esperadas.

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
- **-o <archivo>**: Imprime la salida del programa al archivo especificado

El programa tiene dos códigos de salida: 0 en funcionamiento correcto, y 1 en caso de error, causado por la lectura inválida de un archivo de entrada, o escritura inválida del archivo de salida.

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 (tanto la escritura de un archivo del parámetro `-o` como de la salida estándar) con archivos de salida. Para facilitar la compilación del programa (y del informe) se utilizó un simple Makefile. 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 *L^AT_EX*. La compilación del programa no es ejecutado con la mayor rigurosidad de advertencias debido a la detección de algunas cuando se compila en la plataforma MIPS. Sin embargo, el programa resulta funcionar de la misma manera: todos los casos de prueba pasan correctamente.

6. Anexo A: Código C

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>

//-----
// DEFINITIONS
//-----

#define VERSION "0.1"
const char help_str[] = "Usage:\n"
    " tp0 -h\n"
    " tp0 -V\n"
    " tp0 [options]\n"
    "Options:\n"
    " -V, --version\tPrint version and quit.\n"
    " -h, --help\tPrint this information.\n"
    " -i, --input\tLocation of the input file.\n"
    " -o, --output\tLocation of the output file.\n"
    "Examples:\n"
    " tp0 -i ~/input -o\n";

#define SPACE_SIZE 65
#define SPACE_INDEX 123
#define EMPTY -1
const char* ENTER = "\n";
char space[SPACE_SIZE];
int spaceIndex[SPACE_INDEX];

//-----
// CHARGE SPACE
//-----
// Del 97 al 122 estan las letras de a-z
// Del 65 al 90 estan las letras de A-Z
// Del 48 al 57 estan los numeros de 0-9
// '-' es 45
// '_' es 95
void chargeSpace() {
    int pos = 0;
    for (int i = 0; i < SPACE_INDEX; i++) spaceIndex[i] = EMPTY;
    //-----
    for (int i = 97; i <= 122; i++) {
        space[pos] = (char)i;
        spaceIndex[i] = pos;
        pos++;
    }
    //-----
    for (int i = 65; i <= 90; i++) {
        space[pos] = (char)i;
        spaceIndex[i] = pos;
        pos++;
    }
    //-----
    for (int i = 48; i <= 57; i++) {
        space[pos] = (char)i;
```

```

        spaceIndex[i] = pos;
        pos++;
    }
    //-----
    // incluyo el guion medio
    pos++;
    space[pos] = '-';
    spaceIndex[45] = pos;
    //-----
    // incluyo el guion bajo
    pos++;
    space[pos] = '_';
    spaceIndex[95] = pos;
}
//-----
// BELONGS TO SPACE
//-----
bool belongsToSpace(int aChar) {
    if (aChar >= SPACE_INDEX) return false;
    return spaceIndex[aChar] != EMPTY;
}
//-----
// IS CAPICUA
//-----
bool isCapicua(char* word, size_t size) {
    if (size == 0) return false;
    if (size == 1) return true;
    size_t rightPos = 0, leftPos = size-1;
    while (rightPos < leftPos) {
        int a = tolower(word[rightPos]);
        int b = tolower(word[leftPos]);
        if (a != b) return false;
        rightPos++;
        leftPos--;
    }
    return true;
}
//-----
// READ FILE
//-----
void readFile(FILE* archIn, FILE* archOut) {
    fseek(archIn, 0, SEEK_SET);
    fseek(archOut, 0, SEEK_SET);
    int c = getc(archIn);
    size_t size = 0;
    char word[100];
    while (c != EOF) {
        if (belongsToSpace(c)) {
            word[size] = (char)c;
            size++;
        } else {
            if (isCapicua(word, size)) {
                fwrite(word, sizeof(char)*size, 1, archOut);
                fwrite(ENTER, sizeof(char)*1, 1, archOut);
            }
        }
    }
}

```

```

    }
    size = 0;
}
c = getc(archIn);
}
}
//-----
// Parsea los argumentos
//-----
void arg_parse(int argc, char** argv, FILE** descriptors, int*
clean_exit) {
    int arg = 1;
    const char flags[] = {'i', 'o', 'V', 'h'};
    char flag = 0;
    while (arg < argc) {
        if (!flag && argv[arg][0] == '-') {
            for (int i = 0; i < strlen(flags); i++) {
                if (argv[arg][1] == flags[i]) {
                    flag = argv[arg][1];
                    break;
                }
            }

            if (flag == 'h') {
                printf("%s\n", help_str);
                *clean_exit = 1;
                return;
            }
            if (flag == 'V') {
                printf("tp0: version %s\n", VERSION);
                *clean_exit = 1;
                return;
            }
            if (!flag) {
                printf("Invalid argument: %s", argv[arg]);
                descriptors[0] = NULL;
                return;
            }
        } else {
            if (flag == 'i') {
                descriptors[0] = fopen(argv[arg], "r");
            } else if (flag == 'o') {
                descriptors[1] = fopen(argv[arg], "w");
            }
            flag = 0;
        }
        arg++;
    }
}
//-----
// MAIN
//-----
int main(int argc, char** argv) {
    FILE* fdescriptors[2] = {stdin, stdout};

```

```
int clean_exit = 0;
arg_parse(argc, argv, fdescriptors, &clean_exit);
if (clean_exit) return 0; // finalizacion limpia
if (!fdescriptors[0] || !fdescriptors[1]) return 1;

chargeSpace();
readFile(fdescriptors[0], fdescriptors[1]);
if (fdescriptors[0] != stdin) fclose(fdescriptors[0]);
if (fdescriptors[1] != stdout) fclose(fdescriptors[1]);
return 0;
}
//-----
```

7. Anexo B: Código Assembly MIPS

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

```
gcc -std=c99 -o0 -mrnames
```

Los parámetros `o0` y `mrnames` hacen que el código producido no tenga optimizaciones algunas, y reemplaza los nombres de los registros numéricos por los nombres convencionales usados en manuales de la arquitectura, respectivamente.

```
.file 1 "tp0.c"
.section .mdebug.abi32
.previous
.abicalls
.globl help_str
.rdata
.align 2
.type help_str, @object
.size help_str, 244
help_str:
.ascii "Usage:\n"
.ascii "  tp0 -h\n"
.ascii "  tp0 -V\n"
.ascii "  tp0 [options]\n"
.ascii "Options:\n"
.ascii "  -V, --version\tPrint version and quit.\n"
.ascii "  -h, --help\tPrint this information.\n"
.ascii "  -i, --input\tLocation of the input file.\n"
.ascii "  -o, --output\tLocation of the output file.\n"
.ascii "Examples:\n"
.ascii "  tp0 -i ~/input -o\n\000"
.align 2
$LC0:
.ascii "\n\000"
.globl ENTER
.data
.align 2
.type ENTER, @object
.size ENTER, 4
ENTER:
.word $LC0
.text
.align 2
.globl chargeSpace
.ent chargeSpace
chargeSpace:
.frame $fp,24,$ra # vars= 8, regs= 2/0, args= 0, extra= 8
.mask 0x50000000,-4
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,24
.cprestore 0
sw $fp,20($sp)
```

```

        sw $gp,16($sp)
        move $fp,$sp
        sw $zero,8($fp)
        sw $zero,12($fp)
$L6:
        lw $v0,12($fp)
        slt $v0,$v0,123
        bne $v0,$zero,$L9
        b $L7
$L9:
        lw $v0,12($fp)
        sll $v1,$v0,2
        la $v0,spaceIndex
        addu $v1,$v1,$v0
        li $v0,-1          # 0xffffffffffffffff
        sw $v0,0($v1)
        lw $v0,12($fp)
        addu $v0,$v0,1
        sw $v0,12($fp)
        b $L6
$L7:
        li $v0,97          # 0x61
        sw $v0,12($fp)
$L10:
        lw $v0,12($fp)
        slt $v0,$v0,123
        bne $v0,$zero,$L13
        b $L11
$L13:
        lw $v1,8($fp)
        la $v0,space
        addu $v1,$v1,$v0
        lbu $v0,12($fp)
        sb $v0,0($v1)
        lw $v0,12($fp)
        sll $v1,$v0,2
        la $v0,spaceIndex
        addu $v1,$v1,$v0
        lw $v0,8($fp)
        sw $v0,0($v1)
        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 $L10
$L11:
        li $v0,65          # 0x41
        sw $v0,12($fp)
$L14:
        lw $v0,12($fp)
        slt $v0,$v0,91
        bne $v0,$zero,$L17

```

```

        b $L15
$L17:
        lw $v1,8($fp)
        la $v0,space
        addu $v1,$v1,$v0
        lbu $v0,12($fp)
        sb $v0,0($v1)
        lw $v0,12($fp)
        sll $v1,$v0,2
        la $v0,spaceIndex
        addu $v1,$v1,$v0
        lw $v0,8($fp)
        sw $v0,0($v1)
        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 $L14
$L15:
        li $v0,48          # 0x30
        sw $v0,12($fp)
$L18:
        lw $v0,12($fp)
        slt $v0,$v0,58
        bne $v0,$zero,$L21
        b $L19
$L21:
        lw $v1,8($fp)
        la $v0,space
        addu $v1,$v1,$v0
        lbu $v0,12($fp)
        sb $v0,0($v1)
        lw $v0,12($fp)
        sll $v1,$v0,2
        la $v0,spaceIndex
        addu $v1,$v1,$v0
        lw $v0,8($fp)
        sw $v0,0($v1)
        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 $L18
$L19:
        lw $v0,8($fp)
        addu $v0,$v0,1
        sw $v0,8($fp)
        lw $v1,8($fp)
        la $v0,space
        addu $v1,$v1,$v0

```

```

    li $v0,45          # 0x2d
    sb $v0,0($v1)
    lw $v0,8($fp)
    sw $v0,spaceIndex+180
    lw $v0,8($fp)
    addu $v0,$v0,1
    sw $v0,8($fp)
    lw $v1,8($fp)
    la $v0,space
    addu $v1,$v1,$v0
    li $v0,95          # 0x5f
    sb $v0,0($v1)
    lw $v0,8($fp)
    sw $v0,spaceIndex+380
    move $sp,$fp
    lw $fp,20($sp)
    addu $sp,$sp,24
    j $ra
    .end chargeSpace
    .size chargeSpace, .-chargeSpace
    .align 2
    .globl belongsToSpace
    .ent belongsToSpace
belongsToSpace:
    .frame $fp,24,$ra   # vars= 8, regs= 2/0, args= 0, extra= 8
    .mask 0x50000000,-4
    .fmask 0x00000000,0
    .set noreorder
    .cpld $t9
    .set reorder
    subu $sp,$sp,24
    .cpstore 0
    sw $fp,20($sp)
    sw $gp,16($sp)
    move $fp,$sp
    sw $a0,24($fp)
    lw $v0,24($fp)
    slt $v0,$v0,123
    bne $v0,$zero,$L23
    sw $zero,8($fp)
    b $L22
$L23:
    lw $v0,24($fp)
    sll $v1,$v0,2
    la $v0,spaceIndex
    addu $v0,$v1,$v0
    lw $v1,0($v0)
    li $v0,-1          # 0xffffffffffffffff
    xor $v0,$v1,$v0
    sltu $v0,$zero,$v0
    sw $v0,8($fp)
$L22:
    lw $v0,8($fp)
    move $sp,$fp

```

```

    lw $fp,20($sp)
    addu $sp,$sp,24
    j $ra
    .end belongsToSpace
    .size belongsToSpace, .-belongsToSpace
    .align 2
    .globl isCapicua
    .ent isCapicua
isCapicua:
    .frame $fp,40,$ra    # vars= 24, regs= 2/0, args= 0, extra= 8
    .mask 0x50000000,-4
    .fmask 0x00000000,0
    .set noreorder
    .cplod $t9
    .set reorder
    subu $sp,$sp,40
    .cpstore 0
    sw $fp,36($sp)
    sw $gp,32($sp)
    move $fp,$sp
    sw $a0,40($fp)
    sw $a1,44($fp)
    lw $v0,44($fp)
    bne $v0,$zero,$L25
    sw $zero,24($fp)
    b $L24
$L25:
    lw $v1,44($fp)
    li $v0,1    # 0x1
    bne $v1,$v0,$L26
    li $v0,1    # 0x1
    sw $v0,24($fp)
    b $L24
$L26:
    sw $zero,8($fp)
    lw $v0,44($fp)
    addu $v0,$v0,-1
    sw $v0,12($fp)
$L27:
    lw $v0,8($fp)
    lw $v1,12($fp)
    sltu $v0,$v0,$v1
    bne $v0,$zero,$L29
    b $L28
$L29:
    lw $v1,40($fp)
    lw $v0,8($fp)
    addu $v0,$v1,$v0
    lb $v0,0($v0)
    sll $v1,$v0,1
    lw $v0,_tolower_tab_
    addu $v0,$v1,$v0
    addu $v0,$v0,2
    lh $v0,0($v0)

```

```

        sw $v0,16($fp)
        lw $v1,40($fp)
        lw $v0,12($fp)
        addu $v0,$v1,$v0
        lb $v0,0($v0)
        sll $v1,$v0,1
        lw $v0,_tolower_tab_
        addu $v0,$v1,$v0
        addu $v0,$v0,2
        lh $v0,0($v0)
        sw $v0,20($fp)
        lw $v1,16($fp)
        lw $v0,20($fp)
        beq $v1,$v0,$L30
        sw $zero,24($fp)
        b $L24
$L30:
        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 $L27
$L28:
        li $v0,1      # 0x1
        sw $v0,24($fp)
$L24:
        lw $v0,24($fp)
        move $sp,$fp
        lw $fp,36($sp)
        addu $sp,$sp,40
        j $ra
        .end isCapicua
        .size isCapicua,.-isCapicua
        .align 2
        .globl readFile
        .ent readFile
readFile:
        .frame $fp,160,$ra # vars= 120, regs= 3/0, args= 16, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set noreorder
        .cpld $t9
        .set reorder
        subu $sp,$sp,160
        .cprestore 16
        sw $ra,152($sp)
        sw $fp,148($sp)
        sw $gp,144($sp)
        move $fp,$sp
        sw $a0,160($fp)
        sw $a1,164($fp)
        lw $a0,160($fp)

```

```

    move $a1,$zero
    move $a2,$zero
    la $t9,fseek
    jal $ra,$t9
    lw $a0,164($fp)
    move $a1,$zero
    move $a2,$zero
    la $t9,fseek
    jal $ra,$t9
    lw $v1,160($fp)
    lw $v0,160($fp)
    lw $v0,4($v0)
    addu $v0,$v0,-1
    sw $v0,4($v1)
    bgez $v0,$L32
    lw $a0,160($fp)
    la $t9, __srget
    jal $ra,$t9
    sw $v0,136($fp)
    b $L33
$L32:
    lw $v0,160($fp)
    lw $v1,0($v0)
    move $a0,$v1
    lbu $a0,0($a0)
    sw $a0,136($fp)
    addu $v1,$v1,1
    sw $v1,0($v0)
$L33:
    lw $v0,136($fp)
    sw $v0,24($fp)
    sw $zero,28($fp)
$L34:
    lw $v1,24($fp)
    li $v0,-1 # 0xffffffffffffffff
    bne $v1,$v0,$L36
    b $L31
$L36:
    lw $a0,24($fp)
    la $t9,belongsToSpace
    jal $ra,$t9
    beq $v0,$zero,$L37
    addu $v1,$fp,32
    lw $v0,28($fp)
    addu $v1,$v1,$v0
    lbu $v0,24($fp)
    sb $v0,0($v1)
    lw $v0,28($fp)
    addu $v0,$v0,1
    sw $v0,28($fp)
    b $L38
$L37:
    addu $v0,$fp,32
    move $a0,$v0

```

```

    lw $a1,28($fp)
    la $t9,isCapicua
    jal $ra,$t9
    beq $v0,$zero,$L39
    addu $v0,$fp,32
    move $a0,$v0
    lw $a1,28($fp)
    li $a2,1      # 0x1
    lw $a3,164($fp)
    la $t9,fwrite
    jal $ra,$t9
    lw $a0,ENTER
    li $a1,1      # 0x1
    li $a2,1      # 0x1
    lw $a3,164($fp)
    la $t9,fwrite
    jal $ra,$t9
$L39:
    sw $zero,28($fp)
$L38:
    lw $v1,160($fp)
    lw $v0,160($fp)
    lw $v0,4($v0)
    addu $v0,$v0,-1
    sw $v0,4($v1)
    bgez $v0,$L40
    lw $a0,160($fp)
    la $t9,___srget
    jal $ra,$t9
    sw $v0,140($fp)
    b $L41
$L40:
    lw $v0,160($fp)
    lw $v1,0($v0)
    move $a0,$v1
    lbu $a0,0($a0)
    sw $a0,140($fp)
    addu $v1,$v1,1
    sw $v1,0($v0)
$L41:
    lw $v0,140($fp)
    sw $v0,24($fp)
    b $L34
$L31:
    move $sp,$fp
    lw $ra,152($sp)
    lw $fp,148($sp)
    addu $sp,$sp,160
    j $ra
    .end readFile
    .size readFile, .-readFile
    .rdata
    .align 2
$LC1:

```



```

        .byte 105
        .byte 111
        .byte 86
        .byte 104
        .align 2
$LC2:
        .ascii "%s\n\000"
        .align 2
$LC3:
        .ascii "tp0: version %s\n\000"
        .align 2
$LC4:
        .ascii "dev\000"
        .align 2
$LC5:
        .ascii "Invalid argument: %s\000"
        .align 2
$LC6:
        .ascii "r\000"
        .align 2
$LC7:
        .ascii "w\000"
        .text
        .align 2
        .globl arg_parse
        .ent arg_parse
arg_parse:
        .frame $fp,64,$ra    # vars= 24, regs= 3/0, args= 16, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set noreorder
        .cpld $t9
        .set reorder
        subu $sp,$sp,64
        .cprestore 16
        sw $ra,56($sp)
        sw $fp,52($sp)
        sw $gp,48($sp)
        move $fp,$sp
        sw $a0,64($fp)
        sw $a1,68($fp)
        sw $a2,72($fp)
        sw $a3,76($fp)
        li $v0,1           # 0x1
        sw $v0,24($fp)
        addu $v1,$fp,32
        la $v0,$LC1
        lwl  $a0,3($v0)
        lwr  $a0,0($v0)
        swl  $a0,3($v1)
        swr  $a0,0($v1)
        sb $zero,40($fp)
$L43:
        lw $v0,24($fp)

```

```

        lw $v1,64($fp)
        slt $v0,$v0,$v1
        bne $v0,$zero,$L45
        b $L42
$L45:
        lb $v0,40($fp)
        bne $v0,$zero,$L46
        lw $v0,24($fp)
        sll $v1,$v0,2
        lw $v0,68($fp)
        addu $v0,$v1,$v0
        lw $v0,0($v0)
        lb $v1,0($v0)
        li $v0,45          # 0x2d
        bne $v1,$v0,$L46
        sw $zero,44($fp)
$L47:
        addu $v0,$fp,32
        move $a0,$v0
        la $t9,strlen
        jal $ra,$t9
        move $v1,$v0
        lw $v0,44($fp)
        sltu $v0,$v0,$v1
        bne $v0,$zero,$L50
        b $L48
$L50:
        lw $v0,24($fp)
        sll $v1,$v0,2
        lw $v0,68($fp)
        addu $v0,$v1,$v0
        lw $v0,0($v0)
        addu $a0,$v0,1
        addu $v1,$fp,32
        lw $v0,44($fp)
        addu $v0,$v1,$v0
        lb $v1,0($a0)
        lb $v0,0($v0)
        bne $v1,$v0,$L49
        lw $v0,24($fp)
        sll $v1,$v0,2
        lw $v0,68($fp)
        addu $v0,$v1,$v0
        lw $v0,0($v0)
        addu $v0,$v0,1
        lbu $v0,0($v0)
        sb $v0,40($fp)
        b $L48
$L49:
        lw $v0,44($fp)
        addu $v0,$v0,1
        sw $v0,44($fp)
        b $L47
$L48:

```

```

    lb $v1,40($fp)
    li $v0,104      # 0x68
    bne $v1,$v0,$L52
    la $a0,$LC2
    la $a1,help_str
    la $t9,printf
    jal $ra,$t9
    lw $v1,76($fp)
    li $v0,1        # 0x1
    sw $v0,0($v1)
    b $L42
$L52:
    lb $v1,40($fp)
    li $v0,86       # 0x56
    bne $v1,$v0,$L53
    la $a0,$LC3
    la $a1,$LC4
    la $t9,printf
    jal $ra,$t9
    lw $v1,76($fp)
    li $v0,1        # 0x1
    sw $v0,0($v1)
    b $L42
$L53:
    lb $v0,40($fp)
    bne $v0,$zero,$L55
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,68($fp)
    addu $v0,$v1,$v0
    la $a0,$LC5
    lw $a1,0($v0)
    la $t9,printf
    jal $ra,$t9
    lw $v0,72($fp)
    sw $zero,0($v0)
    b $L42
$L46:
    lb $v1,40($fp)
    li $v0,105      # 0x69
    bne $v1,$v0,$L56
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,68($fp)
    addu $v0,$v1,$v0
    lw $a0,0($v0)
    la $a1,$LC6
    la $t9,fopen
    jal $ra,$t9
    move $v1,$v0
    lw $v0,72($fp)
    sw $v1,0($v0)
    b $L57
$L56:

```

```

    lb $v1,40($fp)
    li $v0,111      # 0x6f
    bne $v1,$v0,$L57
    lw $v0,24($fp)
    sll $v1,$v0,2
    lw $v0,68($fp)
    addu $v0,$v1,$v0
    lw $a0,0($v0)
    la $a1,$LC7
    la $t9,fopen
    jal $ra,$t9
    move $v1,$v0
    lw $v0,72($fp)
    addu $v0,$v0,4
    sw $v1,0($v0)
$L57:
    sb $zero,40($fp)
$L55:
    lw $v0,24($fp)
    addu $v0,$v0,1
    sw $v0,24($fp)
    b $L43
$L42:
    move $sp,$fp
    lw $ra,56($sp)
    lw $fp,52($sp)
    addu $sp,$sp,64
    j $ra
    .end arg_parse
    .size arg_parse, .-arg_parse
    .data
    .align 2
$L43:
    .word __sF
    .word __sF+88
    .text
    .align 2
    .globl main
    .ent main
main:
    .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
    .cpstore 16
    sw $ra,48($sp)
    sw $fp,44($sp)
    sw $gp,40($sp)
    move $fp,$sp
    sw $a0,56($fp)
    sw $a1,60($fp)

```

```

    lw $v0,$LC8
    sw $v0,24($fp)
    lw $v0,$LC8+4
    sw $v0,28($fp)
    sw $zero,32($fp)
    addu $v0,$fp,32
    lw $a0,56($fp)
    lw $a1,60($fp)
    addu $a2,$fp,24
    move $a3,$v0
    la $t9,arg_parse
    jal $ra,$t9
    lw $v0,32($fp)
    beq $v0,$zero,$L60
    sw $zero,36($fp)
    b $L59
$L60:
    lw $v0,24($fp)
    beq $v0,$zero,$L62
    lw $v0,28($fp)
    bne $v0,$zero,$L61
$L62:
    li $v0,1      # 0x1
    sw $v0,36($fp)
    b $L59
$L61:
    la $t9,chargeSpace
    jal $ra,$t9
    lw $a0,24($fp)
    lw $a1,28($fp)
    la $t9,readFile
    jal $ra,$t9
    lw $v1,24($fp)
    la $v0,___sF
    beq $v1,$v0,$L63
    lw $a0,24($fp)
    la $t9,fclose
    jal $ra,$t9
$L63:
    lw $v1,28($fp)
    la $v0,___sF+88
    beq $v1,$v0,$L64
    lw $a0,28($fp)
    la $t9,fclose
    jal $ra,$t9
$L64:
    sw $zero,36($fp)
$L59:
    lw $v0,36($fp)
    move $sp,$fp
    lw $ra,48($sp)
    lw $fp,44($sp)
    addu $sp,$sp,56
    j $ra

```

```
.end main
.size main, .-main

.comm space,65

.comm spaceIndex,492
.ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```
