Universidad de Buenos Aires
Facultad de Ingeniería

66.20 Organización de computadoras

Trabajo práctico 0: Infraestructura básica
1^{er} cuatrimestre de 2015

Paula Saffioti	Padrón: 92001	Email: paula.saffioti@gmail.com
Kaoru Heanna	Padrón: 91891	Email: kaoru.heanna@gmail.com
Julián Scialabba	Padrón: 92181	Email: julian.scialabba@gmail.com

1. Documentación

El objetivo del trabajo práctico es desarrollar una versión del comando tac de UNIX en lenguaje C. La funcionalidad básica del mismo se basa en escribir por stdout el contenido de uno o más archivos invirtiendo el orden de sus líneas.

En la etapa de diseño analizamos de qué manera ibamos a realizar el programa. Identificamos que debíamos trabajar con archivos y memoria dinámica y que la biblioteca standard de C nos proveía la funcionalidad necesaria para estos requerimientos. Luego, pensamos con un mayor nivel de detalle el algoritmo base del programa que se resume en la siguiente lista de pasos:

1. Abrir un archivo pasado por parámetro.
2. Leer el archivo línea por línea hasta el final del archivo y guardarlo en un array. El tamaño de las líneas debe ser dinámico.
3. Recorrer el array en sentido inverso imprimiendo por stdout el contenido de cada línea.

El paso 1 consiste en abrir los archivos pasados por parámetro. Esto lo resolvimos fácilmente con la función fopen. Como resultado de este paso, nos quedó un puntero al archivo para operar con él. En caso de algún error, lo lanzamos por stderr.

El paso 2, sin dudas, es el más complejo del trabajo práctico. La complejidad reside en la lectura del archivo pidiendo memoria de manera dinámica. Identificamos que para leer línea por línea el archivo podiamos hacerlo mediante la lectura de caracter por caracter hasta un fin de línea o la lectura de un bloque de caracteres hasta llegar al fin de línea (fget vs fgets). Hemos optado por la segunda opción por temas de rendimiento y practicidad. Luego de elegir leer el archivo mediante fgets, observamos que dicha función leía una cantidad fija de caracteres por línea y que los archivos con los que íbamos a trabajar esta cantidad iba a ser dinámica. Claramente íbamos a tener que utilizar los métodos de manejo de memoria de c: malloc, calloc, realloc y free.

En detalle de implementación, esto lo resolvimos de la siguiente manera:

1. Por cada nueva línea, inicializamos un array de caracteres que representaba la línea leer. La asignación de memoria dinámica a este array de caracteres lo hicimos mediante la funciona calloc que además de reservar la memoria estipulada la inicializa en 0.
2. Leemos bloques de caracteres mediante fgets en un buffer y lo copiamos en el array de caracteres que representa la línea actual mencionada en el paso 1. En esta copia, tuvimos que redimensionar el array de caracteres mediante un realloc.
3. Si el último caracter del buffer era un fin de línea, se procedía a guardar la línea actual resultante en el array de líneas y se inicializaba un nuevo array de caracteres como en el paso 1.

El paso 3. fue el más simple, ya que con un solo recorrido del array formado anteriormente lo cumplimos sin problemas.

2. Comandos

Para compilar el programa es necesario escribir el siguiente comando en el directorio donde se encuentre main.c:

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

3. Corridas de prueba

Para corroborar el funcionamiento del programa contábamos con una serie de archivos de prueba y con el diccionario de palabras ubicado en /usr/share/dict/words.

Salidas de ejemplo:

return.txt:
of -1 is returned and errno is set to indicate the error. Upon successful completion a value of 0 is returned. Otherwise, a value

basic.txt
4. D
3. C
2. B
1. A

return.txt basic.txt
of -1 is returned and errno is set to indicate the error.
Upon successful completion a value of 0 is returned. Otherwise, a value
4. D
3. C
2. B
1. A

Para poder hacer todas las pruebas de manera rápida escribimos un script que compilaba el programa, luego lo ejecutaba para los archivos mencionados dándolos vuelta dos veces y finalmente verificaba que el archivo resultante fuera igual al original con md5sum.

MD5 de los archivos de prueba procesados dos veces:

empty.txt: d41d8cd98f00b204e9800998ecf8427e
basic: f82584604433de83165f0227f4f06c2c
empty-lines.txt: 0c060d8e86deedcb0f9cd89f6a3f9c93
large-file.txt: 8b2a59a337c384be9b640a38a377eef1
return.txt: 6777acdeb99f2fb0ed9e14fc82899e82
status.txt: 964ccaae773b2f9044dc549ed3636ab6
words: cbbcded3dc3b61ad50c4e36f79c37084

Para el caso de NetBSD, probamos el funcionamiento del programa para todos los archivos a mano.

4. Código fuente

```
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <string.h>

void printHelpInfo() {
    printf("%s", "Usage:\n");
    printf("%s", "      tp0 -h\n");
    printf("%s", "      tp0 -V\n");
    printf("%s", "      tp0 [file ...]\n");
    printf("%s", "Options:\n");
    printf("%s", "      -V, --version Print version and quit.\n")
        ;
    printf("%s", "      -h, --help Print this information and
        quit.\n");
    printf("%s", "Examples:\n");
    printf("%s", "      tp0 foo.txt bar.txt\n");
    printf("%s", "      tp0 gz.txt\n");
}

void printVersionInfo() {
    printf("%s", "tp0 1.1\n");
    printf("%s", "Copyright    2015 FIUBA.\n");
    printf("%s", "Esto es software libre: usted es libre de
        cambiarlo y redistribuirlo.\n");
    printf("%s", "No hay NINGUNA GARANTA, hasta donde permite
        la ley.\n");
    printf("%s", "\n");
    printf("%s", "Escrito por Julian Scialabba, Kaoru Heanna y
        Paula Saffioti.\n");
}

int isEndOfLine(char c) {
    if (c == '\n') {
        return 1;
    }
    return 0;
}

void concatBuffer(char** line, const char* buffer) {
    size_t len1 = *line ? strlen(*line) : 0;
    size_t len2 = buffer ? strlen(buffer) : 0;
    char* concat = realloc(*line, len1 + len2 + 1);
    if (concat) {
        memcpy(concat + len1, buffer, len2 + 1);
        *line = concat;
    }
}

int tacFile(FILE* fp) {
    const int bufIncrSize = 10;
    char ** arrayLines = NULL;
```

```
    int lineCounter = 0;
    char buffer[bufIncrSize];
    char* line = (char *) calloc(bufIncrSize, sizeof (char));
    while (fgets(buffer, bufIncrSize, fp)) {
        concatBuffer(&line, buffer);
        char lastCharacterBuffer = buffer[strlen(buffer) - 1];
        if (isEndOfLine(lastCharacterBuffer)) {
            arrayLines = realloc(arrayLines,
                    (lineCounter + 1) * sizeof (char *));
            arrayLines[lineCounter] = line;
            lineCounter++;
            line = (char *) calloc(bufIncrSize, sizeof (char));
        }
    }

    int i;
    for (i = lineCounter - 1; i >= 0; i--) {
        printf("%s", arrayLines[i]);
        free(arrayLines[i]);

    }
    free(line);
    free(arrayLines);
    return (EXIT_SUCCESS);
}

int main(int argc, char** argv) {

    if ((argc == 2) && ((strcmp(argv[1], "-h") == 0) || (strcmp(
       argv[1], "--help") == 0))) {
        printHelpInfo();
        return (EXIT_SUCCESS);
    }

    if ((argc == 2) && ((strcmp(argv[1], "-V") == 0) || (strcmp(
       argv[1], "--version") == 0))) {
        printVersionInfo();
        return (EXIT_SUCCESS);
    }

    int result;

    if (argc < 2) { //no tengo archivo de entrada, uso standard
       input
        result = tacFile(stdin);
        return (result);
    }

    int i;
```

```
    for (i = 1; i < argc; i++) {
        FILE *fp;
        fp = fopen(argv[i], "r");
        if (fp == NULL) {
            fprintf(stderr, "%s", argv[i]);
            fprintf(stderr, ": nombre de archivo o comando
                inv lido.\n");
            return (EXIT_FAILURE);
        }
        tacFile(fp);
        fclose(fp);
    }
    return (EXIT_SUCCESS);
}
```

5. Código MIPS32

```
        .file   1 "main.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align  2
$LC0:
        .ascii  "%s\000"
        .align  2
$LC1:
        .ascii  "Usage:\n\000"
        .align  2
$LC2:
        .ascii  "\ttp0 -h\n\000"
        .align  2
$LC3:
        .ascii  "\ttp0 -V\n\000"
        .align  2
$LC4:
        .ascii  "\ttp0 [file ...]\n\000"
        .align  2
$LC5:
        .ascii  "Options:\n\000"
        .align  2
$LC6:
        .ascii  "\t-V, --version Print version and quit.\n\000"
        .align  2
$LC7:
        .ascii  "\t-h, --help Print this information and quit.\n
            \000"
        .align  2
$LC8:
```

```
        .ascii  "Examples:\n\000"
        .align  2
$LC9:
        .ascii  "\ttp0 foo.txt bar.txt\n\000"
        .align  2
$LC10:
        .ascii  "\ttp0 gz.txt\n\000"
        .text
        .align  2
        .globl  printHelpInfo
        .ent    printHelpInfo
printHelpInfo:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0,
           args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,$LC0
        la      $a1,$LC1
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC2
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC3
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC4
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC5
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC6
        la      $t9,printf
        jal     $ra,$t9
```

```
        la      $a0,$LC0
        la      $a1,$LC7
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC8
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC9
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC10
        la      $t9,printf
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    printHelpInfo
        .size   printHelpInfo, .-printHelpInfo
        .rdata
        .align  2
$LC11:
        .ascii  "tp0 1.1\n\000"
        .align  2
$LC12:
        .ascii  "Copyright \302\251 2015 FIUBA.\n\000"
        .align  2
$LC13:
        .ascii  "Esto es software libre: usted es libre de
           cambiarlo y re"
        .ascii  "distribuirlo.\n\000"
        .align  2
$LC14:
        .ascii  "No hay NINGUNA GARANT\303\215A, hasta donde
           permite la l"
        .ascii  "ey.\n\000"
        .align  2
$LC15:
        .ascii  "\n\000"
        .align  2
$LC16:
        .ascii  "Escrito por Julian Scialabba, Kaoru Heanna y
           Paula Saffi"
        .ascii  "oti.\n\000"
        .text
```

```
        .align  2
        .globl  printVersionInfo
        .ent    printVersionInfo
printVersionInfo:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0,
           args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,$LC0
        la      $a1,$LC11
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC12
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC13
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC14
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC15
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC0
        la      $a1,$LC16
        la      $t9,printf
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    printVersionInfo
        .size   printVersionInfo, .-printVersionInfo
        .align  2
```

```
        .globl  isEndOfLine
        .ent    isEndOfLine
isEndOfLine:
        .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
        move    $v0,$a0
        sb      $v0,8($fp)
        lb      $v1,8($fp)
        li      $v0,10                  # 0xa
        bne     $v1,$v0,$L20
        li      $v0,1                   # 0x1
        sw      $v0,12($fp)
        b       $L19
$L20:
        sw      $zero,12($fp)
$L19:
        lw      $v0,12($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addu    $sp,$sp,24
        j       $ra
        .end    isEndOfLine
        .size   isEndOfLine, .-isEndOfLine
        .align  2
        .globl  concatBuffer
        .ent    concatBuffer
concatBuffer:
        .frame  $fp,64,$ra              # vars= 24, regs= 3/0,
           args= 16, extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $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)
        lw      $v0,64($fp)
        lw      $v0,0($v0)
        beq     $v0,$zero,$L22
        lw      $v0,64($fp)
        lw      $a0,0($v0)
        la      $t9,strlen
        jal     $ra,$t9
        sw      $v0,36($fp)
        b       $L23
$L22:
        sw      $zero,36($fp)
$L23:
        lw      $v0,36($fp)
        sw      $v0,24($fp)
        lw      $v0,68($fp)
        beq     $v0,$zero,$L24
        lw      $a0,68($fp)
        la      $t9,strlen
        jal     $ra,$t9
        sw      $v0,40($fp)
        b       $L25
$L24:
        sw      $zero,40($fp)
$L25:
        lw      $v0,40($fp)
        sw      $v0,28($fp)
        lw      $a0,64($fp)
        lw      $v1,24($fp)
        lw      $v0,28($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lw      $a0,0($a0)
        move    $a1,$v0
        la      $t9,realloc
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        beq     $v0,$zero,$L21
        lw      $v1,32($fp)
        lw      $v0,24($fp)
        addu    $v1,$v1,$v0
        lw      $v0,28($fp)
        addu    $v0,$v0,1
        move    $a0,$v1
        lw      $a1,68($fp)
        move    $a2,$v0
```

```
        la      $t9,memcpy
        jal     $ra,$t9
        lw      $v1,64($fp)
        lw      $v0,32($fp)
        sw      $v0,0($v1)
$L21:
        move    $sp,$fp
        lw      $ra,56($sp)
        lw      $fp,52($sp)
        addu    $sp,$sp,64
        j       $ra
        .end    concatBuffer
        .size   concatBuffer, .-concatBuffer
        .align  2
        .globl  tacFile
        .ent    tacFile
tacFile:
        .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      $sp,48($fp)
        li      $v0,10                  # 0xa
        sw      $v0,24($fp)
        sw      $zero,28($fp)
        sw      $zero,32($fp)
        lw      $v0,24($fp)
        addu    $v0,$v0,7
        srl     $v0,$v0,3
        sll     $v0,$v0,3
        subu    $sp,$sp,$v0
        addu    $v0,$sp,16
        sw      $v0,52($fp)
        lw      $a0,24($fp)
        li      $a1,1                   # 0x1
        la      $t9,calloc
        jal     $ra,$t9
        sw      $v0,36($fp)
$L28:
```

```
        lw      $a0,52($fp)
        lw      $a1,24($fp)
        lw      $a2,72($fp)
        la      $t9,fgets
        jal     $ra,$t9
        bne     $v0,$zero,$L30
        b       $L29
$L30:
        addu    $v0,$fp,36
        move    $a0,$v0
        lw      $a1,52($fp)
        la      $t9,concatBuffer
        jal     $ra,$t9
        lw      $a0,52($fp)
        la      $t9,strlen
        jal     $ra,$t9
        lw      $v1,52($fp)
        addu    $v0,$v0,$v1
        addu    $v0,$v0,-1
        lbu     $v0,0($v0)
        sb      $v0,40($fp)
        lb      $v0,40($fp)
        move    $a0,$v0
        la      $t9,isEndOfLine
        jal     $ra,$t9
        beq     $v0,$zero,$L28
        lw      $v0,32($fp)
        sll     $v0,$v0,2
        addu    $v0,$v0,4
        lw      $a0,28($fp)
        move    $a1,$v0
        la      $t9,realloc
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,32($fp)
        sll     $v1,$v0,2
        lw      $v0,28($fp)
        addu    $v1,$v1,$v0
        lw      $v0,36($fp)
        sw      $v0,0($v1)
        lw      $v0,32($fp)
        addu    $v0,$v0,1
        sw      $v0,32($fp)
        lw      $a0,24($fp)
        li      $a1,1                   # 0x1
        la      $t9,calloc
        jal     $ra,$t9
        sw      $v0,36($fp)
        b       $L28
```

```
$L29:
        lw      $v0,32($fp)
        addu    $v0,$v0,-1
        sw      $v0,44($fp)
$L32:
        lw      $v0,44($fp)
        bgez    $v0,$L35
        b       $L33
$L35:
        lw      $v0,44($fp)
        sll     $v1,$v0,2
        lw      $v0,28($fp)
        addu    $v0,$v1,$v0
        la      $a0,$LC0
        lw      $a1,0($v0)
        la      $t9,printf
        jal     $ra,$t9
        lw      $v0,44($fp)
        sll     $v1,$v0,2
        lw      $v0,28($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $t9,free
        jal     $ra,$t9
        lw      $v0,44($fp)
        addu    $v0,$v0,-1
        sw      $v0,44($fp)
        b       $L32
$L33:
        lw      $a0,36($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $a0,28($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $sp,48($fp)
        move    $v0,$zero
        move    $sp,$fp
        lw      $ra,64($sp)
        lw      $fp,60($sp)
        addu    $sp,$sp,72
        j       $ra
        .end    tacFile
        .size   tacFile, .-tacFile
        .rdata
        .align  2
$LC17:
        .ascii  "-h\000"
        .align  2
```

```
$LC18:
        .ascii  "--help\000"
        .align  2
$LC19:
        .ascii  "-V\000"
        .align  2
$LC20:
        .ascii  "--version\000"
        .align  2
$LC21:
        .ascii  "r\000"
        .align  2
$LC22:
        .ascii  ": nombre de archivo o comando inv\303\241lido.\n
           \000"
        .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
        .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)
        lw      $v1,56($fp)
        li      $v0,2                   # 0x2
        bne     $v1,$v0,$L37
        lw      $v0,60($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC17
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L38
        lw      $v0,60($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC18
```

```
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L37
$L38:
        la      $t9,printHelpInfo
        jal     $ra,$t9
        sw      $zero,36($fp)
        b       $L36
$L37:
        lw      $v1,56($fp)
        li      $v0,2                   # 0x2
        bne     $v1,$v0,$L39
        lw      $v0,60($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC19
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L40
        lw      $v0,60($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC20
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L39
$L40:
        la      $t9,printVersionInfo
        jal     $ra,$t9
        sw      $zero,36($fp)
        b       $L36
$L39:
        lw      $v0,56($fp)
        slt     $v0,$v0,2
        beq     $v0,$zero,$L41
        la      $a0,__sF
        la      $t9,tacFile
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        sw      $v0,36($fp)
        b       $L36
$L41:
        li      $v0,1                   # 0x1
        sw      $v0,28($fp)
$L42:
        lw      $v0,28($fp)
        lw      $v1,56($fp)
        slt     $v0,$v0,$v1
```

```
        bne     $v0,$zero,$L45
        b       $L43
$L45:
        lw      $v0,28($fp)
        sll     $v1,$v0,2
        lw      $v0,60($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC21
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        bne     $v0,$zero,$L46
        lw      $v0,28($fp)
        sll     $v1,$v0,2
        lw      $v0,60($fp)
        addu    $v0,$v1,$v0
        la      $a0,__sF+176
        la      $a1,$LC0
        lw      $a2,0($v0)
        la      $t9,fprintf
        jal     $ra,$t9
        la      $a0,__sF+176
        la      $a1,$LC22
        la      $t9,fprintf
        jal     $ra,$t9
        li      $v0,1                   # 0x1
        sw      $v0,36($fp)
        b       $L36
$L46:
        lw      $a0,32($fp)
        la      $t9,tacFile
        jal     $ra,$t9
        lw      $a0,32($fp)
        la      $t9,fclose
        jal     $ra,$t9
        lw      $v0,28($fp)
        addu    $v0,$v0,1
        sw      $v0,28($fp)
        b       $L42
$L43:
        sw      $zero,36($fp)
$L36:
        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
.ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```