

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías



División de Tecnologías para la Integración CiberHumana

Ingeniería en Computación

Programación de Bajo Nivel

D02 - IL358 - 209850

4. Manipulación de Archivos

Profesor: José Juan Meza Espinoza

Alumno: Alan Yahir Juárez Rubio

Código: 218517809

Correo: alan.juarez5178@alumnos.udg.mx

Este documento ha sido elaborado con fines estudiantiles.
La información presentada puede contener errores.

20 de octubre de 2024



Índice

1. Introducción	4
2. Implementación	5
2.1. Código	5
2.2. Ejecución del Programa	6
3. Conclusión	12



Índice de figuras

1.	Inicialización del programa	6
2.	Creación y apertura del archivo	7
3.	Escritura del archivo	8
4.	Muestreo de propiedades del archivo	9
5.	Cerrado del archivo	10
6.	Eliminación del archivo	11



Índice de códigos

1.	Manipulación de archivos	5
----	------------------------------------	---



4. Manipulación de Archivos

1. Introducción

La manipulación de archivos en ensamblador es una práctica fundamental, especialmente en ambientes como linux, en donde todo se maneja a través de archivos. La manipulación correcta de archivos te permite tener control acerca del cómo la información es almacenada, recuperada y manipulada. A continuación veremos un código con las fundamentales para la manipulación de archivos.

2. Implementación

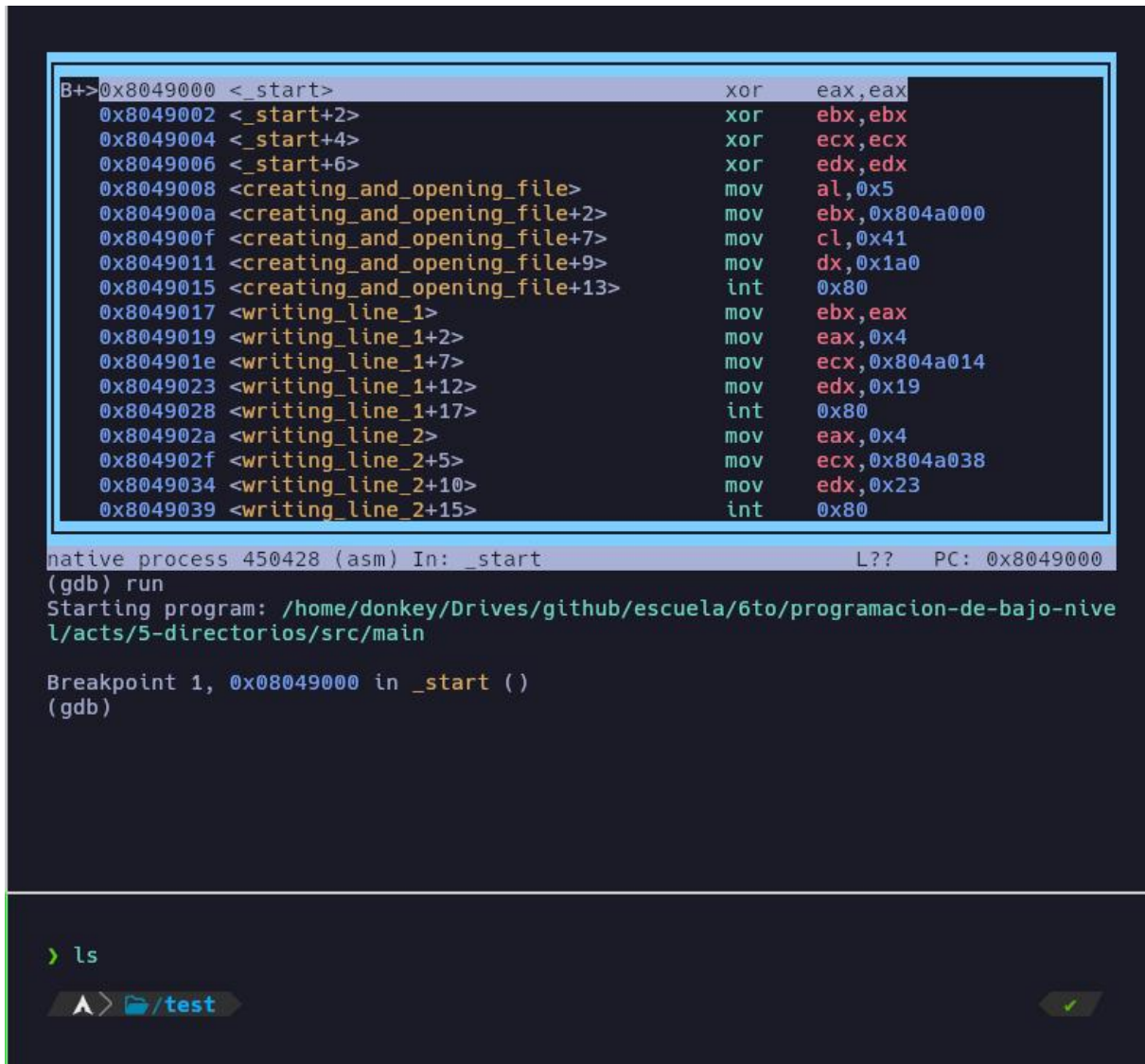
2.1. Código

```
1 section .data
2     file_path DD "./alan.txt", 0
3
4     line_1    DD "Parangaricutirimicuario", 0xA, 0xD, "$"
5     len_line_1 equ 25
6
7     line_2    DD "Hipopotomonstrosequipedaliofobia", 0xA, 0xD, "$"
8     len_line_2 equ 35
9
10 section .text
11     global _start
12
13 _start:
14     XOR eax, eax
15     XOR ebx, ebx
16     XOR ecx, ecx
17     XOR edx, edx
18
19 creating_and_opening_file:
20     MOV al, 5 ; sys_open() call
21     MOV ebx, file_path
22     MOV cl, 0101o ; CW accessss mode
23                     ; 100 stands for create accessss
24                     ; 001 stands for write access
25
26     MOV dx, 0b110_100_000 ; RW permissions for Owner (6)
27                     ; R permissions for Group (4)
28                     ; No permissions for Others (0)
29
30     INT 0x80 ; File descriptor saved in 'eax' after interruption
31
32 writing_line_1:
33     MOV ebx, eax ; File descriptor
34     MOV eax, 4 ; sys_write() call
35     MOV ecx, line_1 ; content
36     MOV edx, len_line_1 ; content_size
37     INT 0x80
38
39 writing_line_2:
40     MOV eax, 4 ; sys_write() call
41     MOV ecx, line_2 ; content
42     MOV edx, len_line_2 ; content_size
43     INT 0x80
44
45 ; also takes file descriptor from 'ebx' (already saved)
46 closing_file:
47     MOV eax, 6 ; sys_close() call
48     INT 0x80
49
50 removing_file:
51     MOV eax, 10 ; sys_unlink() call
52     MOV ebx, file_path
53     INT 0x80
54
55 exit:
56     MOV eax, 1
57     MOV ebx, 0
```

58 INT 0x80

Código 1: Manipulación de archivos

2.2. Ejecución del Programa



```
B+>0x8049000 <_start>          xor    eax,eax
0x8049002 <_start+2>          xor    ebx,ebx
0x8049004 <_start+4>          xor    ecx,ecx
0x8049006 <_start+6>          xor    edx,edx
0x8049008 <creating_and_opening_file> mov    al,0x5
0x804900a <creating_and_opening_file+2> mov    ebx,0x804a000
0x804900f <creating_and_opening_file+7> mov    cl,0x41
0x8049011 <creating_and_opening_file+9> mov    dx,0x1a0
0x8049015 <creating_and_opening_file+13> int    0x80
0x8049017 <writing_line_1>      mov    ebx,eax
0x8049019 <writing_line_1+2>      mov    eax,0x4
0x804901e <writing_line_1+7>      mov    ecx,0x804a014
0x8049023 <writing_line_1+12>     mov    edx,0x19
0x8049028 <writing_line_1+17>     int    0x80
0x804902a <writing_line_2>      mov    eax,0x4
0x804902f <writing_line_2+5>      mov    ecx,0x804a038
0x8049034 <writing_line_2+10>    mov    edx,0x23
0x8049039 <writing_line_2+15>    int    0x80

native process 450428 (asm) In: _start          L??  PC: 0x8049000
(gdb) run
Starting program: /home/donkey/Drives/github/escuela/6to/programacion-de-bajo-nive
l/acts/5-directorios/src/main

Breakpoint 1, 0x08049000 in _start ()
(gdb)

> ls
^> /test
```

Fig. 1: Inicialización del programa

```
B+ 0x8049000 <_start>          xor    eax,eax
0x8049002 <_start+2>          xor    ebx,ebx
0x8049004 <_start+4>          xor    ecx,ecx
0x8049006 <_start+6>          xor    edx,edx
0x8049008 <creating_and_opening_file> mov    al,0x5
0x804900a <creating_and_opening_file+2> mov    ebx,0x804a000
0x804900f <creating_and_opening_file+7> mov    cl,0x41
0x8049011 <creating_and_opening_file+9> mov    dx,0x1a0
0x8049015 <creating_and_opening_file+13> int    0x80
>0x8049017 <writing_line_1>      mov    ebx,eax
0x8049019 <writing_line_1+2>      mov    eax,0x4
0x804901e <writing_line_1+7>      mov    ecx,0x804a014
0x8049023 <writing_line_1+12>     mov    edx,0x19
0x8049028 <writing_line_1+17>     int    0x80
0x804902a <writing_line_2>        mov    eax,0x4
0x804902f <writing_line_2+5>      mov    ecx,0x804a038
0x8049034 <writing_line_2+10>     mov    edx,0x23
0x8049039 <writing_line_2+15>     int    0x80

native process 452098 (asm) In: writing_line_1      L??  PC: 0x8049017
(gdb) n
Single stepping until exit from function _start,
which has no line number information.
0x08049008 in creating_and_opening_file ()
(gdb) n
Single stepping until exit from function creating_and_opening_file,
which has no line number information.
0x08049017 in writing_line_1 ()
(gdb) info registers eax
eax             0x3             3
(gdb)

> ls
> ls
alan.txt

A > /test |
```

Fig. 2: Creación y apertura del archivo


```
>0x804903b <closing_file>    mov     eax,0x6
0x8049040 <closing_file+5>    mov     ebx,edx
0x8049042 <closing_file+7>    int     0x80
0x8049044 <removing_file>        mov     eax,0xa
0x8049049 <removing_file+5>    mov     ebx,0x804a000
0x804904e <removing_file+10> int     0x80
0x8049050 <exit>                  mov     eax,0x1
0x8049055 <exit+5>              mov     ebx,0x0
0x804905a <exit+10>             int     0x80
0x804905c                    add     BYTE PTR [eax],al
0x804905e                    add     BYTE PTR [eax],al
0x8049060                    add     BYTE PTR [eax],al
0x8049062                    add     BYTE PTR [eax],al
0x8049064                    add     BYTE PTR [eax],al
0x8049066                    add     BYTE PTR [eax],al
0x8049068                    add     BYTE PTR [eax],al
0x804906a                    add     BYTE PTR [eax],al
0x804906c                    add     BYTE PTR [eax],al

native process 452098 (asm) In: closing_file      L??  PC: 0x804903b
(gdb) info registers eax
eax             0x3             3
(gdb) n
Single stepping until exit from function writing_line_1,
which has no line number information.
0x0804902a in writing_line_2 ()
(gdb) n
Single stepping until exit from function writing_line_2,
which has no line number information.
0x0804903b in closing_file ()
(gdb)

> ls alan.txt
alan.txt
> cat alan.txt
Parangaricutirimicuario
Hipopotomonstrosequipediaiofobia

^> /test
```

Fig. 3: Escritura del archivo

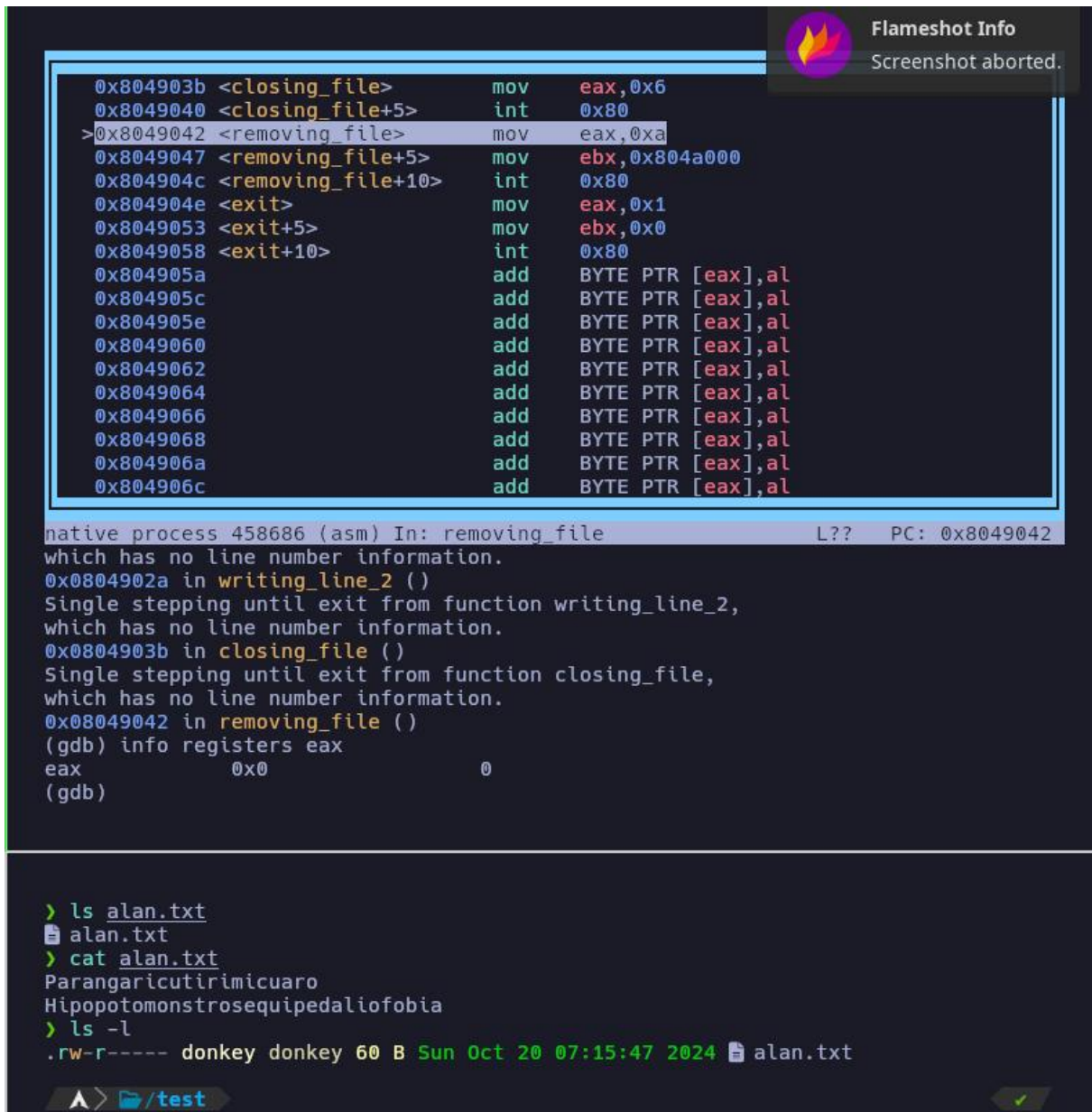
```
>0x804903b <closing_file>    mov     eax,0x6
0x8049040 <closing_file+5>      mov     ebx,edx
0x8049042 <closing_file+7>      int     0x80
0x8049044 <removing_file>         mov     eax,0xa
0x8049049 <removing_file+5>     mov     ebx,0x804a000
0x804904e <removing_file+10>  int     0x80
0x8049050 <exit>                 mov     eax,0x1
0x8049055 <exit+5>             mov     ebx,0x0
0x804905a <exit+10>              int     0x80
0x804905c                      add     BYTE PTR [eax],al
0x804905e                      add     BYTE PTR [eax],al
0x8049060                      add     BYTE PTR [eax],al
0x8049062                      add     BYTE PTR [eax],al
0x8049064                      add     BYTE PTR [eax],al
0x8049066                      add     BYTE PTR [eax],al
0x8049068                      add     BYTE PTR [eax],al
0x804906a                      add     BYTE PTR [eax],al
0x804906c                      add     BYTE PTR [eax],al

native process 452098 (asm) In: closing_file      L??  PC: 0x804903b
(gdb) info registers eax
eax             0x3             3
(gdb) n
Single stepping until exit from function writing_line_1,
which has no line number information.
0x0804902a in writing_line_2 ()
(gdb) n
Single stepping until exit from function writing_line_2,
which has no line number information.
0x0804903b in closing_file ()
(gdb)

> ls alan.txt
alan.txt
> cat alan.txt
Parangaricutirimicuaro
Hipopotomonstrosesquipedaliofobia
> ls -l
-rw-r----- donkey donkey 60 B Sun Oct 20 07:15:47 2024 alan.txt

A> /test
```

Fig. 4: Muestreo de propiedades del archivo



The screenshot shows a debugger window with two panes. The top pane displays assembly code for a function named `removing_file`. The code includes instructions for moving values into `eax` and `ebx`, and a series of `add` instructions for a pointer `[eax]`. The bottom pane shows the debugger's execution flow, including stepping through functions like `writing_line_2` and `closing_file`, and a terminal window showing the execution of `ls` and `cat` commands on a file named `alan.txt`.

```
0x804903b <closing_file>    mov     eax,0x6
0x8049040 <closing_file+5>      int     0x80
>0x8049042 <removing_file>     mov     eax,0xa
0x8049047 <removing_file+5>     mov     ebx,0x804a000
0x804904c <removing_file+10>  int     0x80
0x804904e <exit>                mov     eax,0x1
0x8049053 <exit+5>             mov     ebx,0x0
0x8049058 <exit+10>             int     0x80
0x804905a                add     BYTE PTR [eax],al
0x804905c                add     BYTE PTR [eax],al
0x804905e                add     BYTE PTR [eax],al
0x8049060                add     BYTE PTR [eax],al
0x8049062                add     BYTE PTR [eax],al
0x8049064                add     BYTE PTR [eax],al
0x8049066                add     BYTE PTR [eax],al
0x8049068                add     BYTE PTR [eax],al
0x804906a                add     BYTE PTR [eax],al
0x804906c                add     BYTE PTR [eax],al

native process 458686 (asm) In: removing_file      L??  PC: 0x8049042
which has no line number information.
0x0804902a in writing_line_2 ()
Single stepping until exit from function writing_line_2,
which has no line number information.
0x0804903b in closing_file ()
Single stepping until exit from function closing_file,
which has no line number information.
0x08049042 in removing_file ()
(gdb) info registers eax
eax                0x0                0
(gdb)

> ls alan.txt
alan.txt
> cat alan.txt
Parangaricutirimicuario
Hipopotomonstrosequipediaiofobia
> ls -l
.rw-r----- donkey donkey 60 B Sun Oct 20 07:15:47 2024 alan.txt

A> /test
```

Fig. 5: Cerrado del archivo

```
0x804903b <closing_file>    mov     eax,0x6
0x8049040 <closing_file+5>    int     0x80
0x8049042 <removing_file>     mov     eax,0xa
0x8049047 <removing_file+5>   mov     ebx,0x804a000
0x804904c <removing_file+10> int     0x80
>0x804904e <exit>          mov     eax,0x1
0x8049053 <exit+5>           mov     ebx,0x0
0x8049058 <exit+10>            int     0x80
0x804905a                add     BYTE PTR [eax],al
0x804905c                add     BYTE PTR [eax],al
0x804905e                add     BYTE PTR [eax],al
0x8049060                add     BYTE PTR [eax],al
0x8049062                add     BYTE PTR [eax],al
0x8049064                add     BYTE PTR [eax],al
0x8049066                add     BYTE PTR [eax],al
0x8049068                add     BYTE PTR [eax],al
0x804906a                add     BYTE PTR [eax],al
0x804906c                add     BYTE PTR [eax],al

native process 458686 (asm) In: exit                                L??    PC: 0x804904e
0x0804903b in closing_file ()
Single stepping until exit from function closing_file,
which has no line number information.
0x08049042 in removing_file ()
(gdb) info registers eax
eax                0x0                0
(gdb) n
Single stepping until exit from function removing_file,
which has no line number information.
0x0804904e in exit ()
(gdb)

> ls alan.txt
ls alan.txt
> cat alan.txt
Parangaricutirimicuario
Hipopotomonstrosequipediaiofobia
> ls -l
-rw-r----- donkey donkey 60 B Sun Oct 20 07:15:47 2024 alan.txt
> ls
```

Fig. 6: Eliminación del archivo



3. Conclusión

En retrospectiva, es importante recalcar que la manipulación de archivos, es de vital importancia para un desarrollador cualquier desarrollador, especialmente de ensamblador, debido a que todos los programas requieren la manipulación de datos a través de archivos, para gestionar la información de los mismos programas, usuarios, sistemas y demás.

Para finalizar, el entender cada una de las operaciones mostradas en el código 1 son esenciales entenderlas, especialmente para aquellos desarrolladores que necesitan optimizar el rendimiento, manejar los recursos eficientemente o desarrollar programas que interactúan cercamente al hardware.