Práctica 10

Interrupciones

Luis Eduardo Galindo Amaya (1274895)

Asignatura	Organización de Computadoras (331)
Docente	Arturo Arreola Alvarez
Fecha	04-11-2022

Interrupciones

Luis Eduardo Galindo Amaya (1274895)

04-11-2022

Objetivo

Seleccionar las instrucciones de llamadas al sistema adecuadas, para desarrollar aplicaciones de sistemas basados en microprocesador, mediante el análisis de su funcionalidad, de forma responsable y eficiente.

Desarollo

Actvidad 1

Completar la tabla sobre los parámetros necesarios en las llamadas al sistema operativo para el manejo de archivos en Linux por medio de la interrupción 80h.

Servicio	Param. del servicio	Explicación
Leer	EAX = 3	Numero del servicio[1]
	EBX = 0	Unidad de entrada[2]
	ECX = ptr	puntero a un área de memoria
	EDX = length	Número de caracteres a leer
Escribir	EAX = 4	Numero del servicio
	EBX = 1	unidad de salida[3]
	ECX = ptr	Puntero a un área de memoria
	EDX = lenght	Número máximo de caracteres
Abrir	EAX = 5	Numero del servicio
	EBX = path	Dir. de una cadena de caracteres
	ECX = mode	Modo de acceso[4]
	EDX = permisos	Permisos al archivo

[1] al terminar el servicio el valor se reemmplaza con el numero de caracteres capturados
[2] 0: Entrada estándar o teclado.
[3] 1: salida estándar o terminal.
[4] Más informacion: `https://es.wikipedia.org/wiki/Int_80h`

Actividad 2

Codifique las funciones gets y puts las cuales capturan e imprimen una cadena de caracteres en pantalla, respectivamente. Solo haga uso de la interrupción 80h. No haga uso de la librería proporcionada.

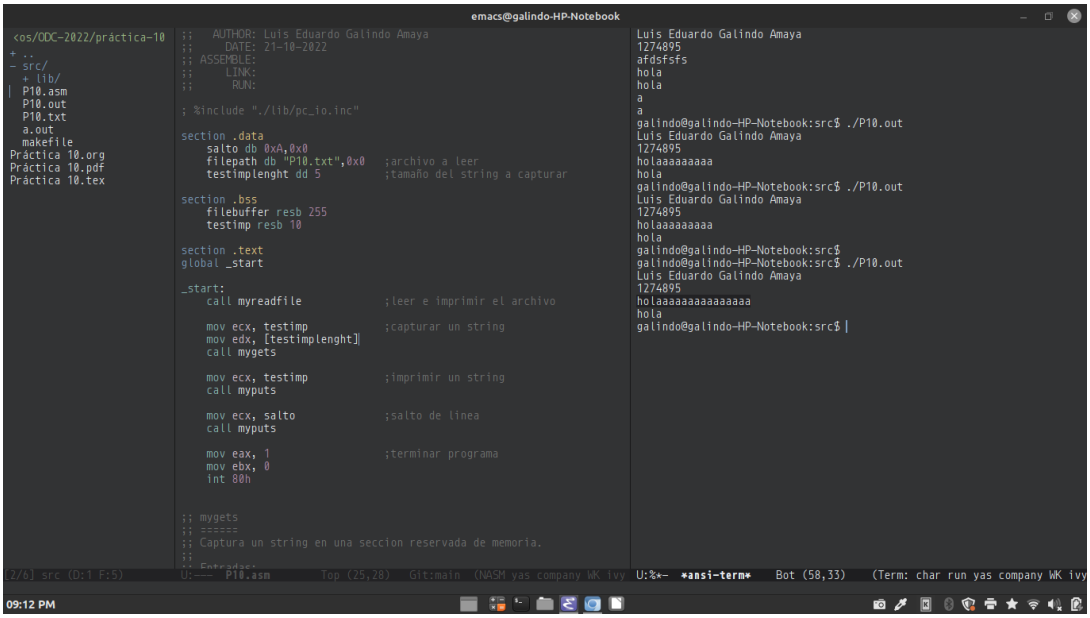
La subrutina gets solo debe poder capturar la cantidad máxima de caracteres que caben en el buffer en el que se guardará la cadena.

La subrutina puts debe ser capaz de detectar la longitud de la cadena a imprimir y pasar este valor a la interrupción 80h para establecer la cantidad de caracteres a imprmir.

Actividad 3

Cree un archivo dentro de la misma carpeta, llamado P10.txt. Dentro de este archivo, coloque su nombre y su matrícula. Desarrolle una subrutina que le permita leer el contenido del archivo e imprimirlo en pantalla utilizando la interrupción 80h. Mencione que datos se deben colocar en cada registro para lograr esto.

Captura



Conclusiones

Cuando trabajamos con interrupciones lo principal cosa que debemos tomar en cuenta es que cada sistema operativo es diferente, por lo que nuestro código va a tener que modificarse de acuerdo a cada sistema operativo lo cual añade una capa mas a las complicaciones que requiere el ensamblador para funcionar.

Dificultades

Con la interrupción de captura tuve que añadir una funcion para vaciar el buffer, ya que pasaba los valores a la siguiente captura.

Código

```
;;    AUTHOR: Luis Eduardo Galindo Amaya
;;      DATE: 21-10-2022
;; ASSEMBLE:
;;      LINK:
;;       RUN:

section .data
    salto db 0xA,0x0
    filepath db "P10.txt",0x0   ;archivo a leer
    testimplenght dd 5          ;tamaño del string a capturar

section .bss
    filebuffer resb 255
    testimp resb 10

section .text
global _start

_start:
    call myreadfile             ;leer e imprimir el archivo

    mov ecx, testimp            ;capturar un string
    mov edx, [testimplenght]
    call mygets

    mov ecx, testimp            ;imprimir un string
    call myputs

    mov ecx, salto              ;salto de linea
```

```
30      call myputs
31
32      mov eax, 1                  ;terminar programa
33      mov ebx, 0
34      int 80h
35
36
37  ;; mygets
38  ;; ======
39  ;; Captura un string en una seccion reservada de memoria.
40  ;;
41  ;; Entradas:
42  ;; - ecx, puntero a el string
43  ;; - edx, tamaño maximo de entrada - 1 (para el terminador)
44  ;;
45  mygets:
46      dec edx
47      mov eax, 3                  ;numero de la interrupción
48      mov ebx, 0                  ;unidad de entrada
49      int 80h
50
51      ; ejecutar el servicio 3, eax contiene la cantidad de
52      ; caracteres capturados
53
54      cmp eax, edx                ;si el string capturado
55      jb .capturaesmenor
56      jmp .capturaesmayor
57
58  .capturaesmenor:                ;es menor al buffer
59
60      dec eax
61      mov byte[ecx + eax], 0
62      ret
63
64  .capturaesmayor:                ;es mayor al buffer
65
66      add ecx, edx                ;guardar la ultima
67      mov esi, ecx                ;posicion capturada
68
69      ;deja la linea vacia para la siguiente entrada
70
71  .clearbuffer:                   ;
72
73      mov eax, 3                  ;numero de la interrupción
74      mov ebx, 0                  ;unidad de entrada
75      mov ecx, esi
76      mov edx, 1
77      int 80h
```

```
78
79      cmp byte[ecx], 10           ;si el contenido en ecx es 0
80      jnz .clearbuffer            ;termina el contador
81
82      mov byte[esi], 0
83      ret
84
85
86  ;; myputs
87  ;; ======
88  ;; Imprime el string en ecx.
89  ;;
90  ;; Entradas:
91  ;; - ecx, puntero a el string
92  ;;
93  myputs:
94      call string_lenght          ;mueve el tamaño del ecx en edx
95      mov eax,4
96      mov ebx,1
97      int 80h
98      ret
99
100
101 ;; string_lenght
102 ;; ============
103 ;; Cuenta el numero de caracteres hasta el terminado de cadena
104 ;; y lo almacena en edx.
105 ;;
106 ;; NOTE:
107 ;; si el string no tiene terminador esto se queda en un bucle
108 ;; infinito.
109 ;;
110 ;; Entradas:
111 ;; - ecx, puntero a el string
112 ;;
113 ;; Salidas:
114 ;; - edx, numero de caracteres
115 ;;
116 string_lenght:
117     mov edx, 0
118     mov eax, ecx
119
120 .contloop:
121     cmp byte[ecx], 0            ;si el contenido en ecx es 0
122     jz .endloop                 ;termina el contador
123
124     inc ecx                     ;inc. la posicion del puntero
125     inc edx                     ;inc. el numero de caracteres
```

```
126     jmp .contloop                   ;continua
127
128 .endloop:
129     mov ecx, eax
130     ret
131
132
133 ;; myreadfile
134 ;; ==========
135 ;; Lee el archivo P10.asm
136 ;;
137 myreadfile:
138     mov eax, 5                      ;Obtener el descriptor
139     mov ebx, filepath
140     mov ecx, 0
141     mov edx, 0
142     int 80h
143
144     mov esi, eax                    ;guardar el descriptor
145
146     mov eax, 3                      ;mueve los primeros 255
147     mov ebx, esi                    ;del archivo a file_buffer
148     mov ecx, filebuffer
149     mov edx, 255
150     int 80h
151
152     mov eax, 6                      ;cierra el archivo
153     mov ebx, esi
154     int 80h
155
156     call myputs                     ;imprime el contenido del
157     ret
```

Fuentes

Lista de interrupciones https://es.wikipedia.org/wiki/Int_80h

Descripciones de las interrupciones http://www.int80h.org/

Leer un archivo e imprimir el contenido https://stackoverflow.com/q/26963871

Mover un valor desde .data a un registro https://stackoverflow.com/a/64005239

Generalidades de NASM https://shorturl.at/lsuHS

Direccion en un arreglo https://reverseengineering.stackexchange.com/a/18711