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.

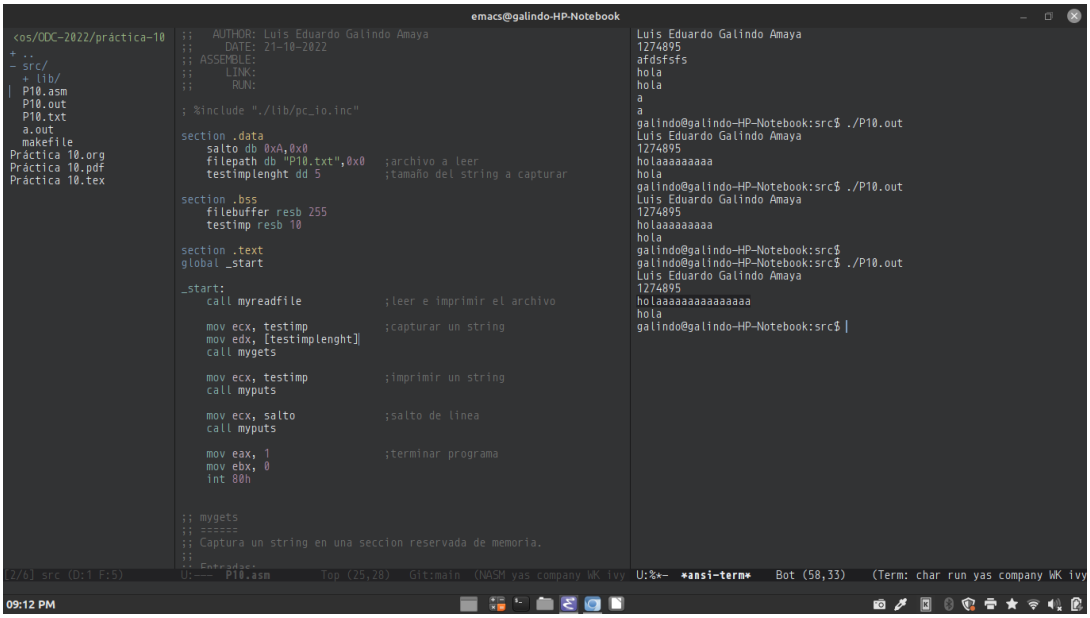
La subrutina gets solo debe poder capturar la cantidad máxima de caracteres que caben en el buffer en el que se guardará la cadena.

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.

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`

Código

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

; %include "./lib/pc_io.inc"

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

section .bss
```

```
15      filebuffer resb 255
16      testimp resb 10
17
18  section .text
19  global _start
20
21  _start:
22      call myreadfile             ;leer e imprimir el archivo
23
24      mov ecx, testimp            ;capturar un string
25      mov edx, [testimplenght]
26      call mygets
27
28      mov ecx, testimp            ;imprimir un string
29      call myputs
30
31      mov ecx, salto              ;salto de linea
32      call myputs
33
34      mov eax, 1                  ;terminar programa
35      mov ebx, 0
36      int 80h
37
38
39  ;; mygets
40  ;; ======
41  ;; Captura un string en una seccion reservada de memoria.
42  ;;
43  ;; Entradas:
44  ;; - ecx, puntero a el string
45  ;; - edx, tamaño maximo de entrada - 1 (para el terminador)
46  ;;
47  mygets:
48      dec edx
49      mov eax, 3                  ;numero de la interrupción
50      mov ebx, 0                  ;unidad de entrada
51      int 80h
52
53      ; ejecutar el servicio 3, eax contiene la cantidad de
54      ; caracteres capturados
55
56      cmp eax, edx                ;si el string capturado
57      jb .capturaesmenor
58      jmp .capturaesmayor
59
60  .capturaesmenor:                ;es menor al buffer
61      dec eax
62      mov byte[ecx + eax], 0
```

```
63      ret
64
65  .capturaesmayor:                ;es mayor al buffer
66      add ecx, edx
67      mov esi, ecx
68
69      ;deja la linea vacia para la siguiente entrada
70
71  .clearbuffer:
72      mov eax, 3                  ;numero de la interrupción
73      mov ebx, 0                  ;unidad de entrada
74      mov ecx, esi
75      mov edx, 1
76      int 80h
77
78      cmp byte[ecx], 10           ;si el contenido en ecx es 0
79      jnz .clearbuffer            ;termina el contador
80
81      mov byte[esi], 0
82      ret
83
84
85  ;; myputs
86  ;; =====
87  ;; Imprime el string en ecx.
88  ;;
89  ;; Entradas:
90  ;; - ecx, puntero a el string
91  ;;
92  myputs:
93      call string_lenght          ;mueve el tamaño del ecx en edx
94      mov eax,4
95      mov ebx,1
96      int 80h
97      ret
98
99
100 ;; string_lenght
101 ;; ============
102 ;; Cuenta el numero de caracteres hasta el terminado de cadena
103 ;; y lo almacena en edx.
104 ;;
105 ;; NOTE:
106 ;; si el string no tiene terminador esto se queda en un bucle
107 ;; infinito.
108 ;;
109 ;; Entradas:
110 ;; - ecx, puntero a el string
```

```
111 ;;
112 ;; Salidas:
113 ;; - edx, numero de caracteres
114 ;;
115 string_lenght:
116     mov edx, 0
117     mov eax, ecx
118
119 .contloop:
120     cmp byte[ecx], 0            ;si el contenido en ecx es 0
121     jz .endloop                 ;termina el contador
122
123     inc ecx                     ;inc. la posicion del puntero
124     inc edx                     ;inc. el numero de caracteres
125     jmp .contloop               ;continua
126
127 .endloop:
128     mov ecx, eax
129     ret
130
131
132 ;; myreadfile
133 ;; ==========
134 ;; Lee el archivo P10.asm
135 ;;
136 myreadfile:
137     mov eax, 5                  ;Obtener el descriptor
138     mov ebx, filepath
139     mov ecx, 0
140     mov edx, 0
141     int 80h
142
143     mov esi, eax                ;guardar el descriptor
144
145     mov eax, 3                  ;mueve los primeros 255
146     mov ebx, esi                ;del archivo a file_buffer
147     mov ecx, filebuffer
148     mov edx, 255
149     int 80h
150
151     mov eax, 6                  ;cierra el archivo
152     mov ebx, esi
153     int 80h
154
155     call myputs                 ;imprime el contenido del
156     ret
```