API Win 32

¿Qué es API Win32?

- Es una interfaz de programación de aplicaciones escrita en C por Microsoft para permitir el acceso a las funciones de Windows.
- Los componentes principales de WinAPI son:
  - WinBase: las funciones del kernel, CreateFile, CreateProcess, etc.
  - WinUser: las funciones GUI, CreateWindow, RegisterClass, etc.
  - WinGDI: las funciones gráficas, Ellipse, SelectObject, etc.
  - Controles comunes: controles estándar, vistas de lista, controles deslizantes, etc.

Documentación

- https://docs.microsoft.com/es-es/windows/win32/apiindex/windows-api-list?redirectedfrom=MSDN
- https://www.geoffchappell.com/studies/windows/win32/kernel32/history/index.htm
- https://www.pinvoke.net/default.aspx/Constants.GENERIC_READ

Como se pasan los parámetros

- Como son funciones escritas en C entonces utilizan el procedimiento estándar de paso de parámetros que veíamos la clase pasada, en el cual los parámetros de entrada se pasan por medio de la pila y el parámetro de salida se devuelve por medio de un registro de propósito general.
- Veamos algunos ejemplos

ExitProcess

ExitProcess es el método preferido para finalizar un proceso.

Esta función proporciona un cierre limpio del proceso.

- ExitProcess: procedure (

  uExitCode:uns32);

  stdcall;

  returns("eax");

  external("__imp__ExitProcess@4");

Parámetros

- uExitCode
  - Especifica el código de salida para el proceso y para todos los subprocesos que finalizan como resultado de esta llamada.
- Valor de Retorno
  - No tiene valor de retorno

Ejemplo

```
.586
.model flat, stdcall
.stack 100h
;Prototipode las funciones
ExitProcess PROTO, dwExitCode:DWORD
.data
mensaje db "mensaje"
textoR db ?
ce dd ?
.code
main PROC
        mov AX,8
        mov BX,11
        add AX,BX
 salir:  push 0
          call ExitProcess,0
main ENDP
END
```

WriteConsole

WriteConsoleW (Unicode) y WriteConsoleA (ANSI)

- WriteConsole: procedure
  (
  hConsoleOutput: dword;
  var lpBuffer: var;
  nNumberOfCharsToWrite: dword;
  var lpNumberOfCharsWritten: dword;
  var lpReserved: var
  );
  stdcall;
  returns("eax");
  external("__imp__WriteConsoleA@20");

Parámetros de entrada

- ***hConsoleOutput***
  Handle del buffer de la pantalla de la consola de salida

- ***lpBuffer***
  Apuntador *al* buffer *que contiene la cadena de salida*

- ***nNumberOfCharsToWrite***
  Especifica el número de caracteres a escribir

- ***lpNumberOfCharsWritten***
  Puntero a una variable que recibe el número de TCHARs realmente escrito. Para la versión ANSI de esta función, este es el número de bytes; Para la versión Unicode, este es el número de caracteres.

- ***lpReserved***
  Reservado; Debe ser NULL.

GetStdHandle

```
GetStdHandle: procedure
   (
   nStdHandle:dword
   );
   stdcall;
   returns( "eax" );
   external( "__imp__GetStdHandle@4" );
```

Parámetros de entrada

- *nStdHandle*
  Especifica el dispositivo estándar para el que devolver el identificador. Este parámetro puede ser uno de los siguientes valores:
  - STD_INPUT_HANDLE(-10) : entrada estándar
  - STD_OUTPUT_HANDLE(-11): salida estándar
  - STD_ERROR_HANDLE(-12): error estándar

Ejemplo

```
_EscribirConsola PROC

;Se obtiene el handle correspondiente
pushd -11    ;se coloca en la pila el parámetro para GetStdHandle
mov EBX, EAX  ;Se pasa a EBX el valor actual de EAX
call GetStdHandle   ; regresa Handle --> EAX

; Se colocan en la pila los parámetrso para WriteConsole
pushd 0
push offset num
pushd ECX
pushd EBX
pushd EAX

call WriteConsoleA
ret
_EscribirConsola ENDP

END
```

```
WriteConsole: procedure
(
hConsoleOutput: dword;
var lpBuffer: var;
nNumberOfCharsToWrite: dword;
var lpNumberOfCharsWritten: dword;
var lpReserved: var
);
```

```
.586
.model flat, stdcall
_EscribirConsola PROTO
ExitProcess PROTO, dwExitCode:DWORD
WriteConsoleA PROTO,hConsoleOutput:dword,lpBuffer:DWORD,nNumberOfCharsToWrite: dword, lpNumberOfCharsWritten:
dword,lpReserved:DWORD
ReadConsoleA PROTO, hConsoleInput: dword, lpBuffer:DWORD, nNumberOfCharsToRead: dword,lpNumberOfCharsRead:
dword,lpReserved:DWORD
GetStdHandle PROTO,  :DWORD
Beep PROTO, :DWORD, :DWORD
.stack 200h
.data
cad db "Prueba de salida", 0
cadDest db  256 dup(0), 0
num db  10
.code
main PROC
MOV ESI, OFFSET cad
MOV EDI, OFFSET cadDest
mov EAX, offset cad
mov ECX, 16
call _EscribirConsola
;Terminar El programa
INVOKE ExitProcess,0
main ENDP
```

ReadConsole

ReadConsoleW (Unicode) y ReadConsoleA (ANSI)

```
ReadConsole: procedure
    (
    hConsoleInput: dword;
    var lpBuffer: var;
    nNumberOfCharsToRead: dword;
    var lpNumberOfCharsRead: dword;
    var lpReserved: var
    );
    stdcall;
    returns( "eax" );
    external( "__imp__ReadConsoleA@20" );
```

Parámetros de entrada

- ***HConsoleInput***
  Handle al búfer de entrada de la consola.

- ***lpBuffer***
  Puntero al búfer que recibe los datos leídos del búfer de entrada de la consola.

- ***nNumberOfCharsToRead***
  Especifica el número de TCHAR a leer. Dado que la función puede leer caracteres Unicode o ANSI, el tamaño del búfer apuntado por el parámetro lpBuffer debe ser al menos de *nNumberOfCharsToRead* *sizeof(TCHAR) bytes

- ***lpNumberOfCharsRead***
  Puntero a una variable que recibe el número de TCHARs realmente leído.

- ***lpReservado***
  Reservado; Debe ser NULL.

Ejercicios

- Escribe una procedimiento que permita leer una secuencia de 4 caracteres desde la entrada estándar.

- Realiza un programa que haga un eco de una secuencia de caracteres leída mediante el procedimiento anterior.

- Realizar un programa que sume dos dígitos y muestre el resultado en pantalla.