MANUAL TECNICO

ARQUITECTURA DE COMPUTADORAS 1

Cristian Daniel Gomez Escobar
202107190

VALIDACIO DE CREDENCIALES

Para validar el acceso primero nos colocamos en el buffer que manejara nuestro método, en este caso será la variable “usuario_capturado” y “clave_capturada”, este será con el que nos estaremos moviendo.

```
validar_acceso:
        ;; abrir archivo de configuración
        mov AH, 3d
        mov AL, 00
        mov DX, offset nombre_conf
        int 21
        mov [handle_conf], AX
```

En este método podemos abrir el archivo de configuración llamado “PRA2.CNF” el cual posee todo lo que necesitaremos para la validación de credenciales, posteriormente guardaremos en el handle_conf.

```
ciclo_lineaXlinea:
        mov DI, offset buffer_linea
        mov AL, 00
        mov [tam_liena_leida], AL
```

Con este método del ciclo_lineaXlinea, procederemos a limpiar el buffer e inicializar el tamaño de la línea leída

```
ciclo_obtener_linea:
        mov AH, 3f
        mov BX, [handle_conf]
        mov CX, 0001
        mov DX, DI
        int 21
        cmp CX, 0000
        je fin_leer_linea
        mov AL, [DI]
        cmp AL, 0a
        je fin_leer_linea
        mov AL, [tam_liena_leida]
        inc AL
        mov [tam_liena_leida], AL
        inc DI
        jmp ciclo_obtener_linea
```

En este procederemos a leer la línea hasta que se detecte un salto de la misma, la cual será la señal de que hemos terminado de leerla.

```
fin_leer_linea:
        mov AL, [tam_liena_leida]
        mov AL, 00
        cmp [estado], AL    ;; verificar la cadena credenciales
        je verificar_cadena_credenciales
        mov AL, 01
        cmp [estado], AL    ;; obtener campo
        je obtener_campo_conf
        mov AL, 02
        cmp [estado], AL    ;; obtener campo
        je obtener_campo_conf
        jmp retorno_exitoso
```

Para finalizar la lectura del archivo incrementaremos los estados actuales para obtener los campos de dicho objeto, esto con el fin del retorno exitoso o del fin de la línea.

```
verificar_cadena_credenciales:
        cmp CX, 0000; se verifica si se leyó
        je retorno_fallido ; si no se leyó, se termina
        mov CH, 00 ; se limpia CH
        mov CL, [tk_creds]; se recupera el tamaño de la cadena
        mov SI, offset tk_creds; se recupera el tamaño de la cadena
        inc SI; se recupera el tamaño de la cadena
        mov DI, offset buffer_linea; se recupera el tamaño de la cadena
        call cadenas_iguales; se recupera el tamaño de la cadena
        cmp DL, 0ff; se verifica si son iguales
        je si_hay_creds; si son iguales, se incrementa el estado
        jmp retorno_fallido; si no son iguales, se termina
```

Con esto podremos verificar si se ha leído la cadena y verificara si las cadenas de las credenciales son iguales para poder continuar con las demás verificaciones o de lo contrario retornara como fallido.

MENU PRINCIPAL

Aquí procederemos a leer la opción que el usuario desea ejecutar, esto lo haremos leyendo el código ASCII que a introducido para verificar dicha acción y desplazarse a la opción deseada.

```
    cmp al, 31;codigo ascii del numero 1 en hexadecimal
    je ingresar_producto_archivo
    cmp al, 33;codigo ascii del numero 3 en hexadecimal
    je mostrar_productos_archivo
    cmp al, 34;codigo ascii del numero 4 en hexadecimal
    je menu_principal
    jmp menu_productos
    ;;
```

```
    mov DI, offset buffer_entrada
    inc DI
    mov AL, [DI]
    cmp AL, 00
    je  pedir_de_nuevo_codigo
    cmp AL, 05
    jb  aceptar_tam_cod  ;; jb --> jump if below
    mov DX, offset nueva_lin
    mov AH, 09
    int 21
    jmp pedir_de_nuevo_codigo
    ;;; mover al campo codigo en la estructura producto
```

Para solicitar el código del producto solicitaremos el tamaño del campo en el buffer de entrada, luego de posicionar el buffer verificaremos que el tamaño sea el que se le ha solicitado al programa, de cumplirse procederá a aceptar la cadena ingresada

```
aceptar_tam_cod:
        mov SI, offset cod_prod
        mov DI, offset buffer_entrada
        inc DI
        mov CH, 00
        mov CL, [DI]
        inc DI  ;; me posiciono en el contenido del buffer
```

Con esto podremos solicitar el tamaño del código posicionando el código en el puntero

```
copiar_cod:
    mov AL, [DI]
    cmp al, 'A'
    jl pedir_de_nuevo_codigo  ; Si no es una letra mayúscula o número válido, volver a pedir la cadena
    cmp al, 'Z'
    jle copiar_codigo
    cmp al, '0'
    jl pedir_de_nuevo_codigo  ; Si no es una letra mayúscula o número válido, volver a pedir la cadena
    cmp al, '9'
    jg pedir_de_nuevo_codigo  ; Si no es una letra mayúscula o número válido, volver a pedir la cadena
```

En esta parte verificaremos con un análisis léxico si el código cumple con lo requerido.

```
copiar_codigo:
        mov AL, [DI]
        mov [SI], AL
        inc SI
        inc DI
        loop copiar_codigo  ;; restarle 1 a CX, verificar que CX no sea 0, si no es 0 va a la etiqueta,
        ;;; la cadena ingresada en la estructura
        ;;;
        mov DX, offset nueva_lin
        mov AH, 09
        int 21
```

Por último copiaremos el código en el archivo solicitado, esta misma lógica se pondrá para los datos del producto que necesitemos ingresar.

ELIMINACION DE PRODUCTOS

```
eliminar_producto_archivo:
        mov DX, offset nueva_lin
        mov AH, 09
        int 21
        print eliminado
        mov DX, offset nueva_lin
        mov AH, 09
        int 21

        mov DX, 0000
        mov [puntero_temp], DX
```

Nos ´posicionamos en el archivo donde están nuestros productos para posteriormente guardar en memoria lo cambios realizados

REPORTE DE ABC

```
generar_ABC:
        mov AH, 3c; crear archivo
        mov CX, 0000 ; atributos
        mov DX, offset nombre_rep2 ; nombre del archivo
        int 21 ; llamada al sistema
        mov [handle_reps], AX ; guardamos el handle del archivo
        mov BX, AX ; guardamos el handle del archivo
        mov AH, 40
        mov CH, 00
        mov CL, [tam_encabezado_html]
        mov DX, offset encabezado_html
        int 21
        mov BX, [handle_reps]
        mov AH, 40
        mov CH, 00
        mov CL, [tam_inicializacion_tabla_ABC]
        mov DX, offset inicializacion_tabla_ABC
```

Generaremos el reporte de la siguiente manera, en donde con nuestra variable nombre_rep2, generaremos el documento html e indicar la cantidad de bytes que voy a escribir, guardare en el handle_reps indicando el puntero de la cadena a escribir.

```
mostrar_productos_archivo_c:
        mov DX, offset nueva_lin ; nueva linea
        mov AH, 09
        int 21
        ;;
        mov AL, 02 ; abrir archivo
        mov AH, 3d ; abrir archivo
        mov DX, offset archivo_prods ; nombre del archivo
        int 21  ; llamada al sistema
        ;;
        mov [handle_prods], AX ; guardamos el handle del archivo
        ;; leemos
```

Procederemos a guardar el handle_prods

```
imprimir_estructura_html_ABC:
        mov BX, [handle_reps] ; guardamos el handle del archivo
        mov AH, 40 ; escribir en archivo
        mov CH, 00 ; cantidad de bytes a escribir
        mov CL, 04 ; cantidad de bytes a escribir
        mov DX, offset tr_html ; puntero a la cadena a escribir
        int 21 ; llamada al sistema
        ;;
        mov BX, [handle_reps] ; guardamos el handle del archivo
        mov AH, 40 ; escribir en archivo
        mov CH, 00 ; cantidad de bytes a escribir
        mov CL, 04 ; cantidad de bytes a escribir
        mov DX, offset td_html ; puntero a la cadena a escribir
        int 21
```

Aquí guardaremos en el handle el archivo y poder escribir dentro de el, en este caso le escribiremos la estructura inicial del contenido de la tabla de cada una de las letras iniciales.

```
mov BX, [handle_reps]
mov AH, 40
mov CH, 00
mov CL, 01
mov DX, offset a
int 21

mov BX, [handle_reps]
mov AH, 40
mov CH, 00
mov CL, 05
mov DX, offset tdc_html
int 21
;;
mov BX, [handle_reps]
mov AH, 40
mov CH, 00
mov CL, 04
mov DX, offset td_html
int 21


mov BX, [handle_reps]
mov AH, 40
mov CH, 00
mov CL, 05
mov DX, offset numero
int 21
```

Aquí es donde escribiremos la letra y su contador correspondiente