A **Apellidos** _____ **Nombre** _____ **DNI** _____

Examen de Arquitectura de Computadores. (Telemática) **Convocatoria de enero, segunda parte: 24-01-2011**

A continuación se muestra el listado de un programa escrito en C encargado de obtener información sobre la unidad de CD-ROM de un computador. Lo que se pretende averiguar es si el computador dispone de una unidad de CD-ROM (asumimos que de existir, tiene asignada la unidad D:) y, en caso afirmativo, hallar el número total de bytes del CD que esté dentro. Para ello se utilizan las funciones del API de WIN32 *GetLogicalDrives* y *GetDiskFreeSpace*, cuya especificación se incluye en el anexo de la última página.

La salida que debe mostrar el programa (en caso de que no falle ninguna función) es la siguiente:

> Informacion sobre la unidad D:
>
> Espacio total = 911360 bytes

Donde el número de bytes mostrados tiene que haberse obtenido de la información proporcionada por la función *GetDiskFreeSpace*.

```
#include <windows.h>
#include <stdio.h>

int main()
{
	BOOL resultado;
	int total;
	char unidad[]="D:\\";

	DWORD unidades;
	DWORD mascara;

	DWORD dwSectPerClust;
	DWORD dwBytesPerSect;
	DWORD dwFreeClusters;
	DWORD dwTotalClusters;

	unidades=GetLogicalDrives();
	//Comprobar si existe la unidad D. Si NO existe, se muestra
	//un mensaje de error. Utilizar la información devuelta por
	//la fucnión GetLogicalDrives.
```

0,75

```
	mascara=8;
	unidades = unidades & mascara;

	if (unidades==0                          ){
		printf("No existe la unidad D\n");
		return -1;
	}
	//Llamar a la función GetDiskFreeSpace para obtener
	//información sobre la unidad D
```

0,75

```
	resultado = GetDiskFreeSpace(unidad, &dwSectPerClust,
	            &dwBytesPerSect, &dwFreeClusters,
	            &dwTotalClusters);

	//Si todo fue bien, muestro el número total de BYTES del disco
	//Antes de mostrar este valor por salida estándar lo calculo
	//y lo guardo en la variabe total
	if (resultado!=0){
```

0,5

```
		total = dwTotalClusters*dwSectPerClust*dwBytesPerSect;

		printf("Informacion sobre la unidad D:\n");
		printf("Espacio total = %d bytes\n",total);
	}
	else
		printf("Error llamando a GetDiskFreeSpace");

    return 0;
}
```

— Responde a las siguientes preguntas.

0,75

¿Por qué en un sistema operativo moderno se puede ejecutar un programa que sea más grande que la memoria física?

Porque los sistemas operativos modernos permiten utilizar parte del disco duro como una extensión de la memoria física, lo que nos permite cargar programas más grandes que la propia memoria.

En un sistema de memoria virtual paginada, ¿cómo se evita que un proceso no escriba en posiciones físicas asignadas a otro proceso?

Utilizando una tabla de páginas distinta para cada proceso, en la cual solo se mapean las páginas propias de cada proceso (además de ciertas partes del SO).

¿Qué almacena la TLB?

Las entradas de la tabla de páginas más frecuentemente utilizadas.

A

— Explica la diferencia fundamental existente entre el mecanismo de planificación de un sistema operativo multiprogramado y el de un sistema operativo de tiempo compartido.

0,5

En los sistemas multiprogramados, los programas solo abandonan la CPU cuando esperan por un periférico.

En los sistemas de tiempo compartido, se asigna a los programas un tiempo máximo de ejecución continuada, que recibe el nombre de quantum. Entonces un programa abandona la CPU si tiene que esperar por un periférico, o bien si ha agotado su quantum de ejecución.

— Contesta a las siguientes preguntas breves:

0,75

¿Qué es un SDK? Indica los elementos habituales que contiene.

Conjunto de herramientas (tales como librerías, ficheros de cabecera, etc.), que permiten desarrollar componentes software que usen los servicios de otro componente software que actúa como proveedor.

Indica la razón por la que un proceso que se encuentra en el estado "en ejecución" pasa al estado "listo"

Porque ha expirado su quantum.

¿Qué ocurre cuando la MMU genera una dirección virtual no asignada (ni a memoria física ni a disco) y cómo las trata el sistema operativo?

Se genera una excepción de protección general, que indica una violación de acceso a memoria. Esto provoca una transferencia de control al SO, el cual elimina el programa causante de la excepción.

— Define los tres objetivos básicos de un sistema operativo.

0,75

Proporcionar una interfaz amigable para la interacción entre el usuario y el computador.

Proporcionar un entorno de funcionamiento para los programas. Así el sistema operativo proporciona un conjunto de servicios que pueden ser solicitados por los programas.

Coordinar el uso de los recursos hardware del computador entre los programas que se encuentran en ejecución en cada instante.

A continuación se muestra el listado de un programa en C. Rellena los huecos presentes en él según las indicaciones de en los comentarios.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>

#define SIZE 4096

int main()
{
	int vector[SIZE];
	int i;
	int * puntero;
	BOOL resultado;

	for (i=0;i<SIZE;i++)
		vector[i]=i;
	//Reserva y compromiso de una zona de memoria en la que
	//se va a escribir. Permitir al SO que elija su ubicación
```

0,75

```
	puntero=VirtualAlloc(NULL,SIZE*4,
	                     MEM_RESERVE|MEM_COMMIT,PAGE_READWRITE);
```

```
	//Si ha fallado la reserva, mostrar un mensaje de error
	if (puntero==NULL){
		printf("Error reservando memoria\n");
		return -1;
	}
```

A **Apellidos** _____ **Nombre** _____ **DNI** _____

Examen de Arquitectura de Computadores. (Telemática) **Convocatoria de enero, segunda parte: 24-01-2011**

```
      //Tratar la zona reservada como un vector de enteros.
      //Utilizarla para guardar una copia del vector "vector"
      //Realizar esta copia mediante un bucle for
      //y SIN UTILIZAR el operador []
```

0,75

```
for (i=0;i<SIZE;i++){
      *(puntero+i)=*(vector+i);
}
```

```
      //Liberar completamente la zona de memoria reservada
      //y comprometida previamente
```

0,5

```
resultado=VirtualFree(puntero,0,MEM_RELEASE);
```

```
      if (resultado==0){
            printf("Error al liberar memoria\n");
            return -2;
      }
   _getch();
   return 0;
}
```

— En una ejecución del programa anterior, la dirección devuelta por la función *VirtualAlloc* fue **00 3C 00 00h**.¿Cuál es la ultima dirección virtual de la zona reservada? Contestar en hexadecimal.

00 3C 3F FF **0,5**

Se dispone de un computador con las siguientes características: maneja direcciones virtuales de 20 bits y direcciones físicas de 16 bits. Utiliza un sistema de gestión de memoria virtual paginada cuyo tamaño de página está fijado en 1024 bytes. Cada posición de memoria almacena 1 byte. La sección de código de los programas ejecutados en este computador comienzan siempre a partir de la página virtual 040h, la sección de datos a partir de la página 100h y la de pila a partir de la 3FFh (la pila crece hacia direcciones menos significativas).

En la siguiente figura se muestra el estado en que se encuentra la tabla de páginas en un determinado instante (solo se muestran las entradas correspondiente a páginas virtuales utilizadas por el programa).

En base a esta información, responder a las siguientes preguntas.

— Si la MMU envía al bus de direcciones la dirección física **10EEh**, ¿qué dirección virtual tuvo que generar la unidad de ejecución? Contestar en hexadecimal.

4 04 EE **0,5**

Tabla de Páginas

Nº Pag. Virtual	Pre-sencia	Read / Write	Nº Pag. Fis. / Offset Dis.
040	No	Read	Offset X
041	Sí	Read	00
042	Sí	Read	10
100	Sí	Read/Write	06
101	Sí	Read/Write	04
102	No	Read/Write	Offset Y
3FE	Sí	Read/Write	12
3FF	Sí	Read/Write	??

— Si la unidad de ejecución genera la dirección virtual **10AA0h**, ¿qué dirección física enviará la MMU al bus de direcciones? Contestar en hexadecimal.

42 A0 **0,5**

— Las dos primeras páginas de la sección de datos albergan un vector de 512 enteros (cada entero ocupa 4 bytes). ¿Qué elemento de ese vector ocupa el rango de direcciones virtuales 4**03F4h – 403F7h**?

Nota: El primer elemento del vector es el que tiene el índice 0.

Elemento 253 **0,75**

— Se sabe que la memoria física instalada en este computador ocupa el 50% del espacio de direcciones físico. Si la primera página de la pila (la más significativa) está mapeada en el marco más significativo de la memoria física, ¿qué número de marco debería aparecer en su correspondiente entrada en la tabla de páginas? (En la figura de la página anterior aparece con dos interrogantes). Contestar en hexadecimal.

1F **0,5**

— ¿Cuál es la dirección virtual más significativa que generaría una excepción de protección de memoria al intentar realizar una escritura sobre ella (**SIN** tener en cuenta las direcciones correspondientes a páginas no utilizadas por el programa)? Contestar en hexadecimal.

1 0B FF **0,5**

GetLogicalDrives

La función **GetLogicalDrives** proporciona una máscara de bits que representa las unidades de disco disponibles en ese momento.

```
DWORD GetLogicalDrives(void);
```

Parámetros

Esta función no recibe parámetros.

Valores de retorno

Si la función tiene éxito, el valor devuelto es una máscara de bits que representa las unidades de disco disponibles en ese momento. El bit de posición 0 (el menos significativo) es la unidad A, el bit de posición 1 es la unidad B, el bit de posición 2 es la unidad C y así sucesivamente. Si el bit correspondiente a una unidad está activo (valor 1) significa que esa unidad está disponible y no lo está en caso contrario. Si la función falla, el valor devuelto es cero.

GetDiskFreeSpace

La función **GetDiskFreeSpace** proporciona información sobre el disco especificado, incluyendo la cantidad de espacio libre en el disco.

```
BOOL GetDiskFreeSpace(
  LPCTSTR lpRootPathName,
  LPDWORD lpSectorsPerCluster,
  LPDWORD lpBytesPerSector,
  LPDWORD lpNumberOfFreeClusters,
  LPDWORD lpTotalNumberOfClusters
);
```

Parámetros

lpRootPathName
[in] Puntero a una cadena que especifica el directorio raíz del disco para el cual se va a proporcionar la información.

lpSectorsPerCluster
[out] Puntero a la variable que recibe el número de sectores por clúster.

lpBytesPerSector
[out] Puntero a la variable que recibe el número de bytes por sector.

lpNumberOfFreeClusters
[out] Puntero a la variable que recibe el número de clústeres libres en el disco.

lpTotalNumberOfClusters
[out] Puntero a la variable que recibe el número total de clústeres del disco.

Valores de retorno

Si la función tiene éxito, el valor devuelto es distinto de cero.

Nota: Un clúster (o unidad de asignación) es un conjunto de sectores contiguos que componen la unidad más pequeña de almacenamiento de un disco.