Prácticas curso 2021-22.

Software de Sistemas. 3º Ingeniería de Computadores.

[Práctica 0. Configurar el entorno de desarrollo para ejercicios/prácticas. 3](#_Toc88125084)

[Práctica 1. C/C++ para sistemas (I). 21](#_Toc88125085)

[Práctica 2. C/C++ para sistemas (II). 27](#_Toc88125086)

[Práctica 3. Bibliotecas estáticas. 33](#_Toc88125087)

[Práctica 4. Bibliotecas dinámicas. 41](#_Toc88125088)

[Práctica 5. Ejercicios Tema 4. 47](#_Toc88125089)

[Práctica 6. Depuración. Manipulando imágenes. 55](#_Toc88125090)

[Práctica 7. Ejercicios control de ejecución (tema 5). 65](#_Toc88125091)

[Práctica 8. Ejercicios de control de concurrencia y sincronización (tema 8). 71](#_Toc88125092)

[Práctica 9. Comunicación entre procesos: tuberías. 77](#_Toc88125093)

[Práctica 10. Comunicación entre procesos: tuberías. Sesión 2. 84](#_Toc88125094)

Última actualización:

16/12/2021.

**Retrasadas fechas de entrega de prácticas 7 y 8 hasta después de las vacaciones de navidad.**

18/11/2021.

Prácticas 7-10. Actualización completa para curso 2021/22 (cambio a repositorio Git).

26/09/2021.

Práctica 0. Actualización completa para curso 2021/22 (cambio a repositorio Git).

28/09/2021.

Práctica 0: Aclarada forma de acceso a máquina virtual con software de prácticas (pg. 7).

Prácticas 1-4: actualizados enlaces, fechas y contenidos a curso 2021/22

11/19/2021.

Práctica 3: Se añade una sesión adicional.

1. Configurar el entorno de desarrollo para ejercicios/prácticas.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20210927/20210929).

27-29 de septiembre de 2021.

Contenido

[0.1. Qué hacer en el laboratorio. 4](#_Toc88125095)

[0.2. Antes de empezar… 5](#_Toc88125096)

[0.2.a. Instalar software. 5](#_Toc88125097)

[0.2.b. Máquinas con el software preinstalado. 8](#_Toc88125098)

[0.3. Pasos para la configuración del software. 8](#_Toc88125099)

[0.3.a. Comprobar links de los repositorios. 8](#_Toc88125100)

[0.3.b. Crear/comprobar el árbol de directorios para prácticas. 9](#_Toc88125101)

[0.3.c. Asociar directorio \_pub con el repositorio público. 9](#_Toc88125102)

[0.3.d. Asociar directorio privado con el repositorio privado. 11](#_Toc88125103)

[0.3.e. Ajustar variables de entorno. 11](#_Toc88125104)

[0.3.a. Ajustar variable de entorno Path. 12](#_Toc88125105)

[0.3.b. Programa principal mínimo C/C++. 13](#_Toc88125106)

[0.4. Antes de cada práctica. 13](#_Toc88125107)

[0.5. Crear y usar una plantilla de proyecto. 14](#_Toc88125108)

[0.5.a. Antes de empezar… 14](#_Toc88125109)

[0.5.b. Ejecutando el asistente Archivo🡪Exportar plantilla… 15](#_Toc88125110)

[0.5.c. Instalar/eliminar una plantilla de Visual Studio. 16](#_Toc88125111)

[0.5.a. Usar una plantilla instalada. 16](#_Toc88125112)

[0.6. Reutilizar código y opciones de un proyecto anterior sin plantilla. 18](#_Toc88125113)

[0.7. Anexo: Qué se puede hacer con el cliente TortoiseGit. Operaciones más comunes. 18](#_Toc88125114)

[0.7.a. Antes de empezar: conceptos básicos. 18](#_Toc88125115)

[0.7.b. Añadir archivos/carpetas al repositorio. 18](#_Toc88125116)

[0.7.c. Eliminar archivos/carpetas al repositorio. 19](#_Toc88125117)

[0.7.d. Actualizar repositorio (commit+push). 19](#_Toc88125118)

[0.7.e. Eliminar una copia local del repositorio (working copy). 20](#_Toc88125119)

* 1. Qué hacer en el laboratorio.
* En la primera parte se comprueba la instalación de los equipos y el manejo básico de las herramientas.
* Se debe traer los equipos de casa con el software ya instalado.
* Para dudas sobre la instalación, enviar un correo a [ss@atc.us.es](mailto:ss@atc.us.es).
  + - En el lab. También se resuelven dudas/problemas de instalación en esos equipos.
* Las personas que no traigan sus equipos pueden usar los de prácticas.
* En la segunda parte vemos parte del tema 2: manejo básico del depurador (sección 2.5).
* Usamos ejercicio 0.1 y 0.2 para ello.

**La práctica 0 contiene información aplicable a todas las prácticas que siguen.**

* 1. Realizar un programa que muestre por pantalla las variables del sistema.  
       
     Ver posibles definiciones de main en la sección *Tema 1🡪Más detallas🡪La función main* en las notas de clase.   
       
     Proyecto inicial en carpeta \_pub\Practicas\P00 del repositorio público, aunque **se recomienda comenzar el proyecto desde cero**[[1]](#footnote-2).  
      Ver documento ***00.05\_AntesDeEmpezar.docx*** y   
      ***00.10\_CrearNuevoProyectoCpp.docx*** en la carpeta [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)/Recetas.

\* Realizar una operación Commit + Push con el proyecto inicial,  
 \* y otra con el ejercicio terminado.

* 1. Modificar el programa anterior para que muestre por pantalla todos los argumentos que se le pasan a main.

\* Realizar una operación Commit + Push con el proyecto inicial,  
 \* y otra con el ejercicio terminado.

**La práctica termina aquí.**

**Ver más detalles sobre las operaciones de configuración instalación en las páginas siguientes.**

## Mas detalles sobre instalación/configuración del software.

* 1. Antes de empezar…

Consideraciones generales:

* Todos los enlaces de los repositorios necesarios están disponibles en EV (página de la asignatura en ev.us.es).
* Caso de que se pida usuario-contraseña al acceder a un repositorio, usar **UVUS**.
  + 1. Instalar software.

#### Visual Studio 2019 (VS2019). Instalación.

**Dónde descargar VS2019:**

* Página oficial de descarga: <https://visualstudio.microsoft.com/es/downloads/>.
* Usar botones "Descarga gratuita" o "Prueba gratuita"
* Las versiones Professional y Enterprise tienen 30 días de evaluación. Después piden clave[[2]](#footnote-3).
* Todos los proyectos/ejemplos de clase se han hecho con la versión Professional.

Se puede usar la cualquiera de las versiones (*community/profesional/enterprise*).

* Solo es necesaria la instalación básica con los siguientes componentes opcional:
* 1/2: Cargas de trabajo-> Desarrollo para el escritorio con C++.
* Solo son necesarios los componentes que aparecen seleccionados en la imagen.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* 2/2: Componentes individuales-> buscar "git" 🡪 seleccionar "GIT para Windows".

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

#### Visual Studio 2019 (VS2019). Primera ejecución.

En la primera ejecución se configuran algunas opciones importantes del entorno de desarrollo.

En la pantalla inicial debemos iniciar sesión con nuestro UVUS completo:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Si aparece esta pantalla, seleccionar la opción "Cuenta profesional o educativa"[[3]](#footnote-4):

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Introducimos nuestras credenciales UVUS[[4]](#footnote-5):

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Seleccionamos configuración inicial:

* **Seleccionar "Configuración de desarrollo:" Visual C++.**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

#### Clientes Git: [TortoiseGit](https://tortoisegit.org/download/).

Con las opciones de instalación de VS2019 de los apartados anteriores ya se instala el soporte Git por defecto de VS2019.

Se recomienda instalar también el cliente Git TortoiseGit, con varias características adicionales interesantes, como pueden ser indicaciones visuales del estado de los repositorios.

Instalación.

Seleccionamos las opciones de instalación por defecto hasta llegar a esta ventana:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

La ruta que debemos copiar es:

C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer\Git\mingw32\bin\git.exe

En la siguiente ventana rellenamos los datos de identificación, con el correo UVUS y nombre.

Tran la instalación, es recomendable reiniciar el PC.

* + 1. Máquinas con el software preinstalado.

~~Las máquinas de los laboratorios de prácticas tienen instalado el software de la sección 0.2.a. Instalar software.~~

* ~~Puede acceder por escritorio remoto en la dirección~~ [~~https://openlabs-devel.us.es/?tab=undefined~~](https://openlabs-devel.us.es/?tab=undefined)~~.~~
* ~~Acceso con uvus.~~

También puede descargar una máquina virtual (Virtualbox) [aquí](https://uses0-my.sharepoint.com/personal/camaya_us_es/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fcamaya%5Fus%5Fes%2FDocuments%2FPubs%2FSS%2DMaquinas%20Virtuales&originalPath=aHR0cHM6Ly91c2VzMC1teS5zaGFyZXBvaW50LmNvbS86ZjovZy9wZXJzb25hbC9jYW1heWFfdXNfZXMvRWlIc0F3WU9zR2hKa3BUV0pCQkxMWm9CblFEeW9wNzNHR3pWWHpUUUZvd2x0UT9ydGltZT1Sc0JpQV8xdTJFZw).

* Acceso al archivo con su correo [uvus@alum.us.es](mailto:uvus@alum.us.es).

También puede descargar una máquina virtual (Virtualbox).

* En enlace solo está disponible a los usuarios de la US:
* Conectar primero a <https://www.office.com/>.
* Acreditarse con su correo de la US: …@alum.us.es.
* Acceder a [este](https://uses0-my.sharepoint.com/personal/camaya_us_es/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fcamaya%5Fus%5Fes%2FDocuments%2FPubs%2FSS%2DMaquinas%20Virtuales&originalPath=aHR0cHM6Ly91c2VzMC1teS5zaGFyZXBvaW50LmNvbS86ZjovZy9wZXJzb25hbC9jYW1heWFfdXNfZXMvRWlIc0F3WU9zR2hKa3BUV0pCQkxMWm9CblFEeW9wNzNHR3pWWHpUUUZvd2x0UT9ydGltZT1Sc0JpQV8xdTJFZw) enlace.
* El usuario de la máquina virtual es ss, clave *Escuela*.
  1. Pasos para la configuración del software.
     1. Comprobar links de los repositorios.

**Debe hacerse al menos una vez por cada usuario que vaya a hacer prácticas en un PC.**

Cada estudiante debe usar 2 enlaces a repositorios para el software de la asignatura:

* **Público**: accesible a todos.
* <https://github.eii.us.es/IC3-SS/_pub.git>
* Código visto en clase.
* Elementos compartidos: bibliotecas, .h, archivos de ejemplo, etc.
* Enlace también disponible en EV (“Información/Enlaces SVN para el curso”).
* Probar que es válido con cualquier navegador.
* Este repositorio es de solo lectura: no se pueden subir modificaciones.
* **Privado**: accesible a estudiante y profesor. Distinto para cada estudiante.
* Disponible a partir del viernes 24 de septiembre de 2021.
* Enlace en EV (Mis calificaciones/URL de repositorio privado, + click en Mostrar comentario:Calificaciones)

Imagen que contiene captura de pantalla

Descripción generada con confianza muy alta

* Probar que el enlace privado es válido con cualquier navegador.
* No usar directamente el enlace, entrar primero en <https://github.eii.us.es/login>.
  + - Se pedirá usuario-contraseña UVUS.
    - Usar después el enlace del repositorio privado[[5]](#footnote-6).
* Este repositorio es de lectura y escritura.
  + 1. Crear/comprobar el árbol de directorios para prácticas.

**Debe hacerse al menos una vez por cada usuario que vaya a hacer prácticas en un PC.**

* Crear la carpeta **SSRoot** en cualquier lugar del sistema de archivos del disco duro.

* + 1. Asociar directorio \_pub con el repositorio público.

Esta operación se denomina **clonar** en terminología Git:

En **TortoiseGit**, seleccionamos la carpeta **SSRoot**, botón derecho del ratón, Git Clone:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Aparece la ventana:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Nos debemos asegurar que:

* La url corresponde a la del repositorio público (se puede comprobar con un navegador).
* La carpeta …\SSRoot\\_pub no existe o está vacía.

Un vez terminado la clonación, aparece una ventana similar a:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se descargarán los archivos desde el repositorio hasta el directorio local[[6]](#footnote-7).

TortoiseGit muestra una marca de check verde en la carpeta indica que los archivos que contiene son idénticos a los del repositorio.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Al directorio local se le denomina **copia de trabajo** (**working copy**) del repositorio.

* + 1. Asociar directorio privado con el repositorio privado.

Las operaciones de configuración son las mismas que en repositorio público, cambiando la carpeta y la url del repositorio.

* Comprobar que el repositorio funciona añadiendo un archivo y realizando una operación **Commit+Push[[7]](#footnote-8)**.

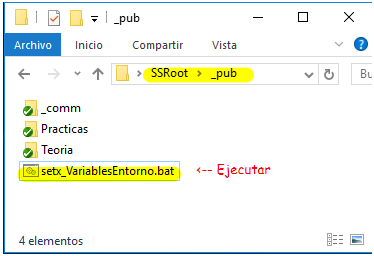
MUY IMPORTANTE.

Información sobre privacidad: Por favor, téngase en cuenta que todas las operaciones que se realizan sobre un repositorio son registradas y visibles a las personas con acceso de lectura (incluido el profesor).

* + 1. Ajustar variables de entorno.

Se usará la variable de entorno **AAComm** para almacenar la carpeta donde están las bibliotecas y archivos comunes a los proyectos C de la asignatura.

Hay varias alternativas para asignar *AAComm*. La más simples es ejecutar el siguiente archivo .bat (en el repositorio público):



Contiene la instrucción:

setx **AAComm** %cd%\\_comm

Que asigna a la variable *AAComm* la ruta en disco de la carpeta \_*comm*.

* + 1. Ajustar variable de entorno Path.

Solo es necesario si se quiere ejecutar operaciones GIT desde la línea de comandos.

* Localizar el directorio donde se encuentra el programa git.exe[[8]](#footnote-9).
* En Windows, ejecutar *Editar las variables de entorno de esta cuenta*.
* En la ventana *Variables de entorno*, seleccionar *Path* y el botón *Editar*…. Después pulsar *Editar Texto…*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente 🡪 Texto, Tabla

Descripción generada automáticamente con confianza media

* En la ventana que aparece, pulsar añadir al valor de la variable la variable la ruta donde esté el programa git.exe y pulsar aceptar:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para probar que funciona, abrir una ventana de comandos ejecutando cmd.exe y teclar git:

Texto

Descripción generada automáticamente

Muy importante: para aplicar un comando git a un repositorio debe ejecutarse git.exe en alguna de los directorios del repositorio.

* + 1. Programa principal mínimo C/C++.

#include <stdio.h>

#include <conio.h>

#include <locale.h>

int main(int argc, char \*argv[], char \*envp[])

{

setlocale(LC\_ALL, "Spanish");

printf("\nPor favor, pulse una tecla para terminar ...");

\_getch();

return 0;

}

* 1. Antes de cada práctica.
* Actualizar la carpeta \_pub con el contenido del **repositorio remoto**.
* Seleccionar la carpeta, TortoiseGit🡪Pull:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* 1. Crear y usar una plantilla de proyecto.

La mayoría de entornos de desarrollo permiten reutilizar de alguna manera la configuración y archivos de un proyecto para crear otros.

En el caso de VS, se hace con las **plantillas de proyecto.**

La forma más simple de crear una plantilla de proyecto es usar el comando *Archivo🡪****Exportar Plantilla…***, que lanza un asistente para crear la plantilla. Es sencillo, pero tiene algunas limitaciones, que veremos más adelante.

Una segunda posibilidad es copiar el proyecto y pegarlo en una nueva carpeta, modificando los archivos para exportar "a mano" las configuraciones de proyectos[[9]](#footnote-10). En proyectos simples, como los que veremos en esta asignatura, es una alternativa a tener en cuenta.

* + 1. Antes de empezar…

Debemos asegurarnos de que:

* Hemos eliminado elementos innecesarios o que no se usen:
* Archivos, código fuente comentado, etc.
* Plataformas (x64) y configuraciones sin uso.
* El proyecto no tiene filtros o archivos dentro de los filtros. Los filtros son las "carpetas" que aparecen en el explorador de soluciones. Por ejemplo, en la siguiente imagen, Project1 no se exportará correctamente:


Imagen sin título.png Texto alternativo generado por el equipo:
bb dema-SaIPOI 
Archivo Editar Ver 
Explorador de soluciones 
Microsoft Visual Studia 
Proyecto Compilar Depurar 
Debug 
Herramier 
Buscar en Explorador de soluciones (Ctrl+) p 
Solución 'bb_dema-SaIPOI' (3 proyectos) 
bb_demo-POI 
Projectl 
Referencias 
Dependencias 
Archivos de código fuente 
Saurce.cpp 
Archivos de encabezado 
Archivos de recursos 
Project2 
• Referencias 
Dependencias 
Archivas de código fuente 
Archivos de encabezado 
Archivos de recursos 
Saurce2.cpp 

Recorte de pantalla realizado: 19/10/2017 23:26

<-- Los archivos aquí      NO se exportan
<-- Los archivos fuera de las     carpetas de filtro Sí se exportan
<-- Mostrar todos los archivos desactivado…


* El proyecto compila y ejecuta correctamente. En este ejemplo, usaremos el proyecto terminado del Ejercicio 1.21. para hacer una plantilla con la que creamos el proyecto inicial de la práctica 2.
* El asistente no permite exportar automáticamente las opciones de depuración del proyecto[[10]](#footnote-11), por lo que, si queremos hacerlo, lo más simple es copiarlas y pegarlas como comentario en el código fuente, y configurarlas a mano en el nuevo proyecto:

Imagen sin título.png Archivo 
Editar 
Ver 
Micrasart Visual StucllO 
Proyecto Compilar Depurar 
Debug 
Herramientas 
x86 
Explorador de soluciones X 
bb_demo-POI .cpp* -E X 
bb dema-POI 
Visua[SVN Ventana Ayuda 
Depurador local de Windows • Automática 
(Amblta global) 
Buscar en Explorador de salu p 
Solución 'bb dema-SaIPOI' ( 
bb_demo-POI 
• Referencias 
Dependencias 
bb dema-POI .cpp 
bb dema-POI .h 
Páginas cle propiedades cle bb_cIemc-POI 
18 
31 
32 
64 
66 
67 
68 
S] int main(int 
argc, char 
argv 
9 static void play(const char *file_name, 
const 
char 
nombre 
Dir. Trabajo: 
Entorno: 
Active (W i n32) 
$ (AAComm) \ eje 
S(TargetPath) 
Automática 
\bin $(Loca1DebuggerEnvironn 
Administrador de configuración... 
Configuración: Todas las config. 
Propiedades de configuración 
General 
Depuración 
Directorias de VC++ 
Vinculadar 
Herramienta Manifiesta 
Generador de documenta! 
Información de examen 
Eventos de compilación 
Pasa de compilación persc 
Análisis de código 
v Plataforma: 
Depurador para iniciar: 
Depurador local de Windows 
Argumentas de comandas 
Directoria de trabaja 
Asociar 
Tipa de depurador 
Entorna 
Fusionar mediante combinación entorna 
Depuración de SQL 
Acelerador predeterminada de AMP 
PATH= S(LocaIDebuggerEnviron 
Dibujos con el lápiz
Dibujos con el lápiz

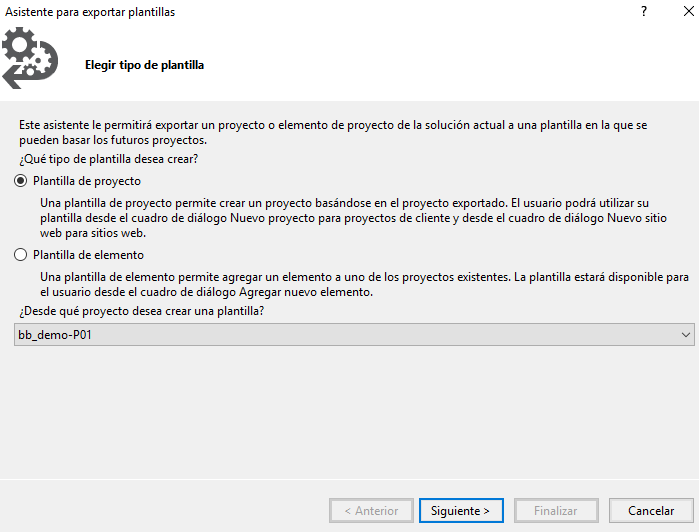


<-- Solo si queremos "exportar" alguna
        configuración de Depuración

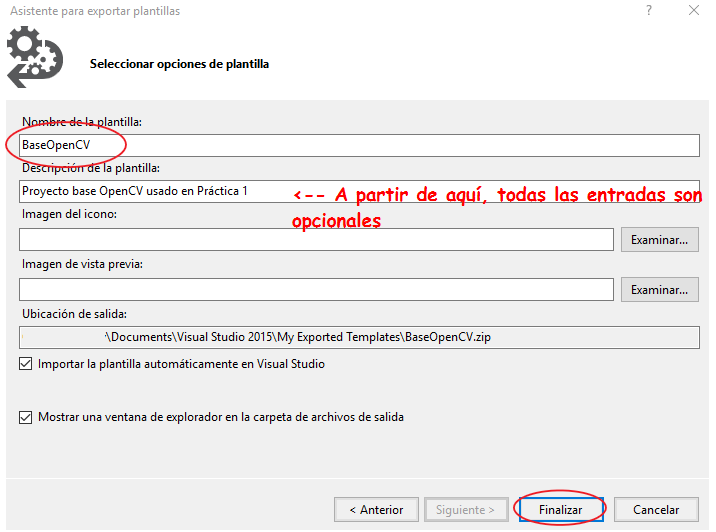

* + 1. Ejecutando el asistente Archivo🡪Exportar plantilla…

Solo puede ejecutarse si tenemos una solución abierta.

Aparece una ventana que permite seleccionar el proyecto en el que se basa la plantilla (si la solución tiene más de un proyecto):



Pulsando *Siguiente*…



La plantilla es un archivo **.zip** que se puede instalar en cualquier Visual Studio 2019.

* + 1. Instalar/eliminar una plantilla de Visual Studio.

Las plantillas instaladas se guardan en la carpeta de documentos del usuario, en la ubicación …Documens*\Visual Studio 2019\Templates\ProjectTemplates*. Para instalar/eliminar una plantilla solo hay que copiar/borrar el archivo .zip de la plantilla en esa ubicación.

* + 1. Usar una plantilla instalada.

La plantilla se puede usar al crear un nuevo proyecto. Por ejemplo, si queremos usar el proyecto inicial de la practica P02 usando la plantilla recién creada:


Imagen sin título.png Texto alternativo generado por el equipo:
Nuevo proyecto 
D Reciente 
Instalado 
Plantillas 
Yisual 
D Windows 
ATL 
CLR 
General 
MFC 
Prueba 
Win32 
D Multiplataforma 
.NET Framewark 4.5.2 
Ordenar por: Predeterminada 
D En línea 
Nombre: 
Ubicación: 
uvus-P02 
Biblioteca estática (Windows universal) 
Visual C+ 
Componente de Windows Runtime (Windows univ...VisuaI C+ 
Proyecto de archivos MAKE 
Visual C+ 
SS-CBasic010 
Visual C+ 
BaseOpenCV 
Visual C+ 
Haga clic aquí para buscar plantillas en línea. 
dema\P02 
Nombre de la solución: uvus 
salP02 
Buscar en Plantillas p • 
Type: Visual C++ 
Proyecto base OpenCV usado en Práctica 
Examinar... 
[9] Crear directoria para la solución 
<-- Seleccionamos plantilla
<-- Cambiamos nombre de proyecto y solución


Una vez creado el proyecto, debemos a mano los nombres de los archivos fuente y las opciones de depuración, si es necesario:

Imagen sin título.png uvus-SaIP02 - Microsoft Visual Studia 
Archivo Editar 
Ver Proyecto Compilar Depurar 
Debug 
Herramientas 
x86 
Inicio rápido (CtrI+Q 
Visua[SVN Ventana Ayuda 
Depurador local de Windows • Automática 
(Amblta global) 
Explorador de soluciones 
Buscar en Explorador de salu 
Solución 'uvus-SaIP02' 
uvus-P02 
Referencias 
bb_demo-P02.cpp -E X 
S] uvus-P02 
(I pr 
1 
16 
64 
65 
66 
67 
I #include "bb_demo-P82. h" 
Dependencias 
bb dema-P02.cpp 
[3 bb dema-P02.h 
Páginas cle propiedades de uvus-P02 
Configuración: Todas las config. 
Propiedades de configuración 
General 
Depuración 
Directorias de VC++ 
Vinculador 
Herramienta Manifiesta 
Generador de documenta! 
Información de examen 
Eventos de compilación 
Pasa de compilación persc 
Análisis de código 
Dir. Trabajo: 
Entorno: 
v Plataforma: 
Depurador para iniciar: 
Active (Win32) 
Depurador local de Windows 
Comanda 
Argumentas de comandas 
Directoria de trabaja 
Asociar 
Tipa de depurador 
Fusionar mediante combinación entorna 
Depuración de SQL 
Acelerador predeterminada de AMP 
$ (AAComm) \ eje 
\bin $(Loca1DebuggerEnvironn 
Administrador de configuración... 
S(TargetPath) 
Automática 
PATH= S(LocaIDebuggerEnviron 
Dibujos con el lápiz
Dibujos con el lápiz
Dibujos con el lápiz
Dibujos con el lápiz
Dibujos con el lápiz
Dibujos con el lápiz
Lo resaltado se ha cambiado "a mano"


* 1. Reutilizar código y opciones de un proyecto anterior sin plantilla.

Una alternativa al uso de plantillas es copiar el proyecto de origen a una nueva carpeta y modificar "a mano" los nombres de archivo y las configuraciones del nuevo proyecto.

Es simple en proyectos pequeños y tiene la ventaja de que "exporta" los archivos .vcxproj.user y no crea nuevas plantillas.

Los detalles de pueden verse en el video/pdf "00.20. Reutilizar código y opciones de un proyecto anterior (sin plantilla)", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw).

* 1. Anexo: Qué se puede hacer con el cliente TortoiseGit. Operaciones más comunes.

(Las funciones que vamos a ver aquí son comunes a casi todos los clientes Git).

TortoiseGit se integra en el Explorador de Windows: a la mayoría de las funciones se accede desde cualquier ventana del explorador pulsando el botón derecho del ratón.

* + 1. Antes de empezar: conceptos básicos.

Se ve en clase de Práctica 0.

**Sistema de gestor de versiones.**

* Para qué sirve: Control de cambios en un desarrollo software.
* Anota qué cambia, cuándo, quién.

Repositorio (remoto).

* Componente software que, desde el punto de vista del usuario, contiene:
* Un conjunto de archivos.
* Historial de revisión de cada archivo que están o han estado en el repositorio.
* Normalmente asociado a proyectos software.

Copia de trabajo (local).

* Copia del repositorio (remoto) donde el usuario puede realizar modificaciones.

Repositorio local.

* En Git, existen dos repositorios: uno local y otro remoto.
  + 1. Añadir archivos/carpetas al repositorio.
* Seleccionar archivo/carpeta, Menú contextual TortoiseGit🡪Add.
* El archivo/carpeta se añadirá al repositorio en la siguiente actualización.
  + 1. Eliminar archivos/carpetas al repositorio.
* Seleccionar archivo/carpeta, Menú contextual TortoiseGit 🡪Delete.
* El archivo/carpeta se eliminará del repositorio en la siguiente actualización.
* También se puede eliminar simplemente borrando el archivo/carpeta con el navegador.
  + 1. Actualizar repositorio (commit+push).

Para integrar las modificaciones de la copia local (working copy) al repositorio local y remoto.

* Menú contextual 🡪*Git Commit ->...*

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

* Aparece la ventana con la lista de archivos que se han modificado respecto a la última versión que aparece en el repositorio.
* Importante:
  + - revisar siempre esta lista. Debe ser coherente con las últimas modificaciones que hemos hecho.
    - Se debe escribir siempre algo en el recuadro *Message:*
    - Pulsar el botón *Commit & Push[[11]](#footnote-12)*.

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

* + 1. Eliminar una copia local del repositorio (working copy).

Se elimina como cualquier otra carpeta de Windows.

1. C/C++ para sistemas (I).
2. Oculto: Numeración ejercicios

4 y 6 de octubre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211004/20211006).

Contenido

[1.1. Expectativas. 21](#_Toc88125120)

[1.1.a. Qué hacer antes de entrar al laboratorio… 21](#_Toc88125121)

[1.2. Evaluación. 22](#_Toc88125122)

[1.3. Más allá de la configuración estándar. 22](#_Toc88125123)

[1.4. Mostrando el tiempo de cada imagen. 25](#_Toc88125124)

[1.5. Ajustando la velocidad de reproducción. 25](#_Toc88125125)

* 1. Expectativas.

Empezar a utilizar:

* La configuración del entorno de prácticas que se explica en la práctica 0.
* El entorno de desarrollo de Visual Studio y, sobre todo, su **depurador**.
* Las características de C/C++ para la programación en bajo nivel.

Entender en detalle el mecanismo de evaluación de prácticas.

Usar el depurador para estudiar la organización de la memoria virtual de programas creados a partir de código C/C++: dónde se almacena el código y los distintos tipos de datos (dependiendo de si son datos globales o locales).

* + 1. Qué hacer antes de entrar al laboratorio…

Revisar sección 0.3. de la práctica 0.

Repasar lo visto en clase o en este enunciado:

* Qué es un IDE[[12]](#footnote-13).
* Qué es un proyecto y un gestor de proyecto[[13]](#footnote-14).
* Las operaciones para generar y compilar una aplicación C/C++ a partir de un proyecto vacío[[14]](#footnote-15).
* Qué son punteros de C/C++ y cómo se utilizan[[15]](#footnote-16).
* Cómo se definen matrices y matrices de caracteres[[16]](#footnote-17).

Traer de casa:

Crear un proyecto con Visual Studio con las características que siguen (Sustituir nusuario por su nombre de usuario de enseñanza virtual)

* La solución debe colocarse en una carpeta **P01** de su disco local.
* Nombre de la solución: nusuario-SolP01
* Nombre del proyecto: nusuario-P01
* Tipo de proyecto: Visual C++, Proyecto vacío[[17]](#footnote-18).
  + Opción "Crear directorio para la solución" activa.
* Nombre del archivo con el código: nusuario-P01.cpp. Dentro de ese archivo:
  + Hacer el código del Ejercicio 1.1.
  + Estos ejercicios se deben traer hechos de casa. Para cualquier consulta, escribir email o ir a tutoría.

Qué se hace en el laboratorio:

Revisamos el código subido en el repositorio.

Hacer funcionar un código que use una biblioteca externa (ejemplo OpenCV).

* 1. Evaluación.
* Las practicas 1 y 2 se evalúan conjuntamente.
* Solo se evalúan si se asiste a ambas.
* Se hace una prueba de evaluación sobre prácticas 1 y 2 en enseñanza virtual:
* **Al inicio de la práctica 3 (20 minutos)**
* No hay prueba previa en las prácticas 1 y 2.
  1. Más allá de la configuración estándar.
  2. Realizar un programa muestre por pantalla un saludo con su nombre y apellidos. Por ejemplo:  
       
      *Hola, soy nombre apellido.*
  3. Añadir el proyecto al repositorio privado.

La práctica en el laboratorio empieza a partir de este punto.

Vamos a usar la biblioteca externa [OpenCV](http://opencv.org/) 3.3 para realizar esta práctica. la documentación de las funciones y estructuras de la biblioteca está en <https://docs.opencv.org/3.3.1/>.

Todos los archivos de OpenCV necesarios para crear y ejecutar los programas de prácticas se encuentran en la carpeta **\_pub/\_comm** del repositorio público.

El programa inicial que vamos a muestra el video contenido en un archivo mp4:

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Based on example 2 page 18 of OReilly Learning OpenCV.pdf:

// Basada en práctica 1 de SPD.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale.h>

#include "opencv/cv.h"

#include "opencv/highgui.h"

int main(int argc, char\* argv[]) {

setlocale(LC\_ALL, "Spanish");  // hace que printf "entienda" tildes, ñ, etc.

printf("\nPulse tecla ESC para cerrar la ventana\n");

// el archivo .mp4 está en \_pub/\_comm/eje del repositorio público

play("P01Eje1.mp4", "Ventana 1");

printf("\nPulse tecla RETORNO para terminar\n");

getchar();

return 0;

}

void play(char\* file\_name, char\* nombre\_ventana)

{

char key = 0;

// tipos de estructuras definidas en OpenCV

CvCapture\* g\_capture = NULL; // Para gestionar la captura de video

IplImage\* newframe, \* oldframe; // para gestionar cada frame

IplImage img;

// Crea una ventana OpenCV para mostrar el video

cvNamedWindow(nombre\_ventana, CV\_WINDOW\_AUTOSIZE);

g\_capture = cvCreateFileCapture(file\_name); // Abre el archivo de video

if (NULL == g\_capture) {

printf("\n\nERROR: No se pudo abrir el archivo de video\n\n");

return;

}

// Bucle de proceso de frames

newframe = cvQueryFrame(g\_capture); // Lee frame del archivo

while (newframe && (key != 27)) {

oldframe = newframe;

cvShowImage(nombre\_ventana, oldframe); // Enseña la imagen antigua

newframe = cvQueryFrame(g\_capture); // Carga la actual

key = cvWaitKey(1); // Espera que se pulse una tecla con un timeout de 1 ms

// Aquí podemos decodificar la tecla

}  // end of:   while(newframe && (key != 27 ) ) {

// Cerrar archivo de video

cvReleaseCapture(&g\_capture);

cvDestroyWindow(nombre\_ventana);

}

* 1. Sustituir el código del Ejercicio 1.1. por el código anterior e intentar compilarlo.
  2. ¿Cómo se hace para que el entorno de desarrollo pare tras la compilación (evita la fase de link)?
  3. Para qué sirve esta característica.
  4. Explicar por qué se da cada uno de los fallos de compilación.
  5. ¿Cómo se arreglan cada uno de ellos[[18]](#footnote-19)?
  6. ¿Por qué la inclusión de stdio.h no provoca error y la de cv.h sí?
  7. ¿Dónde busca el compilador el archivo de un #include?

Vamos a la fase de link (vinculación):

* 1. Cuando se está generando un proyecto, ¿cómo se sabe si un fallo es de compilación o link?
  2. ¿Explicar por qué se da cada uno de los fallos de link?
  3. ¿Cómo se arreglan cada uno de ellos[[19]](#footnote-20)?
  4. ¿Dónde busca el linker las bibliotecas y cómo se le indica cuáles debe usar?

Vamos a intentar ejecutar:

* 1. ¿Cómo se sabe si un programa no se inicia correctamente?
  2. ¿Explicar por qué no se inicia correctamente el programa?
  3. ¿Dónde busca el sistema operativo las DLLs que necesita un programa?   
     ¿Cómo solucionamos el problema de arranque[[20]](#footnote-21)?
  4. Hemos logrado iniciar el programa. Vamos a por el siguiente error.   
       
     ¿Por qué no se localiza correctamente el archivo "P01Eje1.mp4"?   
     ¿Cómo se soluciona[[21]](#footnote-22)?

Enhorabuena, ha logrado copiar y pegar código C/C++ y hacer que funcione.

A partir de aquí, algunos ejercicios para practicar el lenguaje…

* 1. Qué efecto tiene declarar el modificador **const** en la siguiente definición:  
       
     static void play(**const** char \*file\_name, **const** char \* nombre\_ventana)  
     {…  
       
     Pista: intentar modificar en la función algún elemento de las matrices de entrada.
  2. Mostrando el tiempo de cada imagen.
  3. Modificar la función play para que muestre la cadena "Hola" en la cada imagen del video.  
       
     Usar las funciones **cvPutText** de [OpenCV](https://docs.opencv.org/3.3.1/).
  4. Modificar el ejercicio anterior para que se muestre los milisegundos asociados a cada imagen del video.  
       
     Usar la función de openCV **cvGetCaptureProperty** para averiguar los ms asociados a cada frame.  
     Usar las funciones **\_snprintf**, de la biblioteca estándar de C, para construir la cadena.
  5. Ajustando la velocidad de reproducción.
  6. Modificar la función play del ejercicio anterior para que reproduzca a velocidad “real”.

Usar la función de openCV **cvGetCaptureProperty**.

Actualmente, cvWaitKey(1) espera un ms antes de reproducir el siguiente frame.

Con cvGetCaptureProperty podemos leer los ms a los que corresponde cada frame. Por tanto, solo hay que calcular los ms que hay entre el frame actual y el siguiente, y esperar ese tiempo antes de procesarlo.

* 1. Compilando para plataforma x64 (N3).
  2. (N3) Modificar el proyecto del ejercicio anterior para que se compile y ejecute en la plataforma x64.

1. C/C++ para sistemas (II).
2. Oculto: Numeración ejercicios

11 y 13 de octubre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211011/20211011).

Contenido

[2.1. Expectativas. 27](#_Toc88125126)

[2.1.a. Qué hacer antes de entrar al laboratorio… 27](#_Toc88125127)

[2.2. Qué hacer en el laboratorio. 27](#_Toc88125128)

[2.3. Evaluación. 28](#_Toc88125129)

[2.4. Modificar la función play. 28](#_Toc88125130)

[2.5. Usando el depurador para analizar el comportamiento de un programa. 29](#_Toc88125131)

* 1. Expectativas.

Utilizar el depurador para detectar errores y analizar el comportamiento del programa.

* + 1. Qué hacer antes de entrar al laboratorio…

Repasar lo visto en clase:

* La definición de punteros, tablas y cadenas en C/C++.
* Definición y manejo de estructuras y punteros a estructuras[[22]](#footnote-23).
* Manejo del depurador[[23]](#footnote-24).

Partimos del programa realizado en el Ejercicio 1.21. de la práctica anterior terminado.

Podemos usar el proyecto de la práctica anterior o crear un nuevo proyecto usando una plantilla tal como se explica en la sección 0.5.

La práctica empieza a partir de este punto.

Hacer test de práctica 1 (en ev.us.es). Modificado: se hace público el jueves 14 de octubre.

- No entra en evaluación de prácticas.

* 1. Qué hacer en el laboratorio.

- Modificar el programa inicial como se pide en los ejercicios y usarlo, junto con el depurador, para analizar su estructura y comportamiento interno.

* 1. Evaluación.
* Esta práctica no tiene prueba previa.
* Se evalúa junto a la práctica anterior. Ver sección 1.2. , en la práctica anterior.
  1. Modificar la función play.
  2. Modificar el programa para inicial para que imprima en la imagen un texto con su nombre UVUS, el segundo y décima de segundo de cada fotograma. Por ejemplo:
  3. ¿Cuál es el tamaño de la imagen que se está reproduciendo?   
     Usar el depurador (ventana de inspección) para ver el valor de los campos de *newframe* que definen el alto y ancho de la imagen.
  4. Usar una traza (breakpoint actions en VS), para mostrar el tamaño de la ventana por la consola de depuración[[24]](#footnote-25).
  5. Breakpoint condicionales: hacer que el programa pare justo antes de mostrar un frame, pero solo a partir de los 2000 ms de reproducción[[25]](#footnote-26).
  6. Modificar el Ejercicio 2.1. para que el texto “deambule” por la pantalla. Es decir, se debe mover durante la reproducción del video de la forma más continua posible[[26]](#footnote-27).
  7. Usando el depurador para analizar el comportamiento de un programa.

El uso normal de un depurador es analizar posibles errores en un programa. Sin embargo, también se puede usar para analizar detalles de implementación de un lenguaje de programación, un compilador, linker, juego de instrucciones, etc.

Como ejemplo, veremos los pasos que hay que seguir para contestar el siguiente ejercicio:

* 1. Cuando un compilador de lenguaje de alto nivel genera una aplicación, puede organizar la memoria siguiendo este esquema (Unix):  
       
     Posiciones más bajas de memoria 🡪 introduce el código del programa.  
     Posiciones intermedias 🡪 introduce las variables globales.  
     Posiciones más altas de la memoria 🡪 las reserva para la pila.   
       
     ¿Utiliza el compilador de VC++ este esquema? Dibujar un esquema que indique la posición relativa de estas zonas en el programa del ejercicio anterior.

* 1. ¿Qué valores iniciales tiene una matriz y una estructura si se define como global?   
     ¿Y si se define como local?  
     ¿Ocurre lo mismo en Java?
  2. En C/C++, ¿las estructuras se pasan por valor o por referencia?  
       
     Intentar implementar la función:  
       
     void StepDerecha(CvPoint punto);  
       
     … que modifica la posición x del punto que le pasan.
  3. En C, ¿las matrices se pasan por valor o por referencia?  
       
     Usar la funcion upper\_string que sigue (el código está [aquí](https://github.eii.us.es/IC3-SS/_pub/blob/master/Practicas/P02_Codigo_Ejercicio_upper_string.txt)).

#include <stdio.h>

void upper\_string(char[]);

char cad[40] = "cadena Ejemplo ASCII";

int main()

{

printf("\nCadena original: %s\n", cad);

upper\_string(cad);

printf("\nCadena resultado: %s\n", cad);

printf("\nPor favor, pulse una tecla para terminar ...");

\_getch();

return 0;

}

// Copiado de http://www.programmingsimplified.com/c/program/c-program-change-case

void upper\_string(char s[]) {

int c = 0;

while (s[c] != '\0') {

if (s[c] >= 'a' && s[c] <= 'z') {

s[c] = s[c] - 32;

}

c++;

}

}

1. Bibliotecas estáticas.
2. Oculto: Numeración ejercicios

18 y 20 de octubre de 2021: primera sesión + prueba evaluación P01-2.

25 y 27 de octubre de 2021: segunda sesión.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211018/20211019).

Contenido

[3.1. Expectativas. 33](#_Toc88125132)

[3.1.a. Qué hacer antes de entrar al laboratorio… 33](#_Toc88125133)

[3.2. Evaluación. 34](#_Toc88125134)

[3.3. Trabajo previo: usando una biblioteca estática. 34](#_Toc88125135)

[3.3.a. Antes de empezar… 35](#_Toc88125136)

[3.3.b. Añadir la librería al vinculador (linker). 35](#_Toc88125137)

[3.3.c. Usar los símbolos que contiene la biblioteca en nuestro proyecto. 36](#_Toc88125138)

[3.4. Qué hacer en el laboratorio. 36](#_Toc88125139)

[3.5. Utilizar un símbolo C desde código C++. 36](#_Toc88125140)

[3.6. Creando bibliotecas estáticas. 37](#_Toc88125141)

[3.7. Bibliotecas auto-contenidas. Dependencias. 38](#_Toc88125142)

[3.8. Decoración de símbolos. 38](#_Toc88125143)

* 1. Expectativas.
* Programación con bibliotecas estáticas:
* Cómo se usan.
* Cómo se crean.
* Cómo se organiza el código.
* Cómo se prueba.
* Diferencias entre lenguaje C y C++ a nivel de vinculación (bibliotecas y archivos .obj)
  + 1. Qué hacer antes de entrar al laboratorio…

Repasar lo visto en clase o en el enunciado de esta práctica:

* Leer apuntes y realizar los ejercicios N1 y N2 en “**SS-030. Ln- Tema 3. Generación de programas**”.

Qué traer de casa (este programa entra en la evaluación):

* Prueba P03.A: Realizar el proyecto con el código del Ejercicio 3.1. y subirlo al repositorio privado.
* Las especificaciones del proyecto, y cómo hacerlo, están en la sección 3.3. “Trabajo previo: usando una biblioteca estática.”.
  1. Evaluación.

Esta práctica se evalúa junto con la siguiente. 3 pruebas:

Fecha, hora límite

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba | Tipo prueba. | Turnos lunes | Turnos miérc. | Máx/mín. |
| P03.A | Subir código Ejercicio 3.1. a repositorio privado | 18/10/2021, 15:30. | 20/10/2021, 15:30. | 2/1,25 **[[27]](#footnote-28)** |
| P03.B | Subir código hasta el Ejercicio 3.11. | 3/11/2021, 15:30 | 3/11/2021, 15:30 | 4 |
| P03\_4.B | Prueba evaluación en EV sobre prácticas 3 y 4. | 15/11/2021, 15:30 | 10/11/2021, 15:30 | 4 |

* La ponderación de la nota total es 0,8.
* **Solo se evalúan las prácticas si se asiste.** 
  1. Trabajo previo: usando una biblioteca estática.
  2. **Los antecedentes:**Para ilustrar el uso de bibliotecas estáticas vamos a utilizar una biblioteca (que a partir de ahora llamaremos **EjeLib.v0.lib**). EjeLib.v0.lib fue creada por un antiguo alumno que ha pasado a mejor vida[[28]](#footnote-29). El problema es que la información sobre qué había en la biblioteca se ha perdido en gran parte.   
       
     Se ha podido averiguar (de los más viejos del lugar) algunos datos sueltos:  
      \* Se rumorea que la biblioteca contiene una función y una variable global.  
      Sus tipos podrían ser:  
       
      void NombreFuncion(void);  
      char NombreVariable[¿?];  
       
      \* Por desgracia, nadie recuerda cómo se llaman realmente la función ni la variable[[29]](#footnote-30).  
      **\* Sí se sabe que ambas variables se compilaron como un archivo .c.**  
       
     Se pide: hacer un programa que imprima por pantalla la variable global y después llame a la función, ambas definidas en EjeLib??.v0.lib (solo es necesario generar el proyecto, no hace falta que ejecute correctamente). **Cada alumno usará la versión de la biblioteca que se le pasa en el directorio \misArchivos de su repositorio privado**.  
       
     Para este ejercicio usaremos la utilidad DUMPBIN para obtener la información que necesitamos.

Este ejercicio nos va a permitir tener claro…

* … qué información se almacena en una biblioteca estática sobre los símbolos: los nombres de los símbolos (variables y funciones).
* … qué información no se almacena: nada más. No existe ninguna información sobre los tipos de los datos o funciones que contiene.

Las siguientes partes de esta sección pueden servir de guía para hacer el ejercicio.

* + 1. Antes de empezar…

Crear el proyecto.

Crear un proyecto con Visual Studio con las características que siguen (Sustituir nusuario por el nombre de usuario de enseñanza virtual):

* En la carpeta **P03** de su directorio privado
* Nombre de la solución: nusuario-**SolP03**
* Nombre del proyecto: nusuario-**P03-TestLib**
* Tipo de proyecto: Visual C++, Proyecto vacío.
* Opción "Crear directorio para la solución" activa.
* Debe tener al menos un archivo:
* nusuario- **P03TestLib.c**. Dentro de este archivo:
* Código para probar lo que se pide en el Ejercicio 3.1. [[30]](#footnote-31)
* MUY IMPORTANTE: El archivo debe tener extensión **.c**.

En este punto el proyecto tendrá errores de vinculación al generarse.

* El proyecto hecho hasta aquí se evalúa con 0,5 puntos máximo.
  + 1. Añadir la librería al vinculador (linker).

Para que el vinculador use la biblioteca hay que añadir los datos del archivo a las propiedades del vinculador. Los pasos son:

* Copiar la biblioteca al proyecto o solución desde la carpeta **misArchivos** de su repositorio privado[[31]](#footnote-32).
* Añadir la biblioteca al proyecto.
* Los pasos para hacerlo se muestran en el video/pdf " P01.20. Agregar biblioteca .lib a proyecto", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw).
* Un cambio respecto al video: la nueva biblioteca se encuentra en el repositorio privado, no en el público. El nombre y ruta es distinta.
* Si se copia al directorio del proyecto/solución, se pueden usar las [macros](https://docs.microsoft.com/en-us/cpp/build/reference/common-macros-for-build-commands-and-properties) de VS2019 **$(SolutionDir)** ó **$(ProjectDir)**.
* Aunque un archivo en el repositorio privado se puede expresar también respecto a la variable *AAComm* usando la expresión ... (directorio padre):  
    
  **$(AAComm)\..\..\nusuario\misArchivos\EjeLib.v0\_....lib**
* El proyecto hecho hasta aquí se evalúa con 1,5 puntos máximo.
  + 1. Usar los símbolos que contiene la biblioteca en nuestro proyecto.

Para usar un símbolo[[32]](#footnote-33) de una biblioteca debemos saber su nombre y tipo.

El nombre se puede saber usando la utilidad dumpbin.

* Los pasos para hacerlo se muestran en el video/pdf " P03.20.Usar dumpbin para ver los símbolos públicos de un archivo.obj ", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)[[33]](#footnote-34).

Podemos usar esos nombres para definir cualquier variable o función "extern" en nuestro proyecto.

* El proyecto hecho hasta aquí se evalúa con 2 puntos máximo.
  + El proyecto debe utilizar los símbolos definidos en la biblioteca y debe generarse correctamente (compilar + link).
  + No es necesario que se ejecute correctamente.
  1. Qué hacer en el laboratorio.
  2. (N3). Hacer que el programa anterior funcione correctamente usando la función y variable global definida en la biblioteca EjeLib??.v0.lib.

* 1. Utilizar un símbolo C desde código C++.
  2. (N1). Modificar el código fuente de nusuario-P03-TestLib para que compile como código **C++** y probar si funciona correctamente.

Tenemos dos alternativas para hacer esto: cambiar las extensiones de los archivos .c a .cpp, o modificar las propiedades de cada fichero fuente .c[[34]](#footnote-35).

En principio, C++ es una extensión de C, por tanto, es compatible con C a nivel de código fuente.

* 1. (N1/N2). ¿O no? ¿Qué pasa si, en un mismo proyecto, mezclamos archivos .C y .C++?
  2. (N2). ¿Cómo se puede usar una función C (en nusuario-**P03-Static**, por ejemplo) desde un programa C++?[[35]](#footnote-36)
  3. Creando bibliotecas estáticas.

Vamos a crear una biblioteca estática que contenga la función

void Firma\_nusuario(CvArr \* imagen);

Se le pasa un puntero a imagen. Coloca en la esquina inferior izquierda de la imagen su identificador UVUS.

Se recomienda usar la función [cvGetSize](http://docs.opencv.org/2.4/modules/core/doc/old_basic_structures.html?highlight=cvarr#getsize) para determinar el tamaño de la imagen.

* 1. (N1). Se hace como demo en clase: Crear un nuevo proyecto con las siguientes características:  
       
     \* Nombre: nusuario-**P03-Static**. Proyecto vacío. Crear un archivo .cpp con una función “**stub**”:  
        
     \* Las funciones se probarán desde el proyecto nusuario-**P03-TestLib**.

#include <stdio.h>

#include <windows.h>

char msg[100];

int i = 0;

void Firma\_nusuario(CvArr \* imagen)

{

snprintf(msg, sizeof(msg), "Ejecutada Firma\_nusuario #%d veces\n", i++);

OutputDebugString(msg);

}

* 1. **Paso 1:**   
     Modificar las opciones de proyecto de nusuario-**P03-Static** para que se compile como biblioteca estática.
  2. (N1). **Paso 2:**Decidir:  
      \* qué es público,  
      - en qué lenguaje crearemos el símbolo (C o C++) y  
      - qué lenguajes lo usarán (C o C++).  
       
      \* Lo demás, se define como **static**.

Los siguientes pasos se hacen como ejercicios.

* 1. (N1). **Paso 3:**   
     Intentar probar las funciones desde nusuario-**P03-TestLib**.  
       
     \* ¿Qué información necesitamos?
  2. (N1). **Paso 4:**   
     El .h asociado a la biblioteca. Decidir qué se necesita para usar la biblioteca.  
       
     \* Cómo hacer accesible la información del .h al proyecto nusuario-**P03-TestLib**.
  3. Ahora sí: hacer la función **Firma\_nusuario.**
  4. Bibliotecas auto-contenidas. Dependencias.
  5. (N1). ¿Qué es una biblioteca auto-contenida (self-contained library)?   
       
      \* ¿nusuario-P03-TestLib es auto-contenida?
  6. (N2/N3).En realidad, casi ninguna biblioteca es auto-contenida. ¿Por qué?
  7. Decoración de símbolos.
  8. (N1). ¿Cómo se decoran[[36]](#footnote-37) los símbolos C?  
     Modificar el código de nusuario-**P03-TestLib** para mostrar cómo se decora una función C y un entero[[37]](#footnote-38).
  9. (N2) ¿Cómo se decoran los símbolos C++?  
     Declarar las siguientes funciones/variables

int Func1(int);

int Func1(int, int); // C++ permite sobrecarga de funciones. C no.

int Func1(char)  
int Func1(int, char);

* 1. (N3) ¿Para qué sirve la decoración de símbolos en C++?

1. Bibliotecas dinámicas.
2. Oculto: Numeración ejercicios

3 y 8 de noviembre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211025/20211026).

Contenido

[4.1. Expectativas. 41](#_Toc88125144)

[4.1.a. Qué hacer antes de entrar al laboratorio… 42](#_Toc88125145)

[4.2. Evaluación. 42](#_Toc88125146)

[4.3. Crear una biblioteca dinámica en C/C++. 42](#_Toc88125147)

[4.3.a. Método 1. 43](#_Toc88125148)

[4.3.b. Método 2. 43](#_Toc88125149)

[4.4. Usar una DLL. 44](#_Toc88125150)

[4.4.a. Con biblioteca de importación. 44](#_Toc88125151)

[4.4.b. Con LoadLibrary: carga dinámica de una biblioteca dinámica. 44](#_Toc88125152)

[4.5. (N3) Ejemplos de mezcla de lenguajes de programación. 45](#_Toc88125153)

[4.5.a. C# y C. La clase System.Runtime.InteropServices. 45](#_Toc88125154)

[4.5.b. Java y C: JNI (Java Native Interface). 46](#_Toc88125155)

* 1. Expectativas.
* Programación con bibliotecas dinámicas:
* Cómo se usan.
* Cómo se crean.
* Cómo se organiza el código.
* Cómo se prueba.

El desarrollo de la práctica será:

* Primero creamos una biblioteca .dll.
* Tipo de proyecto en VS.
* Componentes de una dll: dllmain y biblioteca de importación.
* Posibles errores en el proceso.
* A continuación, veremos alternativas para usarla desde C/C++.
* Con biblioteca de importación.
* Sin biblioteca de importación (LoadLibrary).
* Posibles errores.
* Por último, veremos un ejemplo simple de mezcla de lenguajes de programación: C# y C++.
  + 1. Qué hacer antes de entrar al laboratorio…

Repasar lo visto en clase o en este enunciado:

* Leer apuntes y realizar los ejercicios N1 y N2 en “**SS-030. Ln- Tema 3. Generación de programas**”.

Qué traer de casa:

* La práctica anterior, hecha al menos hasta el Ejercicio 3.11. (inclusive). Debe estar subido en su repositorio privado **antes de la primera sesión de esta práctica**.
* Veremos cómo se organiza el código cuando queremos realizar una biblioteca dinámica.
  1. Evaluación.
* Ver evaluación de la práctica anterior (se evalúa junto con esta).
  1. Crear una biblioteca dinámica en C/C++.

Usaremos, en todos los ejercicios, la misma solución que en la práctica anterior.

* 1. En la solución usada en la práctica anterior, crear un nuevo proyecto con las siguientes características:  
       
     \* Nombre: nusuario-**P03-Dynamic**. Proyecto vacío.   
      \* Crear un archivo .cpp y copiar las funciones usadas en nusuario-**P03-Static[[38]](#footnote-39)**.   
      \* Crear un archivo nusuario-**P03-Dynamic.h** para las definiciones de la biblioteca.
  2. Realizar las operaciones necesarias para que nusuario-P03-Dynamic se compile como biblioteca dinámica.

Se explicará en pizarra:

* Crear un proyecto para biblioteca dinámica (método 1).
* Qué es una biblioteca de importación.
* Modificaciones para usar una biblioteca dinámica (con biblioteca de importación).
  + 1. Método 1.
* Creamos un proyecto vacío.
* Una vez creado, cambiamos el tipo de proyecto a “Biblioteca dinámica”.
* MUY IMPORTANTE: hacerlo en todas las configuraciones.
* Crear una función [DllMain](https://msdn.microsoft.com/en-us/library/windows/desktop/ms682583%28v=vs.85%29.aspx). Cada vez que se carga o descarga nuestra biblioteca, el SO llama a esta función. Un ejemplo simple:

BOOLAPIENTRYDllMain(HMODULEhModule,

DWORD ul\_reason\_for\_call,

LPVOID lpReserved )

{

switch(ul\_reason\_for\_call)// <-- esto es opcional

{

caseDLL\_PROCESS\_ATTACH:// Posibles causas de entrada en DllMain

OutputDebugString(">> Se ha cargado la DLL XXXXXX \n");//<--opc

break;

caseDLL\_PROCESS\_DETACH:

OutputDebugString("<< Se ha descargado la DLL XXXXXX \n");//<--opc

break;

caseDLL\_THREAD\_ATTACH:

caseDLL\_THREAD\_DETACH:

break;

}

returnTRUE;// <--esto es lo único obligatorio.

}

* Cuando usamos bibliotecas dinámicas, a los símbolos que se pueden compartir con otros módulos se les denomina símbolos exportables o exportaciones (exports).
* La forma de indicar que un símbolo es exportable depende del compilador. En VS:

\_\_declspec(dllexport)intVariableExportada;

...

// \_\_declspec (dllexport) debe estar tanto en la declaración como en la definición del símbolo

\_\_declspec(dllexport)voidFuncionExportada(void){...

* Como resultado de generar el proyecto tendremos dos archivos:
* La biblioteca dinámica (fichero .dll), con el código/datos que se cargará al usar la dll.
* La biblioteca de importación. Es una biblioteca estática (.lib) que permite usar la dll de forma transparente.
* Aunque es opcional, también se debe agrupar la declaración de todos los símbolos exportados en un archivo .h.
  + 1. Método 2.
* Se basa en crear un proyecto de tipo “Aplicación de consola Win32”, subtipo “Biblioteca de vínculos dinámicos”.
* No usaremos este método porque crea un proyecto con [cabeceras precompiladas](http://en.wikipedia.org/wiki/Precompiled_header), tema que no veremos en la asignatura, y automatiza el proceso[[39]](#footnote-40).
* Es el recomendable cuando se usa C++.
  1. Usar una DLL.
     1. Con biblioteca de importación.

1. Cómo se usa una biblioteca de importación.  
     
   Realizar las operaciones necesarias para probar alguna de las funciones de nusuario-**P03-Dynamic** desde **nusuario-P03-TestLib**.  
   1. ¿Son distintas de las necesarias para usar nusuario-P03-Static?

* + 1. Con LoadLibrary: carga dinámica de una biblioteca dinámica[[40]](#footnote-41).
  1. Demo: cómo se usa LoadLibrary para cargar un símbolo desde una dll.

Un ejemplo (llamando a la función *Firma\_nusuario*):

// Prueba de uso de una DLL sin biblioteca de importación

{

HMODULE **HLibreriaDll**; // Handle. Lo veremos en el tema siguiente.

// Puntero a función. Comparar la definición con la definición de Firma\_nusuario en el archivo .h

void (\***pFuncion**)( CvArr\*);

HLibreriaDll = LoadLibrary("**NombreDeLaBiblioteca.dll**"); // “Carga” la dll donde está la func.

if (HLibreriaDll != NULL)

{

pFuncion = (void (\*)( CvArr\*))GetProcAddress(HLibreriaDll,

"**NombreDeLaFuncionEnDLL[[41]](#footnote-42)**");

if (pFuncion != NULL) // Usar el dato/función

(\*pFuncion)(oldframe);

FreeLibrary(HLibreriaDll); // Libera la librería

}

}

* 1. **(N2)**. ¿Cómo se decoran los símbolos de una biblioteca de importación[[42]](#footnote-43)?
  2. **(N1)**. Usar bibliotecas dinámicas no modifica la compilación/link de un ejecutable ¿Tampoco su instalación? ¿Qué pasa si cambiamos de directorio **TestLib.exe**?
  3. **(N2)**. ¿Cuándo se carga la biblioteca dinámica si sólo usamos la biblioteca de importación (Demo 1)[[43]](#footnote-44)?
  4. **(N2/N3).** ¿Cuándo se descarga la biblioteca dinámica?
  5. **(N2)**. ¿Cuándo se carga y se descarga la biblioteca dinámica si sólo usamos LoadLibrary/GetProcessAdresss (Ejercicio 4.4. )?
  6. **(N2)**. Qué sentido tiene usar LoadLibrary en lugar de la biblioteca de importación.
  7. **(N2).** ¿Qué es necesario conocer de un archivo .dll para poder usar cualquiera de sus funciones o datos?
  8. (N3) Ejemplos de mezcla de lenguajes de programación.

Esta parte la veremos como demo si da tiempo.

Algunas normas generales (lenguaje C/C++):

* La situación más habitual es hacer la mayor parte del programa en el lenguaje de “más alto nivel”, con usos puntuales del lenguaje de C/C++.
* Debe haber al menos una clase en el lenguaje de alto nivel que:
* “Encapsule” las operaciones LoadLibrary/GetProcAddress/FreeLibrary.
* Defina algunos tipos intercambiables. Solo un subconjunto de tipos del lenguaje de alto nivel se puede usar en las llamadas a funciones C/C++.
  + 1. C# y C. La clase System.Runtime.InteropServices.

1. (N3) En la solución usada en la práctica anterior, crear un nuevo proyecto con las siguientes características:  
     
   \* Nombre: WindowsFormsApplication1.   
    \* Tipo de proyecto: Visual C#--> Escritorio de Windows 🡪 Aplicación Windows Form.   
     
   Realizar las operaciones necesarias para que el proyecto anterior llame a la función **Toca** cuando se carga la ventana. Toca es una función de nusuario-**P03-Dynamic** con el siguiente prototipo:  
     
    DLL\_EXPORT void Toca(void); // Usa la funcion Beep de Windows para dar un pitido

Este ejemplo es bastante sencillo: solo hay que indicar que la función Toca está definida en una DLL externa:

[System.Runtime.InteropServices.DllImport("nusuario-P03-Dynamic.dll")]

public static extern void **Toca**();

* + 1. Java y C: JNI (Java Native Interface).
  1. **(N3)** Crear un programa Java (IDE Eclipse, por ejemplo) que llame a una función TocaDesdeJava en la biblioteca nusuario-P03-Dynamic:

Para este ejercicio se debe usar la interface [JNI](https://en.wikipedia.org/wiki/Java_Native_Interface) de Java.

En el programa Java:

public class EjemploJNI {

private native void TocaDesdeJava();

public static void main(String[] args) {

// TODO Auto-generated method stub

//System.setProperty("java.library.path",

// "directorio donde está el archivo .dll");

System.*out*.println("Llamando a la función C...");

System.*loadLibrary*("NombreBibliteca");

new EjemploJNI().TocaDesdeJava();

}

}

En el programa C (proyecto nusuario-P03-Dynamic):

#include <jni.h> // <- incluir C:\Program Files\Java\jdk1.8.0\_65\include o equivalente en includepath

NIEXPORT void JNICALL Java\_jmain\_Toca(JNIEnv \*env, jobject thisObj) {

Beep(1000, 1500);

}

1. Ejercicios Tema 4.
2. Oculto: Numeración ejercicios

10 y 15 de noviembre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211110/20211115).

Contenido

[5.1. Expectativas: 47](#_Toc88125156)

[5.1.a. Qué hacer antes de entrar al laboratorio… 47](#_Toc88125157)

[5.2. Evaluación. 48](#_Toc88125158)

[5.3. Qué hacer antes de entrar en el laboratorio. 48](#_Toc88125159)

[5.4. Qué hacer en el laboratorio. 49](#_Toc88125160)

[5.5. Ejercicios. 49](#_Toc88125161)

[5.5.a. Utilidades en SSLib1. 49](#_Toc88125162)

[5.5.b. Instalar la biblioteca (hacer accesible la biblioteca **nusuario-LibSS** a otras soluciones). 50](#_Toc88125163)

[5.5.c. Funciones básicas. 51](#_Toc88125164)

[5.5.d. Funciones simples de ejemplo. 52](#_Toc88125165)

* 1. Expectativas:

En esta práctica se realizan ejercicios simples para comenzar a programar con la API de un sistema operativo.

* + 1. Qué hacer antes de entrar al laboratorio…

Crear un proyecto Visual Studio con las características que siguen

* La solución debe colocarse en la carpeta **P05** de su repositorio privado. En los nombres que siguen, sustituir *nusuario* por su UVUS.
* Nombre de la solución: **nusuario-SolP5**.

Crear 2 proyectos en la solución:

* Proyecto nusuario-LibSS.
* Se creará a partir de su proyecto nusuario-**P03-Static**, con el código correspondiente al **Ejercicio 3.11.** de la Práctica 3. Alternativas:
* Usar una plantilla (ver sección 0.5. Crear y usar una plantilla de proyecto.),
* o copiar el proyecto …P03-Static y modificar los archivos (ver sección 0.6. Reutilizar código y opciones de un proyecto anterior sin plantilla.).
* Añadir 3 archivos adicionales de código:
* nusuario-LibSS.h: tendrá al menos los prototipos de las funciones CheckError y MideRetraso.
* nusuario-CheckError.cpp. Cuerpo de la función CheckError[[44]](#footnote-45).
* nusuario-MideRetraso.cpp. Cuerpo de la función MideRetraso1.
* El proyecto genera una biblioteca estática nusuario-LibSS.lib.
* Proyecto **nusuario-P5**.
* Tipo de proyecto inicial: Visual C++, Proyecto vacío.
* Añadir un archivo de código:
* nusuario-P5-Ej1.cpp. Un programa que haga una llamada a las funciones CheckError y MideRetraso, **usando la biblioteca nusuario-LibSS**.
* El programa no hará nada, pero debe compilar y ejecutar sin fallos.

Este proyecto, junto con la prueba P05.A. de enseñanza virtual, deben estar hechos antes del primer turno de la práctica.

La práctica empieza a partir de este punto.

* 1. Evaluación.

Esta práctica se evalúa junto con la siguiente. 3 pruebas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prueba | Tipo prueba. | Fecha, hora límite | | | Máx/min. |
| Turnos miércoles | Turnos lunes | |
| P05.A | Se publicada el 5/11 (12:30) en EV. | 10/11/21, 15:30. | | 15/11/21, 15:30. | 2/1[[45]](#footnote-46) |
| P05.B | Publicada en EV durante la práctica 5. Se entrega la semana siguiente. | 17/11/21, 21:30. | | 22/11:21, 21:30. | 3 |
| P05\_6.B | Prueba evaluación en EV sobre P06 (al inicio de P07-15 minutos) | 24/11/21. | | 29/11/21. | 5 |

Es muy importante hacer al menos las siguientes operaciones de “Commit” con el código de la práctica:

* Al inicio de la práctica.
* Al final de cada ejercicio.

La ponderación de esta práctica es **1**.

Solo se evalúan las prácticas si se asiste.

* 1. Qué hacer antes de entrar en el laboratorio.
* Revisar tema 4 y realizar evaluación P05.A.
  1. Qué hacer en el laboratorio.
* Todos los ejercicios hasta el Ejercicio 5.8.
* De los ejercicios a partir del Ejercicio 5.8. , hacer al menos un ejercicio nivel N1 y otro N2, más el Ejercicio 5.40.
  1. Ejercicios.
     1. Utilidades en SSLib1.
  2. (N1) Reutilizar el código de la función [ReportError](https://uses0-my.sharepoint.com/:u:/g/personal/camaya_us_es/Ee68rO3tJoVGvoDn4LNAOVkBDlqcMxxxQTS9ZSzl1OxBBQ?e=6appgU), para hacer la función:  
       
     BOOL CheckError (BOOL ErrorCondition, LPCTSTR userMessage, DWORD exitCode);  
       
     **Este ejercicio se hace en el lab.[[46]](#footnote-47)**
  3. **(N1)** ¿Cómo se puede usar ReportError para simplificar el chequeo de errores en [este](https://uses0-my.sharepoint.com/:u:/g/personal/camaya_us_es/ESUYzzglnx5KnDt9jCryxlIBAXtPwBPwwkswnxxB4bTB-Q?e=rvPgF6) programa?  
     (N1) ¿Cómo se puede usar CheckError?
  4. Mejoras en ReportError.  
       
     Realizar las siguientes mejoras en la función ReportError. Cuando se produce un error:  
      \* Da un pitido usando la función de Beep o MessageBeep (Windows).  
      \* Envía el mensaje de error también por la consola de depuración, usando  
      OutputDebugString.

*ReportError* se basa en las funciones [GetLastError](https://msdn.microsoft.com/es-es/library/windows/desktop/ms679360%28v=vs.85%29.aspx) y [FormatMessage](https://msdn.microsoft.com/es-es/library/windows/desktop/ms679351%28v=vs.85%29.aspx). *CheckError* funciona igual que *ReportError* pero sólo cuando *ErrorCondition* es TRUE. Si es FALSE no hace nada. Devuelve el valor del parámetro *ErrorCondition*.

* 1. (N2) Realizar la función:  
       
     double MideRetraso(LPCTSTR pMensaje);

Devuelve el número de segundos desde la última llamada a MideRetraso. La primera vez que se llama devuelve 0.0.

El parámetro pMensaje se usa para enviar un mensaje a la consola de depuración.

Segs =MideRetraso("### Sleep ha tardado # ");

Imprimirá en la consola de depuración (trazas):

### Sleep ha tardado # 1.00435 s.

Si el mensaje que se pasa es NULL o "", no imprime nada.

Usar las funciones QueryPerformanceCounter y QueryPerformanceFrequency, OutputDebugString y \_snprintf.

* 1. (N1) Probarlo para imprimir, en MBytes por segundo, la velocidad de escritura en el disco duro con la siguiente función.

int EscribeK(int NroKBytes, char \* NombreArchivo)

{

static char Buffer[1024];

int counter;

FILE \*ptr\_myfile;

ptr\_myfile=fopen(NombreArchivo,"wb");

if (!ptr\_myfile)

{

printf("Unable to open file!");

return 1;

}

for (counter=0; counter <NroKBytes; counter++)

{

fwrite(Buffer, sizeof(Buffer), 1, ptr\_myfile);

}

fclose(ptr\_myfile);

return 0;

}

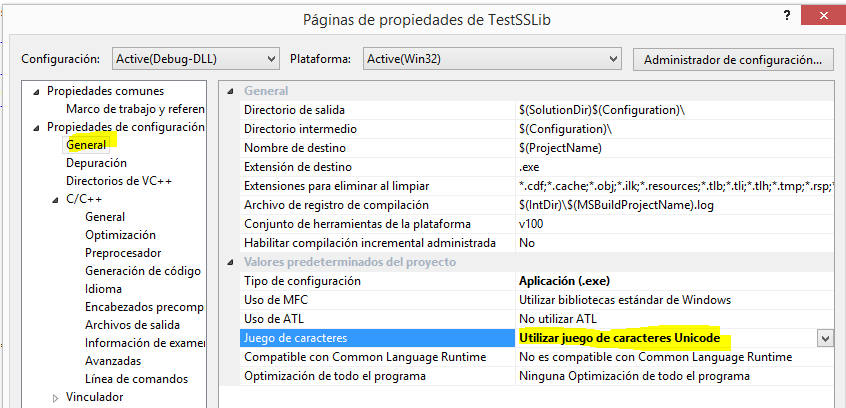
* + 1. Instalar la biblioteca (hacer accesible la biblioteca **nusuario-LibSS** a otras soluciones).

Hasta ahora, siempre que hemos creado una biblioteca se ha usado en la misma solución Visual Studio. Si queremos usa nuestra biblioteca, por ejemplo, *nusuario-LibSS*, desde una solución externa tenemos que hacer accesible sus archivos .h, .lib y .dll[[47]](#footnote-48) a dicha solución. Tenemos varias posibilidades:

* No hacer nada. Los archivos estarán bajo la carpeta de su solución, en este ejemplo *nusuario-SolP5*. Cualquier solución externa que quiera usar nuestra biblioteca debe usar una ruta que incluya la carpeta *nusuario-SolP5* para acceder a los archivos de la biblioteca.   
  Es lo más simple, aunque tendremos algunos inconvenientes:
* Cualquier cambio en la posición o nombre de *nusuario-SolP5* afecta a las rutas almacenadas en las soluciones externas que usen la biblioteca.
* Cualquier cambio en el estado de compilación de la biblioteca *nusuario-LibSS* puede afectar a soluciones externas. Por ejemplo, si limpiamos la solución *nusuario-SolP5* borraremos todos los archivos .exe, .lib y .dll de la solución, incluyendo los de la biblioteca.
* Estos inconvenientes están agravados por el hecho de que el fallo se “produce” cuando manipulamos *nusuario-SolP5*, pero normalmente no se detecta hasta mucho más tarde, cuando usamos la biblioteca *nusuario-LibSS* en otras soluciones.
* **Instalar la biblioteca tras cada compilación correcta**. El término “instalar la biblioteca” implica al menos *copiar los archivos de la biblioteca a una localización conocida* del sistema de archivos[[48]](#footnote-49).
* Supone añadir una operación de copia de archivos tras cada generación de la biblioteca, pero elimina los inconvenientes de no hacer nada.
  1. (N1) Hacer que el proyecto *nusuario-LibSS* “instale” la biblioteca en la carpeta **libs-nusuario** de su repositorio privado.   
       
     \* La carpeta donde están los archivos se puede definir respecto a AAComm como:  
       
      **$(AAComm)\..\..\nusuario\libs-nusuario**   
        
     MUY IMPORTANTE: Tras hacer este ejercicio, debe añadir "a mano" el archivo en …\**nusuario\libs-nusuario** a su repositorio privado.
* Los pasos para hacerlo se muestran en el video "P05.30. Realizar operaciones tras la generación de un proyecto", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw).
  1. (N1) Hacer que el proyecto *nusuario-P5* use la biblioteca desde la carpeta **libs-nusuario**.  
     1. Funciones básicas.

Usar el proyecto *nusuario-P5* para todos los ejercicios, con una configuración distinta por cada ejercicio.

* Los pasos para hacerlo se muestran en el video "P05.40. Crear configuraciones adicionales en una solución.mp4", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw).
  1. (N1) Realizar función una de estas funciones:  
      a) BOOL LlamaAMessageBoxHastaPulsarCancel(CHAR \*msg);   
      b) BOOL LlamaAMessageBoxHastaPulsarContinue(char \* msg);   
      c) BOOL LlamaAMessageBoxHastaPulsarNo(char \* msg);   
     y el código que la prueba, explicando mediante comentarios qué hace, qué devuelve y cómo se hace. El usuario debe ver en alguna parte del mensaje o título cuantas veces se ha llamado a MessageBox.
  2. (N1) Si hacemos la modificación que se muestra en la figura siguiente, qué efecto tiene en el funcionamiento de MessageBox.



* + 1. Funciones simples de ejemplo.

Ver [System Information Functions](http://msdn.microsoft.com/en-us/library/ms724953%28v=vs.85%29.aspx).

* 1. **Realizar las funciones siguientes y el código que las prueba, explicando mediante comentarios qué hace, qué devuelve, qué errores puede tener y cómo se hace.**
  2. (N1)   
     BOOL NombreUsuario(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado);
  3. (N1)   
     BOOL NombreComputador(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado);
  4. (N2)   
     BOOL MessageBoxConNroProcesadoresDelComputador();
  5. (N2)   
     BOOL MessageBoxIndicandoSiElProcesadorEsIntel\_AMD\_Otro();
  6. (N2)   
     BOOL MessageBoxConMemoriaFisicaInstalada();[[49]](#footnote-50)

Más funciones (será por funciones…):

* 1. (N1)   
     WORD NroYNombreDiscosLogicos(CHAR \*dirCadNombres, DWORD tamCadNombres); [[50]](#footnote-51)
  2. (N2)  
     WORD EspacioTotalEnDiscos(DWORD \*pEspacioLibre); [[51]](#footnote-52)
  3. (N1)   
     BOOL DirectorioDeTrabajo(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado); [[52]](#footnote-53)
  4. (N1)  
     BOOL DirectorioWindows(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado); [[53]](#footnote-54)
  5. (N2/N3)  
     BOOL InfoVolumen\_C(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado); [[54]](#footnote-55)
  6. (N2)  
     BOOL FechaYHoraLocal(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado);   
      Usando las funciones GetLocalTime y GetTimeFormat.
  7. (N2)  
     BOOL FechaYHoraSistema(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado);   
     Usando las funciones GetSystemTime y GetTimeFormat.
  8. (N2)   
     BOOL ZonaHoraria(CHAR \*dirCadenaResultado, DWORD tamCadenaResultado); [[55]](#footnote-56)
  9. (N2)   
     BOOL MessageBoxConNroProgramasEnEjecucion();[[56]](#footnote-57)
  10. (N2)   
      BOOL MessageBoxConNroDeviceDriversEnComputador();[[57]](#footnote-58)
  11. (N3)   
      BOOL MessageBoxConDireccionMACDelComputador();[[58]](#footnote-59)
  12. (N2)  
       DWORD MBytesFisicaEnComputador(double \* PorcentajeLibreMemFisica); [[59]](#footnote-60)
  13. ((N2)   
       DWORD MBytesVirtualEnComputador(double \* PorcentajeLibreMemVirtual); [[60]](#footnote-61)
  14. ¿A cuáles de los ejercicios anteriores afecta la modificación que describe el Ejercicio 5.10. ? ¿En qué funciones?

1. Depuración. Manipulando imágenes.
2. Oculto: Numeración ejercicios

17 y 22 de noviembre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211117/20211122).

Contenido

[6.1. Expectativas: 55](#_Toc88125166)

[6.1.a. Qué hacer antes de entrar al laboratorio… 55](#_Toc88125167)

[6.2. Evaluación. 56](#_Toc88125168)

[6.3. Qué hacer en el laboratorio. 56](#_Toc88125169)

[6.4. Ejercicios. 56](#_Toc88125170)

[6.4.a. Ejercicios de procesado de imágenes. 58](#_Toc88125171)

[6.5. Más detalles: Imágenes en OpenCV. 61](#_Toc88125172)

[6.6. Más detalles: Aritmética de saturación. 62](#_Toc88125173)

* 1. Expectativas:

En esta práctica usaremos el depurador para corregir fallos en programas C++.

* + 1. Qué hacer antes de entrar al laboratorio…

Repasar lo visto en clase sobre la sección 4.6. (Programación en Windows desde C/C++) del tema 4.

Muy importante: actualice el contenido de su repositorio \_pub con una operación Update.

Los archivos del curso pasado son DISTINTOS de los de este curso.

Crear un proyecto Visual Studio con las características:

* La solución debe colocarse en la carpeta **P06** de su repositorio privado. En los nombres que siguen, sustituir nusuario por su UVUS.
* Nombre de la solución: **nusuario-SolP06**
* Opción "Crear directorio para la solución" activa.

Crear un proyecto en la solución:

* Proyecto nusuario-P06.
* Tipo de proyecto inicial: Visual C++, Proyecto vacío.
* El proyecto usará su biblioteca **nusuario-LibSS** (archivos .lib y .h, en la carpeta **libs-nusuario** de su repositorio privado), que ha realizado en la práctica anterior.
* Ver Ejercicio 5.8. de la práctica anterior[[61]](#footnote-62).
* Debe funcionar correctamente al menos la función *CheckError*.
* Añadir al proyecto los archivos que aparecen en la carpeta que aparece en la carpeta **\_pub\Practicas\P06\_archivos** del repositorio público[[62]](#footnote-63).
* No es necesario que el proyecto compile ni ejecute correctamente. Se hará en el primer ejercicio.

La práctica empieza a partir de este punto.

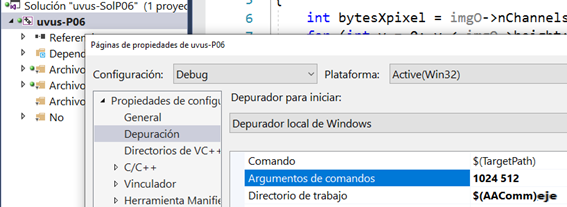
* 1. Evaluación.

Ver evaluación de la práctica anterior (se evalúa junto con esta).

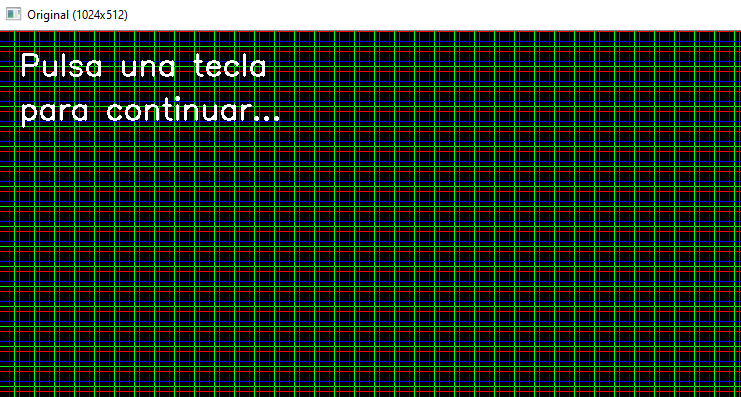
* 1. Qué hacer en el laboratorio.
* Hacer funcionar el programa del primer ejercicio.
* Hacer al menos uno de los ejercicios de la sección 6.4.a.
  1. Ejercicios.

Los argumentos de los programas que siguen se pasan por línea de comandos.

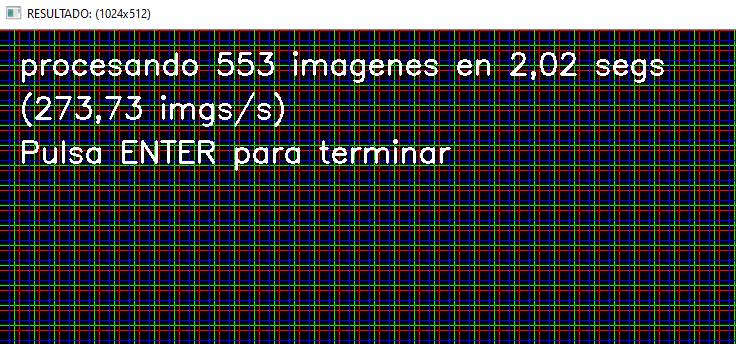
Para hacer que el depurador use argumentos en la línea de comandos, se ponen en la propiedad "Argumentos de comandos" del proyecto:



* 1. Corregir el código del proyecto **nusuario-P06** para que ejecute sin ningún error, y tal como ejecuta el programa …**\_pub\\_comm\bin\P06-TestImg.exe** que está en el repositorio público de la asignatura.   
       
     Si ejecutamos el programa:  
       
     **…>..\_pub\\_comm\bin\P06-TestImg.exe 1024 512**  
       
     Aparece una ventana con una imagen patrón inicial:



Si pulsamos una tecla, aparece una imagen similar a la siguiente:



Este programa sirve para testear funciones de proceso de imágenes y medir su velocidad de ejecución (en imágenes procesadas por segundos).

El código inicial tan solo copia la imagen de origen a la de destino, sin ningún proceso adicional. En el computador de la imagen, la velocidad de proceso es de 273 imágenes por segundo.

* 1. (N3) Fallos latentes  
     Una vez corregidos los fallos del ejercicio anterior, podemos ver que la función ***ImagenAVentana*** tiene un fallo latente que se manifiesta cuando quitamos el comentario de la función printf en este código:  
       
     Imagen que contiene captura de pantalla

     Descripción generada automáticamente  
       
     veremos que cvDestroyWindow no cierra la ventana.   
     ¿Por qué se produce este fallo?
     1. Ejercicios de procesado de imágenes.

De los ejercicios que aparecen en esta sección, uno de ellos aparece casi resuelto en el archivo en Enseñanza Virtual, en la sección ***Información🡪P06: Archivo con fallos para ejercicio de la seccion 6.4.a.*** ~~de su repositorio privado misArchivos\P06.4.a….cpp~~. Corregir los errores que contenga dicho archivo, y hacer que funcione tal como se pide en el ejercicio.

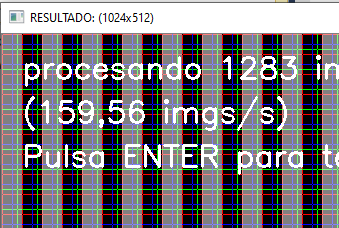
En cada ejercicio, debe

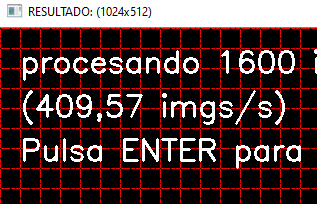
* sustituir el contenido de la función *ProcesaImagen* por una llamada a la función correspondiente. Por ejemplo, en el Ejercicio 6.3. , el contenido de *ProcesaImagen* se sustituye por una llamada a *BandasVerticales*.
* Caso de que se pida, implementar al menos un patrón de prueba.
* Si lo desea, puede incluir en el proyecto una imagen de prueba y cargarla con cvLoadImage.

**En la prueba P05\_6.B. de enseñanza virtual (ver sección 5.2. ) se le pedirá explicar en qué consiste el error y cómo lo ha detectado y corregido**.

NOTA: para poder comparar mejor las imágenes antes y después del proceso, conviene comentar la llamada a *cvDestroyWindow* en Principal.cpp.

Realizar al menos uno de los siguientes ejercicios

* 1. **BandasVerticales.**   
     Muestra bandas verticales de color más claro en la imagen:  
       
     

* 1. **FiltroRojo**.  
     En la imagen resultado se eliminan las componentes verde y azul de los píxeles de la imagen original.   
     Aplicado a la imagen de ejemplo, solo se vería la cuadricula roja:  
       
       
       
     Probar el programa implementando el patrón **Patron\_TodoBlanco**, que genera una imagen original toda en blanco. El resultado sería una imagen toda en rojo.
  2. **InvierteFilas**.  
     Invierte el orden de todas las filas de una imagen; la primera pasa a ser la última…  
       
     Patrones recomendados de prueba: Cualquiera del tipo Patron\_xxx\_BandaHorizontalyyyy, … (el patrón de salida será especular en vertical).  
       
     Si la imagen original se ha generado con el Patron\_Verde\_BandaHorizontal0\_255:  
       
      Imagen original Imagen resultante  
      
  3. **InvierteCols**.  
     Similar al ejercicio anterior, pero invirtiendo el orden de todas las columnas de una imagen: la primera pasa a ser la última…  
     Patrones recomendados de prueba: cualquiera del tipo Patron\_xxxx\_BandaVerticalyyyy.
  4. **CambiaLuminosidad**.  
     Tiene un parámetro adicional, **N**. Cambia la luminosidad de la imagen en un factor N (>=0.0). Dependiendo del valor de **N**, transforma la luminosidad de la imagen. Para hacerlo sólo hay que multiplicar cada componente de color, en cada píxel, por el factor y limitar el resultado al ranco 0-255- Si N es mayor que 1.0, la imagen resultante es más clara. Si es menor que 1.0 será más oscura[[63]](#footnote-64).  
       
     Patrones recomendados de prueba: Cualquiera con cambios de color uniforme (Patron\_XXXX\_BandaVerticalUniforme). Si N es 2.0, la imagen será más clara, si N ew 0.5, más oscura.
  5. **AGris**.  
     Pasa la imagen de color a tonos de grises. Hay que transformar el color de cada píxel de forma que los valores rojo, verde y azul sean iguales (tono de gris). Una forma de hacerlo es sustituyendo los valores R, G y B de cada píxel por la media ponderada de la luminosidad de los colores del píxel, que se calcula como 0.3\*R + 0.55\*G + 0.15\*B[[64]](#footnote-65).  
     Patrones recomendados de prueba: Cualquiera con zonas uniformes de color (Patron\_xxx\_BandaHorizontalUniforme), preferiblemente verde o azul.
  6. **ScrollHorizontal**.  
     Tiene un parámetro adicional, **N**.   
     Scroll horizontal de la imagen. Hay que:  
       
     \* Almacenar N scanlines del principio de la imagen.  
     \* Mover el resto de los scanlines al principio de la imagen.  
     \* Almacenar las N scanlines al final de la imagen.   
       
     Patrones recomendados de prueba: Patron\_xxx\_BandaHorizontalUniforme.
  7. **ScrollVertical**.   
     Scroll vertical de la imagen. Similar al proceso anterior, pero usando columnas en lugar de scanlines.   
     Patrones recomendados de prueba: Patron\_xxx\_BandaVerticalUniforme.
  8. **XExpand**.  
     La imagen resultante es una ampliación horizontal de la mitad izquierda de la imagen original:  
      Imagen original Resultado  
       
     Patrones recomendados de prueba: Patron\_xxxx\_BandaVertical0\_255
  9. **YExpand**.  
     Similar al ejercicio anterior, pero realizando una ampliación vertical de la mitad inferior de la imagen hacia arriba.  
     Patrones recomendados de prueba: Patron\_xxxx\_BandaHorizontal0\_255
  10. XShrink.  
      La imagen resultante es una contracción horizontal de la mitad izquierda de la imagen original:  
       Imagen original Resultado  
        
      La zona a la derecha de la imagen de resultado se puede rellenar con blanco o negro.  
        
      Patrones recomendados de prueba: Patron\_xxxx\_BandaVertical0\_255
  11. **YShrink**.  
      Similar al ejercicio anterior, pero realizando una contracción vertical.  
      Patrones recomendados de prueba: Patron\_xxxx\_BandaHorizontal0\_255
  12. Más detalles: Imágenes en OpenCV.

La biblioteca OpenCV permite manipular imágenes bmp usando estructuras de tipo [**Iplimage**](http://docs.opencv.org/2.4/modules/core/doc/old_basic_structures.html?highlight=iplimage#iplimage) y las funciones asociadas.

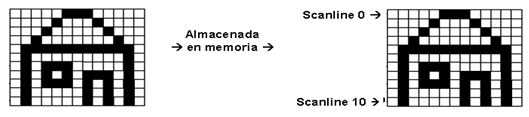
Una estructura *Iplimage* contiene información sobre el tamaño en bytes de la imagen, alto, ancho (en pixels), etc. junto con un puntero al mapa de bits.

La matriz de puntos se almacena en la dirección indicada en el campo *imageData* de la estructura *Iplimage*. Esta matriz de puntos es de tipo cvMat. La organización de la memoria a la que apunta imageData depende del formato de la imagen. En todas las prácticas vamos a usar imágenes con pixeles de 24 bits por pixel. Cada pixel usa el formato RGB (Red, Green, Blue). En RGB, el color de un píxel se expresa como la Combinación de 3 colores básicos: rojo, verde y azul. En este caso, cada píxel tiene 3 bytes:



El byte en la dirección más baja corresponde siempre al componente de color azul.

Para almacenarla en memoria, la imagen se divide la imagen en líneas horizontales (**scanline**). OpenCv almacena los scanlines en memoria empezando de arriba a abajo y de izquierda a derecha:



Si cada punto tiene 24 bits, la forma de almacenamiento en memoria es:



Cada scanline tiene al final una serie de bytes de relleno que sirven para alinear el comienzo de la siguiente scanline a una dirección de palabra de 32 bits.

* 1. Más detalles: Aritmética de saturación.

En un píxel RGB, los componentes de color son enteros de 8 bits que varían entre 0 y 255.

Para hacer operaciones con un componente de color se suele utilizar aritmética de saturación: el resultado de cualquier operación que supere 255 se “satura” a 255, y el cualquier resultado menor que 0 se “satura” a 0.

Por ejemplo, si sumamos 10 a 250 en aritmética de saturación el resultado es 255. En la aritmética modular sin signo con módulo 256[[65]](#footnote-66) que se usa en computadores, el resultado seria 4 (el valor “da la vuelta en 255).

En la práctica, para aplicar aritmética de saturación debemos hacer las operaciones con tipos mayores que unsigned char y con signo. Si el resultado es mayor que 255 se hace 255 y si es menor que 0 se hace 0.

#### Significado físico de la aritmética de saturación.

Para “reproducir” el píxel en una pantalla o cualquier otro dispositivo se necesitan 3 “fuentes” de color (rojo, verde y azul). La luz emitida por estas fuentes se combina en el ojo humano y permite ver el color del píxel[[66]](#footnote-67). Cada componente de color RGB especifica la intensidad lumínica de cada una de sus fuentes, variando desde 0 (no vemos nada) a 255 (intensidad máxima que podemos reproducir/ver).

Por tanto, el significado físico del píxel hace que el resultado de cualquier operación sobre un componente de color esté limitado al rango 0-255. Cualquier aumento de la luminosidad no será superior a la máxima luminosidad que podemos ver/reproducir. Del mismo modo, cualquier disminución de la luminosidad nunca será menor que 0.

1. Ejercicios control de ejecución (tema 5).
2. Oculto: Numeración ejercicios

24 y 29 de noviembre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211124/20211129).

Contenido

[7.1. Expectativas: 65](#_Toc88125174)

[7.1.a. Qué hacer antes de entrar al laboratorio… 65](#_Toc88125175)

[7.2. Evaluación. 66](#_Toc88125176)

[7.3. Qué hacer en el laboratorio. 66](#_Toc88125177)

[7.4. Ejemplo previo. 66](#_Toc88125178)

[7.5. Lanzando hilos. 67](#_Toc88125179)

[7.5.a. UI threads y working threads. 67](#_Toc88125180)

[7.5.b. Ejecución paralela. 68](#_Toc88125181)

* 1. Expectativas:

En esta práctica se realizan ejercicios de creación de hilos, y la sincronización más básica: cómo esperar a que un hilo creado termine.

* + 1. Qué hacer antes de entrar al laboratorio…

Repasar la parte del tema 5 vista en clase:

* Qué son procesos e hilos.
* Cómo se crea un hilo en Windows.

Qué traer de casa:

La solución debe colocarse en la carpeta **P07** de su repositorio privado. En los nombres que siguen, sustituir *nusuario* por su UVUS.

* Nombre de la solución: **nusuario-SolP07**
* Proyecto nusuario-P07.
* Incorporar al proyecto su biblioteca **nusuario-LibSS** (archivos .lib/.h generados en la Práctica 5. ).
* Realizar el programa que se pide en el Ejercicio 7.1.

La práctica empieza a partir de este punto.

* 1. Evaluación.

Esta práctica se evalúa junto con la siguiente. 3 pruebas:

**Retrasadas fechas de entrega hasta después de las vacaciones de navidad.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prueba | Tipo prueba. | Fecha, hora límite | | | Máx/min. |
| Turnos lunes | Turno miércoles | |
| P07.B | Publicada en EV el día de la práctica 7.  **~~Se entrega antes de la P09.~~** | ~~20/12/2021, 15:40~~  **7/1/2022, 15:40** | | ~~22/12/2021, 15:40~~  **7/1/2022, 15:40** | 3 |
| P08.B | Publicada en EV durante la práctica 8. Se entrega la semana siguiente. **~~Se entrega antes de la P09.~~** | ~~20/12/2021, 15:40~~  **7/1/2022, 15:40** | | ~~22/12/2021, 15:40~~  **7/1/2022, 15:40** | 3 |
| P07\_08.B | Prueba evaluación en EV sobre P07 y P08 (al inicio de ~~P09~~ **P10**-15 minutos) | ~~20/12/2021~~  **10/1/2022** | | ~~22/12/2021~~.  **12/1/2022** | 4 |

**No existe prueba previa en esta práctica.**

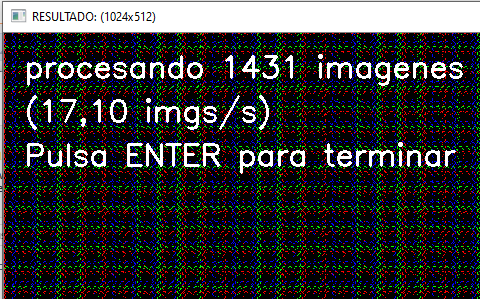
La prueba P07.B consiste en hacer el Ejercicio 7.5. , el Ejercicio 7.8. y el Ejercicio 7.11.

La ponderación de esta práctica es 1,2.

* 1. Qué hacer en el laboratorio.

Hacer los ejercicios a partir del Ejercicio 7.2.

* 1. Ejemplo previo.
  2. **Ejemplo básico.** Hacer un programa que reutilice el código del Ejercicio 6.2. y, antes de mostrar cada fotograma, llame a la función *TrasMalla*, que crea un efecto de “malla” sobre la imagen:



La malla se consigue cambiando los valores de *x* e *y* en cada pixel de la imagen de destino, tal como se muestra en el código de [P07\_TrasMaya.cpp](https://github.eii.us.es/IC3-SS/_pub/blob/master/Practicas/P07_TrasMalla.cpp).

* 1. **Ejemplo básico con imágenes desde video.**Modificar el programa anterior para cumplir las siguientes especificaciones:  
       
     a) La imagen de origen se toma desde un video y va cambiando conforme se va reproduciendo el video. **No es necesario que el video se reproduzca a velocidad real.**   
       
     b) El programa se debe ejecutar especificando el nombre del video como único argumento en la línea de comandos.

**\* La primera versión debe, al menos, procesar sin interrupción los 2400 primeros frames del video[[67]](#footnote-68).**

Recordar, de la primera práctica, que para leer una imagen desde video tenemos que abrir la imagen con:

CvCapture\* g\_capture = cvCreateFileCapture(…);// Abre archivo de video

Y ya podemos leer cada imagen con:

IplImage \* imgOriginal = cvQueryFrame(g\_capture);

* 1. Lanzando hilos.
     1. UI threads y working threads.

Un uso muy habitual de la programación multihilos es independizar la gestión del interfaz de usuario (UI) del proceso propiamente dicho. Se divide el programa en un hilo que gestiona la UI (UI thread) y uno o varios dedicados al resto de procesos (workers threads).

* 1. ¿Por qué dividir UI threads + worker threads? ¿Qué interés tiene está división?
  2. Modificar el programa del Ejercicio 7.2. para que tenga dos hilos:  
       
     \* Un hilo se encarga del interfaz de usuario. Es bastante simple. Se pide al usuario que introduzca un número. Si:  
      \* Es 0: para o continúa la reproducción.  
      \* Es negativo: termina el programa.  
      \* Es **positivo**: modifica el valor de la variable amplitudDistorsion[[68]](#footnote-69) al número introducido.  
      \* Ignora cualquier otro caso.   
     \* Otro hilo se encarga de todo lo demás.  
       
     Cualquier organización es válida mientras los bloqueos de un hilo no afecten la ejecución del otro.

Como interfaz de usuario podemos usar el siguiente código:

void miniMenu() {

int nroCampos, cmd=0;

while (TRUE) {

char ent[20] = "Ninguno";

system("CLS");    // Ejecuta el comando CLS (clear screen) en el shell

printf("Ultimo comando: %d\nY ahora?\n\n", cmd);

printf("0  - Salir\n<0 - Parar/Continuar\n>0 - Modificar amplitud distorsion\n¿Nuevo comando?\n");

nroCampos = scanf("%d", &cmd); // Ojo, esto fall cuando no le pasamos una entrada correcta    // Ver cómo se corrige en clase usando gets\_s y sscanf

if (nroCampos != 1) {

// Corrección:

**while ('\n' != getchar());// vacía el buffer de entrada.**

**// Si scanf tiene un error al convertir la entrada, no la "consume"**

**Beep(1000, 500); // Avisa que ha habido un error**

continue;

}

if (cmd == 0)

ExitProcess(0); // Termina el programa

else if (cmd > 0 && cmd < 100)

amplitudDistorsion = cmd;

else if (cmd < 0) {

// Suspender o rearrancar el hilo

}

}

}

* 1. Modificar el programa anterior sustituyendo la función *TrasMalla* por la función que tiene asignada en la sección 6.4.a. de la Práctica 6.   
        
     También debe adaptar qué hace el programa si el usuario introduce un número positivo.
  2. ¿Por qué no es posible usar *miniMenu* si el programa tiene un solo hilo?
     1. Ejecución paralela.
  3. **Ejemplo básico multihilo (I)**. Realizar la función *TrasMalla\_2x.* Realiza la misma función que TrasMalla, pero con dos hilos. Cada uno de ellos hace la mitad del proceso.   
       
     Medir la diferencia de tiempo entre hacerlo con un hilo y con 2.

Un ejercicio similar se muestra en el video "P07.Ej6.Lanzando 2 hilos para procesar una imagen (curso 2016-17)", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)

* 1. Modificar el programa anterior sustituyendo la funciónTrasMalla por la función que tenga asignado en el Ejercicio 7.5.
  2. **Ejemplo básico multihilo (II)**. Hacer una tercera versión de la función anterior, *TrasMalla\_Nx*, para que haga el proceso usando N hilos (N puede ser cualquier valor).
  3. **Ejemplo básico multihilo(III)**. Probar el programa anterior para distintos valores de N (1 a 20). ¿A partir de qué valor de N empieza a disminuir la velocidad del proceso?
  4. Realizar los dos ejercicios anteriores sustituyendo la función *TrasMalla* por el proceso que tenga asignado en el Ejercicio 7.5.

1. Ejercicios de control de concurrencia y sincronización (tema 8).

1 y 13 de diciembre de 2021.

Turnos 3 y 4 tienen un día adicional el 15 de diciembre, de asistencia libre.

1. 1 Oculto: Numeración ejercicios

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211201/20211205).

Contenido

[8.1. Expectativas: 71](#_Toc88125182)

[8.1.a. Qué hacer antes de entrar al laboratorio… 71](#_Toc88125183)

[8.2. Evaluación. 72](#_Toc88125184)

[8.3. Qué hacer en el laboratorio/casa. 72](#_Toc88125185)

[8.4. Ejercicios. 72](#_Toc88125186)

[8.4.a. Antes de empezar. 72](#_Toc88125187)

[8.4.b. Más problemas con el acceso simultaneo a un recurso. 74](#_Toc88125188)

[8.4.c. Sincronización básica entre hilos. 74](#_Toc88125189)

[8.4.d. Sincronización básica entre procesos. 76](#_Toc88125190)

* 1. Expectativas:

En la práctica anterior se realizaron ejercicios básicos de creación de hilos y procesos; y la sincronización más simple: cómo esperar a que un hilo/proceso creado termine.

En esta se usan mecanismos más complejos de sincronización: eventos, secciones críticas, semáforos, etc.

* + 1. Qué hacer antes de entrar al laboratorio…

Repasar lo visto en clase sobre:

* Creación de hilos y procesos (tema 5).
* Mecanismos de sincronización (tema 6).

Qué traer de casa:

La solución debe colocarse en la carpeta **P08** de su repositorio privado. En los nombres que siguen, sustituir nusuario por su UVUS.

* Nombre de la solución: **nusuario-SolP08**
* Proyecto nusuario-P08.
* Incorporar al proyecto su biblioteca **nusuario-LibSS** (archivos .lib y .h), hechos en la Práctica 5.
* Realizar el código que se pide en Ejercicio 8.1. a Ejercicio 8.3.

La práctica empieza a partir de este punto.

* 1. Evaluación.

Ver evaluación de la práctica anterior (se evalúa junto con esta).

* 1. Qué hacer en el laboratorio/casa.

Hacer al menos 2 ejercicios de cada una de las secciones siguientes, usando distintos mecanismos de control de concurrencia y sincronización vistos en el tema 8.

Importante: actualizar repositorio público antes de empezar la práctica.

* 1. Ejercicios.
     1. Antes de empezar.
  2. **Un problema simple de concurrencia: acceso simultáneo a un recurso.**  
     Crear N hilos que incrementen una variable global.   
     La variable vale inicialmente 0. Cada hilo la incrementa en NRO\_INCREMENTOS veces y termina. El programa principal crea los N hilos y espera a que terminen todos antes de mostrar por pantalla el resultado.  
      \* ¿Se manifiestan los problemas de concurrencia si N vale 10 y NRO\_INCREMENTOS vale 1000?  
      \* ¿Y si N vale 100 o 1000?  
      \* ¿Y si lo que cambiamos es NRO\_INCREMENTOS a 1000\*1000?

El código de cada hilo es:

…

//Cada hilo la incrementa en NRO\_INCREMENTOS veces y termina

for (int i = 0; i < NRO\_INCREMENTOS; i++) {

contador++;

}

return 0;

…

* 1. Arreglar el problema anterior usando alguno de los mecanismos de sincronización vistos en clase.

El recurso compartido a proteger es, en este caso, el código que incrementa la variable global contador.

* 1. MuestraVideoN**.**Modificar el código del Ejercicio 7.4. de la siguiente forma:  
     \* El video se ve simultáneamente en N ventanas.   
     \* Se elimina cualquier referencia a la función TrasMalla y la variable amplitudDistorsion.  
     \* El video se reproduce a velocidad real en cada ventana.   
       
     Hacer N igual a 2 para una primera prueba[[69]](#footnote-70).   
     Probar las alternativas no reentrante y reentrante que se explican a continuación.

Probar 2 alternativas para la reproducción a velocidad real. En ambas se elimina todo el código que use imgOrigen u se sustituye el bloque que hay entre las 2 llamadas a *MideRetraso* por el código que se muestra.

a) Alternativa ***no reentrante***:

MideRetraso(0);

IplImage \* imgResultado = cvQueryFrame(g\_capture);

if (0 == imgResultado)

break;

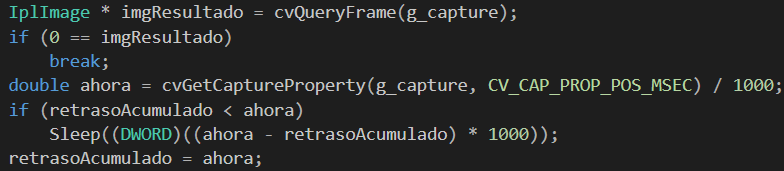
double ahora = cvGetCaptureProperty(g\_capture, CV\_CAP\_PROP\_POS\_MSEC) / 1000;

if (retrasoAcumulado < ahora)

Sleep((DWORD)((ahora - retrasoAcumulado) \* 1000));

retrasoAcumulado += MideRetraso(0);

b) Alternativa ***reentrante***:



**a)** Mide la reproducción con mayor exactitud que **b)** pero no funciona correctamente con más de un hilo[[70]](#footnote-71).

* 1. Sincronización + timeouts.  
     Modificar el código anterior para que el usuario pueda cerrar una ventana introduciendo un número entre 0 y N-1.
  2. Usar algún objeto de sincronización para terminar el hilo que muestra las imágenes.
     1. Más problemas con el acceso simultaneo a un recurso.

IMPORTANTE: Puede encontrar los algoritmos asociados a la mayoría de los ejercicios en el documento **P08.Algoritmos**, en la carpeta "Material" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw).

* 1. **MuestraVideoNSync**.  
     Modificar *MuestraVideoN* para que todas las ventanas estén sincronizadas. Es decir, si el video en una ventana va retrasado respecto a la más adelantada, acelera la reproducción hasta alcanzarla.
  2. **MuestraVideoNEn1**.  
     Modificar el programa MuestraVideoN para que:  
      \* Cada hilo muestre un video distinto.   
      \* Todos los hilos usen la misma ventana para mostrar el video.   
       
     Ojo: el resultado puede ser algo confuso… Empezar con solo 2 videos distintos.   
     Para probar este programa es conveniente usar videos cortos.   
       
     Modificarlo para que muestre, alternativamente, 5 segundos de cada video en la misma ventana.
  3. **MuestraVideoNyCuentaMsecsTotal**.  
     Modificar el programa MuestraVideoN para que el número total de msegundos mostrados en todas las ventanas se guarde en la variable global mSecsTotal. El hilo principal imprime, cada segundo, el valor de mSecsTotal.
  4. ¿Qué recurso/recursos es necesario proteger en cada uno de los ejercicios anteriores?
     1. Sincronización básica entre hilos.
  5. **Tú cuentas, yo imprimo (I).**   
     Crear un hilo que incremente una variable cada segundo. Imprimiremos desde el hilo principal el valor de la variable cada vez que cambie.

Cuestiones a resolver:

* Cómo saben el nuevo hilo y el principal qué variable deben incrementar e imprimir, respectivamente.
* Alternativas.
* Cómo se depuran los hilos.
* ¿El hilo que cuenta, está contando correctamente?
* Qué tipo de objetos se deben usar para sincronización.
* Quitar las funciones de sincronización. ¿Qué pasa?
  1. **Tú cuentas, yo imprimo (II). MuestraVideo1Imprime**.   
     Modificar el programa MuestraVideoN para que haya un solo hilo secundario, que muestra video, y el hilo principal imprima el valor de los segundos de video reproducidos en el hilo secundario. Se imprime solo cuando el valor de los segundos cambia.

Cuestiones a resolver

* Cómo se prueba si la sincronización es correcta.
* Mientras seleccionemos con el ratón el título de la ventana OpenCV, la reproducción del video se detiene. Esto debería detener también la impresión de los segundos.
  1. **Tú cuentas, yo imprimo (III). MuestraVideoNyCuentaMsecsSinc**.  
     Modificar el programa MuestraVideoN para que el número total de msegundos mostrados en todas las ventanas se guarde en la variable global mSecsTotal. El hilo principal imprime mSecsTotal solo cuando su valor cambia contabilizando 5 segundos. Por ejemplo, se imprime si mSecsTotal pasa de 4953 a 5003, o de 9975 a 10009, pero no si pasa de 13984 a 14052.
  2. **Tú la llevas (I)**.  
     Realizar un programa que lanza N hilos simultáneamente, que se presentarán en orden. Cada hilo imprime una frase y termina:  
       
     “Hola, soy el hilo i”  
       
     donde i es el orden en que fue creado.  
     Probar el programa primero con N= 2, después con valores de N muy altos.

Cuestiones a resolver

* Cómo sabe cada nuevo hilo cuál es su orden de creación y cuándo debe imprimir.
* Alternativas.
* Alternativas para la sincronización.
  1. **Tú la llevas (II).**  
     Modificar el programa MuestraVideo para que:  
      \* La primera ventana reproduce un segundo de video y se para. La segunda ventana reproduce un segundo y se para… así hasta la última.  
      \* Cuando se para la última sigue reproduciendo la primera y el ciclo comienza.
     1. Sincronización básica entre procesos.

* 1. **Evitar múltiples ejecuciones**.  
     Queremos hacer que un programa no permita ser ejecutado múltiples veces hasta terminar su ejecución actual.  
     Es decir, si hacemos doble click en su archivo el programa empieza a ejecutar. Si intentamos volver a ejecutar el programa (nuevo doble click en el archivo) antes de que termine la primera ejecución, la nueva instancia nos mostrará un mensaje indicando que no puede continuar y termina.

Cuestiones a resolver

* ¿Qué mecanismo de sincronización debemos usar para hacer esto y cómo se usa?
* Implementar el mecanismo usando el programa:

int main(int argc, char \*argv[]){

int i=0;

printf("\nEntra en proceso....\n");

for (i=0; i<20; i++) {

printf("Ciclo %d\n", i);

Sleep(1000);

}

}

* 1. **Evitar múltiples ejecuciones (II)**.  
     Modificar el programa anterior para que una instancia espere hasta que no haya ninguna otra en ejecución para comenzar a operar.

1. Comunicación entre procesos: tuberías.

Software de Sistemas. 3º IC.

20 y 22 de diciembre de 2021.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20211220/20211222).

Contenido

[9.1. Expectativas: 77](#_Toc88125191)

[9.1.a. Qué hacer antes de entrar al laboratorio… 78](#_Toc88125192)

[9.2. Evaluación. 78](#_Toc88125193)

[9.1. Qué hacer en el laboratorio/casa. 79](#_Toc88125194)

[9.2. El sistema CaseConverterPipe. 79](#_Toc88125195)

[9.2.a. El sistema final. 79](#_Toc88125196)

[9.2.b. Primer paso: un servidor básico, con funcionalidad mínima. 79](#_Toc88125197)

[9.2.c. Completando la funcionalidad. 80](#_Toc88125198)

[9.2.d. Múltiples sesiones. 80](#_Toc88125199)

[9.2.e. Múltiples conexiones. 80](#_Toc88125200)

[9.3. El sistema 7ZipPipe. 80](#_Toc88125201)

[9.3.a. El sistema final. 80](#_Toc88125202)

[9.3.b. Antes de empezar: cómo se ejecuta otro proceso. 81](#_Toc88125203)

[9.3.c. Hacer el servidor/cliente con una sola conexión. 82](#_Toc88125204)

[9.3.d. Comprobar soporte de múltiples sesiones/conexiones. 82](#_Toc88125205)

[9.4. El sistema VideoPipe. 82](#_Toc88125206)

[9.4.a. Hacer el servidor/cliente con una sola conexión. 82](#_Toc88125207)

[9.4.b. Comprobar soporte de múltiples sesiones/conexiones. 83](#_Toc88125208)

1. 1 Oculto: Numeración ejercicios

* 1. Expectativas:

Usaremos las funciones básicas de gestión de tuberías y creación de procesos. El resultado final serán tres grupos de programas que usan tuberías para comunicar entre servidor y clientes:

* Sistema CaseConverterPipe. Un proceso servidor implementa una tubería que permite convertir de mayúscula a minúscula, o viceversa, los datos que se le pasan. Este es un ejemplo de sistema simple, aunque con servicios poco útiles.
* Sistema 7ZipPipe. El servidor recibe archivos, los comprime (y encripta) usando un programa compresor (7zip) externo, y devuelve el resultado al cliente. Es un ejemplo de sistema con tuberías más complejo, pero también con servicios más útiles.
* Sistema VideoPipe. El servidor proporciona imágenes de un video a los clientes.

Está práctica se organiza como una secuencia de ejercicios conectados de forma que al final se obtienen los dos primeros sistemas. El tercero es una combinación de los programas de video vistos hasta ahora con el uso de tuberías.

* + 1. Qué hacer antes de entrar al laboratorio…

Repasar lo visto en clase sobre:

* Creación de hilos y procesos (tema 5).
* Sincronización usando tuberías con nombre (tema 6).
* Vamos a modificar los archivos ***T6\_LoggerPClient.cpp[[71]](#footnote-72)*** y ***T6\_LoggerPServer.cpp[[72]](#footnote-73)*** vistos en clase. Se recomienda revisar cómo funcionan esos programas y sus limitaciones.

A partir del curso 2020/21, veremos esto en la práctica 9.

* Las funciones CreateFile/ReadFile/WriteFile (tema 7, visto en clase de teoría).

Qué traer de casa:

Copiar la solución en la carpeta **T06\T06\_Tuberias-Sol,** del repositorio público (\_pub), a la carpeta **P09** de su repositorio privado. Adaptarla para que se compile y ejecute correctamente en su nueva ubicación.

La solución final tiene 2 proyectos:

* Cliente: contiene el archivo T6\_LoggerPClient.cpp.
* Servidor: contiene el archivo T6\_LoggerPServer.cpp.

No es necesario cambiar el nombre de la solución ni los proyectos.

La práctica comienza a partir de este punto.

* 1. Evaluación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prueba | Tipo prueba. | Fecha, hora. | Máx/min | | |
| Todos los turnos | |  |
| P09.B | Prueba evaluación en enseñanza virtual. Hasta sección 9.3. (incluida). No es necesario subirlas al repositorio. | Al final P10.  10/01/2022, T1 y T2  12/01/2022, T3 y T4 | | 10/6,5[[73]](#footnote-74) |
| P09.C | Bonus track: Hacer sección 9.4. y **subirla al repositorio privado antes de la P09.B**.  Se evalúa en persona durante la P10, solo si la puntuación en P09.B es mayor o igual que 6,5. | Al final P10.  10/01/2022, T1 y T2  10/01/2022, T3 y T4 | | 3 |

No existe prueba previa en esta práctica.

La ponderación de esta práctica es 1,6.

* 1. Qué hacer en el laboratorio/casa.

Hacer los ejercicios de, al menos, las secciones 9.2. y 9.3.

En clase se hacen estos ejercicios para la tubería en T6\_LoggerPClient.cpp

Importante: actualizar repositorio público antes de empezar la práctica.

* 1. El sistema CaseConverterPipe.
     1. El sistema final.

El servidor proporciona una tubería, llamada CaseConverterPipe, que recibe los siguientes comandos:

* 'U' + Bloque de datos. Se interpreta el bloque de datos como caracteres ASCII. Se pasa a mayúsculas el bloque de datos.
* 'L' + Bloque de datos. Se pasa a minúsculas el bloque de datos.
* Cualquier otro valor en el primer byte hace que se ignore el comando.

Para pasar a mayúsculas/minúsculas, usar la función LCMapString. Un ejemplo de uso:

//Pasa a mayusculas LCMapString

LCMapString(LOCALE\_USER\_DEFAULT, LCMAP\_UPPERCASE,

BEntrada, Leidos, BEntrada, Leidos);

* + 1. Primer paso: un servidor básico, con funcionalidad mínima.
  1. Hacer la versión básica de cliente y servidor adaptando el código de T6\_LoggerPClient.cpp y T6\_LoggerPServer.cpp

**En clase: Ejercicio 6.11 y 6.12 de SS-010-Notas de clase…**

* T6\_LoggerPClient11.cpp. Servidor básico:
* Acepta una sola conexión simultánea.
* Acepta una sola sesión[[74]](#footnote-75).
* Cuando termina la sesión, el programa termina.
* T6\_LoggerPClient11.cpp. Una versión básica del cliente:
* Lee el nombre de archivo que se le pasa en línea de comando.
* Convierte a mayúsculas el contenido del archivo usando la tubería CaseConverterPipe.

**En casa:**

* CCPipeServer11.cpp. Servidor básico:
* Solo pasa a mayúsculas (no hay comando, sólo se pasa el bloque de datos).
* Acepta una sola conexión simultánea.
* Acepta una sola sesión[[75]](#footnote-76).
* Cuando termina la sesión, el programa termina.
* CCPipeClient11.cpp. Una versión básica del cliente:
* Lee el nombre de archivo que se le pasa en línea de comando.
* Convierte a mayúsculas el contenido del archivo usando la tubería CaseConverterPipe.
  1. Hacer las siguientes mejoras en el cliente:  
      \* Espera hasta que la tubería esté disponible.   
      \* Tendrá un tiempo máximo de espera (timeout) de 30 segundos.  
     Alternativas:  
      \* Llamar a CreateFile hasta que el servidor responda correctamente.  
      \* Usar un evento (+ fácil, pero hay que cambiar el servidor).  
      \* Usar la función [WaitNamedPipe](https://msdn.microsoft.com/en-us/library/aa365800(v=vs.85).aspx) (+ complicado).

**En clase: Modificamos 6.12 de SS-010-Notas de clase… para implementarlo.**

* + 1. Completando la funcionalidad.
  1. Hacer que el servidor acepte los comandos del programa final: U + Datos, y L + Datos.   
       
      \* ¿Se puede probar sin modificar el cliente?  
       
     Modificar el cliente para probar los nuevos comandos.
     1. Múltiples sesiones.
  2. En qué consiste la limitación a una sesión. En los programas anteriores, el servidor sólo admite una sesión. ¿Cómo se puede probar esto?
  3. Modificar el servidor para que soporte múltiples sesiones.

**En clase: Ejercicio 6.13 y 6.14 de SS-010-Notas de clase…**

* + 1. Múltiples conexiones.
  1. En qué consiste la limitación a una sola conexión. En los programas anteriores, el servidor sólo admite una conexión. ¿Cómo se puede probar esto?: Alternativas.
  2. Modificar el servidor para que soporte un número ilimitado de conexiones.

**En clase: Ejercicio 6.15** **de SS-010-Notas de clase…**

* 1. El sistema 7ZipPipe.
     1. El sistema final.

El servidor proporciona una tubería, llamada **7ZipPipe**, que sirve para comprimir archivos que recibe desde los clientes. Se comporta de la siguiente forma:

* El cliente envía un archivo al servidor usando la tubería 7ZipPipe. Espera la respuesta (contenido del archivo comprimido) y la graba como archivo .7z.

Un esquema más detallado del protocolo es el siguiente:

|  |  |  |
| --- | --- | --- |
| Cliente | 7ZipPipe (tubería) | Servidor |
| Abre tubería.  Envía archivo a comprimir | --> | Lee la tubería y guarda los datos en un archivo propio. |
| Cierra la tubería |  |  |
|  |  |  |
|  |  |  |
| Abre la tubería |  | Ejecuta el programa 7Zip para comprimir el archivo propio. |
| Lee respuesta.  Escribe respuesta en archivo propio. | <-- | Envía el archivo comprimido de vuelta. |
| Cierra la tubería |  |  |

* 1. Qué **problema tiene el protocolo anterior.** Modificarlo para que funcione correctamente.
     1. Antes de empezar: cómo se ejecuta otro proceso.
  2. **Ejecuta7Zip.cpp**. Hacer la función:  
       
      **BOOL Ejecuta7Zip(char \* NArchivo);** // Comprime el archivo NArchivo usando  
      // el compresor 7zip

Por ejemplo, Ejecuta7Zip ("Prueba.dat"), ejecuta 7Zip[[76]](#footnote-77) usando CreateProcess con una configuración equivalente a la siguiente línea de comandos:

c:\...>7z.exe a Prueba.dat.7z Prueba.dat

La función devuelve:

* FALSE, si ha no ha logrado comprimir el archivo.
* TRUE si lo ha comprimido.

Cuestiones a resolver:

* Cómo se usa CreateProcess para ejecutar 7za.exe.
* Cómo se espera a que el proceso creado termine.
* Cómo se sabe si 7za.exe se ha ejecutado correctamente.
* En teoría, un proceso puede indicar si se ha ejecutado correctamente o no usando el “exit code”. ¿Hace eso 7za.exe?
* Si no lo hace ¿alternativas?
  + 1. Hacer el servidor/cliente con una sola conexión.

Podemos aprovechar la estructura de los programas realizados en los pasos anteriores.

* 1. Reutilizar el código de la sección 9.2.d. para hacer el servidor y cliente de 7ZipPipe.   
     Probarlo con una sola conexión.
* 7zipPipeServerii.cpp. Servidor:
* Lee el archivo que le manda el cliente y lo graba.
* Lo comprime llamando Ejecuta7Zip.
* Envía el resultado al cliente.
* **7zipPipeClient.cpp**. Una versión básica del cliente:
* Lee el nombre de archivo que se le pasa en línea de comando.
* Lo envía al servidor.
* Guarda el resultado en un archivo, con el nombre derivado del archivo de entrada (añadiéndole .7z).
  + 1. Comprobar soporte de múltiples sesiones/conexiones.
  1. En el ejercicio anterior, cómo se puede tratar el que el servidor reciba más de un archivo simultáneamente.
  2. El sistema VideoPipe.

El servidor tiene una tubería, llamada **VideoPipe**, que proporciona imágenes de video.

Cuando detecta una conexión, el servidor abre un archivo de video y envía los frames que va leyendo al cliente.

El cliente recibe los frames y los muestra por pantalla.

* + 1. Hacer el servidor/cliente con una sola conexión.

Podemos aprovechar la estructura de los programas realizados en los pasos anteriores.

* 1. Reutilizar el código del Ejercicio 1.21. [[77]](#footnote-78) para hacer el cliente.  
     \* En lugar de leer frames desde el archivo de video, los recibe desde la tubería.
  2. Reutilizar el código del Ejercicio 1.21. y de la sección 9.2.d. para hacer el servidor.   
     \* El código de la función *play* sirve como base para leer una imagen de video y enviarla a la tubería.   
     1. Comprobar soporte de múltiples sesiones/conexiones.
  3. En el ejercicio anterior, cómo se puede tratar el que el servidor atienda múltiples clientes simultáneamente.

1. Comunicación entre procesos: tuberías. Sesión 2.

Segunda sesión de la práctica anterior: 10 y 12 de enero de 2022.

[Turnos](https://calendar.google.com/calendar/embed?mode=WEEK&src=id2uunrd8qt1c5bhiv4gjpn0kc@group.calendar.google.com&ctz=Europe/Madrid&&dates=20220110/20220112).

1. Oculto: Numeración ejercicios
2. Oculto: Numeración ejercicios

🡨 Revisado hasta aquí curso 2021/22

#### Fin del documento.

1. Ver documento 00.05\_AntesDeEmpezar.docx y 00.10\_CrearNuevoProyectoCpp.docx en la carpeta [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)/Recetas. [↑](#footnote-ref-2)
2. Pueden usarse las claves de la licencia del dpto. de ATC. Ver página de la asignatura en <https://ev.us.es>, en la carpeta *Información*. [↑](#footnote-ref-3)
3. Solo aparece si tenemos una cuenta "personal" en Microsoft con el correo …@alumn.us.es, ya que todos los alumnos de la ETSI tienen también una cuenta institucional con su UVUS. [↑](#footnote-ref-4)
4. En algunos computadores pueden no verse correctamente los campos para introducir uvus y contraseña. En tal caso, pulsar las teclas CRTL+ hasta que se vea correctamente.   
   El problema es debido a que, para mostrar esta ventana, se usa un navegador integrado que puede tener un zoom inadecuado. El zoom de ese navegador se cambia pulsando simultáneamente CONTRO+ (Acercar) o CONTRO - (Alejar) [↑](#footnote-ref-5)
5. Esto es por un fallo de diseño en el servidor github.eii.us.es: cuando se accede a un repositorio **privado** sin haber iniciado sesión antes, no pide credenciales, sino que indica que el enlace no existe. [↑](#footnote-ref-6)
6. Los archivos añadidos dependen del estado actual del repositorio: pueden ser distintos a los que aparecen en las imágenes que se muestran en este documento. [↑](#footnote-ref-7)
7. Ver secciones 0.7.a. Añadir archivos/carpetas al repositorio., y 0.6.c. Actualizar repositorio (commit). [↑](#footnote-ref-8)
8. Si se ha instalado Git con VS2019, la ruta depende de la versión de VS2019 instalada (Professional, Enterprise o ¿Community?):  
     
   C:\Program Files (x86)\Microsoft Visual Studio\2019\**Professional**\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer\Git\cmd

   C:\Program Files (x86)\Microsoft Visual Studio\2019\**Enterprise**\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer\Git\cmd

   C:\Program Files (x86)\Microsoft Visual Studio\2019\**Community**\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer\Git\cmd [↑](#footnote-ref-9)
9. Ver detalles en [*SS-Pub*](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)*/Recetas/00.20. Reutilizar código y opciones de un proyecto anterior (sin plantilla)* [↑](#footnote-ref-10)
10. Las opciones de depuración de un proyecto C++ se guardan en un archivo con extensión .user, que no se trata correctamente en el asistente. [↑](#footnote-ref-11)
11. Cuidado con esto: si se pulsa Commit, los cambios se guardan en el repositorio local, pero no se suben al repositorio remot. [↑](#footnote-ref-12)
12. Entorno integrado de desarrollo. Ver Tema 1. [↑](#footnote-ref-13)
13. Ver tema 1. [↑](#footnote-ref-14)
14. Ver documentos (en [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)/Recetas)  
     00.05\_AntesDeEmpezar.docx   
     00.10\_CrearNuevoProyectoCpp.docx en la carpeta. [↑](#footnote-ref-15)
15. Visto en asignaturas anteriores, revisado en tema 1. [↑](#footnote-ref-16)
16. Visto en asignaturas anteriores, revisado en tema 1. [↑](#footnote-ref-17)
17. La información necesaria para crear un proyecto de este tipo puede encontrarse en los archivos "00.05\_AntesDeEmpezar..." y "00.10\_CrearNuevo..." en la carpeta [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)\Recetas. [↑](#footnote-ref-18)
18. Pistas:  
    - Ver [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)\Recetas\P01.15. Configurar los directorios de inclusión…  
    - En general, buscar información usando los códigos mostrados al compilar. Por ejemplo, [C3861](https://www.google.com/search?q=C3861). [↑](#footnote-ref-19)
19. Ver [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)\Recetas\P01.20. Agregar biblioteca .lib a proyecto…. En general, buscar información usando los códigos mostrados al vincular (link). Por ejemplo, [LNK2019](https://www.google.com/search?q=LNK2019). [↑](#footnote-ref-20)
20. Ver [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)\Recetas\P01.30. Modificar PATH durante depuración…. [↑](#footnote-ref-21)
21. Ver [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw)\Recetas\P01.40. Modificar directorio de trabajo en depuración…. [↑](#footnote-ref-22)
22. Ver tema 1, y apuntes de Fundamentos de Programación (1er. curso). [↑](#footnote-ref-23)
23. Ver tema 2. [↑](#footnote-ref-24)
24. Las trazas se llaman en VS *breakpoints actions*. Un ejemplo simple de cómo usarlas puede verse en la documentación de [esta](https://www.visualmicro.com/page/User-Guide.aspx?doc=Working-With-Breakpoints-When-Hit.html) extensión de VS para Arduino (aunque las *actions* no es una característica de la extensión, sino de VS en general). [↑](#footnote-ref-25)
25. Cuidado con la condición de breakpoint: puede no existir ningún fotograma con el tiempo **exacto** de 2000 ms. Cómo usar los breakpoints condicionales [aquí](https://docs.microsoft.com/es-es/visualstudio/debugger/using-breakpoints#breakpoint-conditions). [↑](#footnote-ref-26)
26. Tan solo hay que modificar la posición donde se inserta el texto en cada frame. [↑](#footnote-ref-27)
27. Cuando existe nota mínima, una puntuación menor en esta prueba divide por 2 la nota total de la práctica. [↑](#footnote-ref-28)
28. Ha emigrado a un país con mejores perspectivas. [↑](#footnote-ref-29)
29. En cuanto a los tipos, tampoco están muy seguros, pero la mayoría opinan que eran estos. [↑](#footnote-ref-30)
30. Por ejemplo, imprime en pantalla el valor de la variable y llama a la función. [↑](#footnote-ref-31)
31. El nombre de la biblioteca y su contenido es distinto para cada alumno. [↑](#footnote-ref-32)
32. Nombre de variable o función. [↑](#footnote-ref-33)
33. Aunque el ejemplo del video es un archivo .obj, el mecanismo funciona igual para archivos.lib. [↑](#footnote-ref-34)
34. En Propiedades de configuración del proyecto, en C/C++-->Avanzadas hay que seleccionar Compilar como código de C++ (/TP). [↑](#footnote-ref-35)
35. En C++ se usa el operador extern "C" en el prototipo de una función para indicar que es una función externa C. Más detalles [aquí](http://en.cppreference.com/w/cpp/language/language_linkage). [↑](#footnote-ref-36)
36. Traducción más usada del término [name mangling](https://en.wikipedia.org/wiki/Name_mangling" \l "C). [↑](#footnote-ref-37)
37. Solo hay que declarar una función y una variable con el modificador *extern "C"*, usarlas en el código (por ejemplo, en main), pero no definirlas en ningún sitio. El vinculador dará un error indicando qué símbolo se estaba buscando. [↑](#footnote-ref-38)
38. Cambiar el nombre de las funciones en el nuevo proyecto (añadiendo el sufijo “\_dll”, por ejemplo) para distinguir claramente cuándo se está usando la biblioteca dinámica y cuándo la estática. [↑](#footnote-ref-39)
39. No se ve claramente las diferencias en la generación de bibliotecas estáticas y dinámicas. [↑](#footnote-ref-40)
40. Valga la redundancia. [↑](#footnote-ref-41)
41. Para ver este nombre, hay que ejecutar *DUMPBIN /EXPORTS NombreDeLaBiblioteca.dll*. [↑](#footnote-ref-42)
42. En este ejemplo hay al menos dos formas de verlo:

    * quitar la biblioteca de importación (nusuario-P4-Dynamic.lib) de las opciones del proyecto que la usa (nusuario-P4-TestLib). Aparecerá un error de vinculación al no encontrar el símbolo de importación. En el nombre de ese símbolo se puede ver la decoración.
    * Analizar con DUMPBIN /EXPORTS los símbolos del archivo de la biblioteca de importación del proyecto nusuario-P4-Dynamic (nusuario-P4-Dynamic.lib).

    [↑](#footnote-ref-43)
43. Poner un breakpoint en la primera instrucción de main (nusuario-P4-TestLib) y otro en la primera instrucción de DllMain (proyecto nusuario-P4-Dynamic). ¿A cuál se llega antes? [↑](#footnote-ref-44)
44. La función puede estar vacía, ya que el contenido de la función se hace en la práctica. [↑](#footnote-ref-45)
45. Cuando existe nota mínima, una puntuación menor en esta prueba divide por 2 la nota total de la práctica. [↑](#footnote-ref-46)
46. También está disponible en el video/pdf "P05.20. Funcion CheckError (tema 4)", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw). [↑](#footnote-ref-47)
47. Solo bibliotecas dinámicas. [↑](#footnote-ref-48)
48. Puede implicar también operaciones adicionales, como modificar variables de entorno (PATH, por ejemplo), el registro del computador, etc. En esta práctica usaremos la instalación más simple: solo copiamos los archivos a una posición conocida. [↑](#footnote-ref-49)
49. Ver función GetPhysicallyInstalledSystemMemory. [↑](#footnote-ref-50)
50. Ver funciones GetLogicalDriveStrings y GetLogicalDrives. [↑](#footnote-ref-51)
51. Ver funciones GetDiskFreeSpace y GetDiskFreeSpaceEx. [↑](#footnote-ref-52)
52. Ver función GetCurrentDirectory. [↑](#footnote-ref-53)
53. Ver función GetSystemDirectory. [↑](#footnote-ref-54)
54. Ver función GetVolumeInformation. [↑](#footnote-ref-55)
55. Ver función GetTimeZoneInformation. [↑](#footnote-ref-56)
56. En Windows, a programa en ejecución se le denomina *proceso*. Con la función *EnumProcesses* podemos obtener información para determinar cuántos procesos hay en ejecución en un momento dado (normalmente por encima de 100). También es posible usar la función *PerformanceInformation*. [↑](#footnote-ref-57)
57. En Windows podemos usar la función *EnumDeviceDrivers* podemos obtener información para determinar cuántos drivers hay cargados en un momento dado (normalmente por encima de los 150). [↑](#footnote-ref-58)
58. Los elementos …Data4[2] a Data4[7] del UUID creado son la MAC del computador (si existe). Se puede comprobar ejecutando ipconfig /ALL en la línea de comandos. [↑](#footnote-ref-59)
59. Ver función GlobalMemoryStatusEx. [↑](#footnote-ref-60)
60. Ver función GlobalMemoryStatusEx. [↑](#footnote-ref-61)
61. Si su proyecto funciona correctamente, se recomienda reutilizar las opciones de proyecto de ese ejercicio. [↑](#footnote-ref-62)
62. **No** arrastrar los archivos de la carpeta \_pub/Practicas/P06\_archivos al proyecto. Primero copiar los archivos a la carpeta del proyecto *nusuario-P6,* y después añadirlos al proyecto. [↑](#footnote-ref-63)
63. Ver sección “6.1. Más detalles: Aritmética de saturación.”, al final de este enunciado. [↑](#footnote-ref-64)
64. La ponderación está relacionada con la percepción relativa de cada color para el ojo humano. [↑](#footnote-ref-65)
65. Es la que se usa en C cuando en una operación aritmética todos los datos son de tipo *unsigned char*, por ejemplo. [↑](#footnote-ref-66)
66. Suponiendo que la intensidad sea suficiente (en caso contrario no vemos nada), el píxel sea lo suficientemente pequeño y estemos lo suficientemente lejos (caso contrario, veremos las tres fuentes de luz separadas, no combinadas en un color) [↑](#footnote-ref-67)
67. El programa falla a los 1460 frames, aproximadamente. Veremos por qué en el tema 7 (Gestión de memoria). [↑](#footnote-ref-68)
68. En la versión original de TrasMalla, esta variable está definida dentro de la función. [↑](#footnote-ref-69)
69. Ver el video "P08.Ej03.2019-Video en varias ventanas", en la carpeta "Recetas" de [SS-Pub](https://uses0-my.sharepoint.com/:f:/g/personal/camaya_us_es/Eofg1Tj0tzRCkD1jypkeNJ0Byb8Eua5T73OVMQcKuiFoTw) [↑](#footnote-ref-70)
70. El fallo se manifiesta al mover una de las ventanas. [↑](#footnote-ref-71)
71. Ver carpeta **T06\T06\_Tuberias-Sol \T06\_Cliente\src** en el repositorio publico \_pub. [↑](#footnote-ref-72)
72. Ver carpeta **T06\T06\_Tuberias-Sol \T06\_Servidor\src** en el repositorio publico \_pub. [↑](#footnote-ref-73)
73. La nota mínima en esta práctica afecta a la evaluación de la prueba P09.C: solo se evalúa a quienes superen la puntuación mínima. [↑](#footnote-ref-74)
74. Sesión = secuencia CreateFile... ReadFile/WriteFile (múltiples)... CloseHandle desde un cliente [↑](#footnote-ref-75)
75. Sesión = secuencia CreateFile... ReadFile/WriteFile (múltiples)... CloseHandle desde un cliente [↑](#footnote-ref-76)
76. La versión de 7Zip para usar en línea de comandos puede encontrarse [aquí](http://downloads.sourceforge.net/sevenzip/7za920.zip). [↑](#footnote-ref-77)
77. Ejercicio 1.21. Modificar la función play del ejercicio anterior para que reproduzca a velocidad “real”.  
    es el ejercicio final de la práctica 1, que reproduce video a velocidad real. [↑](#footnote-ref-78)