

Universidad Tecnológica Centroamericana  
UNITEC

30 de agosto del 2019

**Sesión #7**

Laboratorio de Programación III

Javier Cano

# Desarrollo de la práctica

Para el desarrollo satisfactorio del laboratorio, siga al pie de la letra cada instrucción que a continuación se le presenta.

# Objetivos del laboratorio

* Hacer uso de archivos binarios en C++
* Hacer verificaciones por medio de excepciones en C++

# Ejercicio practico

Se le pide realizar un keylogger, no para “hackear” computadoras con Windows que estén expuestas, sino para mantener vigilancia sobre empleados que trabajan en su empresa, y mantener un historial de todos los comandos que realizan en sus computadoras.

La arquitectura recomendada para el programa consiste en:

Una clase **Consola**, que tiene el nombre de la persona que está utilizando la consola y un **Logger**. En la clase **Consola** tiene que estar implementada la funcionalidad de la consola (debe haber un método llamado cmd).

|  |  |
| --- | --- |
| **Atributos de Consola** | **Tipo** |
| usuario | string |
| logger | Logger \* |

La consola debe soportar cualquier comando de Windows (por medio de la función system) y también el comando cd para cambiar de directorio (por medio de la función \_chdir en direct.h).

También debe implementar una clase **Logger**, que es la responsable de llevar un historial de comandos realizados por el usuario. La clase debe tener el nombre del archivo (binario) donde se van a guardar los logs, el nombre del usuario que utiliza la consola, el número de log que se está guardando, al igual que un vector de **Logs**.

|  |  |
| --- | --- |
| **Atributos de Logger** | **Tipo** |
| usuario | string |
| nombre archivo | string |
| numero de log | int |
| vector de logs | vector<Log \*> |
| (opcional) writer | ofstream |

Un **Log** consiste simplemente de un numero entero (un número que lleve cuenta de los comandos que se van ejecutando), el texto del comando que se ejecuta y el nombre del usuario que lo ejecuta.

|  |  |
| --- | --- |
| **Atributos de Log** | **Tipo** |
| usuario | string |
| comando | string |
| numero de log | int |

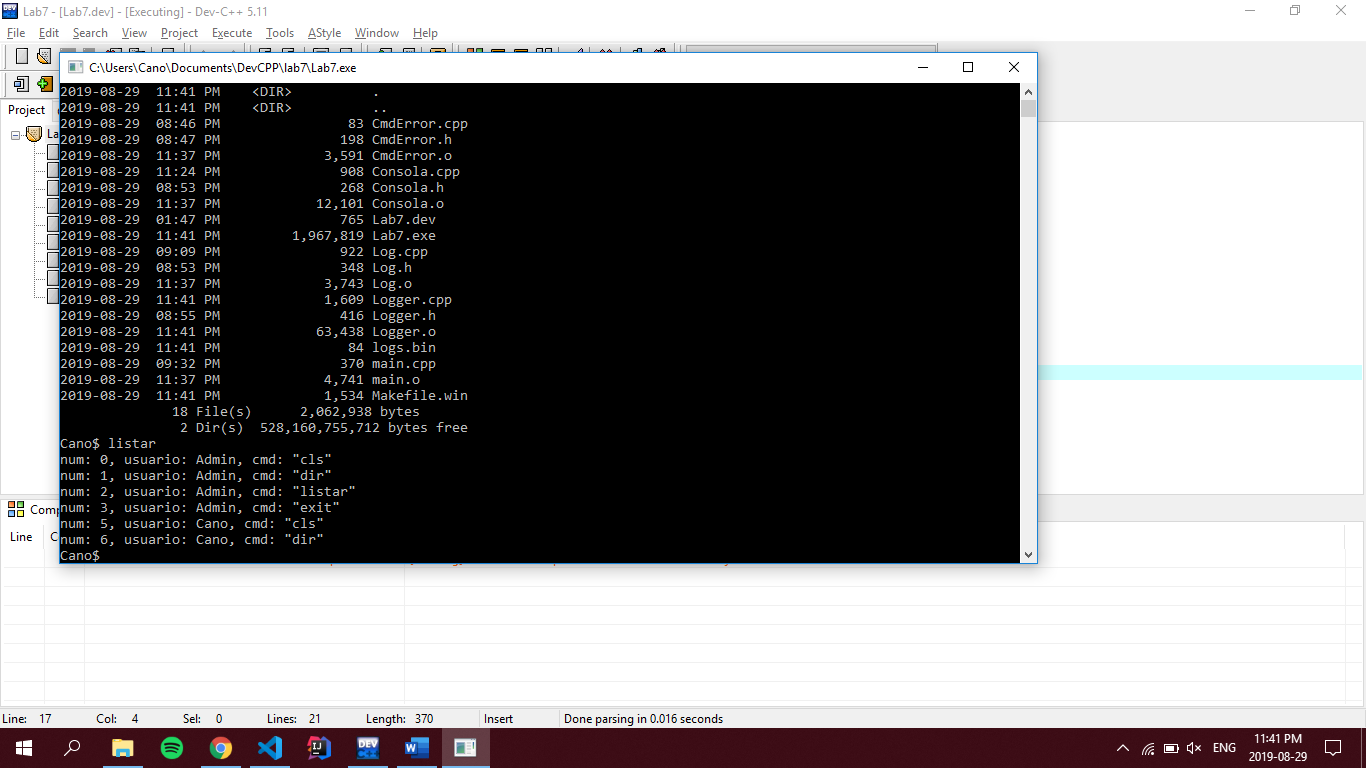
Cada vez que se ejecuta un comando en la consola, el **Logger** toma el texto del comando que se acaba de ejecutar (junto con el número de comando y el nombre del usuario) y lo guarda en el historial (el vector de Logs) como una nueva entrada. Una vez guardado en el vector de Logs, se debe guardar el contenido de este mismo Log en el archivo binario.

Con este archivo binario, al momento de abrir el programa, debemos revisar si existe el archivo, y si existe, cargar todos los contenidos del archivo a el vector de Logs automáticamente.

Además de los comandos de Windows, si se ingresa el comando especial **“listar”** se debe listar todo el contenido del historial del **Logger.**

De igual manera, si se ingresa el comando especial **“exit”** debemos cerrar el programa (tomando en consideración el cerrar archivos abiertos y liberar todo tipo de memoria).

También debe implementar un tipo de excepción, llamada **CmdError** que se dispare en caso de un comando no sea reconocido por el sistema operativo de Windows, y que, al dispararse, haga que el programa se cierre (siempre tomando en cuenta cerrar archivos y liberar memoria).



# Ponderación

|  |  |
| --- | --- |
| Elemento | Puntaje |
| Lectura de archivo binario | 3 |
| Escritura a archivo binario | 2 |
| Resolución del problema | 2 |
| Manejo de Excepciones | 1 |
| Ejecutar comandos de Windows | 1 |
| Liberar Memoria | 1 |

# Comentarios adicionales

Subir el documento como un archivo de **Word (.docx),** incluyendo su nombre completo, número de cuenta y repositorio de GitHub.

**Asegúrense de probar su programa y de que el código sea legible.**

Se aplicarán penalizaciones por quiebres en el código, interrupciones o errores inesperados en el programa y por mala indentación del código. Por cada una de las penalizaciones encontradas se les restara un punto de la nota final de su laboratorio.

Si el código no es legible para otras personas o está mal indentado, entonces los instructores no nos molestaremos por tratar de entender que hace su programa en caso de que se presente un error en el mismo.