Segundo Proyecto de Sistemas Operativos

Monitor de Sistemas

Moreno Morua Luis Alberto
Ramírez Ceseña Angel Orlando

Objetivo:
Aplicar lo aprendido en la clase de sistemas operativos relacionado con: hilos, concurrencia y métodos de sincronización. Para ello realizamos un programa que recopila información del cpu, la memoria y la conexión a internet.

Diseño:
El proyecto fue hecho con C y C++. Usamos la biblioteca <threads.h> para manejar los hilos con C++, y varias APIs de Windows para recopilar la información del sistema.

- Porcentaje de uso del CPU: Con la función GetSystemTimes( ) obtuvimos el tiempo que lleva el sistema ejecutando instrucciones en modo Kernel, Usuario e Idle. Realizamos dos mediciones con al menos un segundo de diferencia, y restamos las cantidades para obtener el tiempo que estuvo en cada modo durante ese segundo. Ahora sólo sumamos el tiempo útil (Tiempo en Kernel + Tiempo en Usuario – Tiempo en Idle) y lo dividimos entre el tiempo total (Tiempo en Kernel + Tiempo en Usuario), y eso lo multiplicamos por 100. ***Nota: El Tiempo en Kernel incluye el tiempo en Idle.***

- Valores de RAM: Estos valores fueron más fáciles de obtener, la mayoría de ellos nos los dio la función GlobalMemoryStatusEx( ). El único valor que tuvimos que calcular fue el de la memoria RAM usada, para ello restamos el valor de RAM libre al total de RAM en el equipo. Ambos datos nos los dio la función, junto con el porcentaje de RAM usada. Después de eso sólo dividimos los datos entre $1x10^9$ para usarlos en Gb.

- Valores de conexión a internet: Con estos valores pretendíamos obtener la cantidad de datos que transmitía y recibía la red por segundo. Aunque tras bastante investigar y probar lo mejor que pudimos obtener fue la cantidad de datagramas recibidos o enviados por segundo. Para ello usamos la función GetIpStatistics( ), que nos regresa una estructura con varios valores sobre la red. Tomamos los que nos interesaban y usando varios if( ) convertimos el valor en kbytes por segundo o Mbytes por segundo. Esto nos permite darnos una idea del uso de nuestra conexión, aunque no muestra el total de datos enviados o recibidos.

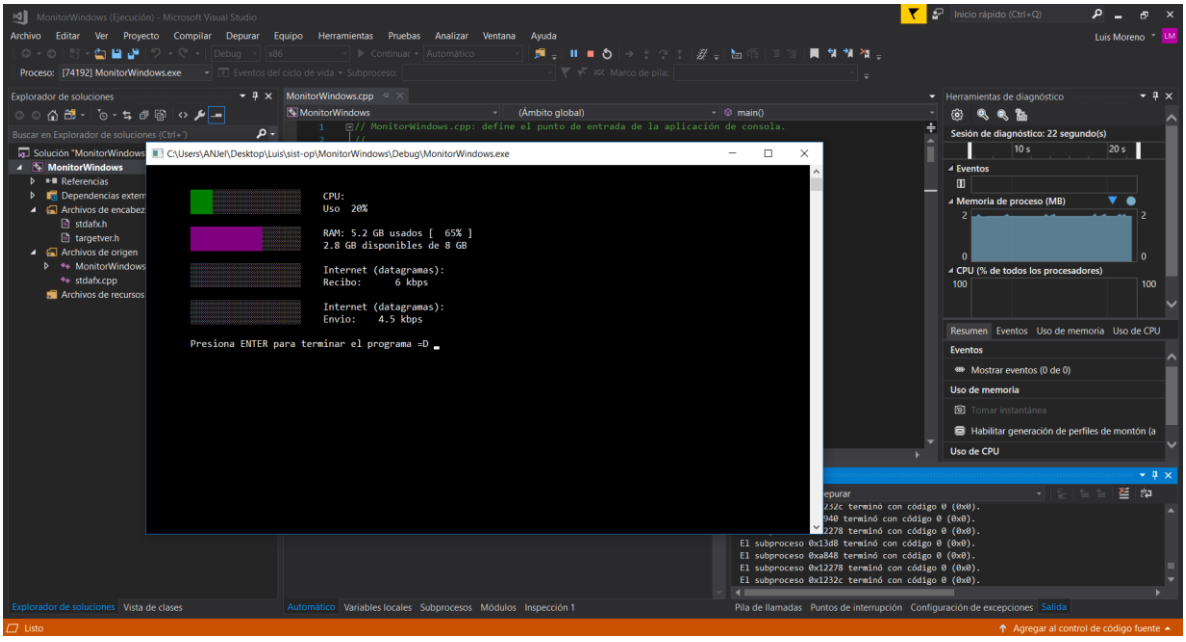
Cuando inicia nuestro programa crea un hilo para cada una de las funciones que se encargan de obtener estos valores y los guardan en varias variables globales. También creamos otro hilo para la función que va a imprimir nuestros valores en pantalla. Estos hilos tienen un

ciclo while( ) que ejecutan la función constantemente hasta que se oprima ENTER. De esta forma creamos un hilo que se acaba hasta que terminamos el programa, y podemos obtener los valores una y otra vez, gracias al ciclo en el que se encuentra nuestra función.
Debido a que la impresión en pantalla se realiza cada medio segundo en automático, necesitamos una forma de mantener los valores constantes mientras se imprime la pantalla. De lo contrario, la impresión nos mostraría una mezcla de los datos antiguos con los recién actualizados. Para ellos creamos la función usoMisDatos( ), que con apoyo de un mutex y varias banderas permite que sólo un proceso use los datos para leerlos al imprimirlos, o reescribirlos al actualizarlos. Las funciones pideImpresion( ) y usoDelCPU( ) la mandan llamar y si alguien la está usando el mutex las mantiene en espera. datagramas( ) y usoDeRam( ) crean un hilo para llamar a la función y se quedan esperando a que una bandera les de la señal de que pueden empezar a modificar los valores.

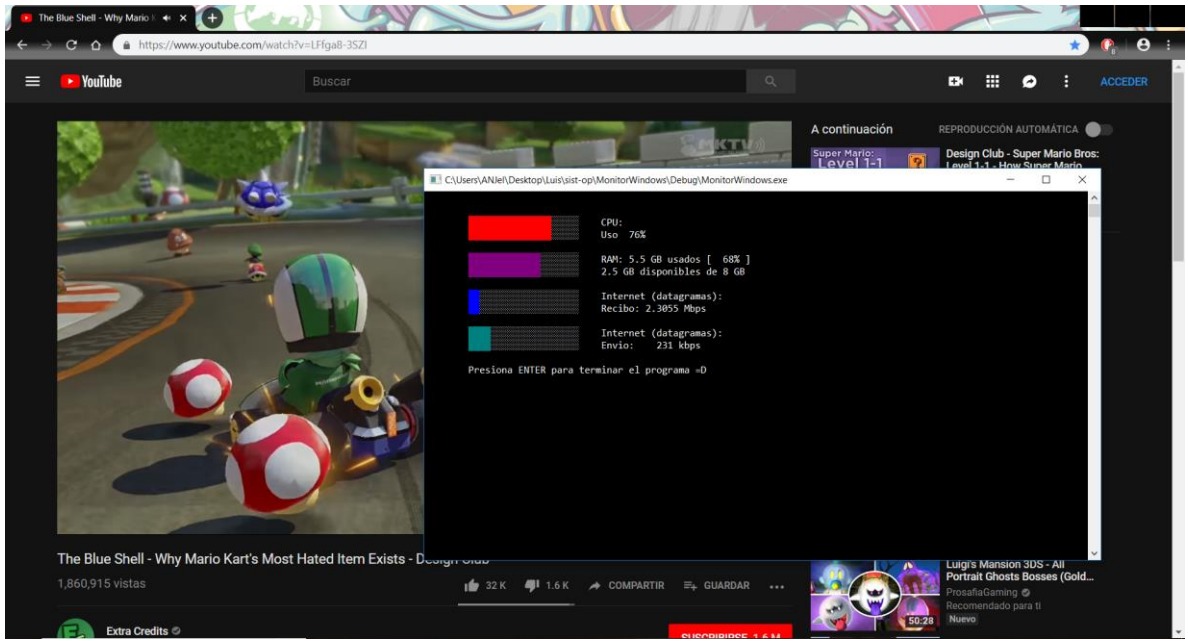
¿Cómo compilarlo?

Para compilarlo usamos Visual Studio 2017, ya que nos permite compilar proyectos con APIs de Windows y la biblioteca threads.h sin tantos problemas. Creamos un proyecto nuevo de Visual C++ en consola, y agregamos el archivo fuente al proyecto, eliminando el que creó por default para que no nos mande error por tener dos archivos con main( ) en un mismo proyecto. O en su defecto copiamos el código del archivo fuente al nuevo archivo creado por Visual Studio, guardamos y compilamos. ***Importante: No eliminar al archivo stdafx.h y stdafx.cpp que crea automáticamente Visual Studio, porque los necesita para compilar y correr el programa.***

Y eso es todo, nuestro programa en consola debería verse así:



Y reaccionar al uso de la computadora:



Fuentes de consulta:

Teoría:

- http://sistop.org/pdf/sistemas_operativos.pdf

Hilos y Mutex en C++:

- https://solarianprogrammer.com/2011/12/16/cpp-11-thread-tutorial/
- https://stackoverflow.com/questions/4989451/mutex-example-tutorial

Para obtener la información del CPU:

- https://github.com/Leo-G/DevopsWiki/wiki/How-Linux-CPU-Usage-Time-and-Percentage-is-calculated
- https://docs.microsoft.com/en-us/windows/desktop/api/sysinfoapi/nf-sysinfoapi-getsystemtime
- https://msdn.microsoft.com/es-es/ms724950

Para obtener la información de la RAM:

- https://docs.microsoft.com/es-es/windows/desktop/api/sysinfoapi/ns-sysinfoapi-_memorystatusex
- https://docs.microsoft.com/es-es/windows/desktop/api/sysinfoapi/nf-sysinfoapi-globalmemorystatusex
- https://helpdesk.commercialnetworkservices.net/index.php?/Knowledgebase/Article/View/185/1/verify-ram-usage

Para obtener la información de la conexión a internet:

- https://docs.microsoft.com/en-us/windows/desktop/api/iphlpapi/nf-iphlpapi-getipstatistics