

Centro Universitario de Ciencias Exactas e Ingenierías

CUCEI



Computación tolerante a fallas 2023B / D06

Uso de hilos y demonios

Profesor: Miche Emanuel López Franco

Gutiérrez Galán Ruben Alejandro Código: 214798315

Hilos:

Los hilos son secuencias independientes de instrucciones que pueden ejecutarse concurrentemente dentro del mismo programa o proceso. A menudo se les llama "subprocesos" o "threads" en inglés.

- **Proceso:** Un proceso es un programa en ejecución en un sistema operativo. Cada proceso tiene su propio espacio de memoria y recursos, como archivos y variables. Un proceso puede contener uno o varios hilos.
- **Hilo:** Un hilo es una secuencia de instrucciones que se ejecuta dentro de un proceso. Los hilos comparten el mismo espacio de memoria y recursos que el proceso padre, lo que significa que pueden comunicarse y compartir datos más fácilmente que los procesos separados.
- **Multihilo:** La programación multihilo implica la creación y gestión de varios hilos dentro de un proceso. Los hilos pueden ejecutarse de forma concurrente, lo que puede mejorar la eficiencia de las aplicaciones al aprovechar los múltiples núcleos de una CPU o al permitir que un hilo continúe su ejecución mientras otro está bloqueado, por ejemplo, esperando entrada/salida.

Ventajas de los hilos:

- **Mayor rendimiento:** Los hilos pueden aprovechar la concurrencia para realizar tareas más rápidamente.
- **Mejora de la capacidad de respuesta:** Los hilos pueden mantener una interfaz de usuario sensible mientras realizan tareas en segundo plano.
- **Facilita la programación paralela:** Los hilos son una forma común de lograr la programación paralela y el procesamiento distribuido.

Desafíos de los hilos:

- **Condiciones de carrera:** Cuando múltiples hilos acceden y modifican compartimentos de memoria compartida, puede ocurrir una condición de carrera si no se sincronizan adecuadamente.
- **Exclusión mutua:** Es necesario utilizar técnicas como semáforos o bloqueos para evitar problemas de acceso concurrente a datos compartidos.
- **Consumo de recursos:** La creación de muchos hilos puede consumir recursos del sistema, y es importante equilibrar la concurrencia con la eficiencia.

Demonios:

Un "demonio" (también conocido como "daemon" en inglés) es un tipo especial de programa que se ejecuta en segundo plano, sin interacción directa con el usuario, y realiza tareas de mantenimiento o servicios específicos para el sistema o las aplicaciones. Aquí tienes algunas características y ejemplos de demonios:

Características de los demonios:

- **Ejecución en segundo plano:** Los demonios se ejecutan como procesos en segundo plano, lo que significa que no tienen una interfaz de usuario visible y no requieren la interacción directa del usuario.
- **Iniciación automática:** Los demonios a menudo se inician automáticamente cuando se inicia el sistema operativo y se ejecutan de manera continua durante toda la vida útil del sistema.
- **Realización de tareas específicas:** Los demonios están diseñados para realizar tareas específicas, como administrar servicios de red, programar tareas, registrar eventos, administrar la impresión, etc.

Ejemplos de demonios comunes:

- **sshd:** El demonio SSH (Secure Shell) permite a los usuarios iniciar sesión de forma segura en un sistema remoto y ejecutar comandos de manera segura.
- **httpd:** El servidor web Apache es un demonio que escucha y responde a las solicitudes de los navegadores web, sirviendo páginas web y recursos a los usuarios.
- **cron:** El demonio cron en sistemas Unix y Linux se utiliza para programar tareas automatizadas en momentos específicos o a intervalos regulares.
- **cupsd:** El demonio CUPS (Common Unix Printing System) se encarga de administrar la impresión en sistemas Unix y Linux.
- **ntpd:** El demonio NTP (Network Time Protocol) sincroniza la hora del sistema con servidores de tiempo externos, asegurando la precisión de la hora del sistema.
- **DBMS (Sistema de Gestión de Bases de Datos):** Los demonios a menudo se ejecutan en segundo plano para administrar sistemas de bases de datos, como MySQL, PostgreSQL o MongoDB.

Programa desarrollado (Continuación del programa recuperable):

El programa que se desarrolló simula el cierre por algún fallo no concreto, en este programa se busca crear un registro nuevo de alumnos, para ello se realiza el llenado de un formulario, durante la el llenado del formulario ocurre el error y se cierra el programa, sin embargo se tiene un autoguardo el cual se ejecuta cada dos segundo (para fines de la simulación se usaron tiempo muy cortos), este autoguardo respalda la información que se ha ingresado en algunos de los campos dentro de un archivo txt. Siempre y cuando exista el archivo se podrá cargar, al iniciar el programa la primera vez el archivo txt no existirá, una vez pase el tiempo establecido para el autoguardado se creará el archivo con los datos ingresados durante esos momentos.

Modificaciones al programa:

Las modificaciones al programa original realmente no son muy grandes, sin embargo, generan permiten un mejor funcionamiento en caso de presentarse errores durante la ejecución.

Función para el guardado de datos en el archivo txt:

Código original:

```
private void respaldo(object sender, EventArgs e){
    //respaldar cada dos segundos
    File.WriteAllLines(archivo_respaldo, new string[] { textBox_nombre.Text, textBox_codigo.Text, textBox_carrera.Text, textBox_contraseña.Text, textBox_universidad.Text });
}
```

Codigo modificado:

```
private void respaldo(object sender, EventArgs e){
    // Utilizar un hilo para realizar la escritura del archivo de respaldo
    Thread respaldoThread = new Thread(() =>
    {
        File.WriteAllLines(archivo_respaldo, new string[] { textBox_nombre.Text, textBox_codigo.Text, textBox_carrera.Text, t
    });

    respaldoThread.Start();
}
```

Este nuevo autoguardado hace uso de hilos para la escritura de los datos recabados en los campos del formulario dentro de un archivo txt.

Función simular cierre:

Codigo original:

```
private void simular_cierre(object sender, EventArgs e){
    //se cerrara despues de 5 segundos
    this.Close();
}
```

Codigo modificado:

```
private void simular_cierre(object sender, EventArgs e){
    // Utilizar un hilo para simular el cierre después de 5 segundos
    Thread cierreThread = new Thread(() =>
    {
        Thread.Sleep(500000); // tiempo de espera para el cierre
        this.Invoke(new Action(() => this.Close())); // Cerrar el formulario en el hilo principal
    });

    cierreThread.Start();
}
```

Al igual que el auto guardado se hace uso de hilos para realizar el cierre de la aplicación pasado un cierto tiempo, en este caso se hace uso de invoke para realizar una acción de cierre (simulando que ocurrió un fallo).

Modificación en algunas variables:

```
private System.Windows.Forms.Timer timer_respaldo;//temporizador de respaldo
private System.Windows.Forms.Timer timer_error_cerrar;//temporizador para simular el cier
private System.Windows.Forms.Timer resourceMonitorTimer;
```

Se realizaron algunas modificaciones al momento de declarar las variables, debido a que ahora se hace uso de hilos (y estos cuentan con un atributo Timer) se genera cierta ambigüedad, por lo cual es necesario que se declare de manera completa el tipo de variable creada, en este caso usamos el Timer pero de Windows forms y no el de Thread.

Nueva función para tener un demonio que nos indique el consumo de memoria:

```
resourceMonitorTimer = new System.Windows.Forms.Timer();  
resourceMonitorTimer.Interval = 30000; // 30 segundos  
resourceMonitorTimer.Tick += ResourceMonitorTimer_Tick;  
resourceMonitorTimer.Start();
```

Vamos a crear temporizador e inicializarlo en el constructor del programa para que realice un análisis de la memoria utilizada por el programa cada cierto tiempo (para fines de muestra está en 30 segundos).

```
private void ResourceMonitorTimer_Tick(object sender, EventArgs e)  
{  
    // Lógica de monitoreo de recursos  
    long memoryUsed = GC.GetTotalMemory(false); // Obtener la memoria utilizada  
    //MessageBox.Show(memoryUsed.ToString());  
    if (memoryUsed > 500000) // Si la memoria utilizada supera 1 GB  
    {  
        // Realiza acciones para liberar memoria o muestra una advertencia al usuario  
        MessageBox.Show("Uso de memoria alto. Realizando limpieza...");  
    }  
}
```

Esta función nos permite monitorear el consumo de memoria que tiene el programa, realiza una consulta de la memoria que esta consumiendo y la compara con un valor de referencia, en este caso está en 500000 equivalente a 0.5MB, si el consumo de memoria en el momento de la ejecución llega a sobrepasar ese limite se enviara un mensaje de alerta al usuario indicando que se esta consumiendo demasiada memoria, que se requiere realizar una limpieza. En este caso solo se muestra el mensaje, pero con este demonio trabajando se podría hacer que este mismo fuerce el cierre del programa para liberar el consumo excesivo o realice alguna función de respaldo de emergencia en prevención de un posible error.

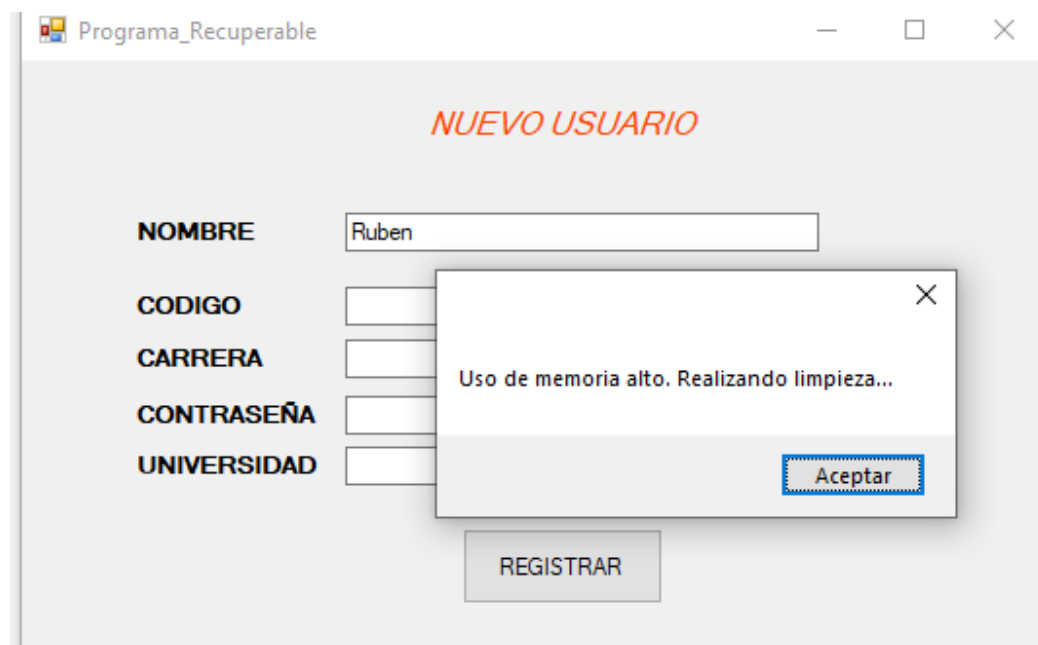
Ejecución del programa:

Programa recién iniciado:






The screenshot shows a window titled "Programa Recuperable" with standard Windows window controls (minimize, maximize, close). The main content area has a light gray background and the title "NUEVO USUARIO" in red, italicized font. Below the title, there are five input fields with labels to their left: "NOMBRE", "CODIGO", "CARRERA", "CONTRASEÑA", and "UNIVERSIDAD". Each field is empty. At the bottom center, there is a button labeled "REGISTRAR" with a blue border.

Iniciando llenado (se llega al límite de memoria usable):



This screenshot shows the same "Programa Recuperable" window, but now the "NOMBRE" field contains the text "Ruben". A modal dialog box is overlaid on top of the form. The dialog box has a close button (X) in the top right corner. The text inside the dialog box reads "Uso de memoria alto. Realizando limpieza...". At the bottom right of the dialog box is a button labeled "Aceptar". The "REGISTRAR" button from the background form is still visible but partially obscured by the dialog box.

Archivo guardado:

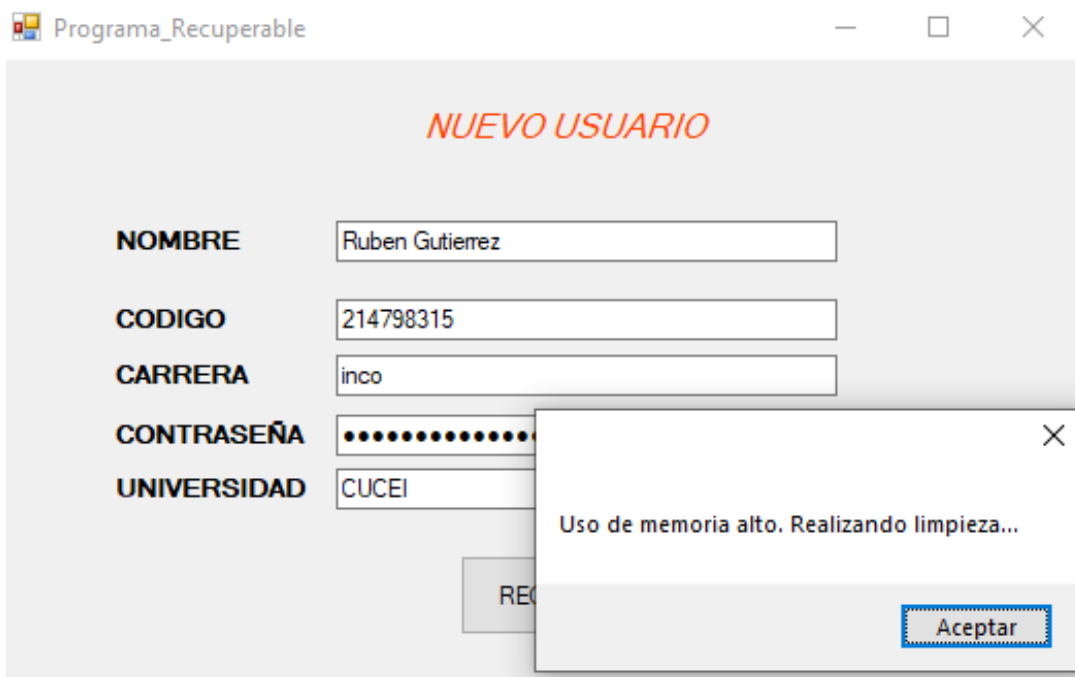
Nombre	Fecha de modificación	Tipo	Tamaño
 Programa_Recuperable.exe	09/09/2023 06:00 p. m.	Aplicación	12 KB
 Programa_Recuperable.pdb	09/09/2023 06:00 p. m.	Program Debug D...	24 KB
 respaldo.txt	09/09/2023 06:03 p. m.	Documento de te...	1 KB

 respaldo.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Ruben

Se alcanzo a llenar todo el formulario después de varios intentos, pero igual se llegó al límite de memoria:



Programa_Recuperable

NUEVO USUARIO

NOMBRE Ruben Gutierrez

CODIGO 214798315

CARRERA inco

CONTRASEÑA

UNIVERSIDAD CUCEI

Uso de memoria alto. Realizando limpieza...

Aceptar

Conclusión:

Los demonios son programas que se ejecutan en segundo plano estos se encargan de realizar algunas tareas específicas ya sea para un sistema operativo o algún programa. Estos no son controlados por el usuario y trabajan sin la necesidad de ser inicializados por el mismo, son muy útiles ya que pueden ser implementados para realizar monitoreos del sistema o realizar acciones tan simples como sincronizar la hora.

En cambio, un hilo es una unidad en ejecución dentro de un proceso que comparte el mismo espacio de memoria y recursos, estos son utilizados para generar concurrencia dentro de un sistema

permitiendo la realización de varias tareas, mejoran el rendimiento y la capacidad de respuesta del programa.

Bibliografía:

- "Fundamentals of Dependable Computing for Software Engineers" de John Knight.
- "Software Fault Tolerance Techniques and Implementation" de Laura L. Pullum.
- "Fault-Tolerant Systems" de Israel Koren y C. Mani Krishna