برنامه‌نویسی سمت سرویس‌دهنده

در

ویندوز

تألیف: مهندس محمد گلشاهی

(عضو هیأت علمی دانشگاه آزاد اسلامی، واحد دزفول)

انتشارات پندار پارس

سرشناسه	:	گلشاهی، محمد، ۱۳۵۸ -
عنوان و نام پدیدآور	:	برنامه‌نویسی سمت سرویس‌دهنده در ویندوز/ تالیف محمد گلشاهی.
مشخصات نشر	:	تهران : پندار پارس، ۱۴۰۱.
مشخصات ظاهری	:	۱۷۹ ص.: مصور، جدول، نمودار.
شابک	:	978-622-7785-09-8
وضعیت فهرست نویسی	:	فیپا
موضوع	:	ویندوز مایکروسافت، سرور Microsoft windows sever
موضوع	:	کامپیوترهای سرویس‌دهنده و سرویس گیرنده Client/server computing سیستم‌های عامل)کامپیوتر(Operating systems (Computers)
رده بندی کنگره	:	QA۷۶/۷۶
رده بندی دیویی	:	۴۴۷۶/۰۰۵
شماره کتابشناسی ملی	:	۸۸۲۶۲۸۰
اطلاعات رکورد کتابشناسی	:	فیپا

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**

تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸همراه: ۰۹۱۲۲۴۵۲۳۴۸ **info@pendarepars.com**

نام کتاب	:	**برنامه‌نویسی سمت سرویس‌دهنده در ویندوز**
ناشر	:	**انتشارات پندار پارس**
تالیف	:	**مهندس محمد گلشاهی**
چاپ ۱ دیجیتال	:	**اردیبهشت ۱۴۰۱**
شمارگان	:	**۱۰۰ نسخه**
طرح جلد	:	**رامین شکرالهی**
چاپ، صحافی	:	**روز**
قیمت	:	**۹۵۰۰۰ تومان**

این کتاب با کاغذ یارانه‌ای تخصیص یافته از سوی وزارت ارشاد چاپ شده است.

شابک : ۸-۰۹-۷۷۸۵-۶۲۲-۹۷۸

مقدمه

برای نوشتن برنامه‌هایی مقیاس‌پذیر و با کارایی بالا که به کاربران زیادی خدمات می‌دهند، نیاز است که با ساختار سرویس‌ها، نحوهٔ مدیریت آنها توسط سیستم‌عامل و برنامه‌نویسی پیشرفتهٔ آنها آشنا شوید. در صورتی که قصد دارید برنامه‌های حرفه‌ای بنویسید که از مدل سرویس-دهنده/مشتری (Client/Server) بهره می‌برند و به طور همزمان تعداد زیادی کاربر را پشتیبانی - می‌کنند، مطالب این کتاب می‌تواند برای شما بسیار مفید باشد.

در این کتاب مفاهیم اصلی سرویس‌ها، نحوهٔ اشکال‌زدایی آنها، روش مدیریت آنها توسط کاربر و سیستم‌عامل و برنامه‌نویسی آنها با استفاده از زبان قدرمند Microsoft Visual C++ .Net بیان شده است. با توجه با اینکه مفاهیم پایهٔ سرویس‌ها و ساختار آنها به طور کامل در این کتاب تشریح شده است، توسعه‌دهندگانی که از زبان‌های برنامه‌نویسی دیگری استفاده می‌کنند، نیز می‌توانند از مفاهیم ارائه شده در این کتاب استفاده نمایند.

شاکله کتاب حاضر برگرفته از کتاب‌ها و منابع معتبر و مستندات شبکه توسعه مایکروسافت (MSDN) است که با تجربیات اینجانب در امر برنامه‌نویسی و سیستم‌عامل آمیخته شده است.

درپایان لازم می‌دانم از مدیر انتشارات پندار پارس و سایر پرسنل انتشارات که زحمت چاپ کتاب را متقبل شده‌اند، صمیمانه تشکر نمایم.

پیشاپیش تمام کاستی‌های آن را می‌پذیرم و هرگونه انتقاد و راهنمایی را از طریق آدرس golshahi@iaud.ac.ir به دیدهٔ منت پذیرا هستم.

محمد گلشاهی

اردیبهشت ۱۴۰۱

تقدیم به همسر مهربانم

فهرست

فصل اول

مفاهیم پایه

در این فصل تعاریف و مفاهیم اولیه سیستم‌عامل ویندوز معرفی می‌شوند. فصل را با تعریف واژه‌هایی مانند **برنامه** (Application)، **پروسس** (Process)، **ریسمان** (Thread)، **وظیفه** (Task)، **کار** (Job)، **چندوظیفه‌ای** (MultiTasking) و **چندریسمانی** (MultiThreading) آغاز می‌کنیم.

در ادامه فصل به بررسی نسخه‌های مختلف ویندوز و تکامل واسط برنامه‌نویسی کاربردی آن می‌پردازیم. سپس، تفاوت بین مد هسته (Kernel mode) و مد کاربر (User mode)، و همچنین نسخه‌های سرور (Server) و کلاینت (Client) ویندوز بیان می‌شود. پس از آن ساختار سیستم‌های چندپردازنده‌ای مورد بررسی قرار می‌گیرد و واژه‌هایی مانند **پردازنده منطقی**، **پردازنده فیزیکی**، **گره** NUMA ، **وابستگی** (Affinity) ، **گروه پردازنده‌ها** و **ابرریسمانی** (Hyper-Threading) تعریف می‌شوند.

در پایان، نحوهٔ اشکال‌زدایی در مد کاربر و مد هسته تشریح شده و ابزارهای اشکال‌زدایی معرفی می‌گردند. همچنین ابزارها و مستندات مورد نیاز کار با سیستم ارائه می‌شوند.

۱-۱ برنامه، پروسس، ریسمان، وظیفه و کار

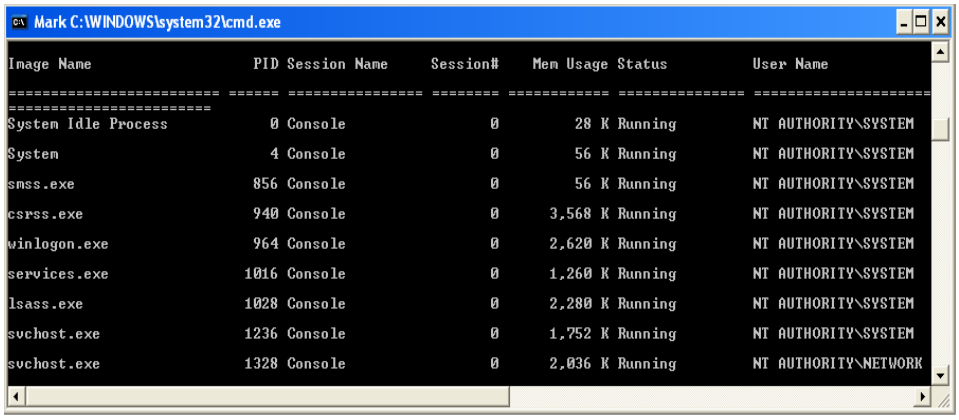
برنامه موجودیتی غیرفعال در حافظهٔ غیرفرار (دیسک سخت، لوح‌های فشرده و غیره) است که شامل یک یا چند فایل است و حداقل یک فایل اجرایی با پسوند **exe.** دارد. به فایل اجرایی برنامه، **تصویر اجرایی** (Executable Image) نیز گفته می‌شود. در صورتی که برنامه توسط کاربر اجرا شود، نمونه‌ای از برنامه و فایل‌های مورد نیازش از حافظهٔ غیرفرار بارگیری شده و به حافظهٔ اصلی انتقال می‌یابند، در این حالت به نمونهٔ بار شدهٔ برنامه در حافظهٔ اصلی، **پروسس** گفته می‌شود. پروسس مانند ظرفی از منابع استفاده شدهٔ برنامه است، که شامل بخش‌های کد، داده و غیره می‌باشد. در حقیقت، این ظرف بخشی از فضای حافظه مجازی است که به طور انحصاری به پروسس تعلق دارد. به هر پروسس یک شناسهٔ منحصربفرد اختصاص داده می‌شود که **شناسه پروسس** (Process Identifier - PID) نام دارد و برای رجوع به یک پروسس از PID و نه نام آن استفاده می‌گردد.

اگر پروسسی در درون خود پروسس دیگری را ایجاد نماید، یک رابطهٔ والد/فرزندی بین آنها ایجاد می‌شود، به طوری که پروسس والد می‌تواند به اجرای همروند با پروسس فرزندش ادامه دهد و یا این که منتظر بماند تا کار پروسس فرزندش تمام شود و سپس اجرایش را از سر گیرد. ممکن است پروسس

والد زودتر از پروسس فرزند خاتمه یابد. در این حالت ممکن است پروسس فرزند به والدی رجوع کند که وجود ندارد.

تمرین: مشاهدهٔ لیست پروسس‌ها با استفاده از دستور خط فرمان tasklist

برای مشاهدهٔ لیست پروسس‌های در حال اجرای سیستم، روش‌های مختلفی وجود دارد. در ویندوز می‌توانید از دستور خط فرمان *tasklist* استفاده کنید. به خط فرمان بروید (cmd.exe را اجرا کنید) و دستور *tasklist /V* را وارد نمایید تا خروجی مشابه با شکل ۱-۱ ببینید.



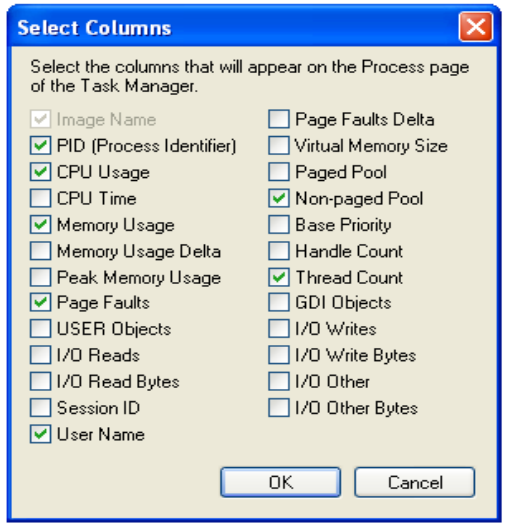
شکل ۱-۱. اجرای دستور tasklist /v در خط فرمان

تمرین: مشاهدهٔ اطلاعات پروسس با استفاده از Task Manager

مدیر وظیفه (Task Manager) یکی از ابزارهای توکار ویندوز است که لیست پروسس‌های موجود در سیستم را نشان می‌دهد. برای اجرای Task Manager چهار روش وجود دارد:

۱- فشردن کلیدهای Ctrl+Shift+Esc
۲- فشردن همزمان کلیدهای Ctrl+Alt+Delete و سپس کلیک بروی دکمهٔ Start Task Manager
۳- اجرای فایل Taskmgr.exe
۴- راست کلیک برروی taskbar و انتخاب آیتم Start Task Manager از منوی محتوایی.

بعد از اجرا Task Manager برای دیدن لیست پروسس‌ها باید زبانهٔ Processes را انتخاب کنید. برای مشاهدهٔ جزییات بیشتر دربارهٔ هر پروسس، از منوی View گزینهٔ Select Columns را انتخاب کنید (در ویندوز 8 باید در زبانهٔ Processes بر روی ستون Name راست کلیک کنید و یا به زبانهٔ Details بروید) و ستون‌های مورد نیازتان را مانند شکل ۱-۲ اضافه نماید.



شکل ۱-۲. اضافه کردن ستون‌های مورد نیاز به Task Manager

این ابزار نمایش داده نمی‌شوند. زبانهٔ Applications لیست پنجره‌های قابل رویت را در همهٔ دسک-تاپ‌های متصل به **ایستگاه پنجرهٔ** (WindowStation) محاوره‌ای نشان می‌دهد. (به طور پیش‌فرض، فقط یک دسک‌تاپ محاوره‌ای وجود دارد – برنامه‌ها می‌توانند با فراخوانی تابع **CreateDesktop** ، دسک‌تاپ‌های بیشتری ایجاد کنند.)

اگر برنامه‌ای بخواهد چند وظیفه را به طور همزمان انجام دهد، باید کد خود را به بخش‌های کوچک‌تری تقسیم نموده، و به هر بخش از آن سهمی از وقت پردازنده را اختصاص دهد. در ویندوز، این بخش‌های کد را **ریسمان** (Thread) می‌نامند. ریسمان کوچکترین واحد اجرایی است که سیستم می‌تواند پردازنده را به آن اختصاص دهد. در واقع، ریسمان تابعی است که توسط برنامه‌نویس در درون یک برنامه تعریف می‌شود. سپس، برنامه‌نویس با فراخوانی یکی از توابع API ، با نام **CreateThread** ، از سیستم عامل درخواست می‌کند تا بخشی از وقت پردازنده را به این تابع اختصاص دهد. در واقع پروسس‌ها اجرا نمی‌شوند؛ بلکه ریسمان‌های درون آنها اجرا می‌گردند. هر پروسس حداقل یک ریسمان دارد که ریسمان اصلی (اولیهٔ) آن است. در ریسمان اصلی، می‌توان ریسمان‌های جدیدی ایجاد نمود، و در درون ریسمان‌های جدیدتر نیز می‌توان ریسمان‌های دیگری تعریف نمود. شکل ۱-۳ پروسسی با دو ریسمان را نشان می‌دهد. همانطوری که در شکل می‌بینید هر ریسمان شامل یک پشته و یک **انبارهٔ محلی ریسمان** (Thread Local Storage - TLS) می‌باشد.

شکل ۱-۳. ساختار پروسس و ریسمان‌های آن

ریسمان‌ها توسط زمانبند سیستم و بر اساس یک الگوریتم زمانبندی (در ویندوز، الگوریتم **نوبت چرخشی اولویت‌دار** - Priority Round-Robin) انتخاب شده و سپس برای اجرا وارد پردازنده می‌شوند، ریسمانی که برای اجرا وارد پردازنده شود، **وظیفه** (Task) نامیده می‌شوند. ویندوز سیستم‌عاملی است که **چندبرنامه‌ای** (Multiprogramming) و **چندوظیفه‌ای** (Multitasking) را پشتیبانی می‌کند. در ویندوز، روش پیاده‌سازی چندوظیفه‌ای، **چندریسمانی** (Multithreading) می‌باشد.

در سیستم‌های عامل مبتنی بر یونیکس از **فیبر** (Fiber) برای پیاده‌سازی چندوظیفه‌ای استفاده می‌شود. فیبرها در مد کاربر تعریف می‌شوند و زمانبندی آنها توسط برنامهٔ ایجاد کننده‌شان، و نه سیستم‌عامل، صورت می‌گیرد. به منظور قابل‌حمل بودن برنامه‌های یونیکس به ویندوز، در ویندوز از فیبرها نیز پشتیبانی می‌شود، ولی قویاً پیشنهاد می‌شود که از ریسمان‌ها به جای فیبرها استفاده نمایید.

اغلب نیاز است که با گروهی از پروسس‌ها به صورت نهادی واحد رفتار شود. ویندوز، **کار** (Job) را ارائه کرد تا توسط آن بتوانید پروسس‌ها را در یک گروه قرار داده، و محدودیت‌های بر روی آن پروسس‌ها اعمال نمایید. بهترین فکری که می‌توانید در مورد یک شیء کار داشته باشید این است که: یک شیء کار ظرفی از پروسس‌ها است که می‌توانید محدودیت‌های مختلفی بر روی آن اعمال نمایید.

در فصل‌های مختلف این کتاب، پروسس‌ها، ریسمان‌ها و کارها به طور کامل تشریح می‌شوند.

۱-۲ تکامل ویندوز و واسط برنامه‌نویسی کاربردی (Windows API) آن

واسط برنامه‌نویسی کاربردی ویندوز یک واسط برنامه‌نویسی سیستمی در مد کاربر می‌باشد که برای سیستم‌های عامل ویندوز طراحی شده است. نسخه‌های مختلف ویندوز از Windows API پشتیبانی می‌کنند. تا قبل از معرفی نسخه‌های ۶۴-بیتی ویندوز، واسط برنامه‌نویسی نسخه‌های ۳۲-بیتی، Win32 API نامیده می‌شد. پسوند 32 در Win32 به این علت بود که آن را از نسخهٔ ۱۶-بیتی API اولیه (واسط برنامه‌نویسی نسخه‌های ۱۶-بیتی ویندوز) متمایز سازد. در این کتاب، هر جا از واژهٔ Windows API استفاده شده است، هر دو نسخهٔ ۳۲-بیتی و ۶۴-بیتی مدنظر می‌باشد.

شاید فکر کنید که وجود نسخه‌های مختلف ویندوز گیج کننده است؛ اما از منظر برنامه‌نویسی، همهٔ آنها مشابه هم هستند، بخصوص این که همهٔ آنها زیرمجموعهٔ API تقریباً مشابهی را پشتیبانی می‌کنند. برنامه‌های توسعه داده شده برای یک سیستم می‌توانند به راحتی در سیستم‌های دیگر اجرا گردند.

در نسخه‌های جدید ویندوز، میکروسافت توابع جدید اندکی به Windows API اضافه نموده است. البته از ابتدا تا کنون، Windows API به طور قابل ملاحظه‌ای ثابت مانده است. زمینه‌های اصلی که در نسخه‌های جدید ویندوز تکامل پیدا کرده‌اند، عبارتند از:

- **مقیاس‌پذیری:** نسخه‌های جدید در محدودهٔ وسیعی از کامپیوترها اجرا می‌شوند: از سرورهای سازمانی با چندین پردازنده تا سیستم‌های ذخیره‌سازی حجیم.
- **کارایی:** نسخه‌های جدیدتر ویندوز در درون بهبود یافته و API های جدید کمی به آنها اضافه شده است که کارایی را بهبود می‌دهند.
- **یکپارچگی:** هر نسخهٔ جدید با تکنولوژی‌های اضافی، مانند چندرسانه‌ای، شبکه‌های بی‌سیم، سرویس‌های وب، Net. و قابلیت‌های اتصال و اجرا (plug-and-play) ارائه می‌گردد.
- **استفادهٔ آسان:** دسک‌تاپ گرافیکی ارتقاء یافته و استفادهٔ از آن در نسخه‌های جدیدتر آسان‌تر می‌شود.
- **API ارتقاء یافته:** با گذشت زمان Windows API ارتقاء یافته است.

API ویندوز شامل هزاران تابع قابل فراخوانی است، که به دسته‌های اصلی زیر تقسیم می‌شوند:

- **سرویس پایه**
- **سرویس مؤلفه**
- **سرویس‌های واسط کاربر**
- **سرویس‌های چندرسانه‌ای و گرافیکی**
- **همکاری و پیام**
- **شبکه‌بندی**
- **سرویس‌های وب**

تمرکز این کتاب بر روی سرویس‌های پایه و کلیدی مانند پروسس‌ها، ریسمان‌ها و کارها می‌باشد.

تاریخچهٔ Win32 API

جالب است بدانید قرار نبوده Win32 به عنوان واسط برنامه‌نویسی اصلی سیستم‌عاملی باشد که بعدها Windows NT نامیده شد. در ابتدا، میکروسافت پروژهٔ Windows NT را به عنوان جایگزین نسخهٔ ۲ سیستم‌عامل OS/2 آغاز کرد. واسط برنامه‌نویسی اصلی ویندوز NT ، واسط ۳۲-بیتی سیستم‌عامل OS/2 با نام **Presentation Manager API** بود. بعد از گذشت یک سال از این پروژه، ویندوز 3.0 میکروسافت با موفقیت سهم عمده‌ای از بازار را در اختیار خود قرار داد. در نتیجه، میکروسافت مسیرش را تغییر داد و ویندوز NT را به جای جایگزینی OS/2 ، جایگزین خانوادهٔ ویندوز نمود. در این جا بود که میکروسافت می‌بایست یک API برای آن تعیین می‌کرد. تا قبل از این، در ویندوز 3.0 واسط برنامه‌نویسی کاربردی فقط به صورت واسطی ۱۶-بیتی بود. اگرچه API جدید ویندوز توابعی را معرفی می‌کند که در ویندوز ۳.۱ نبودند، اما میکروسافت تصمیم گرفت که API جدید را طوری بسازد که با نسخهٔ ۱۶-بیتی آن سازگار باشد. به همین دلیل، نام توابع، معانی و انواع داده را سازگار با نسخهٔ ۱۶-بیتی طراحی نمود تا برنامه‌های ۱۶-بیتی موجود بتوانند بر روی ویندوز NT حمل شوند.

تکنولوژی .NET

چارچوب کاری .NET. میکروسافت یک مدل برنامه‌نویسی جامع و مستحکم است که شامل کتابخانه‌ای از کلاس‌ها با نام FCL و **زمان اجرای زبان عمومی** (Common Language Runtime - CLR) می‌باشد. چارچوب کاری .NET. مجموعهٔ انبوهی از کلاس‌ها، ساختارها ، تایپ‌های شمارشی و غیره است که به صورت مجموعه‌ای از نامکده‌ها سازماندهی شده و بر اساس زمان اجرای زبان عمومی (CLR) پایه‌گذاری می‌شوند.

کتابخانهٔ کلاس پایهٔ .NET ، کلاس‌هایی را برای پیاده‌سازی واسط کاربری، دسترسی به داده‌ها، اتصال به پایگاه داده، رمزنگاری، توسعهٔ برنامه‌های تحت وب، الگوریتم‌های عددی و ارتباطات تحت شبکه ارائه می‌دهد. در جدول زیر نسخه‌های مختلف ترخیص شدهٔ .NET. به همراه ابزارهای توسعه و تاریخ انتشار آنها آورده شده است:

نسل	تاریخ ترخیص	ابزار توسعه	توزیع شده با ویندوز
1.0	2002-02-13	Visual Studio.Net	N/A
1.1	2003-04-24	Visual Studio 2003	Server 2003
2.0	2005-11-07	Visual Studio 2005	Server 2003 R/2
3.0	2006-11-06	Visual Studio 2005	Vista, Server 2008
3.5	2007-11-19	Visual Studio 2008	7 , Server 2008 R2
4.0	2010-04-12	Visual Studio 2010	N/A
4.5	2011-09-13	Visual Studio 2012	8 , Server 2012
4.5.1	2013-10-12	Visual Studio 2013	8.1 , Server 2012 R2
4.6	2015-07-20	Visual Studio 2015	10, Server 2016
4.7.2	2017-04-05	Visual Studio 2017	10,Server 2019

۱-۲-۱ نسخه‌های مختلف سیستم‌عامل ویندوز

تاکنون نسخه‌های مختلفی از ویندوز، شامل ویندوزهای کلاینت و ویندوزهای سرور توسط میکروسافت ارائه شده‌اند. سعی شده است تا نسخه‌های اخیر ویندوز، شامل ویندوز ویستا، 7 ، 8 ، 10 ، سرور 2019 ، سرور ۲۰۱۶، سرور 2012 R2 ، سرویس‌دهندهٔ 2008 R2 و نسخه‌های قدیمی‌تر، مانند ویندوز XP ، سرور 2000 و 2003 پوشش داده شوند. در زمان نوشتن این کتاب، نسخه‌های زیر در حال استفاده بودند:

- **ویندوز 10 :** جدیدترین ویندوز کلاینت می‌باشد که در جولای ۲۰۱۵ وارد بازار شد.
- **ویندوز 8.1 :** یکی از جدیدترین ویندوزها از خانوادهٔ سیستم عامل‌های کلاینت میکروسافت است که در اکتبر ۲۰۱۳ وارد بازار شد.
- **ویندوز 8 :** یکی از نسخه‌های جدید ویندوزهای کلاینت است که در اکتبر ۲۰۱۲ ارائه گردید.
- **ویندوز 7 :** در اکتبر ۲۰۰۹ ارائه شد.
- **ویندوز ویستا:** از سال ۲۰۰۷ ارائه شد و بر روی کامپیوترهای رومیزی، لپ‌تاپ‌ها و نوت بوکها نصب می‌شود.
- **ویندوز XP :** قبل از ویندوز ویستا ارائه شد و هنوز بسیار معروف و محبوب است.
- **ویندوز سرور 2019 :** جدیدترین ویندوز سرور میکروسافت می‌باشد که در نوامبر ۲۰۱۸ وارد بازار شده است.
- **ویندوز سرور 2016** : یکی از قدرتمندترین سیستم‌عامل‌های سرور میکروسافت است که در اکتبر ۲۰۱۶ وارد بازار گردید.
- **ویندوز سرور 2012 R2 :** از جمله سیستم‌عامل‌های سرور میکروسافت می‌باشد که در اکتبر ۲۰۱۳ ترخیص شد.
- **ویندوز سرور 2008 :** بر روی کامپیوترهای سرور نصب می‌شود. کامپیوترهایی که این ویندوز را اجرا می‌کنند از تکنیک‌های چندهسته‌ای با پردازنده‌های مستقل چندگانه بهره می‌گیرند.
- **ویندوز سرویس‌دهندهٔ 2003**: یکی از ویندوزهای سرویس‌دهندهٔ محبوب میکروسافت است که در سال ۲۰۰۳ عرضه شد.
- **ویندوز سرور 2000**: هنوز هم مورد استفاده قرار می‌گیرد. البته میکروسافت از سال ۲۰۱۰ به بعد از آن پشتیبانی نمی‌کند.
- **ویندوز CE :** نسخهٔ ویژه‌ای از ویندوز است که بر روی کامپیوترهای کوچک و قابل حمل

نصب می‌شود و اغلب زیرسیستم‌های ویندوز را ارائه می‌دهد.

۱-۲-۲ نسخه‌های قبلی منسوخ شدهٔ ویندوز

نسخه‌های قدیمی‌تر ویندوز به ندرت استفاده می‌شوند و عموماً پشتیبانی نمی‌شوند. اما در اینجا به طور خلاصه و از نقطه‌نظر تاریخی به آنها اشاره می‌کنیم. اغلب مثال‌های این کتاب ممکن است در این سیستم‌ها نیز کار کنند.

- **ویندوز NT 3.1, 3.5, 3.51 , 4.0** : ویندوز **NT** در اصل برای سرورها و کابران حرفه‌ای طراحی شد. در حالی که ویندوز **9x** برای مصارف شخصی و اداری طراحی شده بود. ویندوز 2000 جایگزین ویندوز NT شد. هرچند، هستهٔ NT هستهٔ پایهٔ نسخه‌های اخیر ویندوز نیز هست، اما با این وجود واژهٔ "Windows NT" دیگر منسوخ شده است.
- **ویندوز 95 ، 98 و ME (در مجموع ویندوز 9x)**: این سیستم‌ها در گذشته سیستم عامل کامپیوترهای رومیزی و لپ‌تاپ‌ها بودند. اما در نهایت ویندوز XP جایگزین این نسخه‌های ویندوز شد.

قبل از این که ویندوز 95 معرفی شود، ویندوز 3.1 سیستم عاملی ۱۶-بیتی بود که بر روی کامپیوترهای شخصی نصب می‌شد. این ویندوز دارای یک **GUI (واسط گرافیکی کاربر** - Graphical User Interface) ساده بود. API ارائه شده توسط این ویندوز، بیشتر ویژگی‌های اصلی سیستم‌عامل، مانند چندوظیفه‌ای، مدیریت حافظه با فضای آدرس بزرگ، و امنیت را پشتیبانی نمی‌کرد.

اگر به عقب‌تر برگردیم، در اوایل دههٔ ۱۹۸۰، سیستم‌عامل DOS برروی کامپیوترهای "IBM PC" نصب می‌شد. DOS سیستم‌عاملی ۱۶-بیتی بود که فقط یک واسط خط فرمان ساده داشت و دارای واسط گرافیکی کاربر نبود. در نسخه‌های مختلف ویندوز، پوستهٔ فرمان هنوز دستورات DOS را پشتیبانی می‌کند. جدول ۱-۱ نسخه‌های مختلف ویندوز، شمارهٔ نسخهٔ داخلی و زمان انتشار آنها را نشان می‌دهد.

نکته به دلیل این که ویندوز 7 و سرور 2008 R2 کد پایهٔ یکسانی دارند، و به طور مشابه، ویندوز 8 و سرور 2012 نیز کد پایهٔ یکسانی دارند، نسخهٔ داخلی و نسخهٔ ساخت آنها با هم برابر است.

۱-۲-۳ ویندوز NT5 ، NT6 و NT10

ویندوزهای سرور 2000 ، XP و سرور 2003 از نسخهٔ ۵ هستهٔ ویندوز NT استفاده می‌کنند. البته نسخهٔ جزئی (minor) آنها (5.x که x نسخهٔ جزئی را مشخص می‌کند) تفاوت دارد. برای مثال، ویندوز XP از هستهٔ NT با نسخهٔ ۵.۱.۲۶۰۰ استفاده می‌کند (۲۶۰۰ نسخهٔ ساخت آن است). از اینرو ویژگی‌های API آنها به نسخهٔ هسته وابسته می‌باشد. می‌توان برای راحتی کار از واژهٔ "NT5" برای رجوع به این

سه نسخه از ویندوز استفاده نمود.

ویندوز 7 و سرور 2008 R2 از هستهٔ نسخهٔ ۶.۱ استفاده می‌کنند، ولی نسخه‌های دیگر سرور 2008 و ویندوز ویستا از هستهٔ نسخهٔ ۶.۰ استفاده می‌کند. ویندوز 8 و سرور 2012 از نسخهٔ ۶.۲ هسته استفاده می‌کنند و ویندوز 8.1 و سرور 2012 R2 نسخهٔ ۶.۳ را اجرا می‌کنند. می‌توان از واژه "NT6" برای مشخص کردن این نسخه از ویندوزها استفاده نمود. ویندوز 10 ، سرور 2016 و سرور 2019 نیز از نسخهٔ 10.0 هسته استفاده می‌کنند.

بعضی از ویژگی‌های مهم ویندوز فقط در هستهٔ NT6 و NT10 وجود دارد. در مستندات MSDN برای هر یک از API های ویندوز، حداقل نسخه‌ای از ویندوز که آن API را پشتیبانی می‌کند، تعیین شده است. در هنگام استفاده از توابع API ، می‌توانید با بررسی این مستندات متوجه شوید که آیا ویندوز مورد نظر شما آن تابع API را پشتیبانی می‌کند یا نه.

جدول ۱-۱. نسخه‌های مختلف سیستم عامل ویندوز

نام محصول	شمارهٔ نسخه	تاریخ انتشار
Windows NT 3.1	3.1	July 1993
Windows NT 3.5	3.5	September 1994
Windows NT 3.51	3.51	May 1995
Windows NT 4.0	4.0	July 1996
Windows 2000 (NT 5.0)	5.0	December 1999
Windows XP	5.1	August 2001
Windows Server 2003	5.2	March 2003
Windows Vista	6.0 (Build 6000)	January 2007
Windows Server 2008	6.0 (Build 6001)	March 2008
Windows 7	6.1 (Build 7600)	October 2009
Windows Server 2008 R2	6.1 (Build 7600)	October 2009
Windows 8	6.2	October 2012
Windows Server 2012	6.2	October 2012
Windows 8.1	6.3*	October 2013
Windows Server 2012 R2	6.3*	October 2013
Windows 10	10.0	2015-2018
Windows Server 2016	10.0	October 2016
Windows Server 2019	10.0	Novamber 2018

۱-۲-۴ تفاوت بین نسخه‌های سرور و کلاینت ویندوز

ویندوز به صورت بسته‌های سرور و کلاینت وارد بازار می‌گردد. برای مثال، شش نسخهٔ کلاینت از ویندوز 7 در بازار موجود است:

- Home Basic
- Home Premium
- Professional
- Ultimate
- Enterprise
- Starter

و هفت نسخهٔ مختلف نیز از ویندوز سرور 2008 R2 وجود دارد:

- Windows Server 2008 R2 Foundation
- Windows Server 2008 R2 Standard
- Windows Server 2008 R2 Enterprise
- Windows Server 2008 R2 Datacenter
- Windows Web Server 2008 R2
- Windows HPC Server 2008 R2
- Windows Server 2008 R2 for Itanium-Based Systems (که آخرین نسخهٔ ویندوز برای پشتیبانی از پردازندهٔ ایتانیوم IA-64 شرکت Intel می‌باشد).

برخی از نسخه‌های ویندوز، نرم‌افزار Windows Media Player را به همراه ندارند. ویرایش‌های Standard ، Enterprise و Datacenter سرور 2008 دارای تکنولوژی "Hyper-V" می‌باشند. موارد زیر تفاوت بین نسخه‌های مختلف را مشخص می‌کند:

- تعداد پردازنده‌هایی که پشتیبانی می‌کنند (برحسب سوکت و نه هسته یا ریسمان).
- میزان حافظهٔ فیزیکی (RAM) که پشتیبانی می‌کنند.
- تعداد اتصالات شبکه‌ای که به طور همزمان پشتیبانی می‌کنند (برای مثال: در نسخه‌های کلاینت ویندوز اجازهٔ حداکثر ۱۰ اتصال همزمان به سرویس‌های فایل و چاپ داده می‌شود.)
- پشتیانی از Media Center
- پشتیبانی از Multi-touch ، Aero و Desktop Compositing
- پشتیبانی از ویژگی‌هایی مانند Windows XP Compatibility Mode ، BitLocker ، VHD Booting ، APPLocker و بیش از ۱۰۰ ویژگی قابل پیکربندی.
- سرویس‌های لایه‌ای که فقط در نسخه‌های سرور وجود دارد (برای مثال، سرویس فهرست

[directory] و سرویس خوشه‌بندی [clustering]).

جدول ۱-۲ تفاوت بین نسخه‌های مختلف ویندوز 7 و سرور 2008 R2 را از لحاظ پشتیبانی از پردازنده و حافظه نشان می‌دهد. برای دیدن تفاوت‌های بیشتر بین این دو ویندوز، صفحهٔ وبی با آدرس *http://www.microsoft.com/7windowsserver2008/en/us/r2-compare-specs.aspx* را مشاهده نمایید.

جدول ۱-۲. تفاوت بین ویندوز 7 و سرور 2008 R2

نسخهٔ ویندوز	تعداد سوکت‌های پشتیبانی شده در ویرایش ۳۲-بیتی	میزان حافظهٔ فیزیکی قابل پشتیبانی در ویرایش ۳۲-بیتی	تعداد سوکت‌های پشتیبانی شده در ویرایش ۶۴-بیتی	میزان حافظهٔ فیزیکی قابل پشتیبانی در ویرایش ایتانیوم	میزان حافظهٔ فیزیکی قابل پشتیبانی در ویرایش x64
Windows 7 Starter	۱	۲GB	در دسترس نیست	در دسترس نیست	۲GB
Windows 7 Home Basic	۱	۴GB	۱	در دسترس نیست	۸GB
Windows 7 Home Premium	۱	۴GB	۱	در دسترس نیست	۱۶GB
Windows 7 Professional	۲	۴GB	۲	در دسترس نیست	۱۹۲GB
Windows 7 Enterprise	۲	۴GB	۲	در دسترس نیست	۱۹۲GB
Windows 7 Ultimate	۲	۴GB	۲	در دسترس نیست	۱۹۲GB
Windows Server 2008 R2 Foundation	در دسترس نیست	در دسترس نیست	۱	در دسترس نیست	۸GB
Windows Web Server 2008 R2	در دسترس نیست	در دسترس نیست	۴	در دسترس نیست	۳۲GB
Windows Server 2008 R2 Standard	در دسترس نیست	در دسترس نیست	۴	در دسترس نیست	۳۲GB
Windows HPC Server 2008 R2	در دسترس نیست	در دسترس نیست	۴	در دسترس نیست	۱۲۸GB
Windows Server 2008 R2 Enterprise	در دسترس نیست	در دسترس نیست	۸	در دسترس نیست	۲۰۴۸GB
Windows Server 2008 R2 Datacenter	در دسترس نیست	در دسترس نیست	۶۴	در دسترس نیست	۲۰۴۸GB

Windows Server 2008 R2 for Itanium-Based Systems	در دسترس نیست	در دسترس نیست	۶۴	۲۰۴۸GB	در دسترس نیست

با وجود این که ویرایش‌های مختلفی از سیستم‌های عامل سرور و کلاینت ویندوز وجود دارد، اما همهٔ آنها از مجموعهٔ مشترکی از فایل‌های هسته، شامل تصویر اجرایی هسته (Ntoskenl.exe و در نسخهٔ PAE : Ntkrnlpa.exe) ، کتابخانهٔ HAL ، درایورهای ابزار (Device drivers) ، برنامه‌های سودمند اصلی و کتابخانه‌های پیوند پویا (DLL ها) استفاده می‌کنند.

سئوالی که پیش می‌آید این است: با وجود ویرایش‌های مختلف ویندوز که تصویر اجرایی یکسانی دارند، سیستم چگونه تشخیص می‌دهد ماشین با کدام ویرایش سیستم عامل راه‌اندازی شده است؟ سیستم با بررسی مقادیر ProductType و ProductSuite در کلید رجیستری HKLM\SYSTEM\CurrentControlSet\ Control\ProductOptions نوع سیستم‌عامل را مشخص می‌کند. مقدار ProductType ، سرور یا کلاینت بودن سیستم را مشخص می‌کند. مقادیر معتبر این فیلد در جدول ۱-۳ لیست شده‌اند. می‌توانید با فراخوانی تابع GetVersionEx این مقدار را در مد کاربر پرس‌وجو نمایید. در مد هسته، درایورهای ابزار می‌توانند با فراخوانی تابع RtlGetVersion این مقدار را بدست آورند.

جدول ۱-۳. مقادیر ProductType در رجیستری

ویرایش ویندوز	مقدار ProductType
ویندوز کلاینت	WinNT
ویندوز سرور – کنترلر حوزه (domain controller)	LanmanNT
ویندوز سرور – فقط سرور (server only)	ServerNT

مقدار دیگری با نام ProductPolicy در رجیستری وجود دارد که کپی همین مقدار در فایل tokens.dat است و تفاوت بین ویرایش‌های مختلف ویندوز و ویژگی‌های فعال آنها را مشخص می‌کند.

سئوال دیگری که ممکن است پیش آید، این است: با وجود این که فایل‌های هسته در نسخه‌های سرور و کلاینت یکسان هستند، چگونه سیستم عامل عملکرد متفاوتی را پیاده‌سازی می‌کند؟ سیستم‌های سرور طوری بهینه‌سازی شده‌اند که اجرای برنامه‌های سرویس‌دهنده (مانند IIS ، SQL Server و دیگر سرویس‌ها) در آنها بالاترین کارایی را داشته باشد، در حالی که نسخهٔ کلاینت (با وجود اینکه قابلیت‌های سرور را دارد) زمان پاسخ را در محاورهٔ بین کاربر و دسک‌تاپ بهینه‌سازی می‌کند. برای مثال، با توجه به مقدار ProductType ، می‌توان چند سیاست تخصیص منابع مختلف را در زمان

راه‌اندازی سیستم اتخاذ نمود: اندازه و تعداد کپه‌های سیستم‌عامل (حوض‌ها - Pools) ، تعداد ریسمان‌های کارگر درون سیستم و اندازهٔ حافظهٔ پنهان (cache). علاوه بر این، تعیین برخی سیاست‌ها در زمان اجرای سیستم، مانند روشی که مدیر حافظه درخواست‌های حافظه‌ای سیستم و پروسس‌ها را موازنه می‌کند، ویرایش‌های کلاینت و سرور ویندوز را از هم متمایز می‌سازد.

۱-۳ مد هسته در مقابل مد کاربر

به منظور جلوگیری از دسترسی و تغییر منابع بحرانی سیستم توسط برنامه‌های کاربر، ویندوز فقط از دو مد دسترسی پردازنده (حتی اگر پردازنده بیشتر از دو مد را پشتیبانی کند) استفاده می‌نماید: **مد کاربر** و **مد هسته**. برنامه‌های کاربردی در مد کاربر اجرا می‌شوند، در حالی که کد سیستم‌عامل (مانند سرویس‌ها و درایورهای ابزار) در مد هسته اجرا می‌گردند. مد هسته به حالتی از اجرا در پردازنده گفته می‌شود که درآن دسترسی به تمام حافظهٔ سیستم و دستورالعمل‌های CPU امکان‌پذیر است. پردازنده با اجرای سیستم عامل در سطح امتیاز بالاتری از برنامه‌های کاربر، چارچوب مورد نیاز را برای طراحان سیستم‌عامل فراهم می‌سازد تا بتوانند از دسترسی بدافزارها به منابع سیستمی، و در نتیجه مختل شدن سیستم جلوگیری کنند.

نکته معماری پردازنده‌های x86 و x64 چهار سطح امتیاز (حلقه) را برای محافظت از کد و دادهٔ سیستم تعریف می‌کنند که از بازنویسی سهوی یا عمدی کدی با امتیاز کمتر جلوگیری می‌کند. ویندوز از سطح امتیاز 0 (حلقهٔ 0) برای مد هسته و سطح امتیاز ۳ (حلقهٔ ۳) برای مد کاربر استفاده می‌نماید. دلیل این که ویندوز فقط از دو سطح استفاده می‌کند این است: بعضی از معماری‌های سخت‌افزاری که در گذشته توسط ویندوز پشتیبانی می‌شدند (مانند Compaq Alpha و Silicon Graphics MIPS) فقط دو سطح امتیاز را پیاده‌سازی می‌کردند.

با توجه به این که در ویندوز هر پروسس فضای آدرس مجازی خودش را دارد، کد هستهٔ سیستم‌عامل و درایورهای ابزار فضای آدرس مجازی یکسانی را به اشتراک می‌گذارند. هر صفحه در حافظهٔ مجازی برچسب‌گذاری می‌شود تا مشخص گردد که کدام مد دسترسی پردازنده می‌تواند از آن بخواند و یا در آن بنویسید. صفحات فضای آدرس سیستم فقط در مد هسته قابل دسترسی هستند، در حالی که همهٔ صفحات فضای آدرس کاربر در مد کاربر قابل دسترسی می‌باشند. صفحات فقط-خواندنی (مانند صفحاتی که داده‌های ایستا را نگهداری می‌کنند) در هیچ مدی قابل نوشتن نیستند. بعلاوه، در پردازنده‌هایی که بتوانند حافظه را به صورت غیر اجرایی (no-execute) محافظت کنند، ویندوز، صفحات داده را به صورت غیرقابل اجرا (non-executable) برچسب‌گذاری می‌کند تا از اجرای کد مخرب در ناحیهٔ دادهٔ پروسس جلوگیری شود.

ویندوزهای ۳۲-بیتی هیچگونه مکانیزم حفاظتی برای جلوگیری از خواندن و نوشتن حافظهٔ سیستم،

توسط مؤلفه‌های در حال اجرا در مد هسته، ارائه نمی‌دهند. به عبارت دیگر، کد سیستم‌عامل و درایورها دسترسی کاملی به فضای حافظهٔ سیستم دارند، و می‌تواند مکانیزم امنیتی ویندوز را برای دسترسی به اشیاء دور بزنند. به دلیل این که بیشتر کد ویندوز در مد هسته اجرا می‌شود، بسیار حیاتی است که مؤلفه‌هایی که در مد هسته اجرا می‌شوند، به دقت طراحی و آزمایش شوند، تا تضمین شود که امنیت سیستم را مختل نکرده، و باعث ناپایداری سیستم نمی‌شوند.

عدم محافظت از صفحات حافظه در مد هسته، ما را نیازمند دقت بیشتری در بارگذاری درایورهای ابزار فروشندگان ثالث می‌سازد. چون درایورهای ابزار پس از نصب در مد هسته اجرا شده و دسترسی کاملی به همهٔ منابع سیستمی دارند. این نقص یکی از دلایل ایجاد مکانیزم **امضای درایور** (driver-signing) در ویندوز بود. اگر کاربری بخواهد یک درایور اتصال و اجرای بدون امضاء را به سیستم اضافه کند، مکانیزم امضای درایور به کاربر هشدار می‌دهد (و حتی می‌تواند از نصب آن ممانعت به عمل آورد). مکانیزم دیگری نیز وجود دارد که **بازبین درایور** (Driver Verifier) نامیده می‌شود و به نویسندگان درایورهای ابزار کمک می‌کند تا اشکال‌هایی (مانند سرریز بافر و یا نشت حافظه) که باعث مشکلات امنیتی یا کاهش قابلیت اطمینان می‌شوند را پیدا کنند.

در نسخه‌های ۶۴-بیتی، سیاست **امضای کد مد هسته** (Kernel Mode Code Signing - KMCS) طوری طراحی شده است تا درایورهای ابزار ۶۴-بیتی (نه فقط درایورهای اتصال و اجرا) را مجبور کند تا با کلید رمزی که توسط یکی از مراجع بزرگ مسئول تصدیق کد واگذار می‌شود، امضا شوند. مدیران سیستم (Administrators) نیز نمی‌توانند صریحاً سیستم را مجبور به نصب درایور بدون امضا کنند. البته می‌توان این رفتار سیستم را در زمان راه‌اندازی تغییر داد. با فشردن کلید F8 در هنگام راه‌اندازی سیستم، و انتخاب گزینهٔ Disable Driver Signature Enforcement (غیرفعال سازی اجرای امضای درایور) از منوی ظاهر شده می‌توان این ویژگی را غیرفعال ساخت. غیرفعال کردن این ویژگی باعث می‌شود که در دسک‌تاپ ویندوز پیامی ظاهر شده که نشان می‌دهد ویژگی «مدیریت حقوق دیجیتال» (Digital Rights Management - DRM) غیرفعال شده است.

برنامه‌های کاربردی در زمان فراخوانی توابع سیستمی از مد کاربر به مد هسته سوئیچ می‌کنند. به عنوان مثال، تابع **ReadFile** در نهایت نیاز به فراخوانی روتین‌های داخلی ویندوز دارد که خواندن داده را از فایل مدیریت می‌کنند. به دلیل این که روتین‌های داخلی ویندوز به ساختمان داده‌های سیستم دسترسی پیدا می‌کنند، باید در مد هسته اجرا شوند. گذار از مد کاربر به مد هسته توسط یکی از دستورالعمل‌های ویژهٔ پردازنده (CPU) انجام می‌شود. این گذار باعث می‌شود که پردازنده به مد هسته سوئیچ کند و کد گسیل سرویس سیستم را در هسته وارد نموده و تابع داخلی مناسبی را در Ntoskrnl.exe یا Win32k.sys فراخوانی کند. قبل از این که کنترل به ریسمان کاربر برگردد، پردازنده به مد کاربر برمی‌گردد. در راه بازگشت به مد کاربر، سیستم‌عامل از خودش محافظت می‌کند.

نکته گذار از مد کاربر به مد هسته (و عکس آن) تأثیری بر زمانبندی ریسمان ندارد؛ زیرا گذار حالت، تعویض متن نیست. گذار از مد کاربر به مد هسته هزینه دارد و باید تا حد امکان از آن پرهیز کرد.

پس با این وجود، ریسمان‌های کاربر بخشی از زمان اجرای خود را در مد کاربر و بخشی از آن را در مد هسته صرف می‌کنند. در برنامه‌های گرافیکی، به دلیل این که حجم انبوهی از عملیات گرافیکی و ترسیم پنجره در مد هسته انجام می‌شود، برنام بیشتر وقت خود را در مد هسته صرف می‌نماید. راه ساده برای آزمایش این موضوع اجرای یک برنامهٔ گرفیکی مانند Microsoft Paint یا Microsoft Chess Titans و مشاهدهٔ زمان صرف شده در مد هسته و کاربر با استفاده از شمارنده‌های کارایی (Performance Counters) لیست شده در جدول ۱-۴ است. اغلب برنامه‌های پیشرفته می‌توانند از تکنولوژی‌های جدیدتری مانند Direct2D استفاده نمایند که حجم عظیم محاسبات را در مد کاربر انجام می‌دهد و فقط دادهٔ خام سطح (surface) را به هسته ارسال می‌کند. با انجام این کار، زمان صرف شده برای گذار از مدهای کاربر به هسته و بالعکس کاهش می‌یابد.

جدول ۱-۴. شمارنده‌های کارایی مرتبط با مد هسته یا کاربر

نام شمارنده	عملکرد
Processor: % Privileged Time	درصد زمانی را مشخص می‌کند که یک CPU (یا همهٔ CPU ها) در دورهٔ زمانی تعیین شده در مد هسته در حال اجرا هستند.
Processor: % User Time	درصد زمانی را مشخص می‌کند که یک CPU (یا همهٔ CPU ها) در دورهٔ زمانی تعیین شده در مد کاربر در حال اجرا هستند.
Process: % Privileged Time	درصد زمانی را مشخص می‌کند که ریسمان‌های یک پروسس در دورهٔ زمانی تعیین شده در مد هسته در حال اجرا هستند.
Process: % User Time	درصد زمانی را مشخص می‌کند که ریسمان‌های یک پروسس در دورهٔ زمانی تعیین شده در مد کاربر در حال اجرا هستند.
Thread: % Privileged Time	درصد زمانی را مشخص می‌کند که یک ریسمان در دورهٔ زمانی تعیین شده در مد هسته در حال اجرا باشد.
Thread: % User Time	درصد زمانی را مشخص می‌کند که یک ریسمان در دورهٔ زمانی تعیین شده در مد کاربر در حال اجرا باشد.

تمرین: مد هسته در مقابل مد کاربر

می‌توانید با استفاده از ابزار Performance Monitor ویندوز، مدت زمانی که سیستم در مد هسته و مد کاربر صرف نموده است را مشاهده کنید. برای انجام این کار مراحل زیر را دنبال کنید:

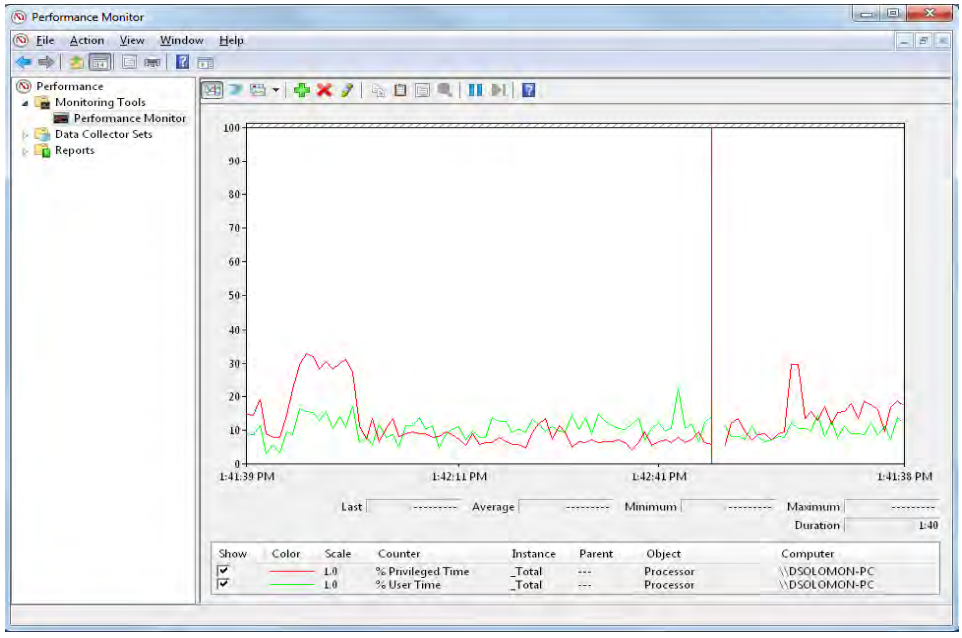
۱- ابزار Performance Monitor را از طریق آیتم Administrative Tools کنترل پانل اجرا کنید. گره Performance | Monitoring Tools را در پانل سمت چپ این برنامه انتخاب نمایید.

۲- در نوار ابزار، روی آیکن + (Add) کلیک نمایید.

۳- از پنجرهٔ Add Counters گره Processor را باز کرده و گزینهٔ Privileged Time % را انتخاب نمایید، سپس کلید Ctrl را نگه داشته و بر روی گزینهٔ User Time % کلیک کنید تا انتخاب شود.

۴- دکمهٔ Add را کلیک کرده و سپس OK را کلیک نمایید.

۵- یک پنجرهٔ خط فرمان باز کنید (cmd.exe را اجرا نمایید) و دستور جستجوی درایو C بر روی شبکه را به صورت dir \\%ComputerName%\c$ /s در آن وارد نمایید.



همچنین، در برخی از ویرایش‌های ویندوز می‌توان با استفاده از Task Manager زمان صرف شده در مد هسته را مشاهده کرد. با کلیک بر روی زبانهٔ Performance در ابزار Task Manager ، و سپس انتخاب آیتم Show Kernel Times از منوی View مدت زمان صرف شده در مد هسته نمایش داده می شود.

در شکل بالا، میلهٔ CPU Usage مقدار نهایی استفاده از CPU را به رنگ سبز نشان می‌دهد و میلهٔ قرمز رنگ بیانگر زمان مصرفی در مد هسته است.

۱-۴ سرویس‌ها، توابع و روتین‌ها

چندین واژه در بین کاربران ویندوز و مستندات برنامه‌نویسی، در زمینه‌های مختلف، معانی متفاوتی دارد. برای مثال، کلمهٔ سرویس را می‌توان به یک روتین قابل فراخوانی در سیستم‌عامل، یک درایور ابزار و یا یک پروسس سرور ارجاع داد. لیست زیر معانی دقیق این کلمات را در این کتاب تشریح می‌کند:

- **توابع API ویندوز:** مستندات این توابع ویندوز موجود است. این توابع، زیرروال‌های قابل فراخوانی واسط برنامه‌نویسی کاربردی ویندوز می‌باشند. برای مثال، توابع CreateProcess ، CreateFile .
- **سرویس‌های محلی سیستم (یا فراخوانی‌های سیستمی):** مستند نشده‌اند، سرویس‌های بنیادی (اصلی) سیستم‌عامل هستند که از مد کاربر قابل فراخوانی می‌باشند. به عنوان مثال، NtCreateUserProcess یکی از سرویس‌های داخلی سیستمی است که تابع CreateProcess در درون خود، آن را برای ایجاد پروسس جدید فراخوانی می‌کند.
- **توابع پشتیبانی هسته (یا روتین‌ها):** زیرروال‌هایی در درون سیستم‌عامل ویندوز هستند که فقط در مد هسته قابل فراخوانی می‌باشند. برای مثال، ExAllocatePoolWithTag روالی است که درایورهای ابزار آن را برای تخصیص حافظه از کپهٔ سیستم ویندوز (که حوض نامیده می‌شود) فراخوانی می‌کنند.
- **سرویس‌های ویندوز:** پروسس‌های هستند که توسط مدیر کنترل سرویس (SCM) ویندوز اجرا می‌شوند. به عنوان مثال، سرویس Plug and Play ویندوز سرویسی است که سیستم را قادر به تشخیص و وفق با تغییرات سخت‌افزاری، بدون نیاز به گرفتن ورودی از کاربر می‌سازد. (توجه: البته، رجیستری درایورهای ابزار را به صورت سرویس تعریف می‌کند، اما در این کتاب کلمهٔ سرویس به آنها رجوع نمی‌کند.)
- **کتابخانهٔ پیوند پویا** (Dynamic Link Library - DLL)**:** مجموعه‌ای از زیرروال‌های لینک شده به یکدیگر در یک فایل باینری می‌باشند که توسط برنامه‌های کاربردی مورد استفاده قرار می‌گیرند. برای مثال، کتابخانهٔ Msvcrt.dll (کتابخانهٔ زمان اجرای C) و Kernel32.dll (یکی از کتابخانه‌های زیرسیستم ویندوز) نمونه‌ای از این کتابخانه‌ها می‌باشند. مؤلفه‌های مد کاربر ویندوز و برنامه‌های کاربردی به طور گسترده از DLL ها استفاده می‌کنند. امتیازی که DLL ها نسبت به کتابخانه‌های پیوند ایستا (فایل‌هایی با پسوند lib.) دارند آن است که برنامه‌ها

می‌توانند DLL ها را به اشتراک بگذارند، و ویندوز تضمین می‌کند که فقط یک کپی از آنها در حافظه موجود باشد. ملاحظه کنید اسمبلی‌های غیراجرایی Net. به صورت DLL کامپایل می‌شوند؛ اما قسمت صدور زیرروال ندارند و به جای آن، CLR متاداده (metadata) را برای دستیابی به انواع و اعضای متناظر تجزیه و تحلیل می‌کند.

۱-۵ نشست‌های چندگانه و Terminal Services

Terminal Services سرویسی است که ویندوز برای نشست‌های مختلف محاوره‌ای کاربران در ماشین ارائه می‌دهد. با استفاده از Terminal Services ویندوز، کاربران دور می‌تواند نشست‌هایی را در ماشینی دیگر برپا کنند، به آن سیستم وارد شده و برنامه‌هایی را در آن سرور اجرا نمایند. ماشین سرور، واسط گرافیکی کاربر (و منابع قابل پیکربندی همچون صدا و کلیپ‌بورد) را به کلاینت انتقال می‌دهد و کلاینت نیز ورودی‌ها را از کاربر گرفته و آنها را به سرور انتقال می‌دهد.

اولین نشست، به عنوان نشست سرویس‌ها در نظر گرفته شده، که به آن نشست 0 نیز گفته می‌شود و شامل پروسس‌های پیاده‌سازی کنندهٔ سرویس‌های سیستم می‌باشد. اولین نشستی که کاربر از طریق کنسول فیزیکی به سیستم وارد می‌شود، نشست ۱ است و نشست‌های دیگری که از طریق برنامهٔ Remote Desktop (Mstsc.exe) یا سوئیچ سریع کاربر (Fast user switching) ایجاد می‌شوند، به ترتیب نشست‌های ۲ ، ۳ و ... می‌باشند.

نسخه‌های مختلف ویندوز، بجز سرورها (نسخه‌های کلاینت: مانند ویندوز 7)، فقط اجازهٔ اتصال یک ماشین راه دور را می‌دهند، و در صورتی که کاربری از طریق کنسول به سیستم وارد شده باشد، ایستگاه کاری آن قفل می‌گردد (زیرا نمی‌توان در یک زمان هم به صورت محلی و هم از راه دور به یک سیستم کلاینت وارد شد). ویرایش‌های از ویندوز که شامل Windows Media Center باشند اجازهٔ یک نشست محاوره‌ای و حداکثر چهار نشست Windows Media Center Extender را می‌دهند.

سرورهای ویندوز از دو اتصال دور به طور همزمان پشتیبانی می‌کنند (برای ساده‌سازی مدیریت دور سیستم - برای مثال، استفاده از ابزارهای مدیریتی که نیازمند ورود به ماشین هستند.) و اگر سروری به صورت Terminal Server پیکربندی شده باشد، بیش از دو نشست دور را پشتیبانی می‌کند.

نسخه‌های کلاینت سیستم‌عامل ویندوز، نشست‌های چندگانهٔ همزمان را پشتیبانی می‌کنند. این پشتیبانی از طریق ویژگی سوئیچ سریع کاربر ارائه می‌شود. زمانی که کاربری بخواهد به جای خروج از سیستم (log off)، نشست خود را قطع کند (برای مثال، با کلیک بر روی Start و انتخاب Switch User از زیرمنوی Shutdown یا با پایین نگهداشتن کلید Windows و فشار کلید L و کلیک بر روی Switch User)، نشست جاری (پروسس‌های در حال اجرا در آن نشست و همهٔ ساختمان داده‌های که اطلاعات نشست را ذخیره می‌کنند) فعال می‌ماند و ویندوز به صفحهٔ ورود به سیستم برمی‌گردد. اگر کاربر