ویندوز اینترنالز

«بخش ۲: مفاهیم پایه و ابزارها»

تاریخ تالیف: جمعه – ۱۸ مهر ۱۳۹۹

تهیه شده توسط تیم فنی آزمایشگاه امنیت کی‌پاد

مفاهیم و ابزارها

در قسمت دوم سلسله مقالات ویندوز اینترنالز، تشریح مفاهیم کلیدی و اصطلاحات پایه سامانه‌عامل ویندوز را ادامه خواهیم داد، از قبیل جاب‌ها، تِرَدها و پروسه‌ها[1]، ساختار درختی پروسه‌ها، تردهای زمانبندی شده در مُد کاربر و فیبرها[2]، و دیگر مفاهیم و همچنین ابزارهای کاربردی ویندوزی را مورد بررسی قرار خواهیم داد.

در قسمت‌های بعدی این سری از مقالات ویندوز اینترنالز، به مفاهیم دیگر مانند رجیستری، حافظه مجازی، دیباگر Windbg، دیباگ کرنل ویندوز و ... خواهیم پرداخت تا مرحله به مرحله با معماری ویندوز و ابزارهای مطرح آن آشنا شویم.

کلیدواژه:

معماری ویندوز، مفاهیم ویندوز، ویندوز اینترنالز

[1] Jobs, Threads and Process
[2] Fibers and User-Mode Scheduler Threads

جاب‌ها، تردها و پروسه‌ها

گرچه برنامه‌ها[3] و پروسه‌ها در یک نگاه سطحی با یکدیگر مشابه به نظر می‌رسند، اما آن‌ها اساساً از یکدیگر متفاوت هستند. یک برنامه، یک دنباله ثابت از دستورالعمل‌های اجرایی است، در حالیکه یک پروسه یک کانتینر[4] برای مجموعه‌ای از منابع[5] آماری و اطلاعاتی استفاده شده توسط برنامه در حال اجرا به شمار می‌رود. به هر صورت، در بالاترین سطح انتزاع، یک پروسه ویندوز شامل موارد زیر می‌شود:

۱. یک فضای آدرس مجازی خصوصی[6] که مجموعه‌ای از آدرس‌های حافظه مجازی است که پروسه می‌تواند از آن‌ها استفاده کند.
۲. یک برنامه اجرایی[7] که کد و دیتا اولیه را تعریف می‌کند که درون فضای آدرس مجازی پروسه مَپ شده است.
۳. یک لیست از هندل‌های باز[8] به منابع سیستمی مختلف از قبیل سمافورها[9]، پورت‌های ارتباطی[10] و فایل‌ها که برای تمامی تِرَدهای پروسه قابل دسترس هستند.
۴. یک کانتَکست امنیتی[11] که یک توکن دسترسی[12] است که کاربر، گروه امنیتی، سطوح دسترسی[13]، حالت مجازی کنترل حساب کاربر[14]، نشست[15] و حالت محدود حساب کاربری[16] در ارتباط با پروسه را شناسایی می‌کند.
۵. یک شناسه پروسه برنامه که Process ID خوانده می‌شود.

[3] Programs
[4] Container
[5] Set of Recources
[6] Private virtual address space
[7] An executable program
[8] Open Handles
[9] Semaphores
[10] Communication Ports
[11] A security context
[12] Access Token
[13] Privileges
[14] User Account Control (UAC) Virtualization State
[15] Session
[16] Limited User Account State

۶. در پایان هر پروسه حداقل شامل یک تِرَد اجرایی است. اگرچه وجود یک پروسه خالی ممکن است، اما کاربردی ندارد.

هر پروسه همچنین به پَرِنت (والد) یا پروسه خالق خودش اشاره می‌کند. اگر والد آن وجود نداشته باشد، اطلاعات مواردی که در بالا ذکر شد، هیچگاه به‌روزرسانی نخواهند شد. همچنین ممکن است یک پروسه به یک والد اشاره کند که اصلا وجود ندارد.

البته این یک مسئله نمی‌باشد، زیرا هیچ‌چیز متکی به این اطلاعات نیست. همچنین شایان ذکر است، در نرم‌افزار Process Explorer، زمان شروع پروسه‌های والد به منظور جلوگیری از پیوست یک پروسه فرزند[1] به شناسه یک پروسه مجدداً استفاده شده[2] محاسبه می‌شود. مثال زیر این نوع رفتار را نشان می‌دهد

تمرین: مشاهده ساختار درختی پروسه‌ها

یک خاصیت منحربفرد درباره یک پروسه که بیشتر ابزارها آن را نشان نمی‌دهند، شناسه والد پروسه است. با این حال، شما می‌توانید با استفاده از Performance Monitor (یا از راه برنامه‌نویسی) با پرس‌وجوی شناسه پروسه والد این اطلاعات را بدست آورید.

```
C:\Users\Cage\Desktop\SysinternalsSuite>pslist.exe /t

pslist v1.3 - Sysinternals PsList
Copyright (C) 2000-2012 Mark Russinovich
Sysinternals - www.sysinternals.com

Process information for WIN-4COSVQ7OEGM:

Name                Pid Pri Thd  Hnd      VM      WS    Priv
Idle                  0   0   4    0       0      24       0
  System              4   8 106  506    2708    1048      44
    smss            296  11   2   32    3044     848     308
csrss               376  13   9  652   37448    3952    1712
  conhost          1868   8   2   53   21752    2540     664
  conhost          3996   8   2   31   21048    2424     596
  conhost          4036   8   1   31   17304    2244     544
csrss               468  13  10  197   48876   12096   16784
  conhost          3804   8   2   35   33448    2820     716
wininit             476  13   3   81   34160    3440     984
  services          572   9  18  273   29816    7560    4656
    VGAuthService   504   8   3   86   53756    9284    5084
    svchost         688   8  11  369   36200    7592    3316
      mobsync      1308   8   8  151   62464    5780    1872
      WmiPrvSE     2128   8  11  311   50820   13232    7736
    ruby            704   8  11  152  292216  191928  203064
```

تصویر ۱: ابزار PsList

[1] Child
[2] Reused

در لیست بالا هر یک از پروسه‌ها دندانه‌گذاری شده است تا ارتباط خودش با والد/فرزند مشخص باشد. شایان ذکر است، پروسه‌هایی که والدشان وجود ندارد، به سمت چپ دندانه‌گذاری شده‌اند (مانند پروسه wininit) چون حتی اگر یک والد بزرگ۳ برای آن‌ها وجود داشته باشد، هیچ راهی به منظور شناسایی ارتباط بین آن‌ها وجود ندارد.

همچنین قابل ذکر است، ویندوز فقط شناسه پروسه خالق را حفظ می‌کند و دیگری ارتباطی بین آن دو وجود ندارد. همچنین به منظور نمایش این واقعیت که ویندوز مسیر بیش از یک شناسه والد را نگه نمی‌دارد، می‌توانید گام‌های آورده شده در قسمت زیر را دنبال کنید:

۱. یک پنجره Command Prompt باز کنید.
۲. در آن دستور title Parent را وارد کنید و بر روی دکمه Enter بفشارید.

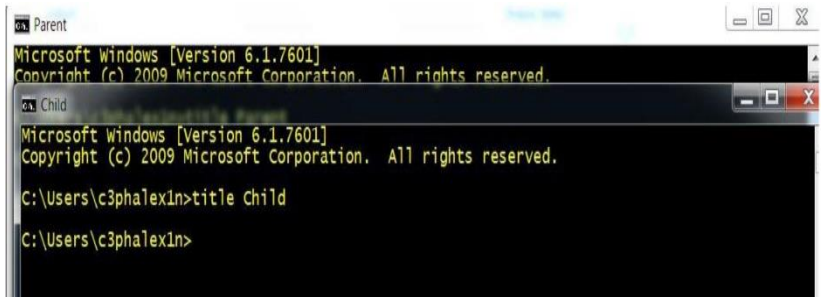
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\c3phalex1n>title Parent

C:\Users\c3phalex1n>
```

تصویر ۲: محیط CMD

۳. سپس دستور start cmd را وارد کنید و بر روی Enter بفشارید.
۴. در پنجره Command Prompt جدید دستور title Child را وارد کرده و بر روی دکمه Enter بفشارید.



تصویر ۳: محیط CMD فرزند

[3] Grandparent

۵. مدیر تَسک‌های سامانه‌عامل ویندوز یا Task Manager را باز کنید.
۶. سپس در پنجره Command Prompt دوم دستور mspaint را وارد کنید و دکمه Enter را بفشارید.
۷. بعد از اینکه نرم‌افزار Paint باز شد، به محیط Command Prompt باز گردید و دستور exit را وارد کنید. (توجه کنید، بعد از اجرای دستور exit برنامه Paint هنوز در حال اجرا باقی می‌ماند).
۸. به محیط Task Manager وارد شوید و بر روی تب Applications کلیک کنید.
۹. بر روی تَسک Parent کلیک راست کرده و گزینه Go To Process را انتخاب کنید.
۱۰. بر روی پروسه cmd.exe کلیک راست کنید و سپس End Process Tree را انتخاب کنید.

پس از انجام این عملیات، اولین پنجره Command Prompt هم ناپدید خواهد شد، اما با این حال، شما هنوز نرم‌افزار Paint را مشاهده خواهید کرد، زیرا آن فرزند بزرگ پروسه Command Prompt بوده است که آن را متوقف کردید و چون پروسه میانی (والد Paint) متوقف شده بود، هیچ پیوندی میان والد و فرزند بزرگ وجود ندارد.

ابزارهای زیادی به منظور مشاهده و تغییر پروسه‌ها و اطلاعات آن‌ها وجود دارد. آزمایشی که در ادامه این قسمت آورده شده است، نمایش‌های مختلفی از اطلاعات پروسه‌ها را که می‌توانید با این ابزارها به دست آورید، ارائه می‌دهد.

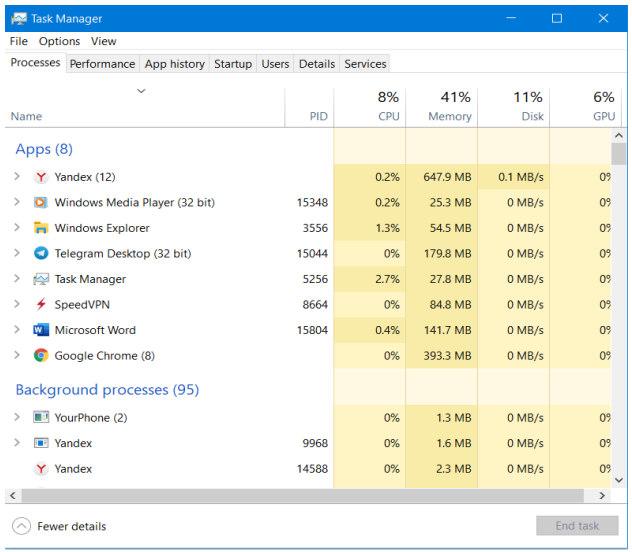
در حالیکه بیشتر این ابزارها درون خود سامانه‌عامل ویندوز و ابزارهای دیباگ آن و Windows SDK وجود دارند، مابقی آن‌ها ابزارهایی هستند که فقط در Sysinternals موجود می‌باشند. بیشتر این ابزارها زیر مجموعه‌های آورلَپ[1] شده از پروسه‌های کرنل و اطلاعات تِرَدها را که گاهی اوقات با نام‌های مختلف شناسایی می‌شوند، نشان می‌دهند.

[1] Overlap

احتمالا بیشترین ابزار مورد استفاده به منظور مشاهده فعالیت پروسه‌ها ابزار Task Manager سامانه‌عامل ویندوز است. (البته از آنجایی که چیزی به نام تَسک یا Task در کرنل ویندوز وجود ندارد، نام این ابزار کمی عجیب است.) آزمایشی که در ادامه این قسمت آورده شده است، تفاوت میان برنامه‌های کاربردی و پروسه‌های لیست شده در ابزار Task Manager را نمایش می‌دهد.

تمرین: مشاهده پروسه‌ها با Task Manager

ابزار Task Manager ویندوز لیست تمامی پروسه‌های در حال اجرا بر روی سامانه‌عامل ویندوز را به شما نمایش می‌دهد. شایان ذکر است، می‌توانید ابزار Task Manager ویندوز را به روش‌های گوناگون از قبیل فشردن کلیدهای ترکیبی Ctrl+Shift+Esc، کلیدهای ترکیبی Ctrl+Shift+Del و کلیک راست بر روی نوارابزار (Taskbar) و انتخاب گزینه Task Manager اجرا کنید.

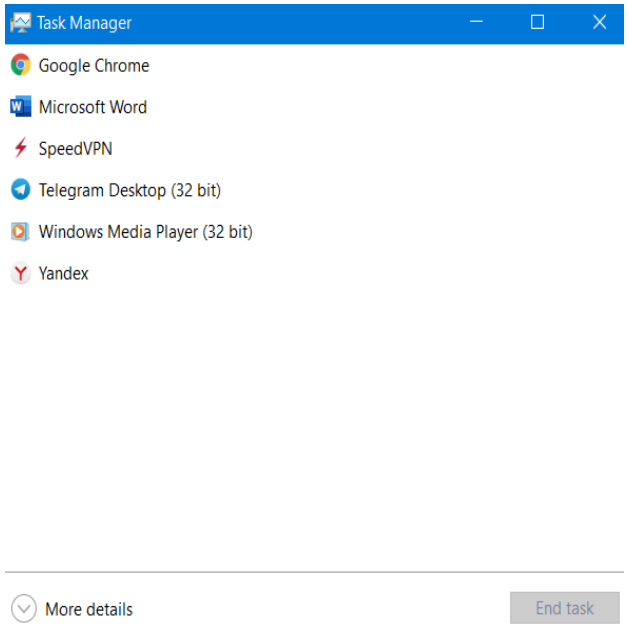


تصویر ٤: محیط Task Manager در حالت نمایش جزئیات

هنگامیکه Task Manager اجرا شد بر روی تب Processes کلیک کنید تا لیست تمامی پروسه‌های در حال اجرا بر روی سامانه‌عامل ویندوز به شما نمایش داده شود. توجه کنید که پروسه‌ها در فیلد نام ایمج[1] که آن‌ها یک نمونه از آن هستند، شناخته می‌شوند.

شایان ذکر است، برخلاف اکثریت آبجکت‌ها در ویندوز، به پروسه نمی‌توان نام‌های گلوبال اختصاص داد. به منظور نمایش جزئیات بیشتر، درون تب Process ابزار Task Manager کلیک راست کنید و اطلاعاتی را که نیازمند هستید، برای نمایش انتخاب کنید.

گرچه تب Process ابزار Task Manager یک لیست از پروسه‌ها را نمایش می‌دهد، اما چیزی که Task Manager در حالت پیش‌فرض نشان می‌دهد، بالاترین پنجره‌های قابل روئیت بر روی تمامی دسکتاپ‌های ایستگاه تعاملی ویندوز[2] را که به آن‌ها متصل هستید، لیست می‌کند.

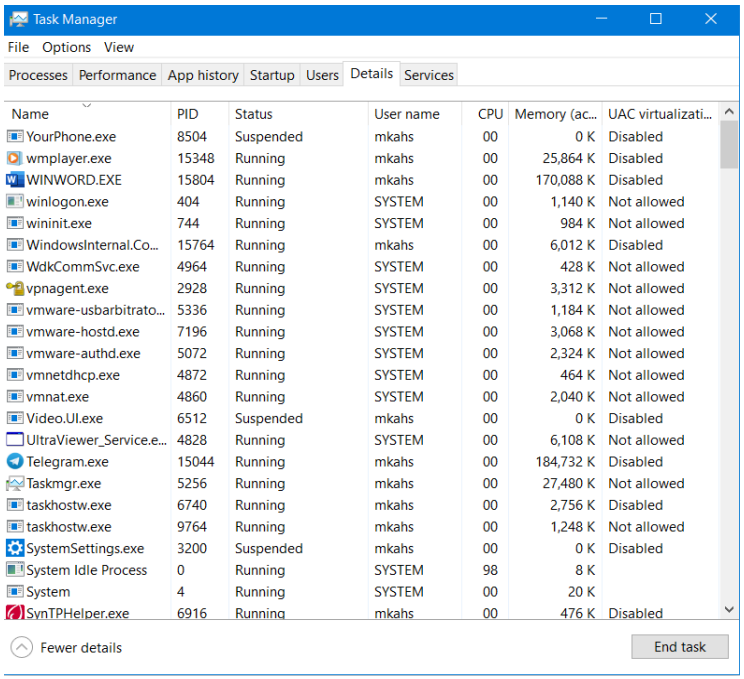


تصویر ۵: ابزار Task Manager در حالت پیش‌فرض

[1] Image Name

در حالت پیش فرض، فقط یک دسکتاپ تعاملی وجود دارد، البته یک برنامه کاربردی می‌تواند با فراخوانی تابع CreateDesktop دسکتاپ‌های بیشتری ایجاد کند که این کار توسط ابزار Sysinternals Desktop صورت می‌گیرد.

اگر در Task Manager گزینه Status در تب Process را فعال کنید، جدول Status در این تب مشخص می‌کند که تِرَد مالک پنجره برنامه‌ها در چه حالتی قرار دارند. Running بدین معنا است که تِرَد برنامه برای ورودی پنجره منتظر باقی مانده است، Not Responding بدین معنا است که تِرَد برای ورودی پنجره منتظر نیست (به عنوان مثال، تِرَد ممکن است در حال اجرا یا در حال انتظار برای ورودی و خروجی (I/O) یا چند آبجکت همگام‌سازی ویندوز باشد.).



تصویر ٦: نمایش جزئیات پروسه‌ها در Task Manager

[2] Interactive Window Station

شایان ذکر است، برای نمایش اطلاعات بیشتر درباره پروسه ها در Task Manager می‌توانید به تب Details بروید. در این تب اطلاعات تکمیلی هر پروسه به تفکیک نمایش داده خواهد شد. در تصویر ۶ جزئیات پروسه‌ها را مشاهده می‌کنید.

نرم‌افزار Process Explorer

نرم‌افزار Process Explorer از مجموعه نرم‌افزارهای Sysinternals جزئیات بیشتری از دیگر ابزارهای موجود درباره پروسه‌ها و تِرَدها نمایش می‌دهد. به همین دلیل در سراسر این سلسله مقالات از این نرم‌افزار به صورت گسترَده‌ای استفاده خواهیم کرد. ویژگی‌های آورده شده در قسمت زیر موارد منحصربفردی هستند که نرم‌افزار Process Explorer می‌تواند نمایش دهد:

۱. توکن امنیتی پروسه[1] (از قبیل لیست گروه‌ها و سطوح دسترسی و حالت مجازی‌سازی)
۲. برجسته‌سازی لیست پروسه و تِرَدهایی که تغییر کرده‌اند
۳. نمایش لیست سرویس‌های درون پروسه‌های میزبانی سرویس از جمله نمایش نام و توضیحات سرویس
۴. نمایش یک لیست از خصوصیت‌ها از قبیل میتیگشن‌ها، سطح محافظت از پروسه.پ
۵. جزئیات یک جاب و پروسه‌هایی که قسمتی از یک جاب هستند
۶. پروسه‌هایی که برنامه‌های دات‌نت و جزئیات مخصوص فریمورک دات‌نت را میزبانی می‌کنند (از قبیل لیست AppDomainها، اسمبلی‌های بارگزاری شده[2] و محاسبه‌کننده پرفُورمَنس CLR3)
۷. پروسه‌هایی که میزبان Windows Runtime هستند (پروسه‌هایی که با عنوان Immersive Process شناخته می‌شوند.).
۸. زمان شروع پروسه‌ها و تِرَدها
۹. لیست کامل فایل‌های مَپ شده در حافظه[4] (نه فقط Dllها.)
۱۰. توانایی ساسپند[5] کردن یک پروسه یا یک تِرَد
۱۱. توانایی از بین بردن یک تِرَد منفرد[6]
۱۲. شناسایی پروسه‌هایی که بیشترین میزان استفاده از CPU را در یک دوره زمانی داشته‌اند.

نرم‌افزار Process Explorer همچنین دسترسی ساده‌ای به اطلاعات آورده شده در زیر به صورت یک‌جا ارائه می‌دهد:

۱. ساختار درختی پروسه‌ها (به همراه توانایی از بین بردن قسمتی از ساختار درختی)
۲. هندل‌های باز در یک پروسه (از جمله هندل‌های بدون نام)
۳. لیست کتاب‌خانه‌های پیوندی پویا (و فایل‌های مَپ شده درون حافظه) در پروسه‌ها
۴. فعالیت تِرَد درون یک پروسه
۵. پشته‌های تِرَد مُد کرنل و مُد کاربر (از جمله آدرس‌های مپ شده در نام‌های استفاده شده توسط فایل DbgHelp.dll که با ابزارهای دیباگ بر روی سیستم نصب می‌شود.)
۶. جزئیات مدیر حافظه از قبیل Peak Commit Charge و Kernel Memory Paged و Nonpaged Pool Limits

به منظور یک آزمایش مقدماتی با استفاده از Process Explorer مثال آورده شده در قسمت زیر را دنبال کنید.

[1] Process Security Token
[2] Loaded Assemblies
[3] CLR Performance Counters

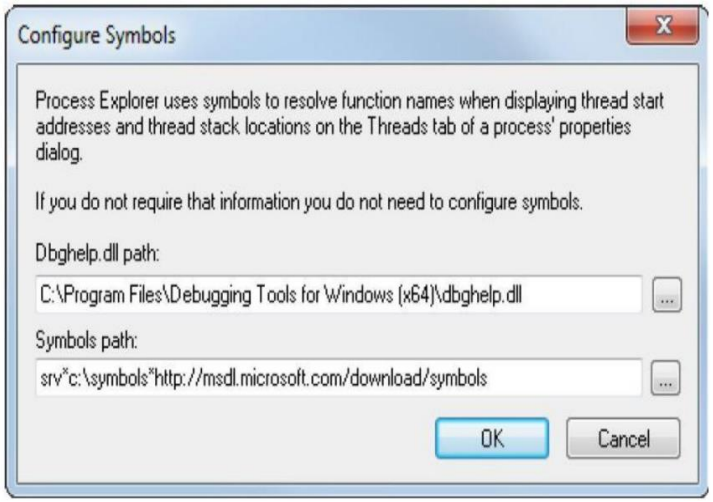
[4] Memory-Mapped Files
[5] Suspand
[6] Individual Thread

تمرین: مشاهده پروسه‌ها با Process Explorer

آخرین نسخه نرم‌افزار Process Explorer را از مجموعه نرم‌افزارهای Sysinternals دانلود و اجرا کنید. برای اولین بار که این برنامه را اجرا می‌کنید، یک پیام دریافت خواهید کرد که می‌گوید سیمبول‌ها[1] پیکربندی نشده‌اند.

اگر این سیمبول‌ها به درستی پیکربندی شوند، نرم‌افزار Process Explorer می‌تواند به اطلاعات سیمبول‌ها به منظور نمایش نام تِرَد شروع کننده تابع و توابع درون پشته فراخوانی تِرَد دسترسی بگیرد (این اطلاعات با کلیک بر روی نام پروسه‌ها در محیط Process Explorer قابل نمایش است).

این ویژگی به منظور کشف اینکه تِرَد درون پروسه چه کاری انجام می‌دهد، بسیار مفید است. به منظور دسترسی به سیمبول‌ها، باید ابزارهای دیباگ ویندوز را بر روی سامانه‌عامل خود نصب کرده باشید (در ادامه این فصل تشریح خواهد شد).



تصویر ۷: محیط پیکربندی سیمبول‌ها

[1] Symbols
[2] Internet Explorer Tab Processes

سپس بر روی منوی Options کلیک کنید و از میان گزینه‌های نمایش داده شده Configure Symbols را انتخاب کنید. زمانیکه Configure Symbols نمایش داده شد، مسیر Dbghelp.dll را برای فیلد Dbghelp.dll مشخص کنید و در قسمت Symbol path مسیر دسترسی به سیمبول‌های دیباگ ویندوز را وارد کنید. در مثال زیر موارد فوق نمایش داده شده است:

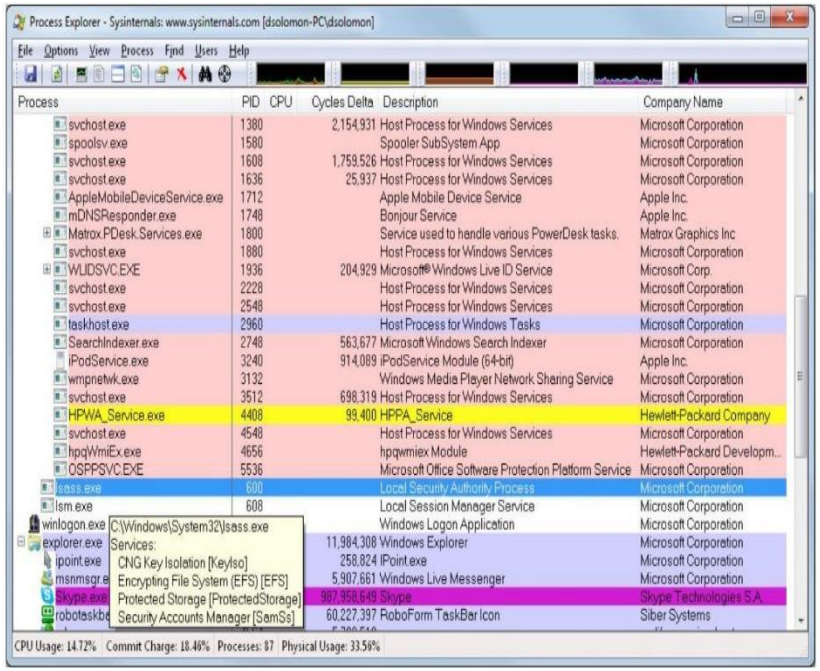
در مثال قبلی، سرور سیمبول‌های استفاده شده به منظور دسترسی بر روی ماشین محلی در پوشه c:\symbols ذخیره شده بود. برای اطلاعات بیشتر به منظور پیکربندی سرور سیمبول‌ها مقاله را مشاهده کنید.

هنگامیکه نرم‌افزار Process Explorer شروع به کار می‌کند، به صورت پیش‌فرض ساختار درختی تمامی پروسه‌ها را نمایش می‌دهد. این نرم‌افزار همچنین یک Lower Pane View دارد که در آن هندل‌های باز و کتاب‌خانه‌های پیوندی پویا و فایل‌های مَپ شده در حافظه نمایش داده می‌شود (این موارد در فصل‌های بعدی این سلسله مقالات مورد بررسی قرار خواهند گرفت).

این برنامه همچنین برای چندین نوع سرویس میزبانی که لیست آن‌ها در زیر آورده شده است، یک Tooltip ارائه می‌کند:

۱. سرویس‌های درون پروسه Svchost.exe، اگر مکانما را بر روی نام این پروسه ببرید در Tooltip سرویس‌های در حال اجرا در این پروسه را مشاهده خواهید کرد.
۲. وظایف آبجکت COM درون پروسه Taskeng.exe (شروع شده توسط Task Scheduler)
۳. آبجکت COM میزبانی شده درون پروسه Dllhost.exe
۴. پروسه تب‌های اینترنت اکسپلورر [2]
۵. پروسه میزبانی کنسول[3]

[3] Console Host Processes



تصویر ۸: محیط Process Explorer

در اینجا چندین گام به منظور مرور برخی از قابلیت‌های اصلی نرم‌افزار Process Explorer آورده شده است:

۱. توجه کنید که سرویس‌های میزبانی پروسه‌ها به صورت پیش‌فرض با رنگ صورتی برجسته می‌شوند و پروسه‌های شما با رنگ آبی برجسته می‌شوند. (البته این رنگ‌ها می‌توانند توسط شما تعویض گردند).

۲. مکانما را بر روی نام ایمج پروسه‌ها ببرید، متوجه خواهید شد که مسیر کامل پروسه‌ها بر روی دیسک سخت نمایش داده می‌شود. همانطور که پیش از این ذکر شد، برخی از انواع پروسه‌ها جزئیات بیشتری در Tooltip خود دارند.

۳. یک تِرَد موجودیتی درون یک پروسه است که زمانبند ویندوز آن را اجرا می‌کند. بدون آن، پروسه برنامه نمی‌تواند اجرا شود. یک تِرَد شامل مولفه‌های مورد نیاز زیر می‌شود:

۴. کانتَکست یک مجموعه از ثبات‌های پردازنده که حالت پردازنده را نمایش می‌دهند.

۵. دو پشته – یک پشته برای زمان اجرا در مُد کرنل و یک پشته دیگر زمانی که در مُد کاربر اجرا می‌شود.

۶. یک محیط ذخیره‌سازی خصوصی که انباره محلی تِرَد[1] خوانده می‌شود که برای زیرسامانه‌ها، کتاب‌خانه‌های زمان اجرا و کتاب‌خانه‌های پیوندی پویا مورد استفاده قرار می‌گیرد.

۷. یک شناسه منحصربفرد که Thread ID خوانده می‌شود (قسمتی از یک ساختمان داخلی[2] که Client ID خوانده می‌شود – شناسه پروسه‌ها و شناسه تِرَدها از فضای نام مشابه‌ای[3] ایجاد می‌شوند، بنابراین آن‌ها هیچگاه آورلَپ یا همپوشانی[4] نمی‌شوند.)

گاهی اوقات تِرَدها، کانتَکست یا توکن امنیتی خودشان را دارند که اغلب توسط برنامه‌های کاربردی سرور چندتِرَدی که کانتَکست امنیتی کلاینت‌ها را جعل هویت می‌کنند، ارائه می‌شوند. محیط انباره خصوصی[5]، پشته‌ها و ثبات‌های فرار[6] کانتَکست تِرَد خوانده می‌شوند.

زیرا این اطلاعات در هر یک از معماری‌ها که ویندوز بر روی آن‌ها اجرا می‌شود، متفاوت است، این ساختمان در صورت لزوم، خاص معماری[7] است. تابع GetThreadContext ویندوز دسترسی به این اطلاعات خاص معماری (بلاک کانتَکست خوانده می‌شود) را ارائه می‌دهد.

[1] Thread Local Storage
[2] Internal Structure
[3] Same Namespace
[4] Overlap

[5] Private Storage Area
[6] Volatile Registers
[7] Architecture-Specific

نکته: تردها یک برنامه ۳۲ بیتی در حال اجرا بر روی نسخه ٦٤ بیتی ویندوز شامل کانتکست ٦٤ بیتی و ۳۲ بیتی می‌شوند که مکانیزم **Wow64** از آنها به منظور تعویض حالت برنامه از حالت اجرای ۳۲ بیتی به حالت ٦٤ بیتی استفاده می‌کند. این تردها دو پشته و دو بلاک کانتکست دارا هستند. با این حال، تابع **Wow64GetThreadContext** کانتکست ۳۲ بیتی بازگشت خواهد داد. در قسمت‌های بعدی بیشتر درباره این مکانیزم صحبت خواهیم کرد.

فیبرها[1]

از آنجایکه تعویض اجرای یک ترد به ترد دیگر موجب درگیری کرنل می‌شود، این عملیات می‌تواند برای ویندوز هزینه‌بر باشد، مخصوصا اگر دو ترد به شکل متناوب با همدیگر تعویض شوند. ویندوز به منظور کاهش هزینه تعویض تردها با یکدیگر دو مکانیزم فیبرها و زمانبندی مُد کاربر را طراحی و عملیاتی کرده است.

فیبرها به یک برنامه کاربردی اجازه می‌دهند، اجرای ترد خودشان را زمانبندی کنند، به‌جای اینکه متکی بر مکانیزم زمانبندی سامانه‌عامل ویندوز باشند. همچنین فیبرها تردهای سبک وزن[2] خوانده می‌شوند و در اصلاح زمانبندی آنها برای کرنل نامرئی یا مخفی هستند زیرا آنها در مُد کاربر درون Kernel32.dll پیاده‌سازی شده‌اند.

به منظور استفاده از فیبرها، ابتدا باید تابع ConvertThreadtoFiber فراخوانی شود. این تابع یک ترد را به یک فیبر اجرایی تبدیل می‌کند. پس از آن، فیبری که به تازگی ایجاد شده است، می‌تواند با فراخوانی CreateFiber فیبرهای اضافی دیگری ایجاد کند. (هر فیبر می‌تواند مجموعه فیبرهای خودش را دارا باشد).

یک فیبر تا زمانیکه از طریق تابع SwitchToFiber به صورت دستی انتخاب نشود، برخلاف یک ترد اجرا نمی‌شود. فیبر جدید تا زمانیکه وجود داشته باشد، اجرا می‌شود یا تا زمانی‌که SwitchToFiber مجدد فراخوانی شود. برای اطلاعات بیشتر در این مورد مستندات Windows SDK را مشاهده کنید.

نکته: استفاده از فیبرها همواره یک ایده خوب نیست. به این دلیل که فیبرها برای کرنل مخفی هستند. همچنین فیبرها در اشتراک فضای محلی تردها (**TLS**) مشکل دارند، زیرا چندین فیبر می‌توانند در یک ترد اجرا شوند. اگرچه فیبرها دارای فضای ذخیره‌سازی محلی (**FLS**) برای هستند، اما این مورد نمی‌تواند تمامی مسائل مرتبط با اشترات فضای آدرس را در فیبرها حل کند و در غیر این صورت فیبرهای **I/O** با پرفورمنس بسیار پایین کار خواهند کرد. علاوه بر این‌ها، فیبرها نمی‌توانند به صورت موازی بر روی یک پردازنده اجرا شوند. در بسیاری از سناریوهای برنامه‌نویسی ویندوز، بهتر است اجازه دهیم خود ویندوز مسئله زمانبندی تردها را مدیریت کند تا اینکه ما خود انجام دهیم و روند اجرایی برنامه‌ها را خراب کنیم.

تردهای زمانبندی شده در مد کاربر

تردها زمانبندی شده در مُد کاربر که فقط بر روی نسخه‌های ۶۴ بیتی سامانه‌عامل ویندوز موجود هستند، ویژگی‌های مشابه فیبرها را بدون معایب آن‌ها ارائه می‌دهند. تردها زمانبندی شده در مُد کاربر، یک ترد مشابه در کرنل سامانه‌عامل دارا هستند و همچنین برای کرنل سامانه‌عامل مخفی نمی‌باشند. این موضوع اجازه می‌دهد، تردها زمانبندی شده در مُد کاربر یا به اختصار تردهای UMS فراخوانی‌های سیستمی، اشتراک‌گذاری منابع و... را انجام دهند.

به هر حال، تا زمانیکه دو یا تعداد بیشتری از تردهای UMS فقط نیاز به انجام کار در مُد کاربر دارند، آن‌ها می‌توانند در فواصل معین محتوای اجرایی خودشان را با یکدیگر تعویض کنند، بدون اینکه زمانبند کرنل[3] سامانه‌عامل درگیر جزئیات شود.

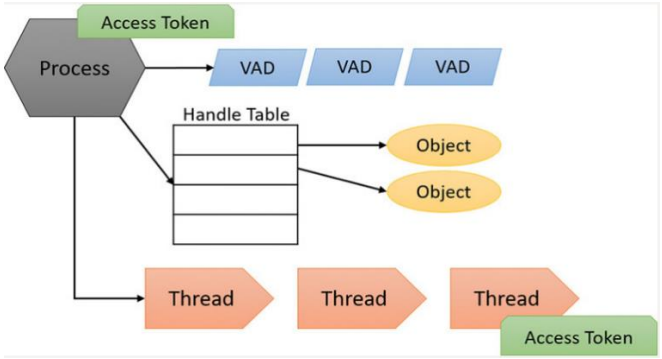
[1] Fibers and User-Mode Scheduler Threads
[2] lightweight threads
[3] Windows Scheduler

از چشم‌انداز کرنل[1] سامانه‌عامل ویندوز، ترد مشابه اجرا شده در کرنل هنوز در حال اجرا است و چیزی عوض نشده است. هنگامیکه یک ترد زمانبندی شده در مُد کاربر، قصد دارد یک عملیات انجام دهد که نیاز به وارد شدن به کرنل دارد، از قبیل فراخوانی یک تابع سیستمی، با ترد مُد کرنل خودش تعویض می‌شود.

گرچه تردها کانتَکست اجرایی خودشان را دارند، اما با این حال هر ترد درون پروسه‌ها، فضای آدرس مجازی پروسه را به اشتراک می‌گذارد (علاوه بر این، تمامی منابع مرتبط با پروسه را هم به اشتراک می‌گذارد)، این بدین معنا است که هر ترد دسترسی خواندن و نوشتن کامل بر روی فضای آدرس مجازی پروسه‌ها دارد.

تردها نمی‌توانند به صورت تصادفی فضای آدرس یک پروسه دیگر را مورد ارجاع قرار دهند، مگر اینکه پروسه مذکور قسمتی از فضای آدرس خود را به عنوان قسمت حافظه اشتراکی موجود کرده باشد یا یک پروسه حقوق باز کردن یک پروسه دیگر را به منظور استفاده از توابعی از قبیل ReadProcessMemory و WriteProcessMemry را داشته باشد.

علاوه بر فضای آدرس خصوصی و یک یا چند ترد، هر پروسه یک محتوای امنیتی و یک لیست از هندل‌های باز به آبجکت‌های کرنل از قبیل فایل‌ها، بخش‌های اشتراکی حافظه یا یک آبجکت همگام‌ساز از قبیل موتکس، رویدادها یا سمافورها دارد.



تصویر ۹: یک پروسه و منابع آن

[1] Kernel's Perspective
[2] Access Token

کانتَکست امنیتی هر پروسه در یک آبجکت ذخیره می‌شود که توکن دسترسی۲ خوانده می‌شود. پروسه به این توکن دسترسی می‌گیرد که شامل شناسه امنیتی و اعتبارنامه پروسه است. در حالت پیش‌فرض، تِرَدها توکن دسترسی مختص به خود را ندارند، اما آن‌ها می‌توانند برای خود یک توکن اخذ کنند، در نتیجه این عملیات تِرَدهای منفرد می‌توانند محتوای امنیتی یک پروسه دیگر را جعل کنند – از جمله پروسه‌های یک سیستم ویندوز راه‌دور – بدن اینکه بر روی دیگر تِرَدها درون پروسه تاثیری گذاشته شود.

دِسکریپتور آدرس مجازی[3] ساختمان داده‌ای است که مدیر حافظه از آن به منظور حفظ آدرس‌های مجازی در حال استفاده توسط پروسه‌ها استفاده می‌کند.

جاب‌ها

ویندوز یک مدل توسعه داده شده از پروسه با نام آبجکت جاب یا Job ارائه می‌کند. تابع اصلی این آبجکت اجازه می‌دهد گروهی از پروسه‌ها به عنوان یک واحد منفرد مدیریت و دستکاری شوند. این آبجکت همچنین اجازه کنترل برخی خاصیت‌ها و ارائه محدودیت برای پروسه یا پروسه‌های در ارتباط با آن را ارائه می‌دهد.

در برخی از شرایط، آبجکت جاب به دلیل عدم وجود یک ساختار درختی برای پروسه‌ها در ویندوز غرامت می‌دهد، اما با این حال هنوز در بیشتر شرایط ساختار درختی ویندوز از ساختار درختی پروسه Unix قدرتمندتر است. در مورد ساختمان داده‌های داخلی از جمله جاب‌ها، پروسه‌ها و تردها، همچنین مکانیزم پروسه‌ها و ایجاد تردها و الگوریتم‌های زمانبندی ترد در قسمت‌های بعدی این سلسله مقالات اطلاعات بیشتری به دست خواهید آورد.

پایان

در این قسمت از سلسله مقالات ویندوز اینترنالز با مفاهیم و ابزارهای کاربردی دیگر ویندوز از قبیل پروسه‌ها، تردها، فیبرها، تردهای زمانبندی کاربر، جاب‌ها و ... آشنا شدیم. از همین روی،

[3] Virtual Address Descriptor (VAD)

در ادامه خواهیم توانست به جزئیات سطح پایین سامانه‌عامل ویندوز از قبیل حافظه مجازی، دیباگ کرنل ویندوز، عملکرد MMU و ... بپردازیم و این موارد را با جزئیات دقیق‌تر مورد بررسی قرار بدهیم.