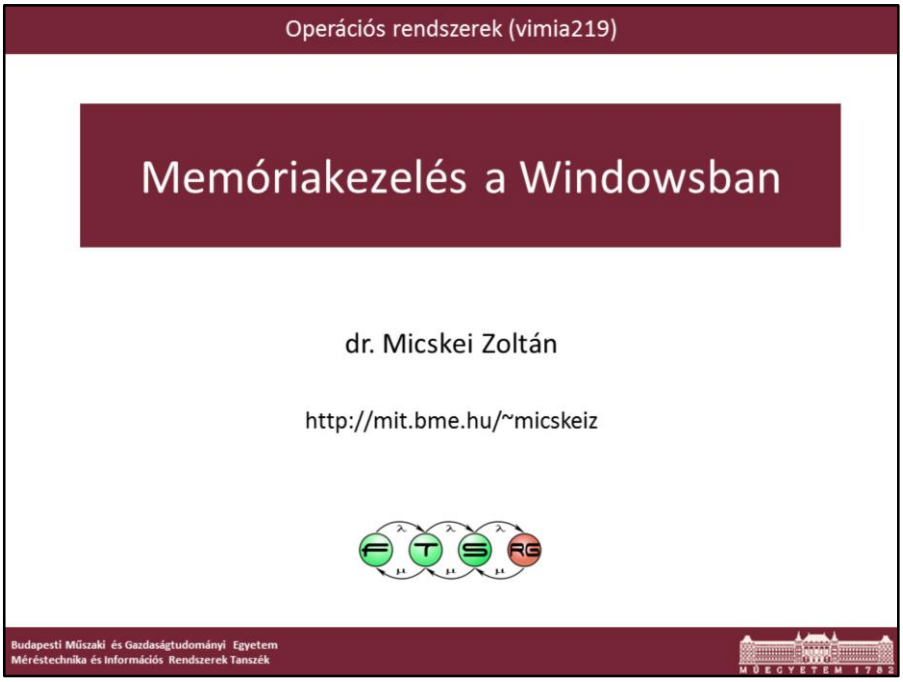
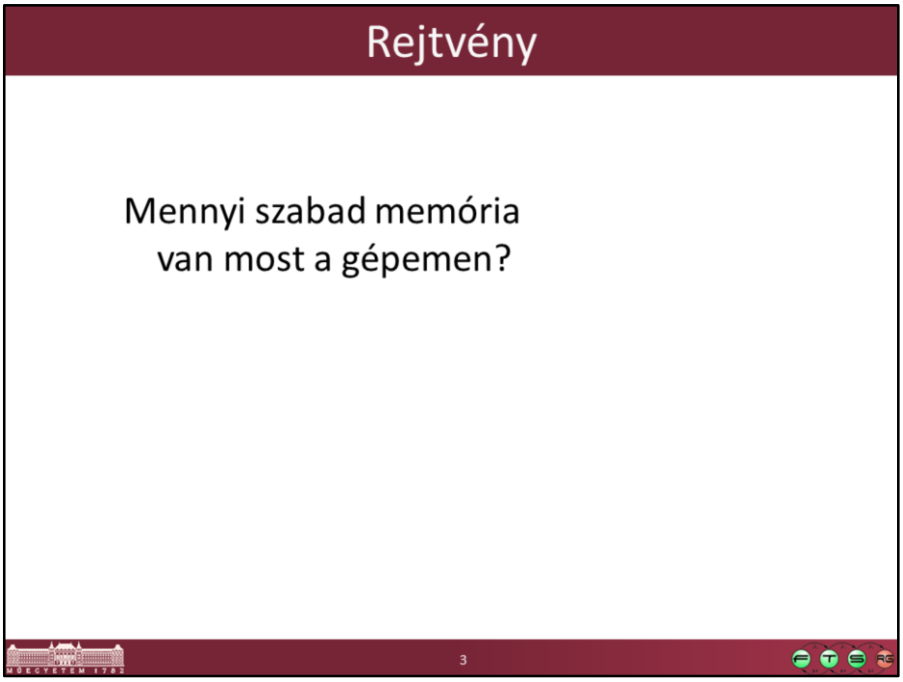


Utolsó módosítás: 2014. 04. 15.

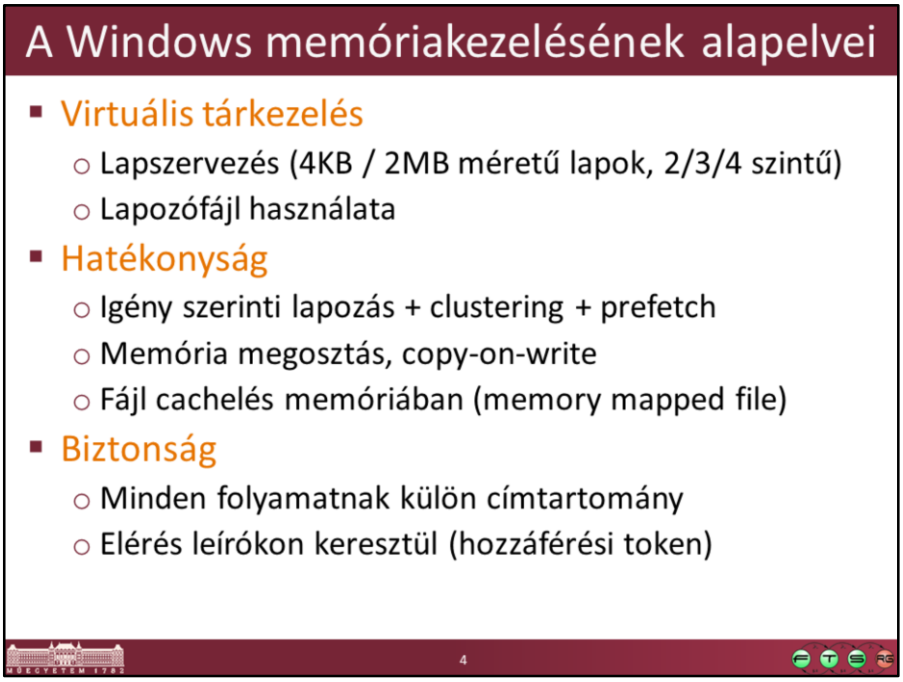
Copyright Notice

- These materials are part of the *Windows Operating System Internals Curriculum Development Kit,* developed by David A. Solomon and Mark E. Russinovich with Andreas Polze
- Microsoft has licensed these materials from David Solomon Expert Seminars, Inc. for distribution to academic organizations solely for use in academic environments (and not for commercial use)
- http://www.academicresourcecenter.net/curriculum/pfv.aspx?ID=6191
-

A fóliák részben a Windows Operating System Internals Curriculum Development Kit alapján készültek.

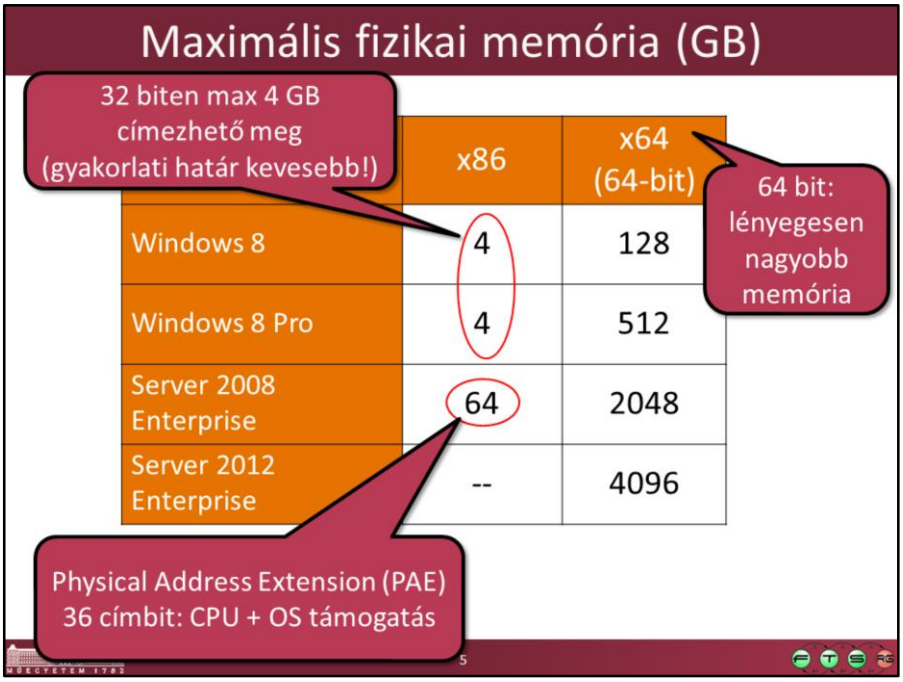


Nem is olyan triviális megválaszolni!
- Feladatkezelő számait óvatosan kell kezelni!
- Mit értek szabad alatt?
- Ha azt írja, hogy 10 MB van szabadon, akkor miért tudok elindítani egy olyan programot, ami 15 MB memóriát foglal le?
- Cache-nek használt memória hova számít?
- Ha a program, amit el akarok indítani, sok megosztott memóriát használ, akkor lehet, hogy sokkal kevesebbet foglalna most, mint akkor, ha csak egyedül fut.



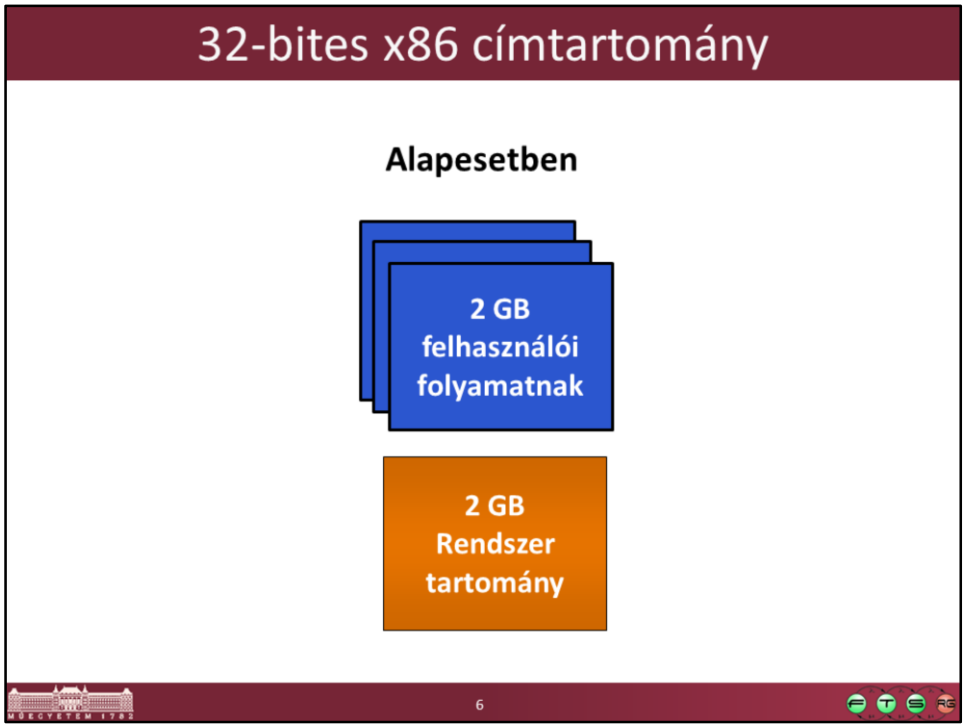
Laptáblák:
- x86: 2 szintű laptáblák, x86 + PAE: 3 szintű, x64: 4 szintű

Lapozás:
- Kezdetben igény szerinti lapozás volt
- Clustering: ezt kiegészítették azzal, hogy a kért lap környékén lévő pár lapot is behozta (lokalitás elv)
- Prefetch: XP óta van egy prefetcher, ami rögzíti, hogy a programok induláskor miket szoktak igényelni, és azokat előre behozza (részletesebben lásd az előadás második felében)
- Vista óta további prefetch jellegű szolgáltatások (SuperFetch, ReadyBoost…)



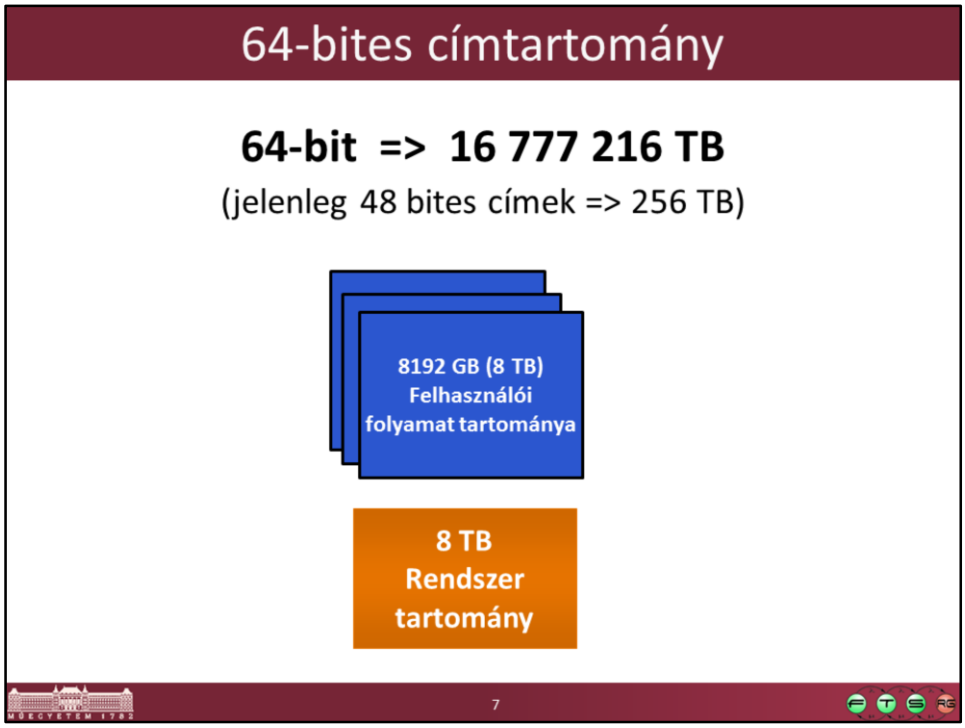
- 32 bites rendszereken (ahol nincs PAE), 4 GB fizikai memória esetén is néha kevesebbet lát az OS, mert a memóriatartomány felső részére I/O eszközöket szoktak berakni (pl. ha van egy nagy memóriájú videokártyánk, akkor lehet, hogy csak 3,5 GB-ot lát az OS).
- *Physical Address Extension* (PAE): bizonyos 32 bites processzorokban meglévő támogatás, ilyenkor 32 helyett 36 címbit van. Ha ezt az operációs rendszer ki tudja használni, akkor képes 64 GB fizikai memóriát is kezelni. Figyelem: a folyamatok címtere továbbra is 32 bites marad, csak az OS tud több fizikai memórialapot kiosztani a különböző folyamatoknak.
    - Kliens Windowsok nem használják a gépben lévő PAE támogatást, mert az a tapasztalat, hogy a kliensekben lévő eszközök meghajtói nem kezelik le rendesen a 4 GB-nál több fizikai memóriát.
    - Lásd még: *Physical Address Extension*, http://msdn.microsoft.com/en-us/library/aa366796%28v=VS.85%29.aspx

Érdekességek:
- *Memory Limits for Windows Releases*, http://msdn.microsoft.com/en-us/library/aa366778%28VS.85%29.aspx
- *Pushing the Limits of Windows: Physical Memory*, http://blogs.technet.com/markrussinovich/archive/2008/07/21/3092070.aspx



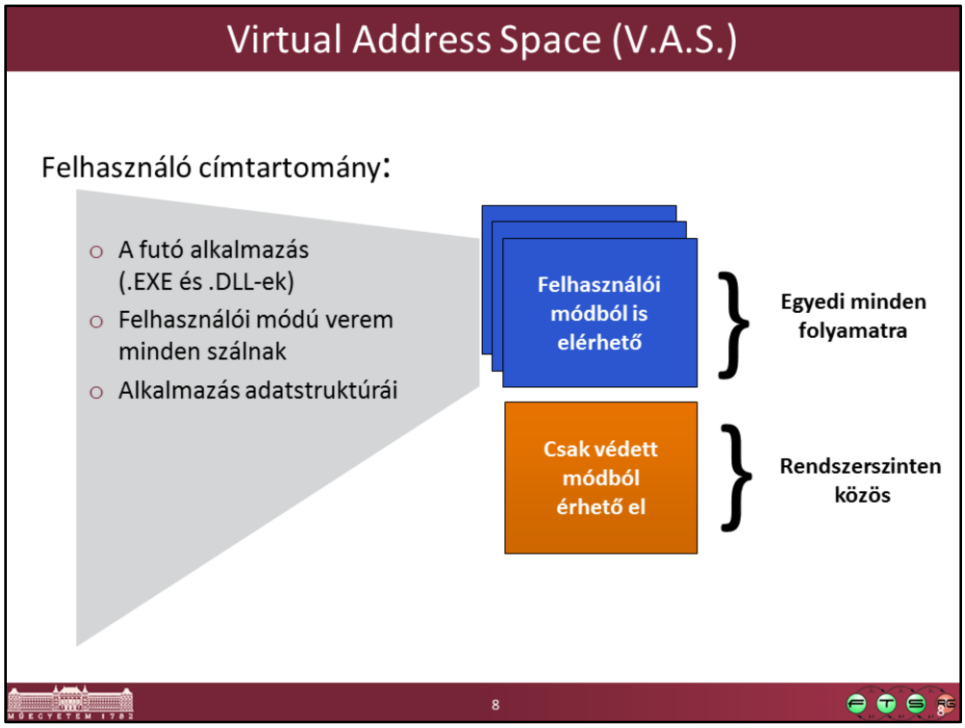
- Alapesetben 32 bites rendszeren egy felhasználói folyamat maximum 2 GB-os címtartományt használhat fel a saját kódjának és adatának tárolására. A címtartomány másik részén mindig a rendszer memóriaterület látszik.

Érdekesség:
- Ha a boot.ini-ben bekapcsoljuk a /3GB kapcsolót, akkor az arány 3 GB – 1 GB-ra változik (ez egy kényszer megoldás volt, akkor került bele, amikor már kevés volt a 2 GB egy folyamatnak, pl. egy nagy adatbázis-kezelőnek, de még nem tértek át a 64 bites OS-re)
    - Vista óta ezt 4GT-nek hívják, lásd: *4-Gigabyte Tuning*, http://msdn.microsoft.com/en-us/library/bb613473%28VS.85%29.aspx
    - A jelentősége egyre kisebb, amióta a 64 bites verziók egyre elterjedtebbek, hisz ott (még) nincs ilyen gond.



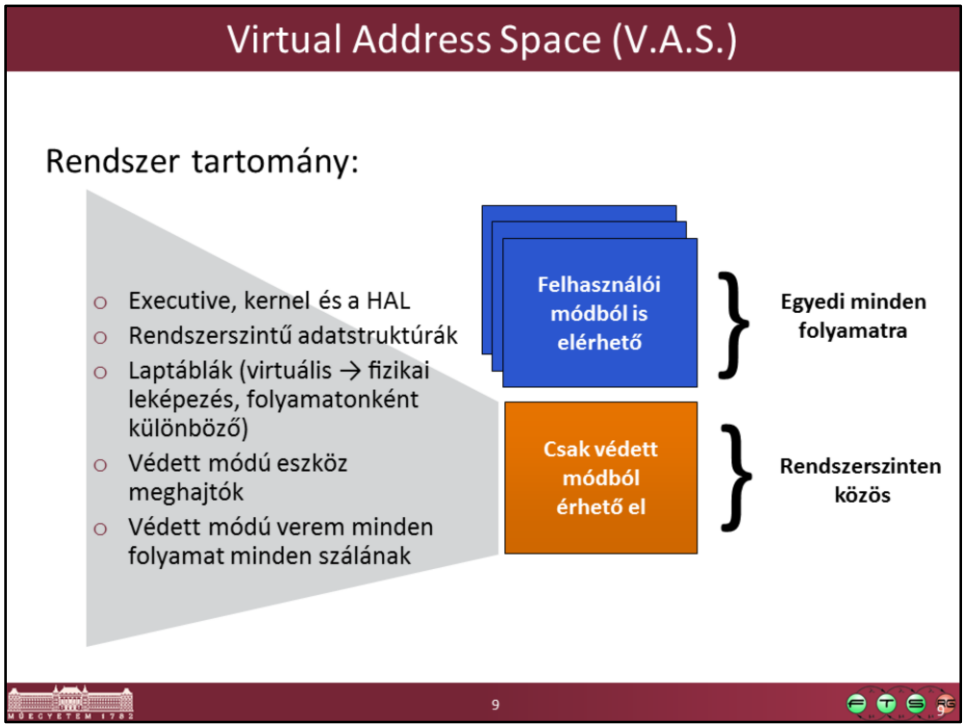
Látványosan nagyobb a címtér 64 bites esetén, ez a fő motivációja a 32 bitről áttérésnek.

Windows 8.1-ben ezt tovább növelték, ott már 128 TB mindkét rész, így elvileg a teljes 48 bites címtartomány használható.



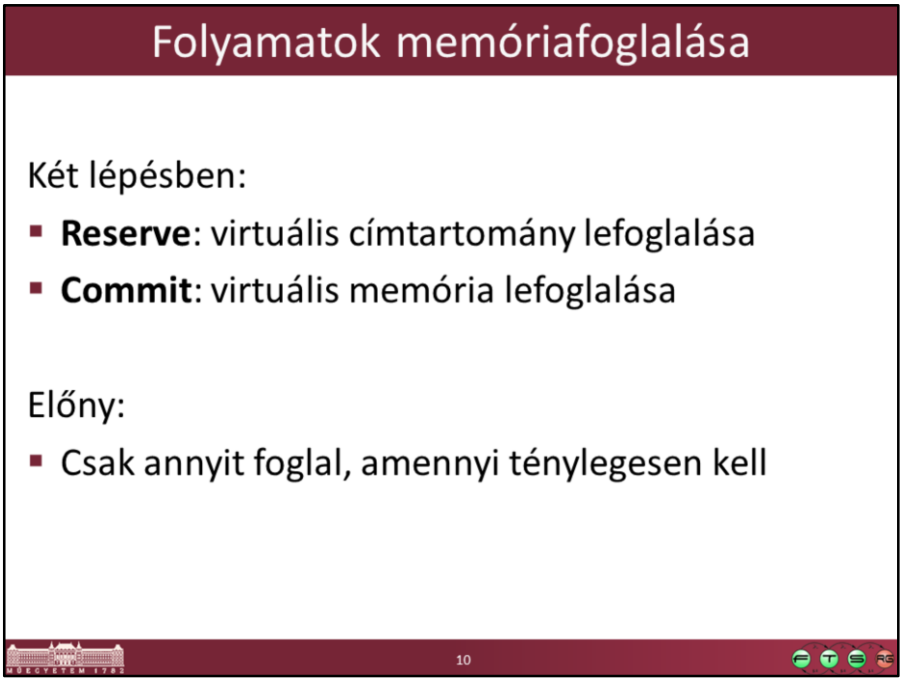
A felhasználói módú virtuális címtartomány minden folyamatra különböző, minden folyamatnak saját címtere van.
Másik folyamat címterét csak úgy érheti el, ha előbb a Windows biztonsági rendszere leellenőrzi, hogy van-e megfelelő jogosultsága rá.

8

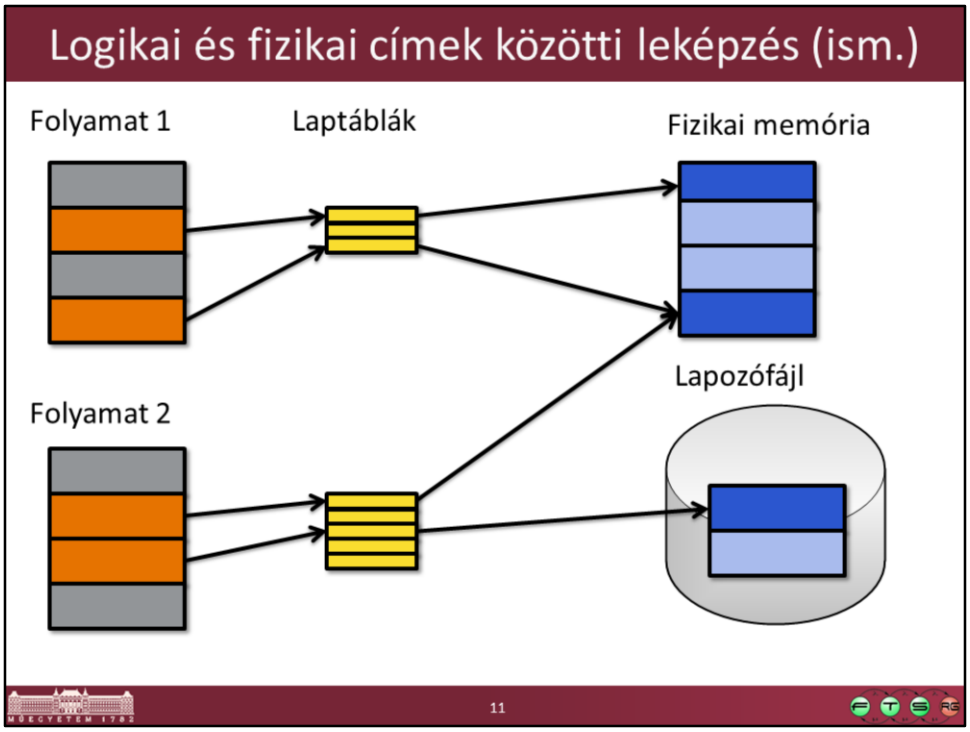


No user process can touch kernel memory

- Page protection in process page tables prevent this
- OS pages only accessible from “kernel mode”
- Threads change from user to kernel mode and back (via a secure interface) to execute kernel code, this does not affect scheduling (not a context switch)



- A commited memóriának kell helyet fenn tartani, a reserve még csak szándék jelzése.
- Ez a foglalási módszer a Windows API használata esetén látszik. A foglalás típusát a VirtualAlloc() memóriafoglaló függvény *dwAllocationType* paraméterében lehet jelezni (MEM_COMMIT vagy MEM_RESERVE).



Az ábra csak szemléltető jellegű, a valóság bonyolultabb ennél. Pl. a laptáblákat is tárolni kell valahol, azok is részei a folyamat virtuális címterének; a lapozófájlban lévő lapokra nem direktbe mutat rá a laptábla…

x86 címfordítás (PAE nélkül)

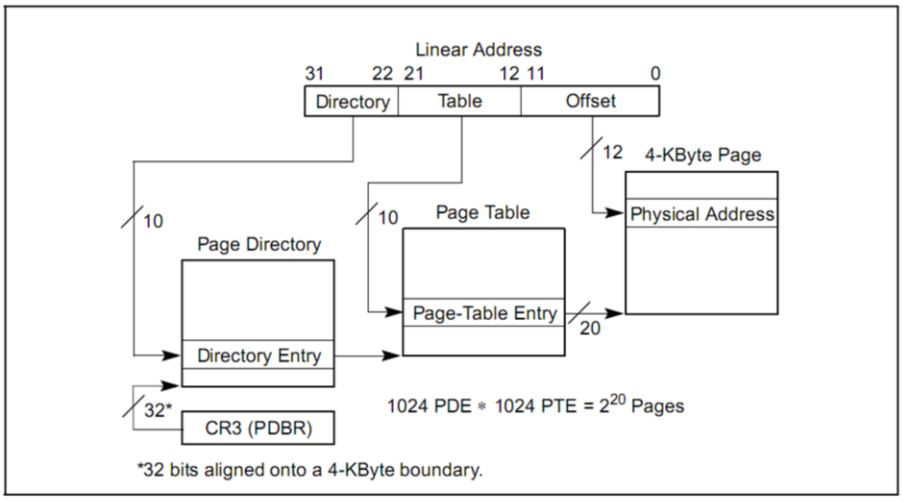
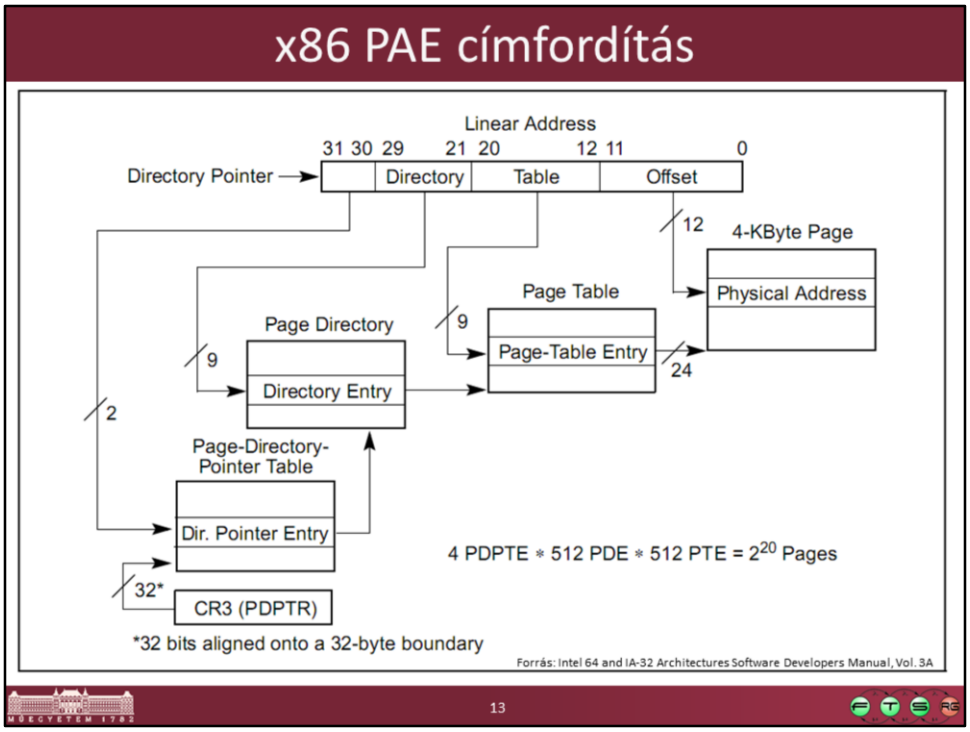


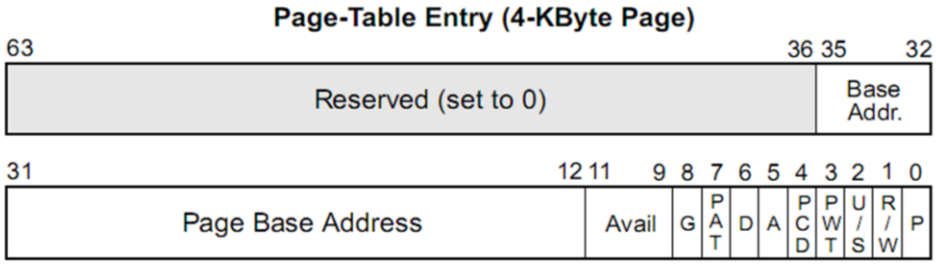
Figure 3-12. Linear Address Translation (4-KByte Pages)

Forrás: Intel 64 and IA-32 Architectures Software Developers Manual, http://www.intel.com/products/processor/manuals/



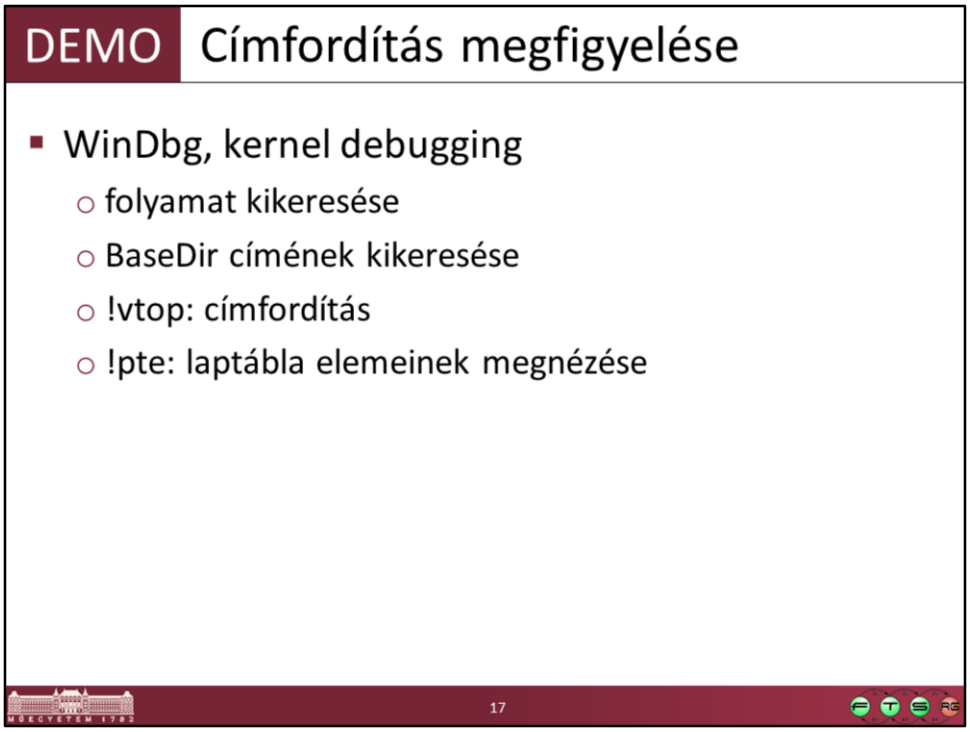
Hiába van PAE, a folyamatunk továbbra is csak 2^20 fizikai memória keretet tud megcímezni, így a 12 bites offsettel együtt ez továbbra is csak 4GB memória. A különbség ott van a PAE-nél, hogy a fizikai memórialap címzésére 24 bitet használunk, így az OS a folyamatok 4 GB-os címtereit 64 GB (= 2^(24+12)) fizikai címtartományba tudja elhelyezni.

x86 PAE esetén a PTE



- 64 bites, 24 bit a lap címének
- Flagek:
  - P – present, A – access, D – dirty, U/S – user/system, R/W – read/write…

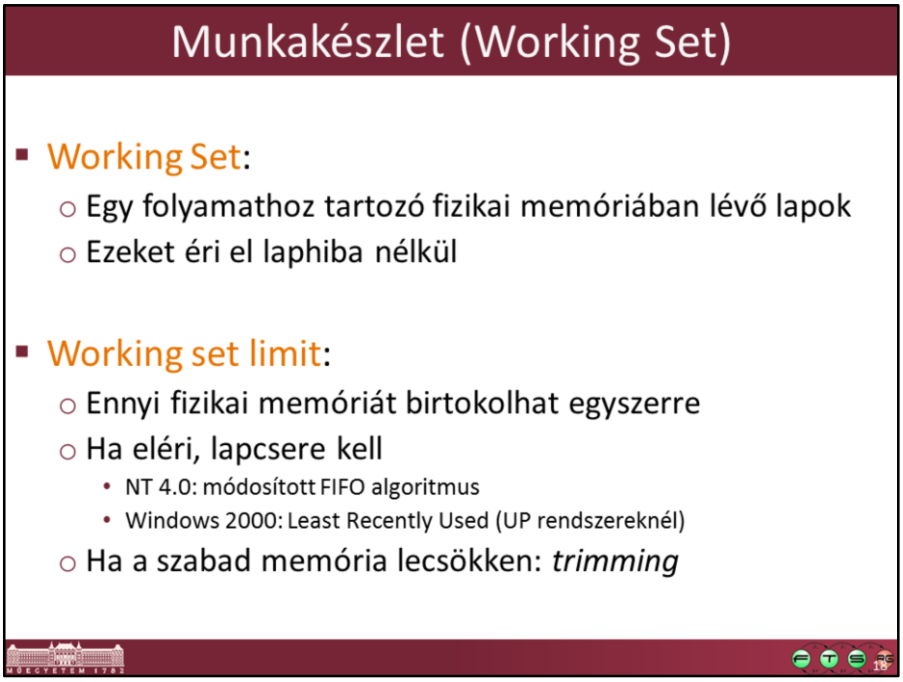
Forrás: Intel 64 and IA-32 Architectures Software Developers Manual, http://www.intel.com/products/processor/manuals/



Local kernel debuggerben
!process 0 0
keressük ki a windbg.exe-t
!process <process address>
egyik szál start address mezőjét kikeresni
!vtop 0 <start address>
hibát dob, mert rossz folyamat kontextusában próbáljuk meg a címfordítást
.process /p <process address>
!vtop 0 <start address>
így már sikeres
db <virtual address>
!db <physical address>
elvileg ugyanazt kell látni mindkét helyen
!pfn <pfn from physical address>

Részletesebb leírás:
- *Virtuális - fizikai címfordítás Windows alatt x64-en*,
http://micskeiz.wordpress.com/2009/05/09/virtualis-fizikai-cimforditas-windows-alatt-x64-en/

Vagy itt:
-*Understanding !PTE , Part 1: Let's get physical*,
http://blogs.msdn.com/ntdebugging/archive/2010/02/05/understanding-pte-part-1-let-s-get-physical.aspx
-*Understanding !PTE, Part2: Flags and Large Pages ,*
http://blogs.msdn.com/ntdebugging/archive/2010/04/14/understanding-pte-part2-flags-and-large-pages.aspx

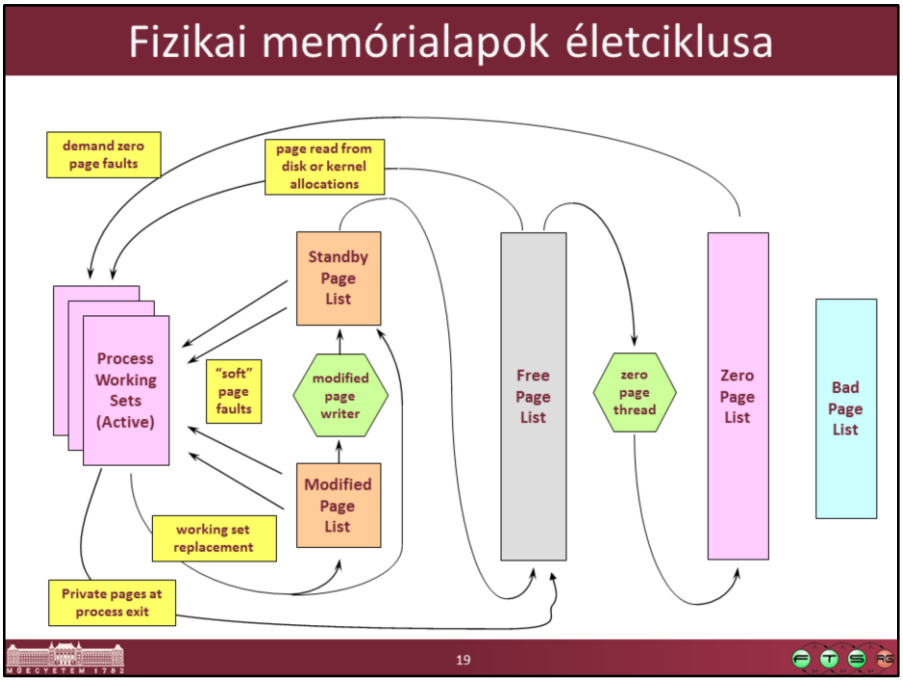


Working set: Implemented as array of working set list entries (WSLE)

Soft vs. Hard laphiba:

Soft page fault: memóriában volt még a lap
Hard page fault: lemezről kell beolvasni

Trimming: egy rendszer szál megvizsgálja a folyamatokat, és akinek sok fizikai memórialapja van vagy régóta nem használta őket, azoktól kezd elvenni párat. Ezt addig folyatja, amíg „elég” szabad memória nem lesz a rendszerben.



Status Description

Active/valid: Page is part of working set (sys/proc), valid PTE points to it

Transition: Page not owned by a working set, not on any paging list, I/O is in progress on this page

Standby: Page belonged to a working set but was removed; not modified

Modified: Removed from working set, modified, not yet written to disk

Free: Page is free but has dirty data in it – cannot be given to user process – C2 security requirement

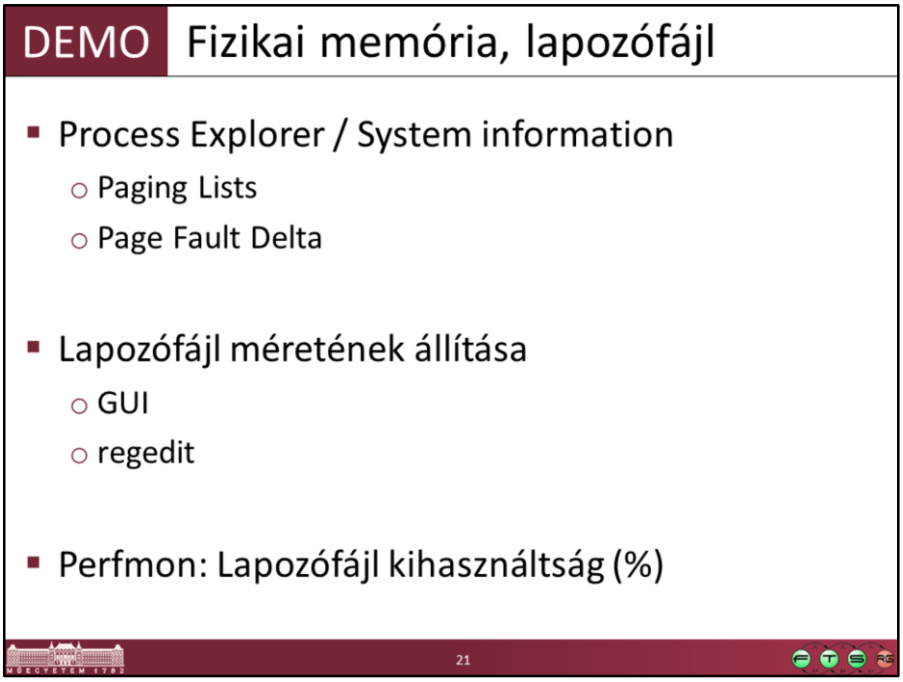
Zeroed: Page is free and has been initialized by zero page thread

Bad: Page has generated parity or other hardware errors

Lapozófájl (page file)

- Mi kerül bele?
  - Csak a *módosított* adat, kód nem
- Mikor kerül bele?
  - Ha van szabad memória, akkor is lehet
  - Folyamatok nem foglalhatnak bármennyi memóriát
  - Tartalék az új/többi folyamatnak
- Meghajtónként egy darab
  - Ajánlott nem a rendszerlemezre rakni
  - De maradjon egy kicsi ott is a memory dumpnak
- Ajánlott méret
  - 1 vagy 1,5-szer a fizikai memória (?), Fix méret (?)

Lásd még*: How to configure paging files for optimization and recovery in Windows XP* (http://support.microsoft.com/kb/314482/en-us)



Lapozófájl állítása: GUI: System Properties / Advanced / Performance, Settings / Advanced / Virtual Memory, Settings
Registry: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management

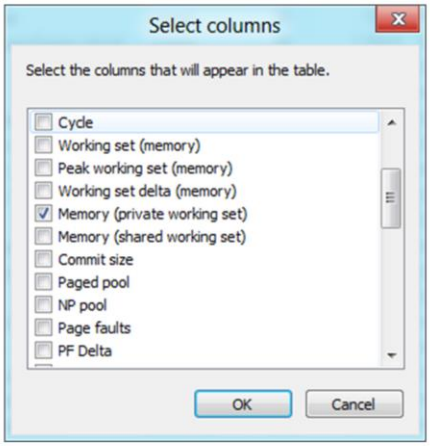
Memóriahasználat megfigyelése

Egyszerűnek tűnő kérdés:

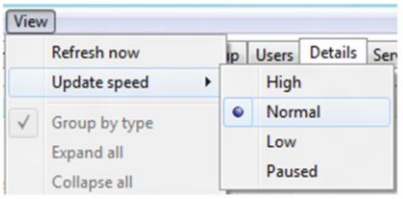
Mi mennyi memóriát foglal a gépen?

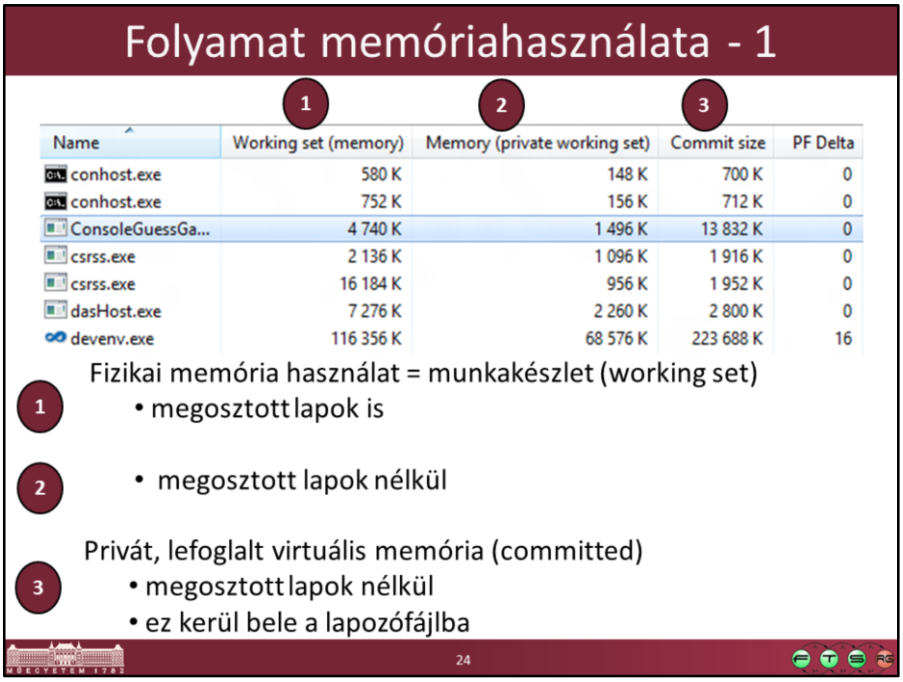
Folyamat memóriahasználata - 0

- Feladatkezelő oszlopai



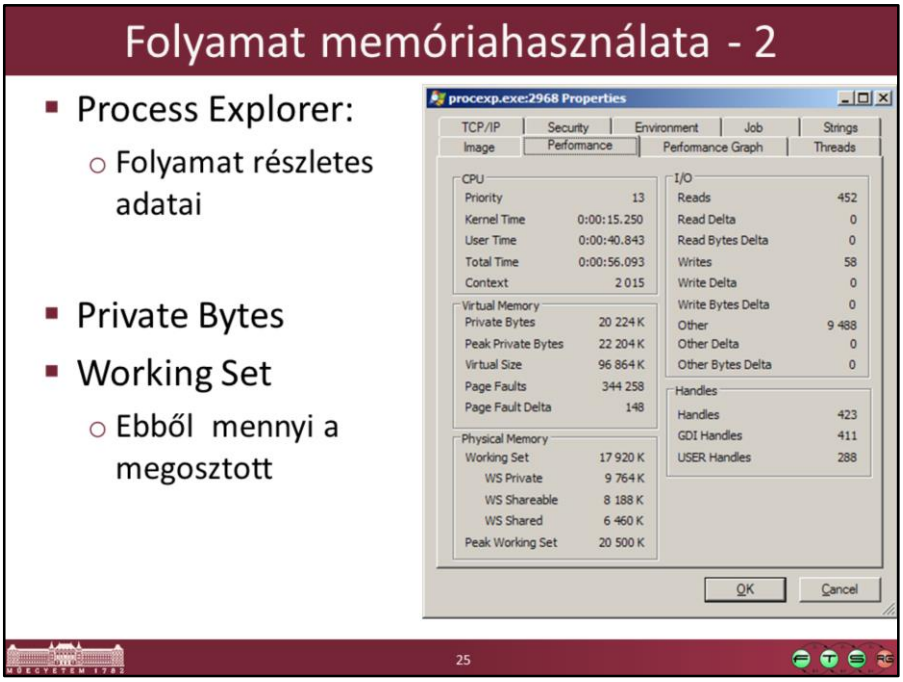
- Frissítés gyakorisága





A megosztott lapok beleszámolása illetve nem számolása miatt állhat elő az, hogy egyes folyamatoknál:
• a virtuális memória a nagyobb (okai lehetnek: kevesebb megosztott memória használata; memóriaterülete kikerült a lapozófájlba;)
• a munkakészlet a nagyobb (pl. sok megosztott dll-t használata)

• Ha összeadjuk a Mem usage oszlop tartalmát, az nem a folyamatok által használt tényleges fizikai memória lesz, mert az osztott lapokat többször számoltuk, és ebben nincs benne a kernel memória.

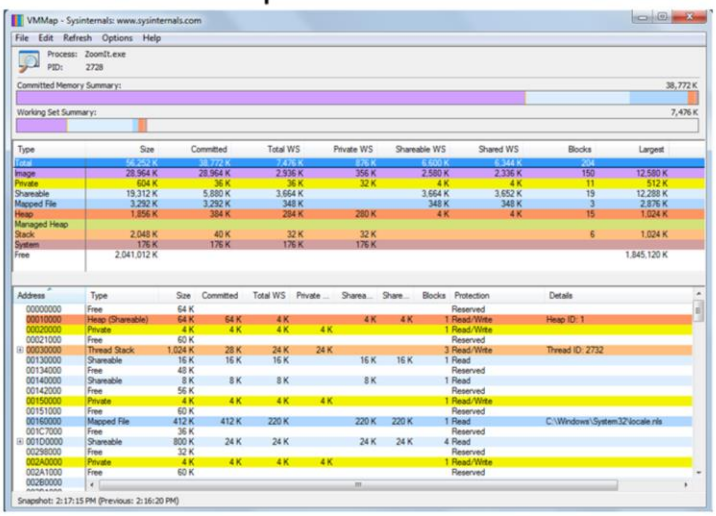


Process Explorer

Kernel debugger: !vm, !memusage

DEMO Folyamat memóriaterülete

- Sysinternals VMMap



A VMMap segítségével megnézhetjük, hogy egy adott folyamatnak milyen elemek vannak jelenleg a címterében:

- Hogy oszlik meg kód/adat/halom/verem között
- Adat esetén mennyi a reserved és mennyi a ténylegesen committed, majd ebből mennyi van aktuálisan a fizikai memóriában

A teljes rendszer memóriahasználata

1

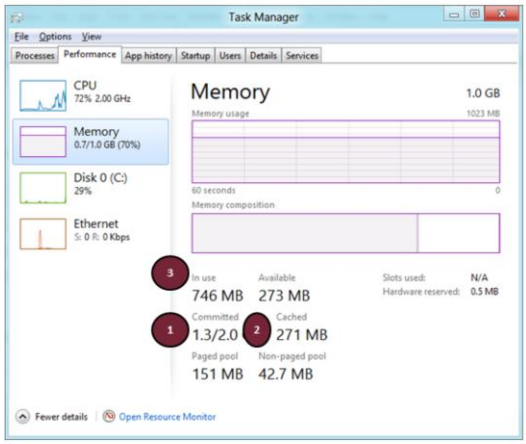
- Összes lefoglalt virtuális memória
- Ennyit kéne kiírni a lapozófájlba, de nem biztos, hogy ennyi van kiírva

2

Előjegyzési küszöb: összes fizikai memória + lapozófájlok aktuális mérete

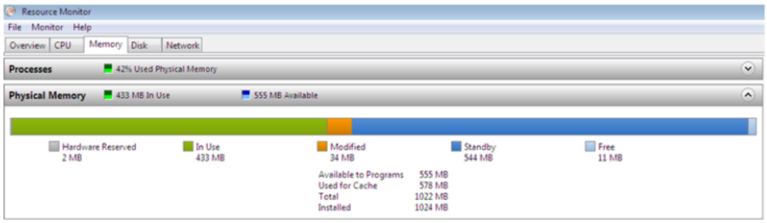
3

~ Aktív memórialapok száma

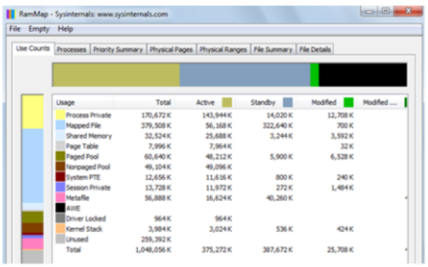


DEMO Resource Monitor

- Memórialapok állapotának gyors áttekintése:



- Sysinternals RamMap: részletek

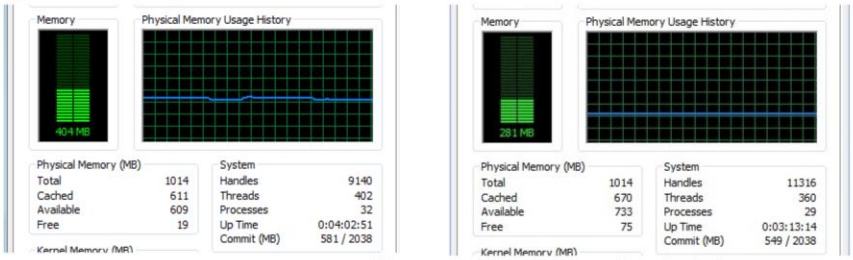


- Memórialapok eloszlása
- Indítsunk el egy memóriaintenzív alkalmazást (pl. virtuális gép), és nézzük, hogyan változik az eloszlás, hogyan nő közben a hard page fault / sec

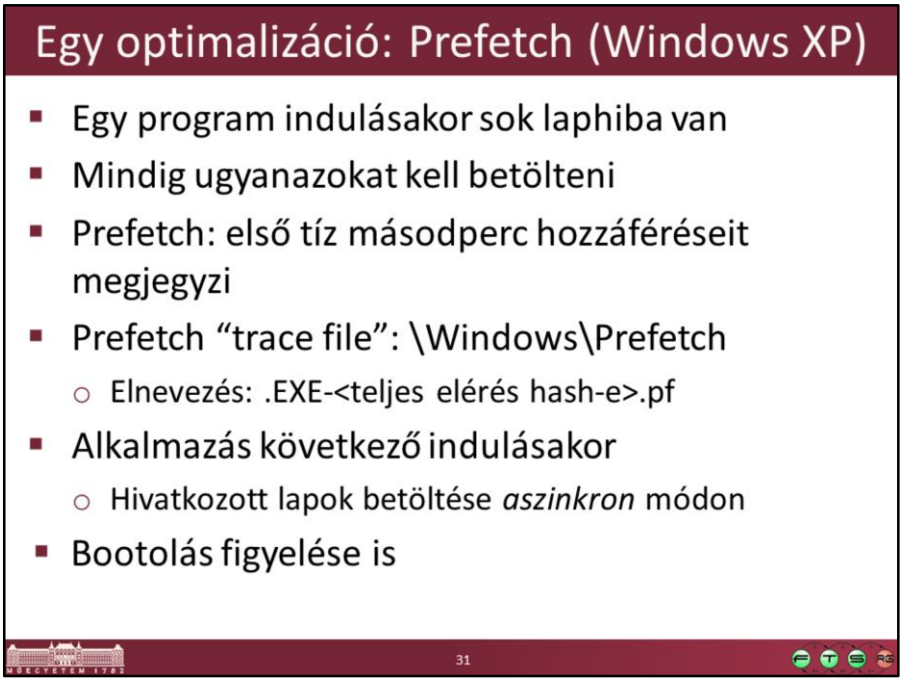
További optimalizációk

Windows 8: memóriahasználat csökkentése

- Memory combining
  - azonos tartalom keresése a háttérben
- Szolgáltatások csökkentése
  - + „Start on demand” indítási mód
- „Hot” és „cold” adatstruktúrák szétválasztása



Lásd: *Reducing runtime memory in Windows 8,* http://blogs.msdn.com/b/b8/archive/2011/10/07/reducing-runtime-memory-in-windows-8.aspx

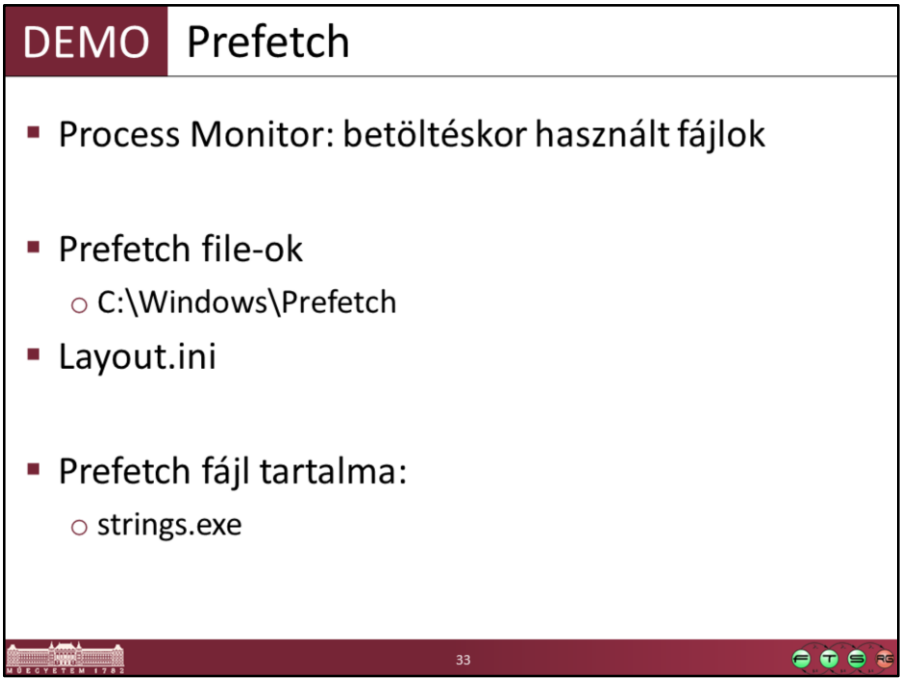


A gondot az okozza, hogy prefetch nélkül először pár lapot az egyik fájlból, majd pár lapot a másik dll-ből tölt be (ahogy éppen az utasítások jönnek a program kódban), és így a merevlemezen a fejnek össze-vissza kell mozognia.

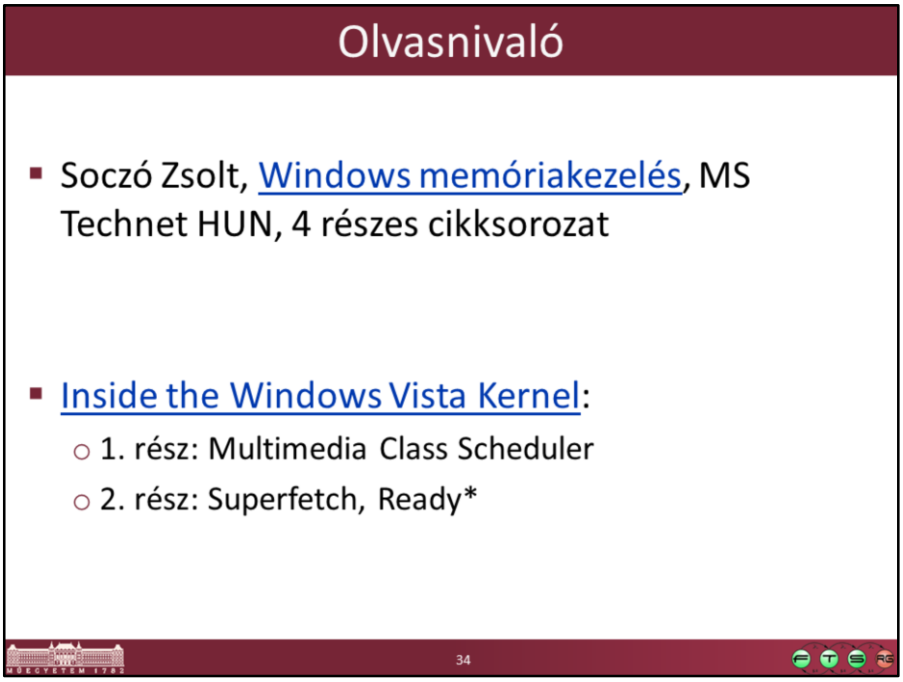
Részletek: Windows XP Kernel Improvements, http://msdn.microsoft.com/en-us/magazine/cc302206.aspx

Forrás: http://technet.microsoft.com/en-us/magazine/cc162480.aspx

A significant change to the Memory Manager is in the way that it manages physical memory. The Standby List management used by previous versions of Windows has two limitations. First, the prioritization of pages relies only on the recent past behavior of processes and does not anticipate their future memory requirements. Second, the data used for prioritization is limited to the list of pages owned by a process at any given point in time. These shortcomings can result in scenarios like the "after lunch syndrome," where you leave your computer for a while and a memory-intensive system application runs (such as an antivirus scan or disk defragmentation). This application forces the code and data that your active applications had cached in memory to be overwritten by the memory-intensive activities. When you return, you experience sluggish performance as applications have to request their data and code from disk.
Windows XP introduced prefetching support that improved boot and application startup performance by performing large disk I/Os to preload memory with code and file system data that it expected, based on previous boots and application launches. Windows Vista goes a big step further with SuperFetch, a memory management scheme that enhances the least-recently accessed approach with historical information and proactive memory management.
SuperFetch is implemented in %SystemRoot%\System32\Sysmain.dll as a Windows service that runs inside a Service Host process (%SystemRoot%\System32\Svchost.exe). The scheme relies on support from the Memory Manager so that it can retrieve page usage histories as well as direct the Memory Manager to preload data and code from files on disk or from a paging file into the Standby List and assign priorities to pages. The SuperFetch service essentially extends page-tracking to data and code that was once in memory, but that the Memory Manager has reused to make room for new data and code. It stores this information in scenario files with a .db extension in the %SystemRoot%\Prefetch directory alongside standard prefetch files used to optimize application launch. Using this deep knowledge of memory usage, SuperFetch can preload data and code when physical memory becomes available.
Whenever memory becomes free-for example, when an application exits or releases memory-SuperFetch asks the Memory Manager to fetch data and code that was recently evicted. This is done at a rate of a few pages per second with Very Low priority I/Os so that the preloading does not impact the user or other active applications. Therefore, if you leave your computer to go to lunch and a memory-intensive background task causes the code and data from your active applications to be evicted from memory while you're gone, SuperFetch can often bring all or most of it back into memory before you return. SuperFetch also includes specific scenario support for hibernation, standby, Fast User Switching (FUS), and application launch. When the system hibernates, for example, SuperFetch stores data and code in the hibernation file that it expects (based on previous hibernations) will be accessed during the subsequent resume. In contrast, when you resume Windows XP, previously cached data must be reread from the disk when it is referenced.



defrag c: -b
Layout.ini-nek megfelelően az induláshoz szükséges fájlokat megpróbálja folyamatosa helyre rendezni, hogy gyorsabb legyen az indulás



Technet HUN cikkek:

- Windows memóriakezelés - 1. rész - Memóriakezelési alapok, http://www.microsoft.com/hun/technet/article/?id=7870c80b-7d94-4732-8220-256b625a3061
- Windows memóriakezelés - 2. rész - A virtuális és a fizikai memória kapcsolata, http://www.microsoft.com/hun/technet/article/?id=d0da1b41-169a-4e8c-a793-d0af5dc31d37
- Windows memóriakezelés - 3. rész - Paged és Nonpaged Pool, http://www.microsoft.com/hun/technet/article/?id=d698795b-d9a0-4c78-bd65-3be12d0780dc
- Windows memóriakezelés - 4. rész - Page File és Working Set, http://www.microsoft.com/hun/technet/article/?id=dcf95237-3966-4213-a66c-88102d8d1cbf
- Windows memóriakezelés - 5. rész - Memóriakezelést gyorsító szolgáltatások, http://www.microsoft.com/hun/technet/article/?id=a93f3d98-a3ff-42fb-a57d-47e2387250aa

Inside the Windows Vista Kernel

- Part 1: http://technet.microsoft.com/en-us/magazine/2007.02.vistakernel.aspx
- Part 2: http://technet.microsoft.com/en-us/magazine/2007.03.vistakernel.aspx
- Part 3: http://technet.microsoft.com/en-us/magazine/2007.04.vistakernel.aspx

Összefoglalás

- Virtuális tárkezelés, lapszervezés
- Többszintű optimalizáció
- Memóriahasználat vizsgálata
  - Feladatkezelő: gyors áttekintés
  - Process Explorer, Perfmon, VMMap stb.: részletek