1.344 Ft

II. Évfolyam 3. szám

ISSN 15865185 03

tech.net

A Microsoft Magyarország szakmai magazinja

XML

Developer • XMLgessünk
Microsoft Research • Háló a falon
Biztonság • Nevem senki

Tanulni sohasem késő!

(V. I. Lenin)

Ismernek Önök szoftverfejlesztőt? Vagy esetleg Önök pont azok? Akkor bizonyára tudják, hogy a szoftverfejlesztők általában megszállott emberek. Nehezebbnél lehetetlenebb feladatokkal kell szembenézniük nap mint nap, és eközben olyan problémákba futnak bele, amelyekhez úgy érzik, nincs segítségük. Éjszakákat töltenek egy megoldhatatlannak tűnő feladat megoldásán, hogy a rákövetkező nap már egy újabbon törhessék a fejüket. A lényeg a kihívás, a jutalom, a megoldott probléma felett érzett dicsőség, illetve az elkészült, működő alkalmazás által nyújtott alkotói öröm.

Az a szerencsés fejlesztő, akinek nem lebegnek a határidő bárdjai a nyaka felett valóban ezt érzi, és élvezi a fejlesztést. Azonban a határidők mindig CurrentDay-1–re vannak datálva. A forrongó, öles léptekkel haladó piac kíméletlen tempót diktál, és emellett, természetesen minden fejlesztő cég a legújabb technológiát szeretné latba vetni a fejlesztések során. Még így is sokszor előfordul, hogy mire elkészül a szuper csoda rendszer, már 25 új technológia döngeti a kapukat, még hatékonyabb, még egyszerűbb, még sokkal többet tudó lehetőségekkel kecsegtetve. 10 évvel ezelőtt egy C programozó a nyelvtudásával úgy érezhette, hogy előtte hever a szakma, és minden számítástechnikai problémát képes megoldani. Egy picit azért megváltozott a világ, nem? Delphi, Basic, Java, Python, Perl, PHP, csak hogy a legismertebb nyelveket emlegessem. És akkor még nem is mertem szólni a C#-ról, amibe akár tetszik akár nem, hamarosan minden Microsoft technológiával foglalkozó fejlesztőnek bele kell kóstolnia.

De a nyelv, a szintakszis csak egy dolog, amit pár nap alatt el lehet sajátítani. Az igazi kihívást mindig a nyelvek mögött álló eljárásgyűjtemények, programkönyvtárak, programozási keretrendszerek megismerése jelenti. Ezekkel már éveket el lehet tölteni, és mégsem ismerünk minden benne található lehetőséget. A mai szoftverek írásakor eléggé háttérbe szorult az algoritmusok ismeretének jelentősége, sokkal inkább fontos, hogy a programtervező megtalálja azokat az eszközöket, amelyek egy adott probléma megoldására a leghatékonyabbak.

Azaz a fejlesztő amellett, hogy dolgozik, még valamikor meg is kellene tanulnia az új eszközökben rejlő lehetőségeket, hogy egyáltalán tudja, mikor mihez nyúljon *(a fején kívül)*. Gyakorlatból mondom, hogy az éles projektek által diktált tempó és a tudás megszerzésére fordítandó idő hadilábon áll egymással, és igen nehéz egy fejlesztőnek vagy projektmenedzsernek a kettőt összeegyeztetni. A határidő hajtja, de a feladatot csak akkor tudja megoldani, ha közben valahogyan megtanulja az új technológiákat. De arra meg nincs ideje, ráadásul, ha egy projektben 15 embernek kell ugyanazt megtanulni, akkor mind a 15 ember rá kell szánja az időt az információk összegyűjtésére, értelmezésére.

Azt mondják, hogy az Interneten minden fenn van, és ingyen bármit meg lehet róla tanulni. Ez így igaz, ha valaki szabadidejében tanul. Azonban munkaidőben ketyeg az óra, és az ügyfél – joggal – nem akar azért fizetni, hogy a megoldásszállító vagy a saját emberei a munkaidejük felét „improduktív" tanulással töltsék. Nyilván egy embernél ez nem annyira problémás, csakhogy minden fejlesztőnek rá kell szánnia az idejét a tanulásra, és sok ember, sok pénz.

Szinte minden elérhető a web-en, ami nagy segítség, de ez azt is jelenti, hogy ki kell bányászni a hasznos információt a sok lényegtelen közül. Lehet hogy egy problémára a válasz 2 programsor vagy egy függvény nevének az ismerete, de mire megvan a függvény, három nap elment a keresgéléssel. Sajnos az a tipikus a szoftverfejlesztésben, hogy naponta jönnek ilyen egyperces problémák, majd a kétnapos keresések az információtengerben.

E dilemma forrása az információáradat strukturálatlansága. Persze mind a Microsoft, mind más cégek rendszerezik az általuk kibocsátott információt, technológiai ismereteket, de az ő csoportosításuk egyfajta logikát tükröz, amire vagy ráérez az információéhes programozó, vagy nem.

A probléma az, hogy természetes intelligenciára van szükség a keresési folyamatban, az információbányászatban, amelyet nem tudnak helyettesíteni a keresőprogramok. Lehet, hogy 50 cikk szól egy problémakörről, de abból esetleg csak 3 tartalmaz értékes információt, a többi csak ugyanazokat a tényeket ismételgeti. A lényeges információk kiválogatása és csoportosítása az, amivel minden egyes programozónak meg kell birkóznia, és el kell töltenie vele az idejét. De miért kell ugyanarra a feladatra mindenkinek időt rászánni? Hol van itt a hatékonyság?

És itt jönnek a képbe az oktatóközpontok. Az oktatók feladata az, hogy minden lehetséges forrásból összeszedjék a lényeges információkat, és kiválogassák azokat az értékeket, amelyek hatékonyan támogatják a fejlesztők, informatikai szakemberek munkáját.

A NetAcademia célkitűzésének tekinti, hogy összefogja a magyar fejlesztői társadalmat, és olyan professzionális képzésekkel, fórumokkal és szakembergárdával segítse a fejlesztők munkáját, amelyek segítségével a programozók, szoftvertervezők rövid idő alatt igen magas szinten művelhetik a munkájukat. A fejlesztők jól járnak, mert pár napos képzéssel olyan technológiákat lesznek képesek használatba venni, amelyekről addig még csak nem is hallottak. A munkáltatók örülnek, mert az alkalmazottak nem az Interneten töltik az idejüket megoldások után keresgélve, hanem a feladatra koncentrálnak, így több idejük marad a tényleges munkára. S végül a megrendelők örülhetnek legjobban, mert modern, határidőre elkészülő, jól megtervezett és hatékony alkalmazást kapnak.

Nem ígérjük, hogy a tanfolyamainkat elvégezve programozózsenit faragunk mindenkiből. Nem ígérjük, hogy a hallgatók egy-egy tanfolyamot elvégezve az MSDN használata nélkül álmukból felkeltve is fújni fogják a Common Run Time Library összes függvényét. Ígérjük viszont, hogy hallgatóink a tanfolyamok után képesek lesznek kiválasztani a feladathoz legmegfelelőbb technológiát, tudni fogják mihez nyúljanak egy-egy probléma megoldásához, és nem fognak hetekig vakvágányon haladni információhiány miatt.

A legjobbakat tanítjuk, a legjobb, legújabb dolgokra.

Soczó Zsolt MCSE, MCSD, MCDBA
Zsolt.Soczo@netacademia.net

2001 Március

tech.net

A Microsoft Magyarország Szakmai Magazinja

Szerkesztőség
Főszerkesztő: **Fóti Marcell**
marcellf@netacademia.net
Főszerkesztő-helyettes: **Fülöp Miklós**
mick@netacademia.net
Szerkesztőség címe:
1105 Budapest, Ihász utca 13.
Tel.: 263-2732
technet@netacademia.net
Nyilvános levelezési lista:
tech.net@lyris.netacademia.net

Kiadja és terjeszti
a **NetAcademia Kft.**
Terjesztési, előfizetési információ:
Tel.: 263-2732
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft

Előfizethető megrendelőlevélben a szerkesztőségnél:
1105 Budapest, Ihász utca 13.
Fax: 261-7145
http://technet.netacademia.net/subs

Hirdetésfelvétel:
Bársonykalapács Marketing
Felelős: **Udvarev Rita**
Tel./Fax: 214-0923
velvethammer@ahol.com
1027 Budapest, Fő utca 67. V. 1.
Grafikai tervezés, kivitelezés, nyomdai előkészítés:
Bársonykalapács Marketing
Művészeti vezető: **Balogh Zoltán**

Nyomda:
Cerberus Kft.
1066 Budapest
Lovag u. 14.
Felelős vezető: **Schmidt Gábor**

ISSN 1586-5185

SBS és BackOffice 2000

Megjelent a Small Business Server 2000

A Microsoft bejelentette a Small Business Server 2000 megjelenését **[1]**, amely a kisvállalatok számára készült hálózati megoldás harmadik generációs verziója. A korábbi vevők és a technológiaszolgáltatók visszajelzéseinek megfelelően a Small Business Server eszközei előre tervezhető, integrált telepítést biztosítanak, lehetővé teszik, hogy a cégek kihasználják az Internet nyújtotta lehetőségeket, erősítik a cég vevőkapcsolatait és növelik az alkalmazottak termelékenységét. A Small Business Server 2000 kimondottan az 50-nél kevesebb PC-vel rendelkező vállalatok számára készült. Tartalmazza a Windows 2000 Server-t és a .NET Server megoldások Windows 2000-es generációját *(email, fax, adatbázis-kezelés és biztonságos, megosztott Internet hozzáférés)*.

Katy Hunter, a Windows .NET Server Division termékmenedzsere szerint „a Small Business Server 2000 lehetővé teszi, hogy a kisvállalat vezetősége egyetlen termék licencelésével úgy elégítse ki a hálózati, kommunikációs, együttműködési, mobilitási és Internetes igényeket, hogy a kifizetett árért igen nagy értékű ellenszolgáltatást kap. Ez a termék a vezetőség rendelkezésére bocsátja azokat az eszközöket, amelyekkel kihasználhatók az Internet lehetőségei, hogy így a cég jobban kiszolgálhassa a vevőket."

A Small Business Server a következő összetevőkből áll **[2]**:

- Windows 2000 Server, Standard Edition, Service Pack 1-gyel
- Exchange 2000 Server, Standard Edition
- Microsoft Internet Security and Acceleration Server 2000, Standard
- Microsoft SQL Server 2000, Standard Edition
- Megosztott faxszolgáltatás
- Megosztott modemszolgáltatás
- Microsoft FrontPage 2000
- Windows Terminal Services

Itt a BackOffice Server 2000 is!

[3] A BackOffice Server 2000 a Microsoft Windows® 2000 operációs rendszer legfontosabb kiszolgálócsomagjának új verziója. A BackOffice Server-rel kevesebb költséggel és egyszerűbben hozhatók létre, telepíthetők és kezelhetők a vállalati osztályok és a közepes méretű cégek integrált és méretezhető IT megoldásai. Ez a verzió integrálja a Windows 2000 Server-t és az összetevő termékek legfrissebb Standard verzióit, az infrastrukturális és alkalmazásszolgáltatások széles körét biztosítja, így címtárat, hálózati szolgáltatásokat, Web-alkalmazásokat, adatbázisokat, üzenetkezelést és együttműködést, Internet proxy-t és tűzfalat, host integrációt.

A BackOffice Server 2000 a következő összetevőkből áll: Microsoft Windows 2000 Server, Service Pack 1-gyel

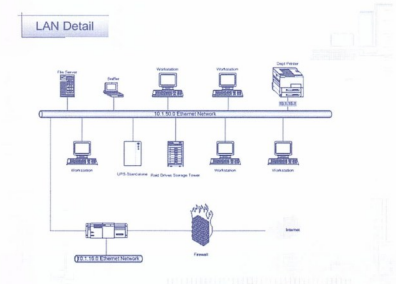
- Microsoft SQL Server 2000, Standard Edition
- Microsoft Exchange 2000 Server
- Microsoft Internet Security and Acceleration Server 2000
- Microsoft Systems Management Server 2.0, Service Pack 2-vel
- Microsoft Host Integration Server 2000
- A Microsoft FrontPage 2000 SR-1 egyfelhasználós verziója
- Egyéb tulajdonságok: intelligens, a helyzethez idomuló telepítő, MultiServer Planning Wizard, BackOffice Server Management Consoles és az ahhoz tartozó Varázslók, Health Monitor 2.1 és Server Status Views and Reports.

Visio 2002 előzetes

Mit kapunk a Visio 2002-vel?

A Visio 2002 béta kipróbálásakor a következő újdonságokkal találkozhatunk:

- Élesebb diagramok. A Visio 2002 szebb grafikát, szöveget és színeket biztosít, valamint lehetővé teszi, hogy egyéb forrásból származó képeket importáljunk és szerkesszünk.
- Microsoft Office család alkalmazás. A Visio 2002 segít abban, hogy a közös funkciókkal és az együttműködési tulajdonságokkal hatékonyabban dolgozhassunk a közismert Office felületen.
- Szoros webintegráció. Az új online erőforrásokkal a felhasználók jobban kihasználhatják a Visio 2002-t. Szebb és hatékonyabb diagrammokat is közzétehetünk a weben.
- Továbbfejlesztett telepítés és karbantartás. Az új továbbfejlesztett tulajdonságok leegyszerűsítik a Visio 2002 telepítését és karbantartását a vállalatban.
- Nagyobb termelékenység. A Visio 2002 továbbfejlesztett tulajdonságlistázást és adatbázis kapcsolatot, hatékonyabb keresési lehetőségeket és használhatóbb munkakörnyezetet nyújt.
- Testreszabhatóság. A fejlesztők a COM add-in-ek, az új XML fájlformátum és a több mint 90 új automatizálási módszer segítségével rugalmasabban hozhatnak létre egyedi Visio alkalmazásokat.

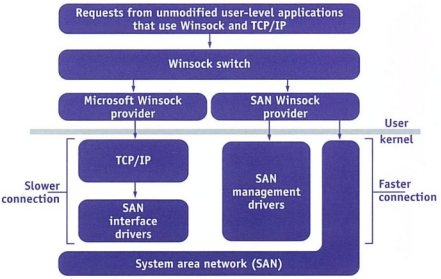


Egy hálózati diagram a Visio 2002-ben. További ábrák a Visio 2002 galériában láthatók [4]

Érdemes ellátogatni a Visio 2002 webhelyére **[5]**, ahol a béta meg is rendelhető.

Hamarosan itt a Windows 2000 Service Pack 2

A hivatalos források szerint 2001 első negyedévében, azaz még márciusban megjelenik a Windows 2000 Service Pack 2. Ez azért is valószínű, mert a weben egyre több helyen található meg az SP2-ben javított hibák listája, egy példa: **[6]** Azt már előre lehet tudni, hogy a Service Pack 2 telepítésével immár minden Windows a 128 bites biztonsági alrendszert használja majd, és nem is lesz lehetőség visszatérni az 56 bites változathoz. Az SP2 CD az SP1-hez hasonlóan tartalmazza majd a külön már régebben letölthető **[7]** Terminal Services Advanced Client-et *(TSAC)*, ami a webes és MMC-be illesztett terminálügyfél használatát teszi lehetővé. Egyes madarak azt csicsergik, hogy az SP2 jóvoltából az eddig csak a Datacenter Server-ben található Winsock Direct **[8]** beköltözik az Advanced Server-be is. Az SP2 béta méretei kicsit ijesztőek: a telepített SP2 Professional-on 180, Server-en 235 megabájtott foglal, ha a visszautat is szeretnénk meghagyni magunknak *(uninstall)*, az további 200, illetve 260 megabájtunkba kerül. A telepítés közben átmenetileg további 190/250 megabájtnyi helyre van szükség



WinSock Direct: SP2 után Advanced Server-ben is?

A cikkben található URL-ek:

[1] http://www.microsoft.com/presspass/features/2001/feb01/feb21sbs.asp
[2] http://www.microsoft.com/sbserver/
[3] http://www.microsoft.com/backofficeserver/productinfo/overview.htm
[4] http://www.microsoft.com/presspass/events/visio2002/gallery.asp
[5] http://www.microsoft.com/office/visio/2002beta.htm
[6] http://www.activewin.com/win2000/sp2bugs.shtml
[7] http://www.microsoft.com/windows2000/downloads/recommended/TSAC/default.asp
[8] http://www.microsoft.com/WINDOWS2000/en/datacenter/help/WSD_Def.htm

Microsoft Network Monitor *(IV. rész)*

E havi NetMonozásunkat nem a NetMonnal a kézben kezdjük. A TCP csatorna jellegzetességeinek megfigyeléséhez előbb át kell esnünk egy kis elméleten. A múltkori részben többször elénk került *(az IP fejlécben, az Ethernet keretben, az ICMP Echoban)* a csomagok integritását végző ellenőrzőösszeg, a checksum. Nem igazán fejtettem ki a védelem mibenlétét, de nem is lett volna miről írni: a checksumkezelés igen primitív algoritmus alapján zajlik. A hálózati forgalmat generáló és irányító eszközökön minden egyes beérkezett csomagon kiszámítódik az ellenőrzőösszeg. Ha a kiszámított érték megegyezik azzal, ami a csomagban is olvasható, a csomag életben maradhat, ha pedig nem egyezik, vesszen a hibás, deformálódott egyed.

El lehet tűnődni azon, hogy vajon ez a hibakezelés elég intelligens-e, vajon mindig egyértelműen törölni kell-e a hibás csomagokat *(a Voice Over IP például „jobban megy" még hibás csomagokkal is, mint újraküldött hibátlanokkal, mert a csomagismétlés késleltetése sokkal zavaróbb a beszédértésben, mint holmi bithibák)*.

Maradjunk annyiban, hogy az alapértelmezett checksumkezelés elég drasztikus módon ugyan, de eltávolítja a hibás csomagokat a hálózatról.
Vizsgáljuk meg, mekkora az a maximális hiba, amit a checksum még elvisel. Ez az összegszámító algoritmustól függ. Az Ethernet kártyák által használt CRC *(Cyclic Redundance Code)* például akár egyetlenegy bit átbillenését is észleli.

- Ha átbillen egy bit – elromlik az ellenőrzőösszeg
- Ha elromlik az ellenőrzőösszeg – eldobódik a csomag
- Ha eldobódik a csomag – elvész az átvinni kívánt adatmennyiségből egy jókora *(1514 bájt)* darab.

Ki, és hogyan fogja ezt észlelni, korrigálni, pótolni? Az IP? Ugyan! Azt már a Tracert esetében is láttuk, hogy ha kicsi TTL-lel indítottuk útjára, elhullott, mint szivacsos agyhártyagyulladásban szenvedő nagy négylábú barom. Bithiba esetén sem lesz jobb a helyzet – elhullik.
A kieső csomagok pótlására akár maga a hálózati szolgáltatást igénybevevő alkalmazás is vállalkozhatna – ha tudatában lenne ennek a nehéz feladatnak. De vegyünk egy Excelt. Annak bizony egyre megy, hogy a FONTOS.XLS-t vajon helyi lemezre, vagy hálózati kiszolgálóra mentjük! A második jelöltünk e feladat ellátására az operációs rendszer *(pl. Windows 2000)* lehetne, ami részben igaz is: az operációs rendszer egyik meghajtója, a tcpip.sys felel a csomagelvesztések menet közbeni korrekciójáért, a hiányzó „alkatrész" pótlásáért. Ennek érdekében a feladó oldalán a tcpip.sys minden egyes csomagot besorszámoz, hogy a fogadó oldalán csücsülő tcpip.sys a sorszámok alapján pontosan észlelhesse, ha kimarad egy-egy csomag. Az eredményességről vagy eredménytelenségről pedig rendszeres visszajelzést küld a feladónak *(ACK, acknowledgement)*.

A TCP csatorna

Mit hívunk TCP csatornának? Csak nem ezt a teljesen primitív sorszámozós rendszert? De igen. És bármilyen furcsa lehet, a maga „bonyolultságában" ez a rendszer meglepően jó hatásfokkal működik, s messze többet tud, mint csomagelveszítéseket korrigálni. Észreveszi ugyanis a csomagsorrend cserét is, valamint a csomagok megduplázódása sem marad rejtve.

Kérdés, melyre a tech.net@lyris.netacademia.net című levlistán várom a válaszokat: hogyan képzelhető el csomagduplázódás? Talán dadog valamelyik router?

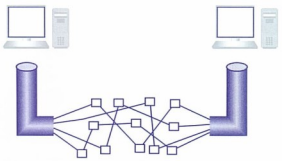
A TCP csatornát úgy célszerű elképzelnünk, mint egy csőpostát, amely két számítógépet úgy köt össze, hogy amit a cső egyik végén bedobok, az a másik végén épen, egyben bukkan fel.

Egy TCP csatorna „idealista" ábrázolása

A cső feladata a szállítás garantálása, illetve annak jelentése, hogy ha a szállítás nem valósítható meg. Sokszor mondják, hogy a TCP/IP katonai rendszer. Ezt a badar tévhitet a TCP csatorna azon viselkedése is erősíti, hogy ha romlik a vonalak minősége, akkor a cső lelassul *(iszonyatosan le tud lassulni!)*, de NEM adja fel! Mintha bizony államtitkot kellene eljuttatnia a végpontra! Addig kínlódik, új utakat keres *(nek helyette a routerek)*, ismételget, amíg van benne szusz. Ha pedig minden lehetőség kimerült, akkor jelenti alásan, hogy a haditervet nem sikerült teljesítenie. Olyan nincs, hogy a TCP félrevezetne minket, és úgy tenne, mintha minden rendben menne.

Itt most megint elfilozofálhatnánk azon, hogy vajon minden esetben érdemes-e bármi áron, vérző fejjel leszállítani az üzenetet, vagy értelmesebb lenne talán feladni a próbálkozást, ha az átbocsátóképesség X bit/s alá csökken. QoS, Quality of Service!

Miből áll tehát a TCP csatorna? Ezernyi sorszámozott csomagból, melyek oda-vissza szaladgálnak a kommunikáló felek között. Némelyikben adat van, némelyikben csak egy ACK, és mindegyikük *(legalábbis Etherneten)* maximum 1514 bájt nagyságú. Egyedi csomagok ezrei...
Vajon azonos útvonalon haladnak az Interneten át? Nyilván nem. Helyesebben: akár azonos útvonalon is haladhatnak, de ezt a közbenső útválasztók és vonalak pillanatnyi „hangulata" erőteljesen befolyásolja. Az előbbi, idealista ábránd helyett a valóságot sokkal hűségesebben adja vissza az alábbi ábra:



Egy TCP csatorna „valósághű" ábrázolása

Ha még azt is figyelembe vesszük, hogy egy gépen a nyitott TCP csatornák száma nem 1, hanem akármennyi, akkor az ábrán úgy néznének ki a gépek, mint egy-egy polip: egyik karja a www.netacademia.net-re tekeredik, másik a www.microsoft.com-ra, harmadik a tartományvezérlőre stb.

Portok

Ha egy számítógépnek ennyi karja lehet, akkor vajon miért nem zavarodik bele abba, melyik karja merre nyúlik? Vegyük először egy internetező-böngésző felhasználó esetét. A célgép IP címe vajon segít a sok-sok csatorna egyedi azonosításában? Csak addig segít, amíg egy címre csak egyetlenegy TCP kapcsolatot *(csatornát)* nyitunk. Ez azonban nem mindig van így, hisz senki sem tilthatja meg nekünk, hogy egy weblapra öt böngészőt nyissunk! Vagy vegyük egy webkiszolgáló esetét. Egyszerre mondjuk tízezer kapcsolatot szolgál ki. Honnan tudja, hogy melyik csőbe mit kell dobni? A feladók IP címe segít? A válasz ugyanaz: ha elindítok tizennyolc böngészőt egyazon gépről ugyanarra a webkiszolgálóra, akkor e tizennyolc csatornát hogyan fogjuk tudni megkülönböztetni egymástól?

Még egy eszmefuttatás: ha egy Internetre kihelyezett gépen nemcsak webkiszolgáló fut, hanem FTP, Telnet, POP3 és még ki tudja mi minden, vajon hogyan címzem meg az egyes alkalmazásokat? Hogy van az, hogy a böngésző mindig a webkiszolgálót szólítja meg? Mindig? Tévedés! Tessék így kezdeni a begépelt URL-t: ftp://www.netacademia.net.

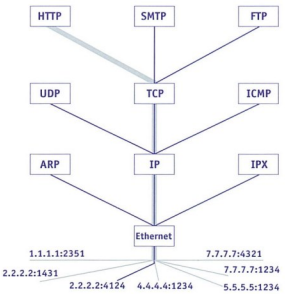
Mi az ördög? Ftpzik? Ftpzik!

A csatornát általában nem tudjuk a végpontjainak IP címével egyértelműen megadni. Szükség van még egy azonosítóra, amely segít megkülönböztetni az azonos gépről érkező kapcsolatokat egymástól, segít elválasztani a HTTP kérést az SMTP-től, segít egy távoli gépen „eltalálni" a sok-sok futó szolgáltatás közül pont azt, amire vágyunk: ez a portszám.

1. Villámkérdés: Hányas porton megy a webszerver?
2. Villámkérdés: Hányas porton megy az Internet Explorer?

1. villámválasz

Az első kérdésre mindenki fújja a választ: 80-as. Így igaz. De miért pont a 80-ason találjuk a webkiszolgálókat? A 80-as úgynevezett well-known *(jól ismert, közismert)* port, és azért pont nyolcvan, hogy a világ összes publikus számítógépén mindig megtaláljuk a webszolgáltatást. Nem kell gondolkodni, paraméterezni és programozni: ha egy gépen a nyolcvanas számú ajtón *(porton)* megyünk be, akkor ideális, szétnem-konfigurált esetben a webkiszolálón vagyunk.

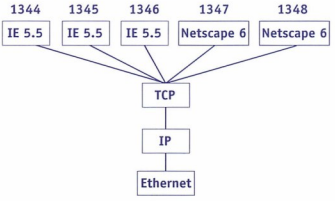


Portok, kapcsolatok

Hány kapcsolat építhető egy portra? A fenti ábra tanúsága szerint egészen pontosan X. Az egyetlen követelmény, hogy a kapcsolatok megkülönböztethetők legyenek. Mivel ezen az ábrán *(szinte)* mindegyik ugyanarra a portra fut be, ezért a szerveroldali végük alapján nem különböztethetők meg – így nyilván az alsó, „rojtos" végük különbözik. Különböző IP címekről jönnek, vagy ha azonos címről, akkor külön portról. A 80-as port egyébként nem szentírás, a legtöbb szolgáltatás átállítható, más portra helyezhető. De ezzel el is veszítjük, ha csak valaki meg nem súgja a helyes portszámot. Ha valaki pl. a 8080-as portra tesz webszolgáltatást, akkor azt így kell becserkészni: http://www.matav.hu:8080. Ez a módja annak, hogy a böngészőnk más porton kopogtasson.

2. villámválasz

A második kérdésre is sokan rávágják: 80-as :-O No ez nem igaz! Hisz ha nyitok két Explorert pont ugyanarra a webkiszolgálóra, sőt, ugyanarra a lapra, ezt a két kapcsolatot is meg kell tudnom különböztetni, hisz az egyik lapon teljesen másra kattinthatok, mint a másikon, s a két böngészőm ebbe nem zavarodhat bele! A böngészők és egyéb ügyfélszoftverek úgynevezett kliensporton „ülnek", aminek száma egyedi, és 1024-nél nagyobb. Az alábbi ábrán egy olyan gépet látunk, amelyen egyszerre X darab IE, és Y darab Netscape Navigator fut párhuzamosan, békességben.



„Ötszáz" böngésző egy gépen: egyedi portszámokon futnak!

Kijelentésem igazságát a

```
netstat -p TCP
```

utasítás begépelésével ellenőrizhetik bátrabb olvasóim.

Tűzfalak, proxyk

Az eddigi ismeretekkel felvértezve megkísérelhetjük megérteni, mit is jelent az, amikor a tűzfal 80-as portja nyitva van. A tűzfalakon átmenő forgalomnak ugyanis meghatározott iránya van: lehet befelé, és kifelé igyekvő csomagunk. Ezen túl érdemes megfigyelnünk az így áthaladó csomag feladó és fogadó portszámát. Csak a 80-as porti forgalomnak négy változata van!

		Feladó portja	Fogadó portja	A forgalom típusa
1	Befelé	80	kliensport	weblapkérésre külső kiszolgáló válasza
2	Kifelé	80	kliensport	a belső webkiszolgáló válasza
3	Befelé	kliensport	80	a belső webkiszolgálót kívülről zaklatják
4	Kifelé	kliensport	80	weblapkérés bentről, kinti címre

Ebből a négyből azok a tűzfalak, amelyek kizárólag a kifelé irányuló webböngészést engedik, csak az első és a negyedik csomagtípust eresztik át, sőt, az okosabbak csak olyan elsőt *(külső választ)*, amely válasz egy negyedikre *(weblapkérés)*. Ez a tűzfaltípus tehát a rajta átívelő TCP csatornák kinyitására és elzárására képes. Ha a csatornát engedélyezi, akkor visszavonul, és az átmenő forgalmat már nem ellenőrzi. Jöhetnek a vírusok!

Egy jobb tűzfal nemcsak nyit/zár, hanem megpróbálja az átmenő forgalmat értelmezni, szűrni, hogy a nyitott csatornákon át ne lehessen mást forgalmazni, mint ami oda „való": a 80-ason csak HTTP, a 25-ön kizárólag SMTP mehessen stb. Az ilyen típusú tűzfalakat más néven proxynak is hívják. Ilyenkor a kliens nem a végcéllal épít ki TCP csatornát, hanem a tűzfallal. Az minden csomagot kiszed a csőből, és ha értelmesnek találja, továbbküldi kifelé, és egy, a kliens helyett *(proxy=helyettesítő)* általa felépített TCP csatornába dobálja bele az adatokat. Ilyenkor a feladó és a fogadó valójában sohasem találkozik közvetlenül egymással.

Szekvenciaszámok, háromutas kézfogás

Az utolsó kérdés a TCP sorszámozásának megvizsgálása. Említettem, hogy a TCP minden egyes csomagot besorszámoz, mielőtt elküldené. Nem említettem, de sejthető, hogy a visszirányú hálózati forgalom is sorszámozott: minden ACK egyedi sorszámmal bír. Mielőtt azonban a felek megkezdhetnék a kommunikációt, meg kell ismerniük egymás kezdősorszámát. Nem, nem egytől indulnak, hanem random számokat használnak – ennek biztonságtechnikai okai vannak. Azt a folyamatot, amikor a partnerek kicserélik egymás szekvenciaszámát, kézfogásnak hívjuk. Én inkább csip-csip csókának hívnám, mert olyan ez a három csomag, mintha egymás kezére tennék egymás kezét, de sajnos a W3C nem fogadta el e terminológiai javaslatomat :(Az angol elnevezés továbbra is three-way handshake, pedig a chip-chip-choka mennyivel találóbb lenne! Körülbelül így zajlik a folyamat:

1. A feladó kapcsolatot kezdeményez a fogadógéppel a megcélzott *(mondjuk 80-as)* porton, és elküldi saját kezdősorszámát. Ezt a csomagot világosan meg lehet különböztetni a Network Monitorban a benne bebillentett „S" flagről. *(S, SYN=Synchronize Sequence Numbers)*. Így fog kinézni a Network Monitorban:

   ....S.

2. A fogadó *(ha van a megszólított porton szolgáltatás)* pozitív választ küld, egy ACK formájában. Az ACK mindig azt mutatja, hogy a partner melyik szekvenciaszámtól folytathatja az adást. Így ez az ACK az előbb megkapott szám plusz egyet *(S+1)* fog tartalmazni. Jelentése: oké, innentől jöhet! Továbbá – sávtakarékossági célból még ugyanebben a csomagban – ő is elküldi saját kezdő szekvenciaszámát, amivel pedig a tőle kiinduló csomagokat sorszámozni fogja:

   .A..S.

3. Végül az eredeti kezdeményező küld egy ACK-ot, s ezzel a csatorna felépült.

   .A....

E három csomag után következhet az oly fontos adatok átvitele. Miután az összes adat átment, a csatorna bontása szintén csip-csip csóka módszerrel történik. Ekkor nem az „S", hanem az „F" flag segít, melynek jelentése: No more data from sender. A lebontás így néz ki:

.....F

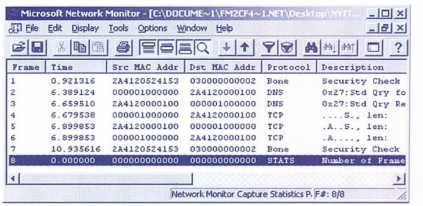
.A...F

.A.....

Ezzel a felek illendőképpen elbúcsúztak egymástól, ásó, kapa, nagyharang. Lássuk mindezt élőben!

Nyitott 80-as port

Az első NetMon fájl, amit mutatok, a nyitott80.cap nevet viseli, és helye szokásosan az újság weblapjának feladatok alkönyvtárában található. 1,5k csupán, érdemes letölteni, hogy vizsgálatainkat együtt folytathassuk!



A TCP csatorna felépítése

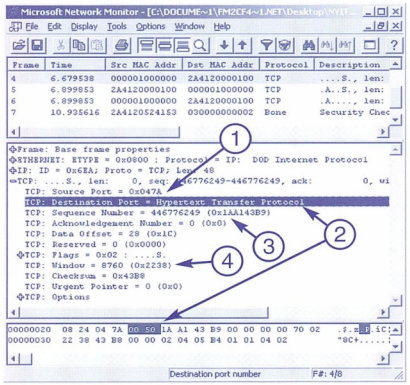
Az ábrán jól látható az előbb magyarázott háromutas kézfogás, ahol a description mezőben egymás alatt három sorban ezt látjuk:

....S.

.A..S.

.A....

Az előtte lévő két DNS sor egy Standard Query és egy Response. Ezeket ismertnek veszem csakúgy, mint a Bone protokollt. Hogy miért ismertek? Mert aki nem a negyedik résztől kezdte olvasni sorozatomat, annak ez már a könyökén jön ki. Nyissuk meg a ....S. csomagot, lássuk mi minden van benne!



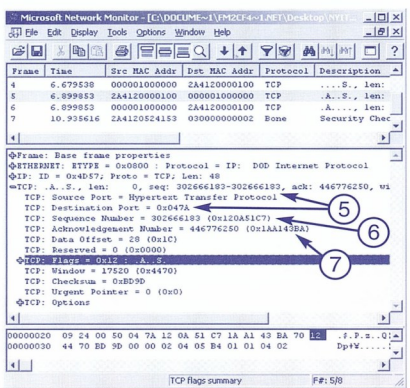
Synchronize Sequence Numbers

Ethernetben IP-ben TCP. A feladó portszáma ①, a 0x047A egy kliensport, hisz ez a szám nagyobb, mint 1024. A célgépen pedig a sokat emlegetett 80-as portot, vagyis egy webkiszolgálót céloztunk meg. Ezt a portot a NetMon is ismeri *(nem hiába well-known)*, és oda is írja szépen a protokoll nevét ②, míg a hexa panelen egy 0x50(=80!) látványának örvendezhetünk. A szekvenciaszámról lesír, hogy random érték ③, az ACK pedig azért nulla, mert ez a legelső csomag ezen csatorna életében, nem tartunk még a visszajelzéseknél.

A flag bitjei a következők lehetnek:

..0..... = Urgent data
...0.... = Acknowledgement field significant
....0... = Push function
.....0.. = Reset
......0. = Synchronize sequence numbers
.......0 = No more data from sender

A TCP-nek is van ellenőrzőösszege, ahogy az eddigi összes rétegnek volt. A Window ④ érték – ha nagyon pongyolán szeretnék fogalmazni – annyi jelent, hogy 8760 bájt küldhető át a hálón visszajelzés, ACK megvárása nélkül. Lássuk a választ!



Synchronize Sequence Numbers válasz

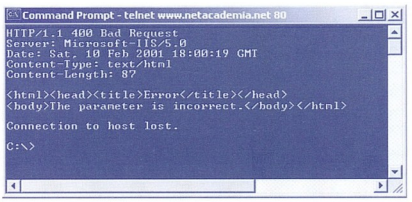
A portoknál megfigyelhető, hogy az előbbi feladóból fogadó lett, s a fogadóból feladó ⑤: a webkiszolgáló válasza a 80-as portról jön, és a kliensportra megy. A válaszoló webgép is generál egy szekvenciaszámot, amivel a tőle kimenő csomagokat majd sorszámozni fogja ⑥. Érdekes összehasonlítani az itteni ACK számát ⑦ az előző csomag szekvenciaszámával: hát nem pont egyel több? S mit várhatunk a harmadik csomagtól? A feladó ismét a kliens, s „leackolja" az itteni szekvenciaszámot, így ott ez olvasható:

```
Acknowledgement Number = 302666184 (0x120A51C8)
```

Vajon hogyan fordulhatott elő, hogy e három csomag után végetért a móka, és nem hullott a kliensre egy Default.HTM? Sőt, semmi sem hullott! Ennek az az oka, hogy a fenti forgalmat nem Explorerrel generáltam, hanem telnettel: rácsatlakoztam a távoli gép 80-as portjára, így:

```
Telnet www.netacademia.net 80
```

Mivel a telnet.exe nem kifejezetten webböngésző, nem küld be a világon semmit a kész csatornába, így az idővel lebomlik. A bomlás virága az alábbi DOS ablakba böffintett HTML hibaüzenet.

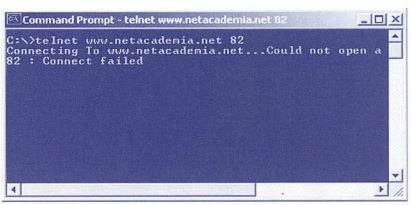


„Betelnetelek" a 80-as porton

Valójában a telnet.exe igen jó TCP segédeszköz, ugyanis nem csinál mást, mint felépít egy TCP csatornát kliensportról egy távoli gép tetszőleges portjára, majd nem szól bele egy mukkot sem, teljesen ránk bízza a csatorna adattal történő ellátását.

Zárt ajtók

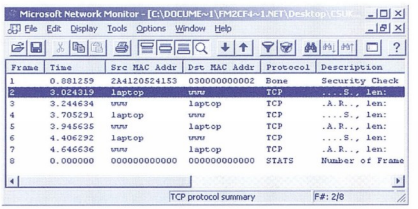
Most vizsgáljuk meg, hogyan néz ki az a hálózati forgalom, amikor olyan portra próbálunk meg csatlakozni, amely nincs nyitva.



„Betelnetelek" a 82-es, zárt porton

Azt hihetnénk, hogy a zárt port meg sem mukkan, de nem így van, mert ha válaszra sem méltatna minket, akkor hálózati hibára gyanakodva még X-szer megkísérelnénk a kapcsolatfelvételt. Emiatt tehát – az udvariasság jegyében – el-

várható, hogy ha bekopogunk mondjuk a 82-es porton, akkor – mint Nyuszi a Micimackóban – visszaválaszolnak, hogy „nincs itthon senki". Ebből tudhatjuk, hogy tényleg üres a ház – ennek dacára három csatlakozási kísérlet történik, amiknek az az oka, hogy sajnos nem derül ki egyértelműen a kopogtató számára, hogy mi az elutasítás oka. Az egy és osztható Reset Connection üzenet egyaránt jelenthet csukott portot és szoftveres elutasítást *(például ha nincs reverse bejegyzés az IP címünkhöz)*. Ennek hálózati forgalma a weben a *csukott82.cap* fájlban található, s így fest:



Reset connection

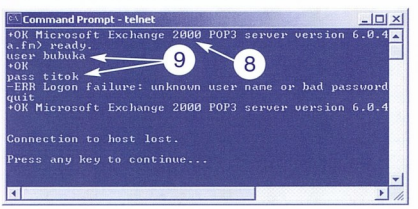
A szokásos S kérésre ebben az esetben R *(Reset connection)* a válasz, amit ezután nem is követ harmadik csomag. Tekintettel arra, hogy az elutasítás egyetlen bittel történt, sajnos indoklást nem találunk a csomagban.

....S.
.A.R..

POP3

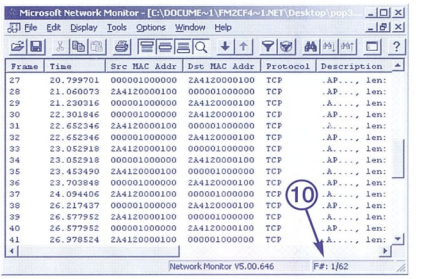
Most próbáljuk ki egy valódi alkalmazás működését Telnet segítségével! Próbáljuk meg elolvasni leveleinket POP3 protokollal, ám ügyfélprogram nélkül, puszta kézzel! *(Nem fogom végigvezetni, csak a bejelentkezésig vizsgáljuk meg a hálózati forgalmat. A fájl neve: pop3.cap)*

```
Telnet mail.netacademia.net 110
```



POP3 , puszta kézzel

A létrejövő TCP csatornában a kiszolgáló elküldi üdvözlő üzenetét ⑧, majd várakozó álláspontra helyezkedik, parancsra vár. A decemberi számunkban megjelent POP3 szabványismertető alapján haladva először megadjuk a felhasználó nevét majd jelszavát ⑨. Ilyen felhasználó sajnos nincsen, de sebaj. Most nézzük a NetMont!

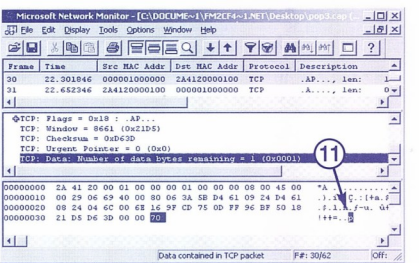


A pop3.cap fájl

Te jó ég! Ez a két parancs 62 csomagot generált ⑩! Hiszen összesen 3-4 kérdés és válasz történt! Mi lehet ez?

Terminálemuláció

Eddig nem került szóba a telnet.exe valódi funkciója, származása. Természetesen tud TCP csatornát nyitni, de ezt azért teszi, hogy utána a TCP csatornát pontosan ugyanarra a célra lehessen használni, mint a régi terminálok „hálózati" kapcsolatát, a soros portot – vagyis karakterek átvitelére! A Telnet.exe úgy emulálja a terminálfunkciót, mintha a TCP csatorna egy vacak RS-232 kábel lenne. Vagyis, ha a felhasználó leüt egy karaktert, az már mehet is át a „kábelen". Igen ám, de itt egy virtuális „kábelről" van szó, melynek csatornajellegét az ide-oda küldött ACK-ok tartják fenn. Így a karakterenkénti átküldés eléggé sajátságos sávszélesség-gazdálkodást eredményez: minden 55 bájtos TCP csomagban EGYETLEN bájtnyi hasznos adat utazik!



Number of data bytes remaining

Csak nézzük meg szúrópróbaszerűen a Number of data bytes remaining-et mondjuk a 30. keretben! Egy nyamvadt „p" betű a hasznos adat ⑪! A többiben pedig rendre „a", „s", „s", „ ", „t", „i", „t", „o", „k". Azaz „pass titok". Ejnye. Titkosítatlanul megy át a hálózaton a jelszavunk? Igen, nagyon sok alapalkalmazásnál ez így történik, ami ellen a csatorna titkosításával védekezhetünk – de az SSL egy másik, hosszú történet. Ami pedig a sávszélesség okos ihasználását illeti – a telnet ebben nem jeleskedik.

7 bites az Internet?

A fenti kísérlet rávilágít egy érdekes dologra. Mint ahogy a TCP/IP nem katonai rendszer, úgy az a közvélekedés is színtiszta badarság, hogy az Internet bizonyos zugai 7 bitesek lennének. Mitől terjedt el ez a tévhit? Valami igazságalapja kell, hogy legyen! Van is. Bizonyos alkalmazásprotokollok nem binárisan kódolt üzeneteket váltanak, hanem egyszerű szöveges parancsokkal kommunikálnak. A POP3 is jó példa erre, azonban az SMTP tökéletes. Ha ugyanis betelnetelnénk egy SMTP kiszolgálóba a 25-ös porton, akkor *(nagy vonalakban)* így tudnánk puszta kézzel levelet küldeni:

```
->HELO mail.netacademia.net
<-OK
->MAIL FROM: marcellf@netacademia.net
<-OK
->RCPT TO: akarki@sehol.com
<-OK
->DATA
…és jöhet a levél.
```

A fenti utasításokkal lehet levelet küldeni mindenféle Outlook és egyéb úri huncutság nélkül. Mivel az SMTP a soremelést veszi a parancs végének, ennek a „karakternek" különleges szerepe van mind az SMTP, mind a POP3, IMAP4 kommunikációban – és még sok más protokollban. Tulajdonképpen minden „különleges" karakternek „különleges" szerepe van, mivel a fejlesztők nem igazán gondoltak sem a magyar nyelvre, sem a később elterjedt csatolásküldésre *(virus.exe)*. Mivel a régi SMTP-t kidobni nem lehetett, a szabványfejlesztők érdekes módon vágták át a gordiuszi csomót. Minden olyan adat, ami nem írható le az angol ABC kis- és nagybetűivel – LEGYEN leírható az angol ABC kis-, és nagybetűivel!! Bölcs döntés nemde? Ezt hívják kompatibilitásnak. S lőn UUENCODE és MIME kódolás, melyeknek pont az a feladatuk, hogy a virus.exe csatolt fájlt enterekkel zárt, fix hosszúságú sorokból álló szöveges alakra hozzák, hogy az SMTP ne boruljon fel tőle! Hoppá! Ipartörténeti érdekesség!

Mire nem jó a TCP csatorna?

Végezetül essen szó arról is, hogy a TCP csatorna nem mindenható, egyáltalán nem minden esetben érdemes használni, sőt olyan forgalomtípusok is léteznek, melyeket képtelenség csatornába zárni. Lássuk a két esetet:

1. Nem érdemes TCP csatornát használni viszonylag kevés adat átvitele esetén, amikor többe kerülne a leves, mint a hús. Gondoljunk bele: a TCP csatorna felépítése 3 csomagot igényel, s a bontás sem „olcsóbb". Ez a 6 keret az ára a megbízhatóságnak. De mi van akkor, ha csak egy-két csomagom van, melyek akár el is veszhetnek? Vagy még ha nem is veszhetnek el, az egyszerű megismétlés bőségesen elég? Gondoljunk a DNS-re: egy kérdés – egy válasz. Nincs szükség tördelésre, szekvenciaszámokra stb. Ha TCP csatornában küldenénk az egyszerű DNS lekérdezéseket, két csomag helyett nyolcat kellene használni!
2. Nem lehet TCP csatornát használni olyankor, amikor kettőnél több kommunikáló fél van, mert a TCP csatornának csak két vége lehet. Azaz nem csatornázunk Broadcast esetén, illetve multicast adatfolyam átvitelekor sem.

Ha nem jó, vagy nem kell a TCP, akkor használunk UDP-t *(User Datagram Protocol)*, ami tulajdonképpen a TCP „light" verziója. Portszámhasználat ilyenkor is van, ám nincs szekvenciaszám, ACK és chip-chip-choka. Viszont mivel pontosan ugyanazon a programozói felületen keresztül érhető el *(WinSock)*, rendkívül kényelmesen használható.

Fóti Marcell
Marcellf@netacademia.net

Az MMC testreszabása

Sok cikkben olvasható az a megállapítás, hogy a Windows 2000 Server verziójának talán egyik leghasznosabb és legizgalmasabb újítása az Active Directory. A Windows NT 4.0 címtárával *(NTDS – Windows NT Directory Service)* összehasonlítva az Active Directory számos újdonságot tartalmaz. Oldalakon keresztül sorolhatnánk a példákat, de talán elegendő a két legalapvetőbb eltérést megemlíteni: a Windows 2000 címtára objektumokból áll, s ezek hierarchikus tárolórendszerben *(szervezeti egységek, Organizational Units)* helyezhetők el. Az operációs rendszer objektumként kezeli például a felhasználói azonosítókat, nyomtatókat, munkaállomásokat stb. Minden objektum számos tulajdonsággal rendelkezik, s minden objektum rá jellemző tulajdonsághalmazzal bír. A felhasználói azonosító tulajdonságai közé tartozik például a vezetéknév, keresztnév, elektronikus levelezési cím stb. Mivel a felhasználóknak lehetőségük van tulajdonság alapján objektumokat megkeresni, nem szükséges megtanulniuk a hálózati erőforrások sokszor nehezen megjegyezhető nevét.

A Nagy Feladat

A rendszergazdák is találhatnak sok hasznos újítást a Windows 2000 címtárában. Nagyon sokszor előfordul, hogy többszöri téves jelszóbeírás miatt az operációs rendszer letiltja a felhasználó azonosítóját *(a felhasználó „kiüti" magát - a szerk.)*. Ilyen esetben a folyamatos munkavégzés biztosítása érdekében a rendszergazdának rövid időn belül engedélyeznie kell az azonosító használatát, különben a felhasználó vagy hazamegy, vagy elkéri egy másik kollégájának jelszavát, és azzal dolgozik tovább *(súlyosan veszélyeztetve ezzel az esetleg érvényben lévő naplózási rendszabályokat)*. Mivel a „visszabillentés" nem tartozik a legkedveltebb rendszergazdai tevékenységek közé, ezért kézenfekvő ötletnek tűnik néhány vállalkozó kedvű munkatárs megbízása ezzel a feladattal. Már a Windows NT 4.0 operációs rendszer címtára is lehetőséget teremtett számunkra, hogy az általunk kiválasztott személyeknek olyan jogosultságot adjunk, amelynek révén el tudnak látni bizonyos rendszergazdai feladatokat. A letiltott azonosító engedélyezéséhez például fel kell vennünk a szóban forgó munkatársakat az Account Operators csoportba. Ezzel azonban a kívántnál sokkal több jogosultságot nyújtottunk. Ugyanis az Account Operators csoport tagjai felhasználói azonosítókat is létre tudnak hozni, le tudnak törölni stb. Az Active Directory erre a problémára is kínál megoldást. A későbbiekben látni fogjuk, hogy az objektumok esetében széleskörűen lehet szabályozni a hozzáférési jogosultságot. Ezáltal megakadályozhatjuk, hogy egy bizonyos rendszergazdai feladattal megbízott dolgozó a kellő hozzáértés hiányában véletlenül fennakadást okozzon a számítástechnikai rendszer működésében.

Az objektumok kezelése és a jogosultsági lista

A Windows NT 4.0-ás környezetben a rendszergazdákat önálló, egymástól független programok segítették feladatuk ellátásában *(például User Manager, Server Manager stb.)*. Ezzel szemben a Windows 2000 operációs rendszerben található Microsoft Management Console *(MMC)* egységes kezelői felületet szolgáltat a rendszergazdai eszközök számára. Az MMC önmagában nem alkalmas a rendszerkörnyezet beállításainak megváltoztatására, működésmódját a megfelelő snap-inek betöltésével határozhatjuk meg. Például az úgynevezett Active Directory Users and Computers snap-in révén az MMC alkalmassá válik az Active Directoryban található számítógépes és felhasználói objektumok kezelésére. Ez azt is jelenti, hogy - megfelelő hozzáférési jogosultsággal - a felhasználói és számítógépes objektumok bármely tulajdonsága megváltoztatható.

Minden objektum önálló jogosultsági listával *(Access Control List – ACL)* rendelkezik. Ezáltal széles körben tudjuk szabályozni, hogy kinek biztosítunk hozzáférést az egyes objektumokhoz, illetve objektumcsoportokhoz. Az Active Directoryban a jogosultságok bizonyos értelemben különböznek például a fájlok esetében megszokottaktól. Az objektumok esetében ugyanis nemcsak a hozzáférési szintet tudjuk szabályozni *(teljes hozzáférés, írási, olvasási jog stb.)*, hanem ezen túlmenően az objektum egyes tulajdonságaihoz *(attribútumaihoz)* is rendelhetünk írási, illetve olvasási jogot. A Nagy Feladatban említett kizárás például a felhasználóobjektumok „LockoutTime" jellemzőjének átállítódása miatt következik be, s a „visszabillentés" is ezen attribútum felülírásából áll. Az objektumok jogosultsági listáját a következőképpen tudjuk megtekinteni. Az Active Directory Users and Computers snap-in felhasználói felületén jelöljük ki az egyik felhasználói azonosítót. Az objektum tulajdonságait az Action menü Properties parancsának segítségével tudjuk megjeleníteni. A párbeszédablakon a Security fül Advanced nyomógombja révén tekinthetjük meg részletesen, hogy az objektum ACL-jében milyen bejegyzések találhatóak. Fontos felhívni a figyelmet arra, hogy a Security fül csak akkor jelenik meg a tulajdonságok között, ha a Users and Computers snap-in View menü Advanced Features beállítását engedélyezzük.

Minden bejegyzés esetében a View/Edit nyomógombbal tudjuk megtekinteni, illetve szükség esetén megváltoztatni a felhasználók, illetve felhasználói csoportok az objektumhoz, illetve az objektum egyes tulajdonságaihoz rendelt hozzáférési jogosultságát.

Előfordul, hogy az ACL szerkesztésekor nem látható az objektum összes tulajdonsága. Jó példa erre a User objektum, amely több mint kétszázötven tulajdonsággal rendelkezik, így alapértelmezésben csak a fontosabbak jelennek meg – a „visszabillentési" jog *(lockoutTime attribútum kezelése)* sajnos alapértelmezésben nem!

Hol a jog?

Van kiút a nehéz helyzetből: minden tartományvezérlő %systemroot%\system32 könyvtárában megtalálható a Dssec.dat nevű file, amely az objektumok rejtett tulajdonságainak listáját tartalmazza. A Dssec.dat egyszerű textállo-

mány, így például a Notepad segítségével könnyen szerkeszthető. A fájlban szögletes zárójelek közé zárva találjuk az egyes objektumok nevét *(például [user])*. Ha az ezt követő sorban a „@=7" bejegyzés szerepel, akkor rejtett objektummal állunk szemben. Ez azt jelenti, hogy külön ehhez az objektumhoz egyáltalán nem tudunk jogosultságot adni. Szükség esetén úgy tudjuk az objektumot a legegyszerűbben megjeleníteni, hogy kitöröljük a „@=7" bejegyzést.
A Dssec.dat fájlban a rejtett tulajdonságok neve után is az „=7" bejegyzés áll. Csak akkor lesznek láthatóak az ACL szerkesztésekor, ha a 7-es számot megváltoztatjuk bármilyen más értékre, vagy a tulajdonság nevével együtt kitöröljük a fájlból. *(Miért pont 7 ? – a szerk.)* Ezáltal lehetőségünk nyílik, hogy ezekhez a tulajdonságokhoz külön-külön is jogosultságot adjunk.

A letiltott azonosító engedélyezésének kiosztása

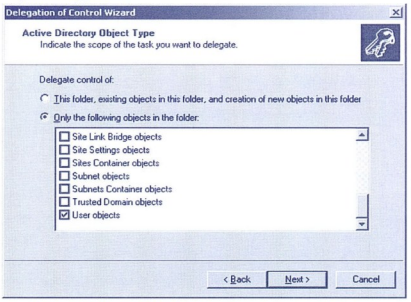
A rendszergazdai feladatok, így a letiltott azonosító engedélyezésének a kiosztása is két lépésből áll:

- Az objektumokhoz hozzáférési jogosultságot kell adnunk a felhasználónak vagy felhasználói csoportnak. A jog legyen a lehető legszűkebb, hogy a feladattal megbízott felhasználó ne tehessen kárt a címtárban.
- Ki kell alakítanunk a célfeladatnak megfelelő, korlátozott funkciókészletű felhasználói felületet. A felhasználói felület is legyen szűkre szabott, hogy a felhasználót ne zavarják meg olyan virtuális lehetőségek, amelyekkel valójában nem élhet, hisz joga úgysincs hozzá. Senki sem szereti látni az „Access denied" üzenetet!

A jogosultságot legkézenfekvőbben a Delegation of Control Wizard segítségével tudjuk kiosztani. Ezen kívül azonban lehetőségünk van arra is, hogy közvetlenül az objektum ACL-jéhez adjuk hozzá az általunk kiszemelt felhasználót vagy felhasználói csoportot. Mielőtt részletesebben is áttekintenénk a jogosultság-kiosztás mindkét módját, lehetővé kell tennünk, hogy a letiltott azonosító engedélyezéséhez a többi tulajdonságtól függetlenül is lehessen jogosultságot adni. A korábban leírtaknak megfelelően ezt úgy tudjuk megvalósítani, hogy a Dssec.dat állományból kitöröljük a „lockoutTime=7" bejegyzést vagy a 7-es számot megváltoztatjuk bármilyen más értékre.
Most nézzük meg a Delegation of Control Wizard működését. A varázsló az Active Directory Users and Computers snap-in bármely mappájának gyorsmenüjéből elérhető a Delegate Control parancs révén. Figyelembe kell vennünk azonban, hogy az így kiosztott hozzáférési jogosultság csak a szóban forgó mappa alatt található objektumokra vonatkozik. A Delegation of Control varázslóval szervezeti egység, sőt tartományi szinten is tudunk jogosultságot kiosztani. Ez utóbbi esetben például - mivel a jogok alapértelmezésben öröklődnek - a kiosztott jogosultság a tartomány összes objektumát érinti.

Figyelem! A Delegation of Control varázsló csak hozzáadni tud jogosultságokat a címtár különböző pontjaihoz, törölni és módosítani már nem képes saját korábbi outputját sem. Éppen ezért fontos a jogosultságállítás kézi módszereinek ismerete is – előbb-utóbb menthetetlenül szükségünk lesz erre a tudásra.

A Delegation of Control varázsló használatakor először meg kell adnunk azt a felhasználói azonosítót vagy csoportot, akinek, illetve amelynek jogosultságot szeretnénk adni. Ezt követően meg kell határoznunk, hogy mely objektumcsoportra vonatkozzon a hozzáférési jogosultság. Alapértelmezésben a beállításaink az adott mappában lévő összes objektumra érvényesek lesznek. A másik lehetőségünk, hogy mi jelöljük ki a megfelelő objektumcsoportot. Bármelyik megoldást választjuk is, a beállítás természetesen érinteni fogja a mappa újonnan létrehozott objektumait is. Mivel a letiltott felhasználói azonosító engedélyezéséhez szeretnénk jogot adni, ezért a Delegation of Control varázsló párbeszédablakán válasszuk ki a User objects melletti jelölőnégyzetet.



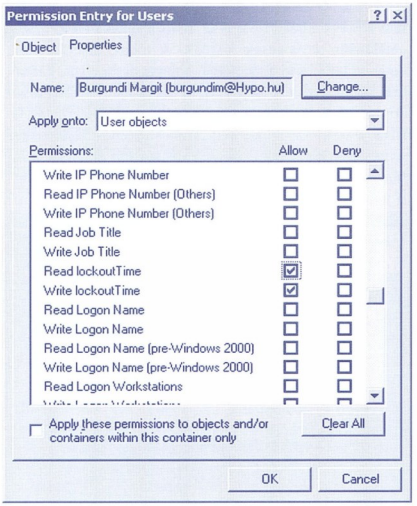
Csak a felhasználói objektumokra van szükség

Ezzel a varázsló későbbi képernyőin megjelenő jogosultságok körét leszűkítettük a felhasználókon érvényesíthető jogokra. A továbblépést követően a hozzáférési jogosultságot határozhatjuk meg. A párbeszédablakban alapértelmezésben az általános jogosultságok jelennek meg *(General jelölőnégyzet)*. Nekünk viszont az egyes piciny tulajdonságokhoz, attribútumokhoz tartozó jogosultságokra van szükségünk, amelyeket a Property-specific jelölőnégyzet kiválasztásával tudunk megjeleníteni. A felsorolt jogosultságok közül a „Read lockoutTime" és a „Write lockoutTime" bejegyzéseket kell kijelölnünk.
A Finish gombra kattintva a Delegation of Control varázsló aktualizálja a változtatásokat az ACL-ben.

Ugyanez kézzel...

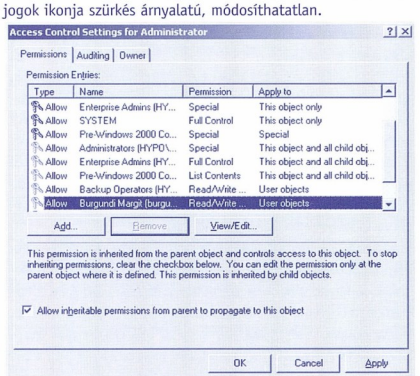
Mint korábban már volt róla szó, közvetlenül a jogosultsági listához is hozzáadhatjuk a megfelelő felhasználói azonosítót vagy csoportot. Első lépésben az Active Directory Users and Computers snap-in felületén jelenítsük meg például a Users mappa tulajdonságait *(Action menü Properties parancs)*, majd kattintsunk a Security fül alján található Advanced nyomógombra. Ekkor megjelenik a Users mappára vonatkozó jogosultsági beállítás.
A párbeszédablak Add gombjával tudjuk hozzáadni a jogosultsági listához azokat felhasználói azonosítókat, illetve csoportokat, amelyek még nem szerepelnek az ACL-ben. Az ezt követően megjelenő párbeszédablakban határozhatjuk meg az egész objektumra *(Object fül)* vagy az objektum egyes tulajdonságaira *(Properies fül)* vonatkozó jogokat. Az általunk keresett jogok a Properties fül alatt találhatóak, amelyek azonban csak azután jelennek meg, miután kijelöltük a megfelelő

objektumcsoportot. Először tehát a párbeszédablak felső harmadában található legördíthető listából *(Apply onto)* válasszuk ki a User objects-et, majd engedélyezzük a „Read lockoutTime" és a „Write lockoutTime" jogosultságokat.



A beállított jogok öröklődnek a mappa tartalmára

Fontos, hogy a párbeszédablak alján található jelölőnégyzetet *(Apply these permissions to objects and/or containers within this container only)* az alapértelmezésnek megfelelően üresen hagyjuk. Ellenkező esetben az újonnan beállított jogok csak a Users mappára lesznek érvényesek, a mappában található objektumokra nem öröklődnek. A változtatások jóváhagyása után nézzük meg valamelyik felhasználói objektum jogosultsági listáját. Az öröklődés szabályainak megfelelően ezen a szinten is meg kell jelennie a Users mappán az imént beállított jogosultságnak. Az ACL minden bejegyzésének elején kulcsokat ábrázoló kis ikon látható. Az úgynevezett szülőobjektumtól örökölt jogok ikonja szürkés árnyalatú, módosíthatatlan.



Az örökölt jogok szürkék, nem módosíthatók

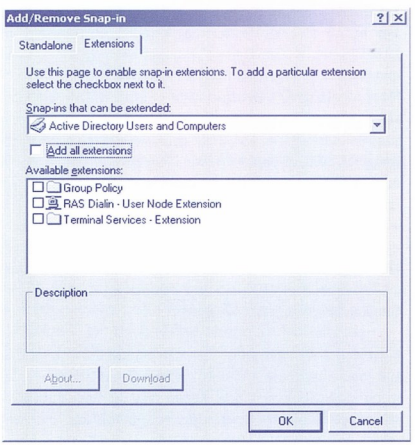
A hierarchia felsőbb szintjéről származó bejegyzések csak akkor jelennek meg, ha az ACL beállításairól tájékoztató párbeszédablak alján található jelölőnégyzet *(Allow inheritable permissions from parent to propogate to this object)* kijelölt állapotban van. Az a jogosultság tehát, amelyet a Users mappa szintjén osztottunk ki, csak azoknál a felhasználói objektumoknál NEM lesz érvényes, amelyeknél az öröklődést szabályozó jelölőnégyzet üres.

A jogosultság meghatározása után még egy feladatunk van: biztosítanunk kell a letiltott azonosítók feloldásával megbízott munkatársak számára a megfelelő*(en korlátozott)* felhasználói felületet.

A Taskpad View és az MMC testre szabása

A letiltott azonosító feloldásához szükséges felület biztosításának az a legkézenfekvőbb módja, ha a felhasználó kliensére telepítjük a Windows 2000 Server rendszergazdai eszközeit. Az Active Directory Users and Computers snap-in azonban túl komoly eszköz erre a feladatra. A nagyszámú objektum, parancs és menü megnehezítheti a felhasználói felületen való tájékozódást a számítástechnikában nem különösebben képzett munkatársak számára. Szerencsére a Microsoft Managemet Console nagyon jól testre szabható a célfeladatnak megfelelően. Ezáltal lehetővé válik, hogy csak azokat a funkciókat tegyük elérhetővé, amelyekre ténylegesen szükség van. A mi esetünkben a felhasználói felület nagyon egyszerű lesz. A felhasználói objektumokon kívül ugyanis csak a letiltott azonosító feloldására szolgáló parancs elérhetőségét kell biztosítanunk, minden más elrejthető.

Először is indítsuk el Microsoft Managemet Console-t *(mmc.exe)*, majd a Console menü Add/Remove snap-in parancsának segítségével töltsük be az Active Directory Users and Computers snap-int. Ez a snap-in három kiterjesztéssel is rendelkezik *(Group Policy, RAS Dialin, Terminal Services)*. Mivel ezekre a kiterjesztésekre a későbbiekben nem lesz szükségünk, ezért betöltésüket tiltsuk le az Extensions fül alatt az Add/Remove Snap-in párbeszédablakban.

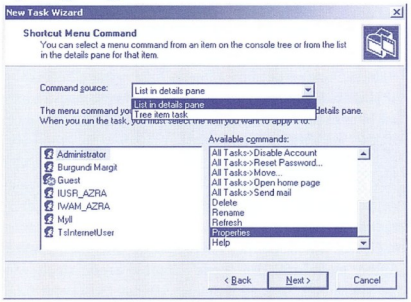


Ezekre a csingilingikre nincs szükségünk a Nagy Feladathoz

Ezt követően rejtsük el azokat az objektumokat, amelyekkel a rendszergazdai feladattal megbízott munkatársaknak nem lesz dolguk. Erre szolgál a View menü Filter Options parancsa. A megjelenő párbeszédablakban három beállítás közül választhatunk. Alapértelmezésben minden objektum látható lesz *(Show all types of objects)*. Ezen kívül lehetőségünk van arra, hogy magunk jelöljük ki azokat az objektumtípusokat amelyeket szeretnénk megjeleníteni *(Show only the following types of objects)*. Sőt egyedi lekérdezési feltételekkel is meghatározhatjuk a felhasználói felületen keresztül elérhető objektumok körét *(Create custom filter)*. Számunkra a második választási lehetőség kínál megfelelő megoldást, ezért kattintsunk Show only the following types of objects rádiógombra, és jelöljük ki a Users felirat mellett álló jelölőnégyzetet. Ha megnyomjuk az OK gombot, tapasztalni fogjuk, hogy csak a felhasználói azonosítók jelennek meg a snap-in felületén.
Mivel a mi esetünkben minden felhasználói objektum a Users mappában található, ezért érdemes az összes többi mappát elrejteni. Ezt úgy tudjuk elérni, hogy a Users mappa kijelölése után az Action menü New Window from here parancsára kattintunk. Ilyenkor egy új MMC ablak jön létre, amelyben már csak a számunkra fontos mappa található.

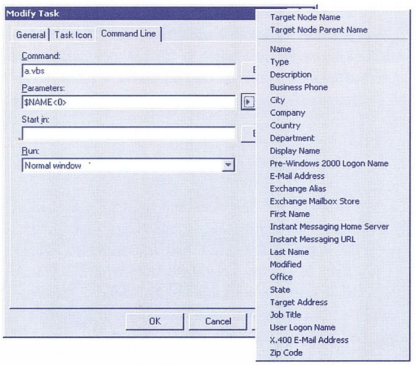
Sajnos nem minden esetben alkalmazható ilyen hatékonyan ez a beállítási lehetőség. Például ha a felhasználói azonosítók szervezeti egységekben *(OU)* helyezkednek el, és a hierarchia csúcsán több OU is található, akkor nem így tudjuk elrejteni az olyan mappákat *(például Computers)*, amelyekre nincs szükség, hanem – érdekes módon – a Read jog megvonásával. Ez azért tűnhet szokatlannak, mert szemben az NTFS-sel, ahol a Read jog megvonása nem tünteti ez előlünk a fájlokat és könyvtárakat, csak nem tudjuk azokat használni, az Active Directoryban a Read jog „láthatási" jog is egyben!

A felhasználói felület kialakításakor lehetőségünk van arra, hogy a menükben szereplő parancsokhoz parancsikonokat hozzunk létre. Ezáltal megkönnyíthetjük a rendszergazdai feladattal megbízott munkatársak dolgát. Parancsikont Taskpad View elkészítésekor hozhatunk létre. Jelöljük ki a Users mappát, majd az Action menü New Taskpad View parancsával indítsuk el a varázslót. Az üdvözlő képernyő után meghatározhatjuk, hogy a felhasználói felületen miképpen jelenjenek meg az objektumok. Lehetőségünk van vízszintes *(Horizontal list)*, illetve függőleges *(Vertical list)* elrendezést választani. Mindkét esetben eldönthetjük, hogy az objektumok listája a felhasználói felület mekkora hányadát foglalja el *(List size)*. Az alapértelmezett értéktől *(Large)* csak akkor érdemes eltérni, ha olyan sok ikont hozunk létre, hogy nem fér ki a képernyőre. A vízszintes és függőleges elrendezés mellett az objektumokat akár el is rejthetjük *(No list)*. Ez utóbbi esetben csak az ikonok lesznek láthatóak. Az ezt követő párbeszédablakban meghatározhatjuk, hogy a Taskpad View csak a kijelölt egységre *(például egy bizonyos mappára)* vagy az összes, a kijelölttel azonos típusú egységre vonatkozzon *(például minden mappára)*. Alapértelmezésben ez utóbbit kínálja fel a varázsló *(All tree items that are the same type as the selected tree item)*, amely beállítást érdemes meghagyni. A következő lépésben a Taskpad View nevét kell megadnunk, amely közvetlenül az objektumok listája fölött jelenik majd meg. Legvégül pedig a Taskpad View varázsló felkínálja a New Task wizard elindítását.
A New Task varázslóval lehet az általunk kiválasztott parancshoz gombot rendelni. Első lépésként a parancs típusát kell meghatároznunk. Mivel a gomb egy menüparancsot fog képviselni, ezért hagyjuk meg azt a beállítást, amit a varázsló alapértelmezésben felkínál *(Menu command)*. A következő párbeszédablakban kell megadnunk a szóban forgó parancs nevét. Mivel a letiltott felhasználói azonosító engedélyezése a Properties menüparancson keresztül érhető el, ezért a párbeszédablak jobb oldalán található listából *(Available commands)* ezt az utasítást válasszuk ki. Mielőtt továbblépnénk, győződjünk meg arról, hogy a párbeszédablak felső részén található kisablakban *(Command source)* a „List in details pane" felirat áll-e. Ugyanis csak ebben az esetben fog a Properties parancs a felhasználói azonosítókra vonatkozni. A „Tree item task" felirat esetén az ikon a kijelölt mappa vagy szervezeti egység tulajdonságait fogja megjeleníteni.



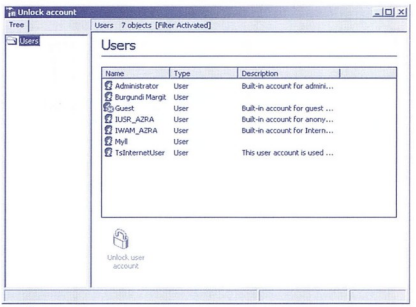
Mire vonatkozik a parancs? Az egész fára?

A következő lépésben a gomb nevét, ezt követően pedig a hozzá tartozó ikont kell meghatároznunk. Egy fontos célt nem tudtunk elérni sajnos a snap-in testreszabásával: a teljes Properties párbeszédpanelt tudjuk csak odaadni a felhasználónak, amellyel szegény mégiscsak zavarba ejtően sok fülhöz, pipához és mezőhöz jut. Szerencsére a nyomógomb nemcsak menüpont aktiválására képes, hanem például VBScriptet is futtathatunk a segítségével, amely csendben elvégezné a feladatot a kijelölt objektumokon – ennek megírására most terjedelmi okokból nem térünk ki, ehhez hasonló teljeskörű címtármódosító scriptek a tavaly októberi számban részletesen szerepeltek. Az alábbi ábra szépen mutatja, hogy hányféle bemenő paraméterrel segíthetjük a script tényekedését.



A bemenő paraméterek választéka script futtatása esetén

Ezzel azonban még nem fejeztük be teljesen a feladatra szabott rendszergazdai eszköz elkészítését. A felhasználói felületet illetően ugyanis érdemes még néhány korlátozással élnünk. Fontos lehet például a beállítások megváltoztatási lehetőségének letiltása. Alapértelmezésben ugyanis az általunk elmentett MMC fájl Author módban fog megnyílni, ami lényegében azt jelenti, hogy a felhasználó minden beállítást meg tud változtatni. Ha az MMC Console menü Options parancsára kattintunk, akkor a megjelenő párbeszédablakban több hasznos korlátozást is be tudunk állítani. Először is a megnyitási módot *(Console mode)* állítsuk át „Author mode"-ról „User mode – limited access single window"-ra. Ennek hatására a felhasználónak csak az MMC fájl elmentésekor látható ablak fog megjelenni, és új ablakokat sem tud létrehozni. Ezen kívül lehetőségünk van letiltani a jobb egérgomb lenyomásakor megjelenő gyorsmenüt *(Enable context menus on taskpads in this console jelölőnégyzet)*, a nézetek megváltoztatásának lehetőségét *(Allow the user to customize views jelölőnégyzet)*, valamint megakadályozhatjuk, hogy a felhasználók elmentsék az esetleges változtatásokat *(Do not save changes to this console jelölőnégyzet)*. Sőt a felhasználói felületet tovább egyszerűsíthetjük a View menü Customize parancsának segítségével. Elrejthetők többek között az MMC és a snap-in menüi csakúgy, mint az eszközsor, az állapotsor stb. A mi esetünkben sem az eszközsorra, sem a menüsorra nincs szükség, ezért a megjelenítésüket érdemes letiltani. A végeredmény egy listaablak, melyben a kiemelt felhasználó kiválaszthatja pórul járt társát, majd rákattintva a lakat ikonra, a megjelenő Properties ablakban kiválasztja a megfelelő jelölőnégyzetet, és elvégzi a rábízott Nagy Feladatot.



És kész!

Fontos felhívni a figyelmet arra, hogy minden korlátozás ellenére az elmentett MMC fájlt *(*.MSC)* a felhasználók meg tudják nyitni Author módban *(például My Computer File menü Author parancs)*. Lehetőségük nyílik ezáltal a beállításaink megváltoztatására. Ez azonban nem jelenti azt, hogy hiábavaló volt eddigi fáradozásunk. A Windows 2000 csoportos házirendje erre a problémára is kínál megoldást az Administrative Templates\Windows Components\Microsoft Management Console mappában található „Restrict the user from entering author mode" beállítás révén. Érdemes lehet az MSC fájlt a felhasználók számára csak olvasható módon megosztott hálózati könyvtárba helyezni, így központi helyen, egy példányban a megváltoztatás kockázata nélkül fel tudjuk kínálni a kész „rendszergazdai" eszközöket a kijelölt felhasználóknak.

Végezetül meg kell említeni, hogy a rendszergazdai feladattal megbízott felhasználó munkaállomására először telepíteni kell a rendszergazdai eszközöket, csak ezután tudja majd használni az általunk elkészített MMC fájlt. Az eszközöket a Windows 2000 Server CD I386 könyvtárában található adminpak.msi segítségével tudjuk telepíteni.

Tomasz Balázs
balazs.tomasz@hu.hypovereinsbank.com
pszichológus

Használjunk USB-t és 1394-et

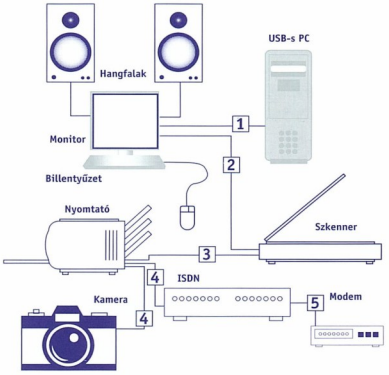
Közeleg az idő, amikor PC-ink hátuljáról eltűnnek a soros és a párhuzamos portok csatlakozói, megszűnik a kábeláradat. Nem lesz PS/2-es port, nem lesz külön hangkártya- és modemkimenet, nem fog fájni a fejünk az elfogyott IRQ-k miatt. Lesz viszont egy vagy két *(néhol négy)* USB konnektor, egy IEEE-1394 csatlakozó, egy monitor csatlakozó és egy helyi hálózati kimenet. Semmi több.

Mi is az a(z) USB?

Mint a neve is mutatja az USB *(Universal Serial Bus)* egy busz, melyre maximum 126 *(!)* eszköz csatlakoztatható. Figyelem, ezek az eszközök összesen egyetlen egy megszakítási címet fognak elhasználni! Az eszközök bármikor csatlakoztathatók vagy leköthetők a buszra*(-ról)*, az operációs rendszer automatikusan érzékeli és „beállítja" ezeket. Nincs jumperelés, nincs újraindítás, minden Plug and Play.

Hogyan kell használni?

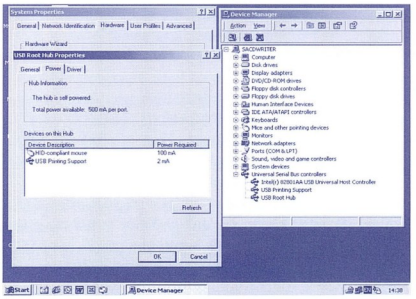
Kábelezési szabályok: A kábelek legfeljebb 5 méter hosszúak lehetnek, melyek segítségével maximum egy 5 kábelből álló láncolatot hozhatunk létre a számítógéptől a legutolsó még használható eszközig. *(Tehát a maximális távolság a géptől 25 m, de a kábelhosszokat hiába vesszük kisebbre, a láncolat akkor is csak öt lépésből állhat.)* A csomópontokban természetesen lehetnek elágazások, ha az eszközök több USB porttal rendelkeznek vagy az eszköz maga egy USB HUB. *(Így lehetséges az elméleti 126-os határ elérése.)* Az alap USB kábelnek különböző a két vége, ezért lehetetlen nem működő „hurkot" létrehozni.



Az ábrán jól látható a maximum 5 láncszem

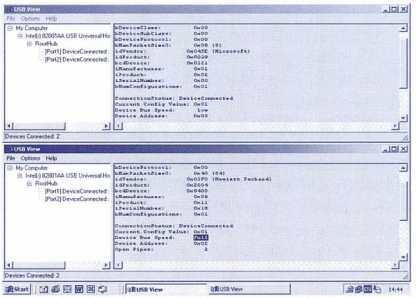
Portok?

Egy USB port lehet aktív vagy passzív. Az aktív port maga biztosítja a rácsatlakoztatott eszköz számára az energiát *(aktív portja természetesen olyan eszköznek lehet, amely az áramot nem a buszról veszi)*. A buszt vezérlő szoftver gondoskodik arról, hogy ha „elfogyott a buszról az összes energia" *(maximum 500 mA)* akkor több eszköz ne akarjon áramot felvenni.



Az egérkénk passzív és sokat fogyaszt, a nyomtató viszont aktívan szerénykedik

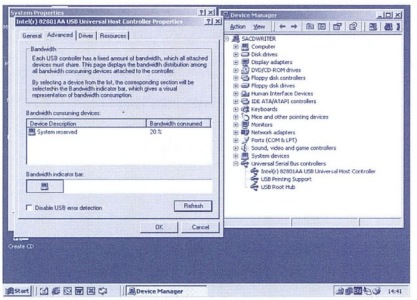
Ennek feloldására való az aktív port. Ha egy aktív portot tartalmazó eszközt kikapcsolunk, akkor onnantól lefelé a láncban lévő eszközök nem biztos, hogy működni fognak, minden az energiafelvételen múlik. Honnan tudhatjuk, hogy egy eszköz mennyi energiát használ *(lásd fent)* vagy milyen sebességgel működik *(high/low)*? Erre szolgál a Windows 98 CD-n egy kis programocska: (\tools\reskit\diagnose\usbview.exe).



Az USBVIEW Windows 2000 alatt is megy...

Az USB a következő operációs rendszerekben támogatott: Windows 95 OSR2.1B; Windows 98; Windows 2000.
És ha öregecske a gép?
USB porttal nem rendelkező számítógépet *(pl. régebbi Pentium modellek)* is bővíthetünk olyan PCI-os kártyával, amely

USB csatolót tartalmaz. USB segítségével akár hálózatot is kiépíthetünk, erre is léteznek már eszközök.



Az 1.1-es USB teljes sávszélessége hamar elfogyhat

USB 2.0

Az USB 2.0-s verziójában *(melyet olyan cégek dolgoztak ki, mint a Compaq, HP, Intel, Lucent, Microsoft, NEC és a Philips)* a sebesség közel negyvenszeresére, azaz 480 Mbit/s-ra emelkedik majd! Jó hír, hogy a gyorsabb adatforgalom a „régi" USB kábeleken zajlik, valamint a meglévő eszközök is ráköthetőek az új buszra *(ez alól kivételek a HUB-ok, amelyek szerepét a már háromféle sebességre képes - 480, 12, 1,5 Mbit/s - 2.0-sok veszik át)*. Ami viszont szükséges: egy USB 2.0-s PCI csatolókártya, vagy egy új alaplap, ami már 2.0-s USB-t támogat, de sajnos ezek megjelenésére még várnunk kell egy pár hónapot. *(Az Intel már az I815-ös chipsetben ígérte a támogatást, de később ez elhalasztódott.)* A szoftveres támogatásra is várni kell, jelenleg még nem létezik 2.0-s USB patch a windows-okhoz. *(A Whistler, „akarom mondani" a Windows XP már támogatni fogja.)*

Technológia	Maximális sebesség (elméletben)
Soros port	230 Kbit/s
USB 1.1 Low	1,5 Mbit/s
USB 1.1 High	12 Mbit/s
FireWire	400 Mbit/s
USB 2.0 High	480 Mbit/s
Ultra SCSI-3	160 Mbyte/s

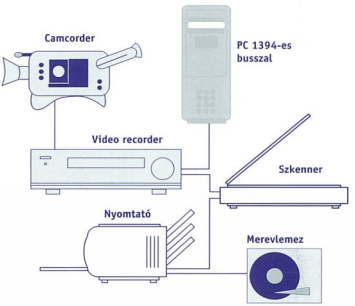
Fire-Wire

Mivel az USB 2.0 még nem elérhető, egy lehetséges alternatíva lehet az IEEE 1394. Ez tulajdonképpen az Apple által kitalált Fire-Wire, amelynek szabványát 1990-ben kibővítették, s ennek eredményeképpen jött létre az IEEE 1394 specifikáció.

Sebesség

A 1394-es busz két sebességen működhet: 200 Mbit/s-on *(S200)* és 400 Mbit/s-on *(S400)*, ez utóbbi már 50 Mbyte/s-ot jelent. *(Jobb, mint egy UW SCSI!)* A nagy adatátviteli sebesség igénye főleg multimédiás applikációknál jelentkezik, tehát azon eszközök lesznek *(vannak)* 1394-es csatlakozóval ellátva, amelyek ebben szerepet játszanak: camcorderek, videofelvevők, merevlemezek. Persze jelenleg is léteznek olyan eszközök *(pl. video grabber kártyák)*, melyekkel mondjuk a videófeldogozás elvégezhető, de az 1394 jóval olcsóbb megoldást jelent.

Az 1394 segítségével könnyedén létrehozhatunk otthon is egy kis videóstúdiót, hiszen az 1394-es buszra csatlakoztatott videókamera és a merevlemez között közvetlen adatforgalom jön létre, a „film" egy szoftverrel megszerkeszthető, majd tárolható az ugyancsak a buszra csatlakozható videofelvevőn.



Az én kis házistúdióm...

Kábelezés

Az IEEE 1394-es buszra maximum 63 eszköz csatlakoztatható egyszerre, a maximális kábelhossz *(két eszköz között)* négy és fél méter lehet. Legfeljebb 16 ilyen kábel fűzhető láncba, tehát a számítógép és a legmesszebb lévő eszköz maximális távolsága 72 méter. Nincs szükség terminálásra, ID jumperelésre *(SCSI)*, az eszközök menet közben csatlakoztathatók vagy leköthetők *(hot plug)*, minden rákapcsolt eszköz automatikusan címződik. Csakúgy, mint az USB esetében, a csillagpontos elrendezéshez HUB-okat használhatunk, de minden olyan eszköz, amelynek egynél több 1394-es csatlakozója van már HUB-ként is funkcionál. Az eszközök áramfelvétele is hasonló az USB-nél látottakhoz, azaz vannak aktív és passzív portok. A passzív porttal ellátott eszköz közvetlenül a buszról veszi fel a működéséhez szükséges energiát, az aktív port pedig a passzív eszközöket tudja ellátni maximum 3 Watt energiával. A használt kábel 6 eres: 2 kell az „áramhoz", 2 kábelen megy az adat, kettőn pedig a szinkronjel. *(Erről többet itt most inkább nem írnék...)*

Csatlakozó

A megfelelő csatlakozó keresésekor a következő szempontokat vették figyelembe: legyen strapabíró, legyen megfelelően árnyékolt és persze olcsó! A választás a Nintendo Gameboy csatlakozójára esett...

Azoknak sem kell elkeseredniük, akiknek a gépén nincs 1394-es csatlakozó, hiszen vásárolható olyan kártya *(általában PCI buszos)*, amely rendelkezik ilyennel.

Végezetül csak érdekességként jegyzem meg, hogy az SCSI-3 specifikáció felveti az IEEE 1394 kábelek használatának alkalmazását is *(hiszen ez jóval olcsóbb)*.

Tóth László
tothl@montana.hu
MCSE, Compaq ASE

A BOOT.INI magánélete

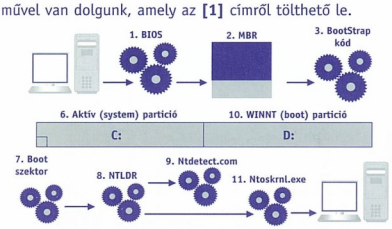
Valószínűleg UNACCESSIBLE_BOOT_DEVICE hibaüzenetet mindannyian láttak már. Ez a hiba szép kék képernyőn szokott megjelenni valamikor az operációs rendszer betöltésének kellős közepén. Ilyenkor hirtelen felindulásból általában elsőként a fejlesztők felmenőit szoktuk szidni, hogy mi az, hogy INACCESSIBLE, ha egyszer már félig bebootolt. A hiba érdekessége, s a sors különös fintora, hogy ebben a szent esetben nem a fejlesztők, és nem is Bill Gates tehet a dologról, hanem ilyen a gép: az Intel architektúra egy szörnyű korlátjába ütköztünk. Mert legyen bár az INACCESSIBLE lemez SCSI vagy IDE, mindenképpen a BIOS okolható a rendszertöltés meghiúsulásáért. Vagy azért, mert nincs is *(például le va tiltva a SCSI vezérlőn)*, vagy azért, mert pont olyan helyről akarunk bootolni, amit a BIOS már nem „lát". Köztudott talán, hogy a BIOS-ok maximum 7,8 gigabájtot képesek kezelni a legnagyobb merevlemezekből is, ennél többet akkor sem látnak, ha márkás gyártó BIOS-ával állunk szemben. Most nem vezetném le bitszintig, hogy miért pont 7,8 giga az „álomhatár", hisz tavaly ezt egyszer már megtettem.

Tehát a probléma gyökere abban keresendő, hogy bár a bootolás elindulhatott a BIOS segítségével *(hisz másképp nem tud elindulni)*, ám befejeződni már nem tudott: egyszer csak kikerült az éppen betöltendő rendszerkomponens a gép látómezejéből.

Egy paradoxont hadd oldjak fel mindjárt. Persze, hogy a Windows 2000 tetszőleges nagyságú *(sajnos ez max. 2 terabájt Intelen, s nem 16 exa, ahogy a prospektus mondja)* partíciót tud kezelni, de ahhoz nem is a BIOS INT 13 megszakításaival fér hozzá, hanem közvetlenül, saját eszközmeghajtójával, melynek neve IDE esetén ATAPI.SYS.

Eddig azonban el kell jutni! A rendszertöltés igenis mindig BIOS-sal indul, hisz az minden gépen mindig elérhető, s csak az oprendszer betöltésének egy későbbi fázisában lehet tőle megszabadulni. Ha azonban ez a váltás, stafétaátadás valami miatt meghiúsul, akkor egy félig kész oprendszert „nyerünk", ami ennyire képes: INACCESSIBLE_BOOT_DEVICE.

A rendszertöltés folyamatának teljes megértéséhez rendhagyó, de lehet, hogy hagyománnyteremtő eszközhöz nyúltam: animációt készítettem *(PPT)*! Az alábbi ábra ennek csontvázát mutatja, de ennél lényegesen szebb, kommentált remekművel van dolgunk, amely az **[1]** címről tölthető le.



A rendszertöltés folyamatának vázlata

Az ominózus kék képernyős hiba akkor következik be, amikor az NTLDR *(ez már NT!)* megpróbálja megkeresni a boot partíciót, tehát az NTOSKRNL.EXE-t tartalmazó lemezterületet. A probléma teljes elkerülésének módja az lehetne, ha a BIOS-tól a lehető legkorábbi pillanatban meg lehetne szabadulni, azaz már az NTOSKRNL.EXE megtalálásához sem kellene BIOS *(ennél korábbi fázisban nem lehet lerázni a BIOS láncait!)*. Van rá mód! Ugyanis az ábrán látható NTLDR is képes lehet saját eszközmeghajtó használatára, ha teljesítjük összes kívánságát:

1. Az eszközmeghajtó az aktív partíció gyökerében legyen
2. Neve legyen szigorúan NTBOOTDD.SYS
3. A BOOT.INI-ben pedig ne MULTI(), hanem SCSI() szerepeljen.

A harmadik kívánság kicsit abszurd nem? Ha a merevlemezem IDE, akkor ugyebár nem SCSI, és vica versa.

Amit tudni akartál a BOOT.INI-ről, de nem merted megkérdezni

A teljes igazság így hangzik:

- A MULTI() nem azt jelenti, hogy a merevlemez IDE, hanem azt, hogy BIOS INT 13 megszakítással kezelendő. SCSI lemezekről is elstartol, ha a kártyán engedélyezve van a BIOS.
- A SCSI() nem azt jelenti, hogy SCSI a lemez, hanem hogy saját eszközmeghajtóval kezelendő. És IDE lemezekkel is használható, ha a BIOS-t kiváltjuk a megfelelő eszközmeghajtó programmal.

Ez mindent megmagyaráz. Éppen olyan megkövült terminológiai bakival állunk szemben, mint amilyen például a SYSTEM és a BOOT partíció megnevezése a Windows 2000 dokumentációban, ugyanis a SYSTEM partíció arról ismerhető fel, hogy azon vannak a BOOT fájlok, míg a BOOT partíción találjuk a WINNT könyvtárat, tehát a SYSTEM-et :-O

Ezzel a tudással felvértezve mindenki elvégezhet egy érdekes kísérletet a saját asztali, SCSI-t sosem látott gépén. Érintetlenül hagyva a meglévő BOOT.INI bejegyzéseket *(hogy legyen legalább egy ép menüpontunk a hekkelés után)*, készíthet egy újat, ami SCSI() kezdetű! Ha a három fenti feltétel teljesül, a Windows 2000 tökéletesen bebootol az NTBOOTDD.SYS meghajtó segítségével. S hogy honnan szdjük az NTBOOTDD.SYS-t? Nos - hogy minden kíváncsit a webre tereljek – az **[1]** címen található kiváló PPT utolsó előtti lapján olvasható a titok megfejtése. Vigyázat, másképp kell előállítani a fájlt Windows NT 4.0 és Windows 2000 esetén!

Fóti Marcell
marcellf@netacademia.net

A cikkben található URL-ek:
[1] http://technet.netacademia.net/feladatok/ppt/bootini.ppt

Exchange – átszervezés

Az üzlet gyakran zavarosabb életünk bármely más területénél. Általánossá vált a cégek gyakori átszervezése, ami a számítástechnikai részleg fő gondjává válhat, hiszen többnyire együtt jár a számítógépek áthelyezésével és néha a teljes tartománymodell átszervezését jelentheti. Bonyolultsága miatt az Exchange Server 5.5 azok közé az összetevők közé tartozik, amelyeket a legnehezebb átszervezni. Utódját, az Exchenge Server 2000-t lényegesen könnyebb, de a meglévő régi rendszerek nagy száma miatt talán mégsem érdektelen szót ejteni a Moveserver.exe-ről, mely lehetővé teszi a mozdíthatatlannak hitt Exchange 5.5 kiszolgáló újratelepítés nélküli áthelyezését más telephelyre *(site)*, illetve szervezetbe *(organization)*. Ez a cikk megpróbálja megkönnyíteni az informatikusok munkáját és bemutatja, hogyan helyezzük át egyik telephelyről a másikra az Exchange Servert.

Tervezés

A sikeres áthelyezés kulcsa a körültekintő tervezés. Az áthelyezéshez használt segédprogram kezelése viszonylag egyszerű és tesztjeink során semmiféle probléma nem merült fel az Inbox, a Calendar, a Contact Manager, a Task List és a többi összetevő áthelyezésénél. Mindazonáltal az áthelyezést végzőnek sok a tennivalója.

Mentés

Amikor kiszolgálókat helyezünk át, mindig fennáll annak a veszélye, hogy valami nem sikerül. Mielőtt hozzákezdünk, mindig készítsünk mentést minden egyes kiszolgálón, amivel dolgozunk.

Az Exchange ügyfelek előkészítése

Mielőtt áthelyezzük a kiszolgálót, meg kell értenünk, hogyan befolyásolja az áthelyezés az ügyfeleket. Általában semmit nem kell változtatnunk az ügyfélen, mert nem változik a kiszolgáló neve. Mint ismeretes, az Exchange Client és az Outlook a kiszolgáló nevét használják a kapcsolódáshoz, nem a telephely nevét. Ennek ellenére bizonyos esetekben szükségessé válhat az ügyfél újrakapcsolódása a megfelelő postaládához. Különösen igaz ez akkor, ha azon a telephelyen, ahova a kiszolgálót áthelyezzük, már van egy másik kiszolgálón ugyanolyan nevű postaláda. Ha a kiszolgálót más tartományba akarjuk áthelyezni, mint amibe az ügyfelek bejelentkeznek, meg kell győződnünk arról, hogy léteznek a megfelelő megbízotti kapcsolatok *(trust relationships)*.

E-mail-címek

Fontos kitérni arra, hogy mi történik azokkal az e-mailekkel, amelyek egy áthelyezett postaládába érkeznek. A belső üzenetek nem változnak, mert minden, az áthelyezéssel kapcsolatos változás automatikusan bekerül a globális címlistába *(feltéve, hogy a replikáció működik)*.

Az internetes üzenetekkel más a helyzet. Mint tudjuk, a kiszolgáló IP-címe a DNS-kiszolgáló listáján a kiszolgáló domain nevével együtt áll. A kiszolgáló áthelyezésekor minden postaláda megtartja eredeti SMTP-címét. Mivel a kiszolgáló neve és IP-címe megmaradt, és mivel van DNS-hivatkozás a kiszolgálóra, az eredeti címnek továbbra is működnie kell. Minden postaládához egy új SMTP-cím is hozzá van rendelve, ami megfelel a hídfő kiszolgáló DNS-bejegyzésének. Ennek a címnek is működnie kell.

A kiszolgálók előkészítése

Miután megterveztük az áthelyezést, elő kell készítenünk a kiszolgálókat. Győződjünk meg arról, hogy mindkét oldalon Exchange 5.5 fut *(legalább 2-es szervizcsomaggal)*. Az eljárás nem működik az Exchange régebbi verzióival vagy 1-es szervizcsomaggal.

Miután ellenőriztük az Exchange verziószámát, engedélyeznünk kell a telephelyek közötti címtárreplikációt. Ez azt jelenti, hogy mindkét telephely tud egymásról és közös globális címlistájuk van. Ehhez létre kell hoznunk egy telephelycsatolót *(Site Connector)* és egy címtárreplikációs csatolót *(Directory Replication Connector)*.

A Site Connector *(telephelycsatoló)* beállítása

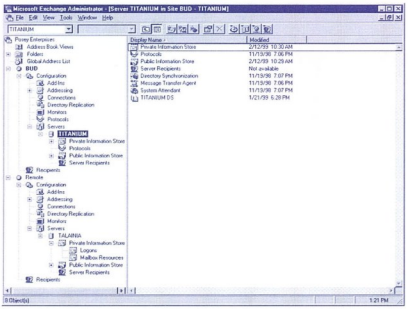
A Site Connector feladata, hogy logikai hivatkozást hozzon létre a két telephely között. A Site Connector beállítása előtt már léteznie kell fizikai összeköttetésnek. Az összekapcsolandó telephelyeknek ugyanabban a szervezetben kell lenniük. Figyeljünk arra, hogy a szervezetnevek esetében nem mindegy, kis- vagy nagybetűt használunk-e. A Site Connector létrehozásához jelöljük ki a „Connections" menü New Other | Site Connector pontját és adjuk meg a megfelelő információkat.

A Site Connector beállítása a legtöbb esetben egyszerű és világos. Mégis hangsúlyoznunk kell egy bizonyos lépést, amelyen sok minden múlhat. Az Exchange telepítésekor a telepítő program kérte, hogy adjunk meg egy bejelentkezési nevet - melynek jogaival az Exchange futni fog - és a hozzá tartozó jelszót. Az Exchange ezt a bejelentkezési nevet arra használja, hogy kommunikáljon a Windows NT-vel. Ennek eredményeképpen mindkét telephelynek tudnia kell a másik bejelentkezési nevét és jelszavát. Ezeket az információkat a Site Connector „Properties" lapján az „Override" fülön lehet megadni. Ne feledjük, hogy ha a két telephely hídfő kiszolgálója különböző Windows NT tartományokban található, akkor a Site Connectornak kétirányú megbízotti kapcsolatot kell megadni a tartományok között.

A Directory Replication Connector *(címtárreplikációs csatoló)* beállítása

A Site Connector létrehozása után be kell állítanunk a Directory Replication Connectort. A Directory Replication Connector létrehozásához válasszuk ki a Directory Replication tárolót, majd a „File" menü New Other | Directory Replication Connector pontját.

A Directory Replication Connector beállítása egyszerű, csupán meg kell adnunk az adatokat a „Directory Replication Connector Properties" adatlapon. Az egyetlen, amire figyelnünk kell, hogy ugyanúgy, mint a Site Connector esetében, itt is mindkét telephelyen be kell állítanunk a replikációs csatolót. A beállításnál a „Schedule" fülön mindkét telephelyen „Always"-et kell kijelölni. Ne törődjünk vele, ha a „Sites" fülön semmi nem látszik. Később, a replikáció kezdetekor ez az információ is automatikusan megjelenik. Onnan tudhatjuk, hogy a replikáció befejeződött, hogy az Exchange Administratorban mindkét telephely látszik, mint azt az alábbi ábra mutatja.



Engedélyeznünk kell a telephelyek közötti címtárreplikációt, mielőtt elindítanánk az áthelyezést.

Az áthelyezés végrehajtása

Ha egy kiszolgálót másik telephelyre akarunk áthelyezni, első lépésként ki kell csomagolnunk a Move Server segédprogramot. Helyezzük az Exchange 5.5 2-es szervizcsomagjának CD-jét a CD-meghajtóba. Hozzunk létre egy Movesrvr nevű könyvtárat a merevlemezen és másoljuk bele a CD \Server\Eng\Server\Support\Movesrvr könyvtárának tartalmát. Ezután indítsuk el a Setupmvi.exe fájlt. Ha a kicsomagoló program megkérdezi, hova akarjuk tenni a fájlokat, adjuk meg neki a merevlemezen található Movesrvr könyvtárat.

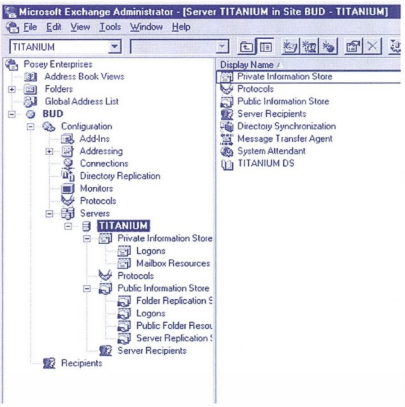
Az Exchange Server elválasztása

Mielőtt használhatnánk a Move Server programot, el kell választanunk az Exchange Servert. Ehhez meg kell szüntetnünk a két telephely között az imént létrehozott címtárreplikációs csatolót, hisz arra már nincs szükség. Kérdezhetnénk, hogy miért hoztuk létre, ha máris megszüntetjük, de tapasztalataink szerint a telephelyeknek legalább egyszer tudniuk kellett egymásról, különben az áthelyezés nem sikerül.

A Directory Replication Connector eltávolításához keressük meg az Exchange Administratorban az Organization | Site | Configuration | Directory Replication tárolót. Jelöljük ki a Directory Replication Connectort majd nyomjuk meg a „Delete" gombot. Ekkor figyelmeztető üzenet jelenik meg, kattintsunk a „Yes" gombra a folytatáshoz. Ezután az Exchange megkérdezi, akarjuk-e a másik telephelyről is törölni a Directory Replication Connectort. Kattintsunk a „Yes" gombra és az Exchange törli a Directory Replication Connectort. Kattintsunk az „OK" gombra, majd tegyük ezt újra, hogy átlépjünk a figyelmeztető üzeneten.

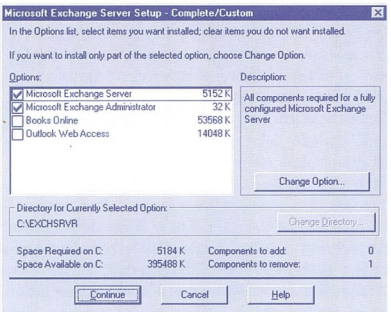
Ha eltávolítottuk a Directory Replication Connectort, törölnünk kell a telephelyeket összekötő csatolót *(Site Connector)* is. Ehhez az Organization | site | Configuration | Connections konténerben jelöljük ki a Site Connectort és nyomjuk meg a „Delete" gombot. Amikor a program megkérdezi, akarjuk-e törölni a Site Connectort, kattintsunk a „Yes" gombra. A Site Connectort a másik telephelyen is ugyanígy törölnünk kell.

A Site Connector eltávolítása után eltűnik a másik telephely, amint az az alábbi ábrán látható. Ez akár egy órát is igénybe vehet. Az F5 gombbal frissíthetjük az Exchange Administrator nézetét, hogy lássuk, eltűnt-e már a másik telephely.

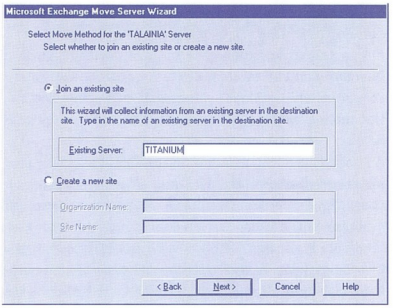


A távoli telephelyek végül eltűnnek az Exchange Administratorból.

A következő lépésben ellenőriznünk kell, hogy azon a kiszolgálón, amit át akarunk helyezni, fut-e a Microsoft Exchange Event Service. Ha igen, el kell távolítanunk. Ehhez indítsuk el a Microsoft Exchange Server Setup programot és válasszuk az Add/Remove Components menüpontot. Ha az alábbi ábrán látható párbeszédpanel megjelenik, jelöljük ki a Microsoft Exchange Server sort és kattintsunk a „Change Option" gombra. Ekkor megjelenik a Microsoft Exchange Event Service. Távolítsuk el az „Event Service" kijelölését és kattintsunk az „OK" gombra a folytatáshoz.



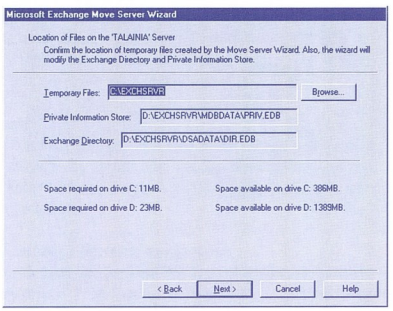
Használjuk a Microsoft Exchange Server Setup programot az Exchange Event Service eltávolításához.



Adjuk meg a kiszolgáló nevét azon a telephelyen, ahová csatlakozni akarunk

A Move Server Wizard használata

Elérkeztünk a tényleges áthelyezéshez. Nyissuk meg a \Movesrvr\Mvexsrvr.exe fájlt, ami elindítja a Microsoft Exchange Move Server Wizardot. Kövessük az utasításokat, amíg a következő ábrán látható ablakhoz érünk. Vegyük figyelembe a kijelzett helyigényt. Ezek a becslések nem mindig pontosak, ezért olyan meghajtót válasszunk, ahol sok szabad hely van. Ha áthelyezés közben elfogy a szabad hely, az beláthatatlan következményekkel járhat.



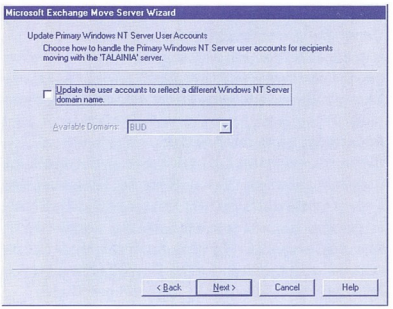
Olyan meghajtót válasszunk, amelyen sok szabad hely van.

Ha kijelöltük a használni kívánt merevlemezeket, kattintsunk a „Next" gombra a folytatáshoz. Ekkor egy olyan ablak jelenik meg, amely hasonlít a Microsoft Exchange Server Setup programjában látottra. Mivel éppen azon vagyunk, hogy egy kiszolgálót egy másik telephely részévé tegyünk, kiválasztjuk a „Join an existing site" lehetőséget, megadjuk a szükséges információt, mint azt a következő ábra mutatja, majd a „Next" gombra kattintunk.

Ezután megjelenik egy ablak, amelyben meg kell erősítenünk az imént megadott információkat. Amennyiben az információk helyesek, kattintsunk a „Yes" gombra a folytatáshoz. Ekkor egy figyelmeztetést látunk, amely felhívja a figyelmet arra, hogy az a telephely, ahova a kiszolgálót át akarjuk helyezni, ugyanannak a szervezetnek a része és a két telephely között nem replikáljuk a címtárinformációkat. Kattintsunk a „Yes" gombra és folytassuk az áthelyezést.

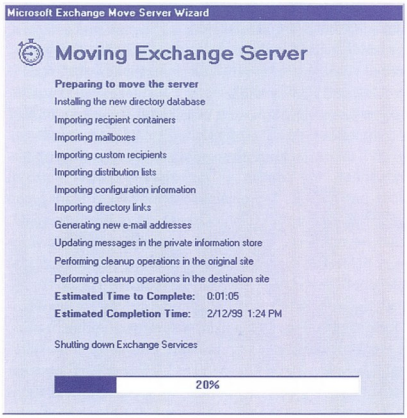
A varázsló ezután kéri az új telephely szolgáltatásfiókjának *(service account)* bejelentkezési nevét és jelszavát. Az információk megadása után kattintsunk a „Next" gombra. Győződjünk meg róla, hogy a „Move All Custom Recipients From This Site" jelölőnégyzet ki van jelölve és kattintsunk a „Next" gombra. A következő ablakban a varázsló megkérdezi, hogyan kezelje a levelezési listákat. Ha nincs semmilyen különösebb óhajunk, jelöljük ki a „Move All Distribution Lists In The Site" sort, majd kattintsunk a „Next" gombra.

A lent bemutatott képernyő jelenik meg előttünk. Mivel minden postaládának van már Windows NT fiókja, itt semmit nem kell tennünk, kattintsunk egyszerűen a „Next" gombra. Ha más tartományt adunk meg, akkor a létező fiókok nem tudják elérni a hozzájuk tartozó postaládákat. Ha a „Next" gombra kattintunk, a program elkezdi keresni az áthelyezendő objektumokat.



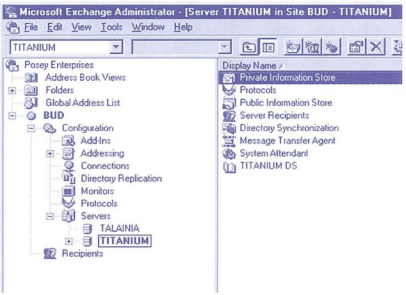
Ne változtassuk meg a Windows NT tartományokat ebben az ablakban, hacsak nincs rá különleges okunk.

A varázsló ebben a szakaszban lehetővé teszi, hogy átnevezzük a már létező objektumokat és ellenőrzi, hogy minden készen áll-e mindkét kiszolgálón, közben tájékoztat bennünket, mit tesz éppen. Ha készen állunk az áthelyezésre, kattintsunk a „Finish", majd az „I Understand" gombra. Az áthelyezés sokáig tarthat, különösen akkor, ha a kiszolgálón sok a felhasználó. Itt látható, hogy a varázsló kijelzi, hol tart a folyamat, így végigkövethetjük:



Ebben az ablakban végigkövethetjük az áthelyezés folyamatát.

Amikor az áthelyezés befejeződött, kattintsunk a „Finished" gombra. Ha megnyitjuk az Exchange Administratort, a kiszolgálót annak a telephelynek tagjaként kell látnunk, ahova áthelyeztük.



A kiszolgálónak most már az új telephelyen kell megjelennie.

Végkövetkeztetés

Ebben a cikkben bemutattuk, hogyan lehet áthelyezni az Exchange kiszolgálót egyik telephelyről a másikra. Feltétlenül tartsuk szem előtt, hogy ezt a feladatot úgy lehet a leggördülékenyebben véghezvinni, ha kivitelezését megfelelő tervezés előzi meg.

Mobile Information Server

Óvatos becslések szerint 2002-re 6-800 millió internetezésre alkalmas mobiltelefon lesz használatban világszerte. A vállalatok szeretnék, hogy alkalmazottaik bárhol és bármikor elérjék a fontos adatokat, a fogyasztói igények pedig a nagysebességű vezeték nélküli hálózatok irányába terelik a fejlesztéseket. Az alkalmazottak gyakran dolgoznak útközben és otthonról, ezért szükségük van a vállalati hálózat mobil eszközről történő elérésére, hiszen így tudnak megalapozott döntéseket hozni, hatékonyan kiszolgálni a vásárlókat és folyamatosan követni az irodában történteket. A mobileszközök terjedésével egyidejűleg azonban szükség lesz arra is, hogy ugyanúgy képesek legyenek megóvni a vállalat adatainak integritását és titkosságát, mint manapság a vezetékes eszközökkel, ezért minden, a vállalati intranetet bővítő mobileszköznek meg kell felelnie a következő feltételeknek:

- A felhasználói azonosítók és jelszavak ne legyenek „lehallgathatók"
- Biztosítható legyen a vállalati adatok bizalmas kezelése
- Tartható legyen az alacsony birtoklási költség *(TCO)*

A Microsoft Mobile Information 2001 Server Enterprise Edition és a Microsoft Mobile Information 2001 Server Carrier Edition újfajta mobilalkalmazás-kiszolgálók, melyeket úgy terveztek, hogy megfeleljenek a vállalatok biztonsági elvárásainak. Ezek a kiszolgálók biztosítják a felhasználóknak a vállalat adatainak *(például intranet és Exchange)* mobil elérését. A Mobile Information Server együttműködik a jelenlegi mobileszközökkel is, de teljes funkcionalitása csak az új generációs, fejlettebb ketyerékkel lesz kihasználható. Jelenlegi változata csak a WAP-os telefonok használatát támogatja, de a későbbi változatokban és külső gyártók kiegészítői segítségével támogatott lesz a WAP mellett a HTML, és az intelligens telefonok, digitális személyi asszisztensek *(PDA-k)*, kétirányú személyi hívók használata is.

A Windows támogatja a szabványos biztonsági protokollok használatát *(a Microsoftnak jelentős szerepe volt abban, hogy a vállalati (vezetékes) hálózatokban végponttól-végpontig biztonságos adatátvitel legyen megvalósítható, például a Point-to-Point Tunneling Protocol (PPTP) és az Internet Security Protocol (IPSec) segítségével)*. Ebben a cikkben azt mutatjuk be, hogyan valósít meg a Microsoft ehhez hasonló biztonsági funkcionalitást a Mobile Information 2001 Server-ben.

Miért is más a „vezeték nélküli"?

A vezeték nélküli átvitel biztonságossá tétele természetéből adódóan igen bonyolult lehet. Amikor egy ilyen modem jeleket visz át, gyakorlatilag bárki elcsípheti az adást. A digitális mobiltelefon rendszerek elterjedése előtt virágzott a telefonhamisítás. Ehhez csak egy adóvevőre és néhány olcsó dekódoló eszközre volt szükség, és már hozzá is lehetett férni bárki telefonszámához. A digitális vezeték nélküli hálózatok jobb védelmet nyújtanak, de mivel maga a hordozó közeg fizikailag nem biztonságos, mindenképp hatékony adatvédelemre van szükség.

A vezeték nélküli hálózatok lassabbak, hajlamosabbak az átviteli hibákra, és késleltetésük is nagyobb, mint vezetékes társaiké. Ebből adódik, hogy a vezetékes hálózatokhoz tervezett megoldásokat nem lehet egy az egyben vezeték nélküli hálózatokban alkalmazni. Például nem mindegyik vezeték nélküli hálózat támogatja a szabványos Internetes protokollok, így például a Transmission Control Protocol *(TCP)* használatát. E problémák enyhítésére a legnagyobb gyártók új protokollokat fejlesztenek, és olyan új megoldásokat hoznak létre, amelyekben a vezeték nélküli és vezetékes hálózatok közti átjárás biztosítására váltókiszolgálók működnek. A Wireless Application Protocol *(WAP)* Forum, amelynek a Microsoft is tagja, bemutatott egy protokollcsomagot, melyet a vezeték nélküli hálózatokra optimalizáltak, és kiegészíti a meglevő szabványos Internetes protokollokat. A vezeték nélküli és vezetékes hálózatok közti átmenet biztosításához a Mobile Information Server protokollátalakítókat tartalmaz, vásárlói pedig egy funkciókban gazdag és integrált átjárót kapnak.

A vezetékes és vezeték nélküli hálózatok közötti kiszolgáló vezeték nélküli protokollokat és biztonsági szabályokat biztosít a kommunikáció vezeték nélküli részéhez, míg hagyományos protokollokat és biztonsági szabályokat a vezetékes részhez. Ezeket a szolgáltatásokat nyújthatja egyetlen, de akár több felügyeleti tartományhoz tartozó kiszolgáló is. A mai vezeték nélküli környezetekben például egy - a vezeték nélküli eszközökhöz adatot szállító - WAP kiszolgálót, és egy Mobile Information Server-t futtató számítógépet jelenthet, amely átjárást biztosít a cég intranete felé. Az ilyen típusú hálózatokban a kommunikációnak több szakasza van:

- A vezeték nélküli eszköztől a WAP kiszolgálóig
- A WAP kiszolgálótól a Mobile Information Server-ig
- A Microsoft Mobile Information Server-től a háttér-kiszolgálókig *(például Exchange, IIS és SQL Server)*.

A következő részben ismertetjük, hogy hogyan is biztosítja a Mobile Information Server ezeken a szakaszokon a kommunikáció biztonságosságát.

A Mobile Information Server biztonsága

A Microsoft célja az, hogy a vezetékes hálózatokban megszokott biztonsági funkciók rendelkezésre álljanak a vezeték nélküli megoldásoknál is. A köztes kiszolgáló, azaz a vezetékes és vezeték nélküli hálózatok közti átjáró jelenléte nem befolyásolhatja a hitelesítést és az adatok titkosságát. A mai WAP-os telefonok nem támogatják a fejlett hitelesítési és titkosítási *(például SSL)* eljárások használatát, ezért a Mobile Information Server első verziója alap *(basic)* hitelesítést és szakaszonkénti titkosítást használ. A külső megoldásszállítók és a Mobile Information Server későbbi változatai -

amint az erre alkalmas telefonok megjelennek - valószínűleg további hitelesítési eljárásokat is támogatni fognak *(például a Digest Authenticationt és az ügyfélbizonyítványokat)*.

Az alábbi ábrán egy tipikus, Mobile Information Server-t tartalmazó hálózat látható, melyen külön ki vannak emelve a titkosítás szakaszai. A Mobile Information Server a két tűzfal közti demilitarizált zónában *(DMZ)* található. Ez az elhelyezés fokozza a biztonságot, mert a külső tűzfal a kívülről nyitott kapcsolatok közül csak a DMZ-ben levő kiszolgálók felé irányulókat engedélyezi, a belső tűzfal pedig csak a DMZ-ben levő kiszolgálók és a háttérkiszolgálók közti kapcsolatokat teszi lehetővé. Ily módon a mobil ügyfelek sosem kapcsolódnak közvetlenül a vállalat belső kiszolgálóihoz. *(Egyszerűbb esetben a DMZ a tűzfal harmadik „lába". Természetesen a fenti megoldás így is működőképes, a két tűzfalas megoldás viszont szemléletesebb.)*

A Mobile Information Server, és a titkosítás szakaszai.

Mindegyik szakaszban más biztonsági eljárást alkalmazunk:

- A WAP eszköz és a WAP kiszolgáló között az azonosítók a Wireless Transport Layer Security-val *(WTLS)* vannak titkosítva. Ez egy vezeték nélküli biztonsági és titkosító protokoll, melyet a WAP Forum fejlesztett ki. A WAP-os telefonok támogatják ennek használatát.
- A WAP kiszolgáló és a Mobile Information Server között a felhasználói azonosítók SSL-lel vannak titkosítva. Az SSL nyílt kulcsú és szimmetrikus kulcsú titkosítás kombinációját használja arra, hogy biztonságos kapcsolatot építsen ki egy a WAP kiszolgáló és a Mobile Information Server között, és az adatok átvitele előtt titkosítsa azokat. Az Internet szabványnak számító SSL-t már ma is gyakran használják érzékeny adatok *(például hitelkártyák száma, jelszavak)* titkosított átvitelére.
- A háttérkiszolgálók és a Mobile Information Server közötti adatbiztonságot egy virtuális magánhálózat *(VPN)* biztosítja, mely Internet Protocol Security-t *(IPSec)* használ. Az IPSec hitelesíti az adat küldőjét és megakadályozza a „spoofing"-ot *(egy elterjedt támadási forma, melynek során az illetéktelen felhasználó megszemélyesít egy jogosult felhasználót, így próbál hozzájutni érzékeny adatokhoz)*. Az IPSec támogatja az adatok titkosítását is, ezzel is segít megakadályozni az illetéktelen hozzáférést.

A felhasználói azonosítók mindig titkosítottak. *(Kivéve akkor, amikor éppen a WAP kiszolgálón vannak, és WTLS-ből éppen SSL-be huppannak át – a szerk.)*.

Hitelesítés

Mivel a Mobile Information 2001 Server alaphitelesítést és már megszokott biztonsági protokollokat használ, biztonsági beállításainak elvégzéséhez gyakorlatilag nem kell semmi újat tanulni. A biztonsági szabályok beállítása és a felhasználók hozzáadása egy MMC beépülő modul segítségével végezhető el. A Mobile Information Server használatához kétféle felhasználói fiók közül választhatunk:

- Mobilazonosítók—speciális külső tartományi fióknevek és jelszók, melyek csak a Mobile Information Serveren keresztüli vezeték nélküli távoli eléréshez használhatók.
- Natív azonosítók—szabványos Windows felhasználói fióknevek és jelszók.

Mobilazonosítók

A mobilazonosítók lehetővé teszik a WAP eszközök speciális, Windows 2000 „felhasználóként" történő kezelését. A mobilazonosítók elkülönített kezelése még a cég adatainak jobb védelmét is elősegíti.

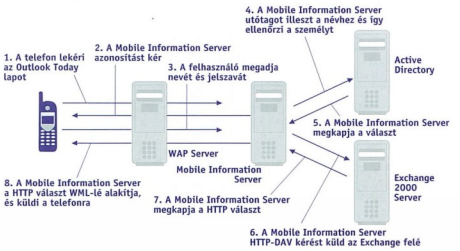
A mobilazonosítók alkalmazásának számos további előnye van:

- A mobil felhasználói fiókok intranethez és Exchange-hez való hozzáférése le van tiltva, ezért használatuk biztonságosabb. A felhasználók csak a saját Exchange postafiókjukat és azokat az intranet oldalakat érik el, melyek mobil elérését a cég jóváhagyja.
- A mobil és Windows felhasználói fiókok és jelszavaik eltérnek egymástól, ezért illetéktelenek nem használhatják a mobilazonosítókat a vállalati erőforrásokhoz való hozzáféréshez.
- A mobil felhasználói fiókok beállításaitól függően használhatók hozzájuk számokból álló jelszavak *(PIN)* a szokásos szám/betű kombinációk helyett. Ez azért jó, mert ezeket könnyebb beírni a mobil eszközök kis méretű billentyűzetén.
- A Mobile Information Server képes a mobil felhasználói fiókok jelszólejáratának ellenőrzésére, ezzel kikényszerít*(het)*i a jelszavak mobil eszközről történő frissítését.

A mobilazonosítók kétféle módon adhatók meg: külön biztonsági csoportban *(security group)* vagy külön tartományban. A mobil felhasználói fiókok beállításaitól függően a Mobile Information Server egy mobilfiók elő- vagy utótagot illeszt a felhasználó Windows fióknevéhez. A felhasználó bejelentkezésekor a Mobile Information Server automatikusan hozzáilleszti az elő- vagy utótagot a fióknévhez, ami szükségtelenné teszi, hogy a felhasználók több fióknevet használjanak. A felhasználóknak nem is kell tudniuk, hogy a mobil felhasználói fiókjuk eltér a Windows-ostól, az elő- vagy utótag számukra láthatatlan.

A következő ábrán látható, hogy hogyan ellenőrzi a Mobile Information Server a mobil fiókazonosítókat, és hogyan jeleníti meg az Exchange-en található tartalmat (jelen esetben a felhasználó Outlook Today lapját). A folyamat akkor kezdődik, amikor a felhasználó HTTP tartalmat kér. A Mobile Information Server kihívást *(challenge)* küld az ügyfélnek, melyben kéri a felhasználói azonosítót. Amikor az ügyfél elküldi az azonosítóit, a Mobile Information Server hozzáteszi a mobilfiók utótagot a felhasználó nevéhez, és ellenőrzi az azonosítókat az Active Directory-ban. Miután megkapta a felhasználó bizton-

sági jellemzőit, a Mobile Information Server átalakítja a HTTP kérést HTTP-Distributed Authoring and Versioning *(HTTP-DAV)* kéréssé, és továbbítja ezt az Exchange kiszolgálónak. Az Exchange elküldi a HTTP-DAV választ a Mobile Information Server-nek, ami ismét átalakítja azt, de most Wireless Markup Language *(WML)* formátumúra, amit egy WAP-os telefon meg tud jeleníteni.



Az Outlook Today elérése mobilazonosítók használatával.

Natív azonosítók

Ha telepítés közben nem adunk meg mobil utótagot vagy másik tartományt, a Mobile Information Server alapértelmezésben a Windows tartományt, felhasználói azonosítókat és jelszavakat fogja használni. Natív azonosítók használata esetén a felhasználók csak számítógépükről változtathatnak jelszót, mert a Mobile Information Server-rel nincs mód a szabványos Windows felhasználói fiókok jelszavának megváltoztatására.

A natív azonosítók használata leegyszerűsíti a felhasználói fiókok kezelését, mert lehetővé teszi az egyszeri bejelentkezést minden felhasználó számára, attól függetlenül, hogy milyen elérési módot és eszközt használ. Néhányan a szabványos tartományi azonosítók mobil környezetben való használatát bizonyára fölösleges biztonsági kockázatnak tartják, hiszen a tűzfalon kívül a jelszavak nem mindenhol titkosítottak. A kétféle hitelesítési mód biztosításával a Mobile Information Server lehetővé teszi, hogy a cégek kiválasszák a számukra előnyösebb azonosítótípust. A mobilazonosítók használata biztonságosabb, a natív azonosítóké viszont kényelmesebb.

A mobilalkalmazások és a biztonság

Várható, hogy kifejlesztésre kerülnek olyan új mobilkészülékek és alkalmazások, melyek a Mobile Information Server biztonsági modelljét használják. Néhány külső gyártó által készített készülék és alkalmazás biztosan képes lesz olyan Internetes protokollokat és jobb hitelesítési eljárásokat (például Digest Authentication) használni, melyeket a WAP eszközök jelenleg nem támogatnak. A Digest Authentication a felhasználók jelszavait titkosított formában küldi a mobil eszközről a Mobile Information Server-re, így ezzel a hitelesítési eljárással fölöslegessé válik a külön mobilazonosítók használata, hiszen a jelszavak sehol nem jelennek meg titkosítatlan formában.

A Microsoft Mobile Information 2001 Server szoftverfejlesztő készletében *(SDK)* megtalálható az a dokumentációkat és standard library COM modulokat tartalmazó eszközkészlet, melynek segítségével a fejlesztők megvalósíthatják az általuk készített alkalmazásokban az adatvédelmi és biztonsági funkciókat.

Jövőkép

A Mobile Information 2001 Server a mai vezeték nélküli eszközökkel megvalósítható legmagasabb szintű adatvédelmet és biztonságot nyújtja. A Mobile Information 2001 Server jövőbeni verziói továbbra is az Internetes és vezeték nélküli szabványok kombinációját fogják használni *(például SSL és WTLS)*, és támogatni fogják a például a következőket:

- Passport: lehetővé teszi, hogy felhasználói a vele együttműködő webhelyeket *(például MSN, Hotmail, Expedia és buy.com)* egyetlen felhasználói név és jelszó segítségével érjék el. A Passport tartalmaz egy opcionális „pénztárca" *(„wallet")* szolgáltatást, mellyel a felhasználók hitelkártyájuk számát, szállítási címüket és más olyan bizalmas információt tárolhatnak, mely az elektronikus vásárlások lebonyolításához szükséges. Minden passport tranzakció biztonságos SSL kapcsolatokon keresztül zajlik.
- Nyilvános kulcsú infrastruktúra *(PKI)*: olyan összetevők csoportja, melyek együttműködésével megvalósíthatók a digitális aláírások és a bizalmas adatok titkosítása.
- Intelligens kártyák: a PKI egyik legfontosabb eleme, mely biztosítja a bizalmas adatok megváltoztathatatlan tárolását, és lehetővé teszi a felhasználói azonosítók asztali gépek és mobil eszközök közti átvitelét.
- Microsoft Mobile Explorer: mikroböngésző, mely képes WML és HTML tartalom megjelenítésére, és végponttól-végpontig terjedő biztonságos adatátvitelt biztosít az SSL segítségével.

A Microsoft úgy látja, hogy az adatok biztonsága a vezeték nélküli Internet fejlődésével a végfelhasználóknak egyre fontosabb lesz. A Microsoft továbbra is igyekezni fog, hogy a Mobile Information 2001 Server következő verziói egyre biztonságosabbak legyenek, ennek érdekében alkalmazni fogja a legújabb biztonsági módszereket, hogy felhasználóinak biztonságos, titkos és hitelesített hozzáférést biztosítson bármely háttérkiszolgálón levő tartalomhoz.

További információ: http://www.microsoft.com/miserver.

Nevem: Senki

Sokan nem tudják, hogy minden Windows NT-ben *(és a Windows 2000-ben is)* található egy különleges felhasználó, akit úgy hívnak: „". Természetesen idézőjel nélkül. Szokták emlegetni anonymous felhasználóként is, de nem azonos a webkiszolgáló anonymous böngészőjével, az IUSR_<számítógépnév>, illetve a másik hasonló, IWAM_<számítógépnév> felhasználóval, ezért elterjedt a „null user" elnevezés is. „Senkit" hiába is keresnénk a felhasználók listájában, mert nincs sehol. Illetve valahol mégiscsak van: tagja az Everyone *(Mindenki)* csoportnak.

Kell ez nekünk?

Bizony, ha nem is nekünk, de – sajnos – szükség van rá. Számos olyan művelet, funkció létezik a Windows világban *(hálózatban)*, amihez az kell, hogy két vagy több számítógép együttműködjön. A null user-t *(„normális" esetben)* maga az operációs rendszer használja, amikor különféle okokból be kell jelentkeznie egy távoli számítógépre. *(Ilyen eset például maga a bejelentkezés (NetLogon), a megbízotti kapcsolat (trust) felépítése, és még sok más is.)* A távoli eljáráshívás (Remote Procedure Call, RPC) lehetővé teszi, hogy egy felhasználó, vagy akár program távoli számítógépen található programokkal kommunikáljon, persze csak a hálózati bejelentkezés után, a megfelelő jogosultságú felhasználó nevében. Amikor pedig a számítógép nem egy adott felhasználó nevében tevékenykedik, kénytelen a null user-t választani.
Lényeg, ami lényeg, a megfelelő eljárások távoli meghívása által a null user képes hozzáférni a felhasználók nevéhez, biztonsági beállításaihoz *(jelszóházirendhez, stb.)*, a számítógépre telepített, éppen futó rendszerszolgáltatások nevéhez, a megosztott könyvtárak listájához, és még sok minden máshoz is.

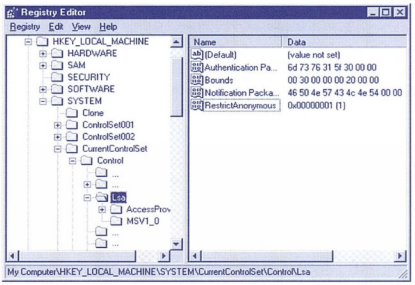
Valakik már eleget voltunk...

... legyünk inkább egy kicsit senkik – mondja a hacker. És tulajdonképpen nem is kell nagyon összetörnie magát az ügy érdekében. Amikor egy távoli számítógép erőforrásaihoz először szeretnénk hozzáférni, be kell jelentkeznünk. Az első bejelentkezés után *(míg csak ki nem jelentkezünk, vagy ki nem rúgnak)* az előzőleg megadott felhasználónévvel tevékenykedhetünk. Ha megpróbálnánk mást választani, jönne a jól ismert hibaüzenet: „The credentials supplied conflict with an existing set of credentials". Azaz, vagyunk, akik vagyunk, ne akarjunk más lenni.
Pontosan ez az, amit a „senkiség" érdekében kihasználhatunk: ha egyszer null user-ként bejelentkezünk a távoli számítógépre, a későbbi parancsaink már a null user nevében futnak le.
Próbálkozzunk egy kicsit! Legyen az áldozat számítógép neve „victim". Először jelentkezzünk be:

```
net use * \\victim\ipc$ "" /user:""
```

A parancs szintaxisa talán ismerős, az \ipc$ nevű rejtett megosztás minden Windows NT-n *(és Windows 2000-en)* megtalálható, a gépeken futó processzek közötti kommunikációs csatornák kiépítésére való belépési pont. Kénytelenek vagyunk ezt használni, mert a „hagyományos" megosztásokhoz a null user kevés lenne. Így viszont sikerrel járhatunk. És hogy ez mire jó?

Miután a null user az Everyone csoport tagja, a bejelentkezés után mindent megtehetünk, amit az Everyone megtehet. S milyen alapértelmezett jogosultsági beállításokat találunk a Windows NT alatt? Bizony, aki kapja, marja, legtöbbször a klasszikus Everyone – Full Control érvényesül.
Pontosan ezért találták ki a Windows NT 4 Service Pack 3-ban az Authenticated Users csoportot. Ez a csoport mindössze annyiban különbözik az Everyone-tól, hogy a null user nincsen benne. Ezért, amikor csak tehetjük, az Everyone helyett használjuk inkább az Authenticated Users-t. Számos lyukat befoltozunk ezzel, de sajnos azért marad még tennivaló.
A gond ugyanis az, hogy a korábban már felsorolt, biztonsági jellegű adatok lekérdezésére való függvények a távoli eljáráshíváson keresztül a null user égisze alatt továbbra is elérhetők. Szinte az összes biztonsági ellenőrző *(és kalóz-)* eszköz vigan listázza a felhasználóink nevét, hogy kinek mikor jár le *(és lejár-e egyáltalán)* a jelszava, és az is azonnal kiderül, ha az igazi Administrator fiókot átnevezték valami másra.
A Windows NT 4.0 Service Pack 3 szerencsére bevezetett egy másik újítást is, amit RestrictAnonymous néven ismerünk. Ez egy *(DWORD)* típusú registry érték, amit ha a HKLM\SYSTEM\CurrentControlSet\Control\LSA kulcs alatt létrehozunk, és értékét 1-re állítjuk, megnehezíti a biztonsági adatok és erőforrások lekérdezését.



Windows NT 4-en a RestrictAnonymous érték létrehozása és 1-re állítása létfontosságú

A beállítás hatására bizonyos távoli eljárások a null user számára nem lesznek elérhetők *(egyszerű jogosultsággal kapnak az eljárások, ahova a null user nem került bele)*. Így – gondolhatnánk – elbúcsúzhatunk a felhasználók, megosztások, biztonsági erőforrások távoli kilistázásától.

Semmi sem az, aminek látszik...

Timothy M. Mullen **[1]** *(és nyilván még sokan mások)* ugyanis vették a fáradtságot, és kipróbálták, hogy valójában mit tilt, és mit nem a RestrictAnonymous. Nyilvánvaló, hogy mindent nem tilthat, hiszen ez a funkcionalitás a Windows NT hálózati működésének alapját képezi. Kiderült, hogy például a felhasználónevet és a hozzátartozó biztonsági azonosítót *(SID)* összerendelő függvények továbbra is működnek. Ha valakinek a birtokában van a user2sid és sid2user nevű eszköz, kipró-

bálhatja, hogy a RestrictAnonymous beállítás dacára is sikeresen használhatók. Szabadon maradt ezen kívül egy olyan eljárás is, ami egy adott azonosítójú felhasználó biztonsági adatait adja vissza... A **[2]** címről letölthető userinfo.exe nevű kis eszköz a következő választ adja egy ismert nevű felhasználó lekérdezésére *(Windows 2000-nél is!)*:

```
C:\>userinfo \\victim user1

        UserInfo v1.5 - thor@hammerofgod.com

        Querying Controller \\victim

        USER INFO
        Username:         user1
        Full Name:        Elso felhasznalo
        Comment:          (ures jelszoval)
        User Comment:
        User ID:          1004
        Primary Grp:      513
        Privs:            User Privs
        OperatorPrivs:    No explicit OP Privs

        SYSTEM FLAGS (Flag dword is 66049)
        User's pwd never expires.

        MISC INFO
        Password age:     Sat Mar 10 22:46:47 2001
        LastLogon:        Thu Jan 01 00:00:00 1970
        LastLogoff:       Thu Jan 01 00:00:00 1970
        Acct Expires:     Mon Dec 31 23:00:00 2001
        Max Storage:      Unlimited
        Workstations:
        UnitsperWeek:     168
        Bad pw Count:     0
        Num logons:       0
        Country code:     0
        Code page:        0
        Profile:          \\suicide\user1
        ScriptPath:       userlon.cmd
        Homedir drive:
        Home Dir:         c:\users\user1
        PasswordExp:      0

        Logon hours at controller, GMT:
        Hours-          123456789O1N123456789O1M
        Sunday          000000000000000000000000
        Monday          000000011111111110000000
        Tuesday         000000011111111110000000
        Wednesday       000000011111111110000000
        Thursday        000000011111111110000000
        Friday          000000011111111110000000
        Saturday        000000000000000000000000
```

A korlátozott hozzáféréshez képest elég bőbeszédű adathalmaz, nem? :-) És még nincs vége.

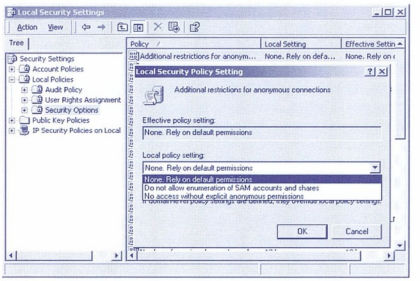
Aki ismeri a SID felépítését, az tudja, hogy a SID egy sokjegyű azonosító, ami tartalmazza a tartomány és a felhasználó azonosítóját egyaránt. Ugyanazon tartomány felhasználóinak SID-je csak az úgynevezett relatív ID-ben *(RID)* különbözik, az *(igazi)* adminisztrátor RID-je, bárhogy is hívják, 500, a létrehozott felhasználók RID-je valahol 1001-nél kezdődik és az újabb felhasználók létrehozásával folyamatosan nő.
Kíváncsiak vagyunk a felhasználók listájára? Fogjunk egy ismert felhasználónevet *(administrator, guest)*, keressük ki a hozzá tartozó SID-et, vágjuk le a végét, majd ragasszunk hozzá tetszőleges RID-t, és kérjünk információt a felhasználóról, ha létezik. Ha nem, próbálkozzunk más RID-vel. Pofonegyszerű, nemde? A **[3]** címről letölthető userdump.exe pontosan ezt teszi. (Itt jegyezném meg, hogy az adminisztrátor és guest felhasználó átnevezése ezt a trükköt megnehezíti. Sebaj, akkor használjunk egy csoportnevet, vagy akár a számítógép nevét :-))...

Mit tehetünk akkor?

Windows NT 4 hálózatban használjunk tűzfalat. Mivel a távoli eljáráshívás *(RPC)* a 139-es TCP portot használja, e port blokkolása megoldja a problémát. Legalábbis a tűzfalon kívülről érkező támadások esetén. További lehetőség a NetBIOS over TCP/IP letiltása, de akkor el kell búcsúznunk a Windows hálózati funkcióktól. Egy átmeneti megoldás lehet, hogy egy virtuális hálózati kártyát *(MS Loopback Adapter)* telepítünk a gépre, és csak arra engedélyezzük a NetBIOS használatát. A NetBIOS szolgáltatások ekkor a számítógépen kívülről ugyan nem érhetők el, de legalább maga az operációs rendszer, és a NetBIOS kommunikációt igénylő helyi komponensek, szolgáltatások nem szenvednek csorbát.

Van remény?

A Windows 2000-ben a biztonsági házirenden keresztül is hozzáférhetünk a RestrictAnonymous beállításhoz:



Anonymous beállítások a Windows 2000 biztonsági házirendjében

Látható, hogy itt egy harmadik lehetőség is van: „No access without explicit anonymous permissions". Ha ezt beállítjuk *(ami egyébként a 2-es RestrictAnonymous érték)*, a null user nem kerül bele az Everyone csoportba, így már annyit sem tehet, mint eddig. Ezt a beállítást viszont csak akkor használhatjuk, ha tiszta Windows 2000 környezetben vagyunk; a Windows 9x, Windows NT ügyfelek többé nem lesznek képesek bejelentkezni a rendszerbe.

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:

[1] http://www.securityfocus.com/focus/microsoft/nt/restrict.html
[2] http://www.securityfocus.com/tools/1930
[3] http://www.securityfocus.com/tools/1931

Transact SQL *(VI. rész)*

Bevezetés

A mai, korszerű adatbázisok egyik legfontosabb jellemzője, hogy sokan használják egyidejűleg. Mivel a felhasználók, alkalmazások egymástól függetlenül próbálják meg módosítani a táblák tartalmát, gyakori a konfliktushelyzet. Ilyenkor kezdenek lelassulni a rosszul megtervezett adatbázisok, és jönnek az időtúllépésről, valamint a misztikus dead-lock-okról szóló hibaüzenetek, nem beszélve a logikailag hibás adatokról. Ebben a részben részletesen kitárgyaljuk a zárolások okait és fajtáit, a következő számunkban pedig a dead-lock-ok misztikus világáról lebbentjük le a fátylat. Cikksorozatunk mostani fejezete elég nehéz, ám annál fontosabb témakörrel foglalkozik, ami nélkül igen nehéz megbízható és hatékony adatbázisokat tervezni az SQL Server 2000-re.

Optimista vagy pesszimista?

Nézzünk meg egy klasszikus ügyfél-kiszolgáló alkalmazást, ami kurzorok használatával módosítja az adatokat. A legtöbb Visual Basic és Visual C++ alkalmazás ilyen.

Tegyük fel, hogy van egy adatbázis, amely egy cég alkalmazottait tartja nyilván. Kiss Béla cégen belüli pozíciója megváltozik, kap egy Senior jelzőt a rangja elé. Ezzel együtt a lakcíme is megváltozott, amiről külön értesíti az egyik HR-es hölgyet. Az emberi erőforrás menedzsmenten lelkes emberek dolgoznak, és azonnal nekilátnak a változás adatbázisba rögzítésének. A lelkesedésük nagyobb, mint a munkaszervezettségük, így egyszerre ketten kezdik el módosítani a kérdéses alkalmazott adatait az adatbázisban. Tegyük fel, hogy mindkettejük előtt ki vannak listázva Béla adatai, és nekiállnak módosítani a rekordot. Az első a lakcímet és a rangot, a második csak a rangot írja át. Megnyomják az „Elment" gombot, és tegyük fel, hogy az első hölgy a gyorsabb. Mi történik?

Két eset lehetséges. Egy butább adatbáziskezelő vagy egy rosszul megírt ügyfélalkalmazás esetén az első ügyfél által kért változtatás beíródik az adatbázisba, amit követ a második ügyfél módosítása. Mivel mindketten ugyanazokból a kiinduló adatokból módosított adatokat írják vissza, a második módosítás fejbe csapja az elsőt, azaz a végleges rekordban nem lesz módosítva a lakcím, mert a második hölgy csak a rang mezőt módosította. A probléma nem az, hogy ez így megtörténhet, hanem az, hogy az ügyfélprogramok nem is szereznek róla tudomást, hogy módosításvesztés történt. Az iménti helyzetben felvázolt helyzetet hívjuk az elveszett módosítások problémájának. Hogyan védekezik az ilyen helyzetek ellen egy okos adatbáziskezelő? Amikor egy ügyfélprogram lekér egy adott rekordot az adatbáziskezelőtől, akkor a szerver megjegyzi, hogy valaki letöltötte a rekordot, mert módosítani szeretné. Amikor más ügyfelek is szeretnék ugyanezt megtenni, akkor kétféle dolog történhet. Ha az első ügyfél optimista zárolás felhasználásával kérte le a rekordot, akkor a hasonlóan eljáró további ügyfelek is megkapják a rekordot. Azonban módosítás visszaírási kísérlet esetén az adatbáziskezelő megnézi, hogy megváltozott-e az adatbázisban tárolt sor a korábban lekért állapothoz képest *(annyira nem optimista, hogy vakon megbízzon benne, hogy nem változott* :). Ha igen, akkor a próbálkozónak már csak egy hibaüzenet jár, ami arról tájékoztatja, hogy a módosítani kívánt rekordot már valaki más módosította:

```
Optimistic concurrency check failed. The row was
modified outside of this cursor.
```

Ilyenkor nincs mit tenni, újra le kell kérni a módosított adatokat, újra beírni a változtatásokat, és újra megpróbálni beküldeni a változtatási kérelmet. Ha ezúttal mi voltunk a leggyorsabbak, akkor nyertünk, és a mi módosításunk lesz érvényes. Ha nem, try again... Nyilvánvaló, hogy egy olyan rendszerben, ahol gyakoriak a módosítások, ott nem megfelelő ez az eljárás, mert túl gyakoriak az ütközések.

A másik stratégia úgy gondolkodik, hogy ne ringassuk hiú ábrándokba a második, harmadik, satöbbi ügyfelet, hanem az első alkalmazás, ami módosítani akar egy rekordot lefoglalja azt, és a többiek addig nem is tudják lekérni a rekordot mindaddig, amíg az első fel nem oldja a zárolást. Ezt a stratégiát pesszimista zárolásnak hívjuk. Ez is egy elfogadható hozzáállás, ráadásul egyszerűbb implementálni a várakozást, mint lekezelni a sikertelen módosítást. *(Gyakorlatilag nem kell tenni semmit, mert az adatbázist elérő metódus nem tér vissza addig, amíg a módosítandó rekord fel nem szabadul.)*

Például egy helyfoglaló rendszernél csak ez a módszer tud helyesen működni, hisz optimista esetben az operátor még szabadnak láthat olyan helyeket, amelyeket már rég lefoglaltak más operátorok. Inkább ne is láthassa azokat a helyeket, amelyeket éppen valaki más próbál lefoglalni.

Az eddigi példában olyan helyzetről beszéltem, amikor a rekordokat kurzor segítségével kértük le, és a kapott recordseten keresztül módosítjuk az adatokat. Ez a fajta megoldás a mai világban egyre ritkább, és különösen a Webalkalmazásokban nem ilyen módon kezeljük az adatokat. Azokban általában tárolt eljárások segítségével módosítjuk a sorokat. Ilyenkor már nagyon könnyen fejbe lehet csapni a konkurens módosítások eredményét, hisz az adatok lekérése és a módosított adatok visszaírása közben megszakad az ügyfélprogram *(a Webalkalmazás)* kapcsolata az adatbázissal, így az adatbázisnak még esélye sincs arra, hogy zárolással vagy Voodoo varázslással megóvjon minket a módosítások elvesztésétől.

Tegye a szívére a kezét minden Webalkalmazás fejlesztő! Gondolt már valaha erre a problémára? Vagy csak mechanikusan visszaírja a módosított eredményeket a forrástáblába? Vesszen a lassabb? Az ADO természetesen ilyen helyzetekre is biztosít megoldást, de használjuk ezeket? *(A jövőben mindenképpen áldozunk egy-két cikket a témának.)*

SQL Server tranzakciók

Szakadjuk el egy kicsit a kurzort használó ügyfélprogramoktól, és evezzünk át a tiszta SQL Server megoldásokhoz, valamint a tárolt eljárásokat használó alkalmazásokhoz. Nézzük meg, hogy a tranzakciók során mennyire vagyunk védettek mások adatmódosításai ellen.

Kiindulásként álljon itt egy kérdés. Alapértelmezett beállítások mellett biztos lehetek benne, hogy egy tranzakción belül háborítatlanok maradnak az általam használt táblák, miközben mások is dolgoznak az adatbázisban? Legtöbben azt gondolják, igen. Ha biztos akarok lenni abban, hogy a tábláimat nem változtatják meg a hátam mögött a tranzakcióm alatt, akkor elég BEGIN TRAN és COMMIT TRAN közé rakni az utasításaimat, és minden rendben lesz? Biztos? Egyáltalán nem. Járjuk körbe ezt a témát, mert ennek megértése nélkül senki nem mondhatja el magáról, hogy konzisztens adatbázist tud tervezni.

A zárolások fajtái

Annak érdekében, hogy az SQL Server szabályozni tudja az adatokhoz való párhuzamos hozzáféréseket, a védendő adatokra zárolásokat helyez el. Az SQL Serverben többféle zárolási típus van, és mindegyiknek van egy meghatározott viselkedése. Például más zárolást kell használjon a szerver az adatok olvasása során *(SELECT)*, hisz ilyenkor általában csak azt kell megakadályozni, hogy más tranzakció módosítsa az éppen olvasás alatt álló adatokat. Ezzel szemben például egy adatmódosító tranzakció közepette nem lenne szerencsés engedni a többi tranzakciót, hogy olvassa az éppen módosítás alatt álló adatokat, plane, hogy módosítsa ugyanazt. Nyilván ehhez másféle zárolásra van szükség. Tekintsük át a legfontosabb zárolási típusokat!

Mint említettük az adatok olvasása során meg kell akadályozni, hogy az éppen kiolvasott adatokat mások módosítsák az olvasási művelet közben, de meg kell engedni, hogy mások is olvashassák, hisz az veszélytelen a mi tranzakciónkra nézve. Ehhez az SQL Server Shared lock-okat helyez el az olvasott adatokra *(a könnyebb követhetőség kedvéért nem fordítottam le a zárolások nevét, és az egyszerűbb olvashatóság miatt a zárolás eredetijét, a lock-ot is meghagyom)*.

Ha egyszerre több tranzakció is olvassa ugyanazt az adatot, akkor mindegyik elhelyezi a maga Shared lock-ját a rekordokon, és addig rajta is tartja, amíg nem végez az olvasással. Az adatmódosító utasítások *(INSERT, DELETE és UPDATE)* alatt nem szabad másnak olvasni a módosítandó adatokat, ilyenkor a szerver Exclusive lock-ot helyez el a sorokon. Az Exclusive lock mellett más nem helyezhet el semmilyen zárolást a sorokra, meg kell várnia, míg az adatmódosítás befejeződik, és a tranzakciót így vagy úgy, de be nem fejezik. A legtöbb esetben ezzel az esettel kerülnek szembe az adatbázisfejlesztők és üzemeltetők, azaz, hogy egy hosszú ideig tartó adatmódosító tranzakció zárol egy bizonyos adatmennyiséget, így az egyéb adatolvasó vagy módosító tranzakcióknak várni kell a módosítás befejezéséig. Ezt sokan tévesen deadlock-nak azonosítják, pedig ennek semmi köze nincs ahhoz. Egyszerűen csak egy hosszú idejű tranzakció blokkolja a többi tranzakció munkáját. Az SQL Server Enterprise Manager Management, Current Activity, Lock/Process ID alatt találhatjuk meg a szerveren a zárolásokat megjelenítő grafikus alkalmazást. Ennek segítségével azonosítható az a tranzakció, ami blokkolja a többit *(felkiáltójeles emberke ikon)*. Ezen a nyomon elindulva meg lehet keresni, és át lehet írni a bűnös tranzakciót. Aki nem szereti a grafikus felületeket, annak az sp_lock tárolt eljárást ajánlom a zárolások megfigyelésére.

Az UPDATE rendhagyó művelet a többi háromhoz képest, mert az első fázisban fel kell olvasnia a módosítandó adatokat, a másodikban pedig módosítani azt. Emiatt az olvasási részben Shared lock-ot kell elhelyezzen az adatokon, a módosítás során pedig Exclusive lock-ot. Az ő kettős természete miatt kapott is egy saját zárolási típust, amit Update lock-nak hívnak. Az UPDATE az adat olvasási fázisban Update lock-ot rak a sorokra, és a tényleges módosítás megkezdése előtt felemeli azt Exclusive lock-ra. Azért nem Shared lock-ot használ, mert az Update lock nem engedi meg, hogy mások is igényeljenek Update lock-ot ugyanazokra az adatokra, így nem tudja más megmódosítani az adatokat a felolvasás és a módosítás között. A dead-lock-ok megelőzésében nagyon fontos szerepe van az Update lock-nak, amiről a következő számban írok bővebben.

Schema Modification lock-ot az adatbázis szerkezetét módosító utasítások *(például ALTER TABLE)* helyeznek el a megfelelő objektumokon, hogy közben nehogy mások is megpróbálják ugyanazt módosítani.

A lekérdezések fordítása közben a szerver Schema Stability lock-al akadályozza meg a lekérdezésben szereplő táblák és egyéb objektumok szerkezetének módosítását.

A zárolások finomsága

Eddig elég homályosan fogalmaztam meg, hogy az SQL Server valójában mekkora adatmennyiségeket zárol a tranzakciók során. Most nézzük meg, hogy milyen egységekben tud adatokat zárolni a szerver.

A legfinomabb zárolási egység a sor. Ez képes egyetlen rekord zárolására, azaz miközben egy sort módosítunk, egy másik tranzakció képes a mellette található sor *(reklord)* olvasására vagy módosítására.

Ha egy lapon *(8 kByte-os egység, amely a sorokat tartalmazza)* sok sort kellene zárolni, akkor a szerver inkább zárolja a teljes lapot, ahelyett, hogy sok sor-zárolást kellene nyilvántartania. Egyes esetekben, amikor olyan sok módosítás történik, hogy az szinte egy egész tábla tartalmát érinti, a szerver inkább zárolja az egész táblát, semmint egyedi lapokat, ezzel a zárolások nyilvántartásához szükséges erőforrásokat spórolva.

Az SQL Server automatikusan választja ki, hogy mikor milyen finomságú zárolásra van szükség. A tranzakció által érintett sorok számától függően keres olyan szintű zárolást, ami még elég finom ahhoz, hogy ne korlátozza jelentősen a többi tranzakció futását, de ne is kelljen nagyon sok lock-ot nyilvántartania. A szerver egy tranzakció lefutása közben is képes változtatni a zárolás finomságát. Lehet, hogy elindul sorzárolással, ám a sorok zárolása közben észreveszi, hogy már olyan sok sort kell nyilvántartania, hogy érdemesebb lenne áttérnie az egész tábla zárolására. Ezt a folyamatot, amikor egy finomabb, de nagy számú zárolásról a szerver áttér egy durvább, nagyobb tartományra ható, de kevesebb számosságú zárolásra zárolás eszkalációnak *(Lock Escalation)* hívjuk. Ha tudjuk, hogy a tranzakciónk nagyon sok sort fog érinteni, akkor lehet, hogy érdemes a szervernek súgni, hogy nem érdemes sorzárolástól indulva végiglépkednie a zárolásokon, hanem rögtön kezdje például tábla szintű zárolással. Lehet, hogy így olyan tranzakciókat is blokkolunk, amelyeket sor vagy lap zárolással nem befolyásolnánk, de a kis számú zárolás nyilvántartása miatt a tranzakciónk lehet, hogy sok-

kal gyorsabban fut le, így végeredményben kevesebb blokkolást okozunk a többi tranzakció felé.

Más esetben lehet, hogy az SQL Server egy egész táblát zárolna, és így más tranzakciók nem tudnának abban dolgozni, például adatokat beszúrni. Tipikus példa erre, amikor egy hosszú idejű lekérdezést futtatunk, ami múltbeli adatokkal foglalkozik, miközben záporoznak be a táblába a mai naphoz tartozó sorok. Lehet, hogy a lekérdezés akár a sorok első 99%-át érinti, így a szerver nyilvánvalóan egy darab tábla zárolással lefoglalja a tranzakciónk számára a táblát, ám így az adatokat beszúró alkalmazás nem tud írni sorokat a tábla végébe. Ilyenkor lehet, hogy például lapszintű zárolást erőltetve a tranzakciónk nem 5 perc, hanem fél óra alatt fut le a sok zárolás adminisztrációja miatt, de eközben az adatokat beszúró alkalmazás egy pillanatig sem állt le. Azaz vannak esetek, amikor szélesíteni akarjuk a zárolások tartományát, és vannak, amikor szűkíteni, az alkalmazásunk logikájától függően. Hogyan befolyásolhatjuk az SQL Servert a zárolások finomságát illetően? A kérdésre a lock hint-ek adnak választ, a cikk utolsó részében.

A végére hagytam egy különleges zárolási típust, amely az előbbiekkel ellentétben nem fix méretű zárolást valósít meg. Ez az index-tartományzárolás. Bizonyos esetekben *(SERIALIZABLE tranzakciók, lásd később)* szükség van arra, hogy egy lekérdezés WHERE feltételében definiált határok között ne lehessen új adatokat beszúrni. Például lekérdezzük az 5 és a 13 közötti azonosítójú sorokat, és nem szeretnénk, ha a tranzakciónk alatt valaki más beszúrna új sorokat olyan azonosítóval, amely 5 és 13 közé esik. Ebben az esetben a szerver az indextartomány két végét lezárja Key lock-kal, így a megadott tartományba nem enged új sorokat beszúrni. Ennek a zárolásnak a hossza nyilvánvalóan nem fix, hanem a lekérdezés függvénye. Természetesen ez a zárolás csak akkor tud működni, ha a tartományokat definiáló mezőre van index létrehozva. Ha nincs, akkor a szervernek nincs mit tennie, tábla zárolást kell alkalmaznia.

Zárolás kompatibilitás

Mi történik, ha az egyik tranzakció zárolásokat helyez el bizonyos adatmennyiségen, miközben mások ugyanezt akarják megtenni, ugyanazokra az adatokra? Ez attól függ, hogy milyen zárolás van éppen az adatokon, és milyet igényel egy másik tranzakció.

Vannak zárolások, amelyek szeretik egymást, és vannak, amelyek nem. Nyilvánvaló, hogy a Shared lock szereti a Shared lock-ot, azaz, ha az egyik tranzakció olvassa az adatokat, és emiatt Shared lock-okat helyez el az olvasott sorokon, a másik tranzakció veszélytelenül felolvashatja ugyanazokat a sorokat, azaz ő is elhelyezheti a Shared lock-jait ugyanazokon a sorokon. Ha eközben egy harmadik résztvevő is beszáll, aki módosítani akarja a kétszeresen is zárolt *(Shared módon)* sorokat, akkor neki bizony várnia kell egészen addig, amíg a másik két tranzakció be nem fejezi az adatok olvasását, és le nem veszi a lock-jait. Ez is a klasszikus blokkolás esete, amikor egy adatmódosító utasításnak várnia kell arra, hogy elhelyezhesse az Exclusive lock-jait az adatokon. Miután kivárta a sorát, és felrakta a kizárólagosságát biztosító zárolását, senki más semmilyen zárolást nem tud elhelyezni mindaddig, amíg az be nem fejezi a módosító tranzakciót, és le nem veszi az Exclusive lock-ot. Nyilván ebből adódik e zárolás neve is.

Azaz abban az esetben, ha egy tranzakció szeretne valamilyen zárolást elhelyezni egy adathalmazon, az SQL Server ellenőrzi, hogy a már fennálló zárolások alapján kiadható-e a kért típusú zárolás. Ha igen, akkor megkapja, a zárolás feljegyzésre kerül, és a trónkövetelő tranzakció megkezdheti a munkáját. Amennyiben viszont olyan zárolást kért, ami logikailag nem összeegyeztethető a már meglévőkkel, akkor a zárolást kérő utasítást a szerver mindaddig felfüggeszti, amíg meg nem szűnnek az akadályozó zárolások. Az igényt természetesen feljegyzi, és a többi zárolás fokozatos „lehullása" alatt mindig ránéz, hátha már kiadható a kért zárolás. Miközben az igénylő vár a lock-jára, lehet, hogy más tranzakciók is jelentkeznek zárolási igénnyel, és azok között akár olyan is lehet, ami összeegyeztethető lenne a már fennálló zárolásokkal. Ilyenkor mit tegyen a szerver? Engedje őket, hogy elhelyezzék a saját zárolásaikat, vagy addig ne engedje őket szóhoz jutni, amíg a már régóta várakozó tranzakció meg nem kapja az áhított zárolását? Ha engedi őket, akkor azok lefuthatnak a várakozó előtt, ám előfordulhatna az, hogy a sok újabb és újabb kérő soha nem engedné, hogy a várakozó megkapja a zárolását. Azaz ezzel a stratégiával kiéheztetnénk azokat a tranzakciókat, amelyek olyan zárolásokat kérnek, amelyek általában nem kompatíbilisak a már meglévőkkel. A gyakorlatban ez azt jelentené, hogy egy módosító utasítás soha nem kapná meg az Exclusive lock-ját, ha az egymás után érkező olvasó *(SELECT)*, Shared lock-okat elhelyező utasításokkal operáló tranzakciók időben átlapolják egymást. Nyilván ezt nem engedhetjük meg. Emiatt az SQL Server nem engedi zárolni a további kérőket, amíg a már fennálló zárolási igényeket nem elégítette ki. Ez persze azt is jelenti, hogy egy adatmódosító utasítás után akár hosszú sorokban állhatnak a csak olvasni akaró tranzakciók, akik ugyan nyugodtan olvashatnák a Shared lock-kal védett sorokat, de nem tehetik, mert a módosító utasítás vár az Exclusive lock-jára. Gyönyörű hosszú blokkolási láncok tudnak így kialakulni. Mit lehet tenni ellenük?

A legegyszerűbb védekezés, hogy a módosító tranzakciókat nagyon rövidre tervezzük. Nem szabad egy adatmódosító tranzakcióba felhasználói beavatkozásra váró rutint elhelyezni! Mi van, ha közben elmegy ebédelni? Mire visszaér, az adatbázis adminisztrátor már a tízezredik feltorlódott tranzakciót fogja látni a le nem zárt módosító tranzakció miatt! Természetesen ezt nem szabad megengedni.

A másik eszközünk a zárolás finomságának állítása, azaz nem hagyjuk, hogy a módosító tranzakció túl nagy falatot zároljon le kizárólagosan a táblákból. Erre valók a lock hint-ek, amelyekről hamarosan szólok.

Egy valamiről még nem beszéltem. Honnan tudja az SQL Server, hogy melyik zárolási típus melyik másikkal kompatíbilis? Nos, erre a célra van egy táblázata, és abból olvassa ki. Ezt a táblázatot az SQL Server tervezői alkották meg, figyelembe véve az egyes zárolások természetét, és hogy melyik futhat párhuzamosan a másikkal anélkül, hogy az adatbázis épségét veszélyeztetné. A Books Online a Lock Compatibility című fejezetben ismerteti ezt a táblázatot.

Az Intent lock-ok

Megnéztük, hogy az SQL Server csak akkor helyez el egy újabb zárolást ugyanazon az adaton, ha az igényelt zárolási típus kompatíbilis a már fennállóval. Azonban hogyan hasonlít össze különböző finomságú zárolásokat? Ha van egy Exclusive lock egy soron, akkor rakható Shared lock ugyanarra a táblára? Ilyen kérdőjeles helyzet nagyon sok kialakul, hiszen minden tranzakció más finomságú zárolást használhat. Nézzünk erre egy példát.

Az első tranzakció Shared lock-kal lefoglal 3356 sort egy táblában. Egy másik tranzakció lefoglal 10 lapot Exclusive módon. Van még 23 éppen futó tranzakció, amelyek 12354 darab Shared és 5 darab Exclusive lap szintű lock-ot tartanak a táblán. Ezután egy sokadik tranzakció tábla szinten szeretne Shared lock-ot. Mit tud tenni a szerver, hogy megállapítsa, megkaphatja-e? Végig kell néznie az összes *(3356+10+12354+5 darab)* zárolást, és meg kell keresnie, hogy van-e köztük olyan, amelyik Exclusive módon birtokolja a tábla valamely szeletét. Ha van, akkor nem adhatja ki a tábla szintű Shared lock-ot. Ha közben egy-egy tranzakció befejeződik, és engedi el a zárolásait, akkor a lock manager-nek minden esetben végig kellene nézni az összes még megmaradt zárolást, hogy maradt-e még Exclusive, és ha már nem, akkor kiadható a tábla szintű Exclusive lock. Ez az eljárás igen lassú volna. Ennek elkerülésére az SQL Server trükkösen foglalja le a kisebb finomságú *(sor, lap, extent)* zárolásokat. Ha egy tranzakció elhelyez akár csak egy sornyi zárolást is egy táblán, akkor ezzel együtt a szerver elhelyez egy ugyanolyan típusú *(Shared vagy Exclusive)* lock-ot a sort tartalmazó lapra és táblára is, ám azt csak szándéknyilatkozatként Intent Shared vagy Intent Exclusive-ként megjelölve. Ezek után a teljes táblára Exclusive lock-ot kérő tranzakció igénye könnyen eldönthető, hisz elég megnézni, hogy van-e nem kompatíbilis Intent lock a táblán.

Ez az eljárás nemcsak tábla szinten működik, hanem minden olyan szinten, amikor egy kisebb finomságú zárolást kér egy tranzakció. Így egy Exclusive sor lock-ot kérő tranzakció kap egy „valódi" Exclusive lock-ot a soron, és kap egy lap és tábla szintű Intent Exclusive-et is. Ha az Intent lock-ok elhelyezése közben kiderül, hogy a sort tartalmazó lapon már van egy Shared lock, akkor a sorra sem szabad kiadni az Exclusive lock-ot, mert előfordulhat, hogy belemódosítunk olyan sorba, amit valaki más olvas lap szinten *(pont ezért rakott rá Shared lock-ot)*. Azaz az exkluzív sor-zárolás kiadását megakadályozhatja egy, a sort tartalmazó lapon már létező Shared lock, így az Intent lock-ok elhelyezése *(helyesebben meghatározása)* közben kiderül a zárolási igény kompatibilitási kérdése is.

A tranzakciók elszigeteltségi szintjei

Láttuk, hogy a párhuzamosan futó tranzakciók többé-kevésbé hatnak egymásra, befolyásolják egymás működését. Természetesen egy adatbázisban nem alapozhatunk „többé-kevésbé" szabályokra, valamilyen egzakt módszer kell annak eldöntésére, hogy miközben az egyik tranzakció valamit működik egy táblán, a többi tranzakció ebből mit lát, illetve mit tehet a kérdéses táblával. Ennek a kérdésnek a szabályozásával az ANSI SQL 92-es szabvány részletesen foglalkozik, és ad is ajánlást egy lehetséges megvalósításra. A szabvány a tranzakciók elszigeteltségét négy szintre bontja. Minél inkább haladunk előre a szintekkel, annál kevesebb hatással vannak egymásra a tranzakciók, cserébe annál kisebb az esély a tranzakciók párhuzamos végrehajtására. Az egyik oldalon nyerünk valamit, cserébe a másikon veszítünk. SQL Serverben az elszigeteltségi szinteket a tranzakciók belsejében lehet beállítani a

```
SET TRANSACTION ISOLATION LEVEL szint
```

utasítással. Az utasítás hatására a tranzakcióban szereplő összes SELECT utasítás az adott elszigeteltségi szintnek megfelelően fogja olvasni az adatokat, illetve elhelyezni a zárolásokat a már olvasott adatokon. A tranzakció belsejében bármikor át lehet térni más elszigeteltségi szintre, és onnantól kezdve a SELECT-ek annak megfelelően fognak működni. Ez azonban nem jelenti azt, hogy az előtte levő SELECT-ek által lefoglalt zárolások feloldódnának, csak azt, hogy az ezután kiadottak az új szintnek megfelelően fognak viselkedni. Igazából nem sok szituáció indokolja a szintek váltogatását egy tranzakció során, általában az elején beállítunk egy nekünk megfelelő szintet, és azt használjuk a tranzakció végéig. Lássuk hát a négy szintet!

1. READ UNCOMMITTED *(dirty read)*

Ezen a szinten a tranzakcióban szereplő utasítások bármilyen adatot kiolvashatnak a táblákból, függetlenül attól, hogy az adott sort/lapot/táblát zárolta-e valamely más folyamat. Ez azt is jelenti, hogy olyan adatokat is olvashat, amik még nincsenek véglegesen lerögzítve az adatbázisba, azaz a módosító tranzakció végén még nem volt COMMIT TRAN, és lehet, hogy a következő pillanatban visszavonják. Másképpen fogalmazva fizikailag helyes adatokat fogunk kiolvasni, azonban logikailag nem biztos, hogy helyeset.

Ez az elszigeteltségi szint üzleti tranzakciókban elfogadhatatlan, hisz ott csak akkor fogadhatunk el egy adatot érvényesnek, ha az őt beszúró vagy módosító folyamat véglegesítette a változtatását.

Azonban sokszor nem fontos az adatok hajszálra menő precizitása, de fontos, hogy a tranzakciónk ne blokkoljon más tranzakciókat a sok és hosszú idejű kiolvasás által generált zárolásokkal, valamint, hogy a módosító tranzakciók ne akadályozzák a lekérdezésünk futását. Általában statisztikák és trendek analízise, kimutatások és összesített eredmények számolása során nem baj, ha beveszünk a számításba néhány olyan sort, amelyek esetleg egy másodperc múlva már nem is léteznek, de cserébe gyorsan lefut a tranzakciónk. Ilyenkor nagyon jól jön ez az elszigeteltségi szint.

Nézzük meg, hogy ezen a szinten egy SELECT hatására milyen zárolások keletkeznek az adatbázisban:

```
BEGIN TRAN
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT
  *
FROM
  Employees
WHERE
  LastName LIKE 'B%'
EXEC sp_lock @@SPID

COMMIT TRAN
```

Kimenet:

```
LastName             FirstName
-------------------- ----------
Buchanan             Steven

(1 row(s) affected)

spid dbid ObjId    IndId Type Resource Mode Status
---- ---- ------ ------ ---- -------- ---- ------
51   6    0           0    DB           S    GRANT
```

A kimenetben csak azokat a sorokat hagytam meg, amelyek a 6-os dbid-jű adatbázisra vonatkoznak, ami a vizsgált Northwind. Mit jelent ez a kimenet? Az első zárolásokról szóló sor azt mutatja, hogy szerver elhelyezett egy Shared lock-ot a 6-os adatbázisra *(Northwind)* adatbázis szinten. Ezt csak azért tette, hogy a tranzakció alatt ne forgassák fel alapjaiban az adatbázist, ám semmi más zárolás nem látszik.
Sajnos azt nem látjuk, hogy a kiolvasott sorokat még a SELECT lefutása idejére sem zárolta a szerver, mert mire a végrehajtás az sp_lock-ra kerül, a zárolások *(ha lettek volna)* már rég megszűntek volna. Azt azonban könnyű megfigyelni a következő példában, hogy ezen a szinten lehet nem véglegesített *(csúnya hunglish-el élve nem kommitált)* lapokat olvasni, és hogy a SELECT nem vár az exkluzív zárolások miatt. Futtassuk le az alábbi kódot egy másik Query Analyzer ablakban:

```
BEGIN TRAN

UPDATE
  Employees
SET
  LastName = 'Borzaska'
```

Azaz megkezdünk egy tranzakciót, amiben minden alkalmazott családi nevét Borzaskára állítjuk. A tranzakciót logikailag még nem véglegesítettük, ám a változások fizikailag már rögzítődtek a táblába. Mit lát ebből a korábbi lekérdezésünk *(READ UNCOMMITTED szinten)*?

```
LastName             FirstName
-------------------- ----------
Borzaska             Nancy
Borzaska             Andrew
...
```

Azaz látja a beírt, de még nem véglegesített adatokat! Ezért hívják dirty read-nek ezt a szintet. Viszont láttuk, hogy nem tudtuk megakadályozni az olvasást még egy egész táblára szóló UPDATE-el sem, azaz ezen a szinten az adatbázist olvasó műveletek nem foglalkoznak még az Exclusive lock-okkal sem.
Hogy megnyugodjanak a kedélyek, görgessük vissza az előbbi félbehagyott tranzakciónkat:

```
ROLLBACK TRAN
```

2. READ COMMITTED

Ez az alapértelmezett elszigeteltségi szint az SQL szerverben. Ezen a szinten a SELECT utasítások Shared lock-okat helyeznek el azokon a sorokon, amelyeket éppen olvasnak. Emlékezzünk vissza, a Shared lock egy olyan zárolási típus, amit akárhányan olvashatnak, de senki nem írhat. Azaz a Shared lock megakadályozza, hogy valaki belenyúljon azokba az adatokba, amit a SELECT éppen olvas. Amint a megfelelő sor, lap vagy tábla kiolvasása megtörtént, a zárolások feloldódnak. Amennyiben a SELECT halad előre a sorok olvasásával, és beleütközik egy Exclusive lock-ba, ami azt mondja neki, hogy állj, ne tovább, akkor kénytelen arra várni, hogy az Exclusive lock feloldódjon. Ellenkező esetben visszalépnénk az előző szintre, és olyan adatokat olvasnánk, amelyeket még nem véglegesítettek. Ez a szint azonban arról szól, hogy csak olyan adatokat olvashat az adatbázisból, amelyeket már véglegesítettek, innen a szint neve is. Azaz ezen a szinten logikailag mindig konzisztens adatokat olvasunk ki.
Futtassuk le a korábbi teszt tranzakciónkat ezen a szinten is:

```
BEGIN TRAN
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
...
```

A tranzakciót egyedül lefuttatva a lekérdezés kimenete és a keletkezett zárolások pont úgy néznek ki, mint az előző szinten. Azonban gyökeresen más a helyzet, ha elindítjuk a másik „zavaró" tranzakciónkat is. Azaz futtassuk le az abban található UPDATE-et, de ne hajtsuk végre a ROLLBACK TRAN-t, hanem helyette indítsuk el az első lekérdezést!
Mit látunk? Semmit. A lekérdezés csak fut, csak fut... Mivel a másik tranzakció adatmódosítása Exclusive lock-okat helyezett el a tábla sorain *(sőt ez egész táblán, mert minden sort módosítottunk)*, a SELECT ezen a szinten már figyelembe veszik ezeket a zárolásokat, és addig nem hajlandó kiolvasni az adatokat, amíg a zárolás el nem takarodik a sorokról. Ehhez hajtsuk végre a ROLLBACK TRAN-t a második tranzakcióban! Ekkor az első tranzakció is befejezi a futását, és kiadja az eredeti, módosítás előtti sorokat:

```
LastName             FirstName
-------------------- ----------
Buchanan             Steven
```

Amennyiben a második tranzakció nem visszagörgeti, hanem érvényesíti a tranzakciót COMMIT TRAN-al, természetesen akkor is folytatja a futást az első tranzakció SELECT-je, csak a már módosított adatokat olvasva.
Ezen a szinten semmi nem biztosítja azt, hogy a tranzakción belül ugyanazokkal a feltételekkel visszaolvasva az adatokat ugyanazt az eredményt kapjuk két különböző időpillanatban. Lehet, hogy más tranzakció megváltoztatja az általunk kiolvasandó sorokat a két kiolvasás között, ezt nevezzük nem megismételhető olvasásnak *(non-repeatable read)*. Az is előfordulhat, hogy beszúrnak olyan sorokat a két SELECT közötti időben, amelyek megjelennek a második SELECT eredményhalmazában. Ezeket a megjelent sorokat hívják fantomoknak *(phantoms)*. A következő két szint ezeket a problémákat fogja orvosolni.

3. REPEATABLE READ

Itt már biztosak lehetünk abban, hogy logikailag helyes adatokat olvashatunk ki a táblákból, plusz, hogy egy tranzakción belül ugyanazt az olvasást többször megismé-

telve mindig ugyanazt az eredményt kapjuk vissza. Ezt azt jelenti, hogy a már olvasott sorok tartalma nem fog megváltozni, de nem jelenti azt, hogy nem jelenhetnek meg új sorok más tranzakciók ármány munkájának köszönhetően. Hogyan védekezik az SQL Server a már olvasott sorok módosítása ellen? Úgy, hogy a SELECT-ek által végigolvasott sorokra *(lapokra vagy táblára)* elhelyezett Shared lock-okat nem oldja fel egészen a tranzakció végéig. Ezek után hiába akarja valamelyik másik tranzakció módosítani a már leválogatott sorokat, a Shared lock-ok nem engedik meg, hogy megtegye, egészen a tranzakció befejezéséig. Ezen a szinten már nagyon erősen érezhető a zárolások miatti párhuzamosság csökkenése, hisz egy

```
SELECT * FROM tábla
```

utasítással gyakorlatilag befagyasztjuk az összes olyan tranzakciót, ami a táblán akar módosítani. Azaz csak tényleg olyankor érdemes bevetni, amikor a tranzakción belül többször ki kell olvasni ugyanazokat a sorokat, és fennáll a veszélye, hogy valaki közben módosítja őket.
Ha megnézzük, milyen zárolások keletkeznek ezen a szinten, akkor a következőt látjuk:

spid	dbid	ObjId	IndId	Type	Resource	Mode
51	6	0	0	DB		S
51	6	1977058079	1	KEY	(0500d1d065e9)	S
51	6	1977058079	2	KEY	(6c01b4c53be8)	S
51	6	1977058079	1	PAG	1:136	IS
51	6	1977058079	0	TAB		IS
51	6	1977058079	2	PAG	1:385	IS

Azaz a lock manager elhelyez Shared lock-okat sorokra a kulcsaikon keresztül *(2. és 3. sor)*, valamint Intent Share lock-okat a lekérdezett sort tartalmazó lapokra *(4. és 6. sor)*, valamint a táblára *(5. sor)*. Az Intent Share jelzi más tranzakcióknak, hogy ne is próbáljanak Exclusive lock-ot kérni a kérdéses lapokra vagy az egész táblára, mert úgysem fog sikerülni, hisz az adott „nagy" tartományokon belül vannak olyan sorok, amelyek Shared lock-kal védettek.
Miért van két sor és lap zárolás, amikor a lekérdezés kimenete csak egy sort tartalmaz? Láthatjuk, hogy különböző indexekhez *(IndId oszlop)* tartoznak a zárolások. Az Employees táblán három index is van, ezek közül kettőt használt a lekérdezés. A LastName-re szűrtünk, ehhez a LastName oszlopra definiált Nonclustered index-et használta a szerver *(ez könnyen ellenőrizhető a végrehajtási terv megtekintésével is)*. Miután megtalálta a LastName index táblában a megfelelő sorokat *(jelen esetben 1 sor)*, a Nonclustered index, mint sorazonosító segítségével kiolvassa a megfelelő sor tartalmát. Ahhoz, hogy biztosítsa a zárolást bármelyik indexet használó tranzakció elől, kénytelen zárolni mindkét index által lefoglalt sorokat és lapokat.

4. SERIALIZABLE

Nagyon hasonlít a REPEATABLE READ szintre, csak itt meg kell akadályozni azt is, hogy ugyanazt a SELECT-et megismételve új sorok jelenjenek meg az eredményhalmazban. Ehhez a szervernek le kell zárolni a teljes lehetséges tartományt, amelyet a SELECT WHERE feltétele jelöl ki. Az SQL Server a tartomány zárolására a már említett key-range lock-ot használja.

Nézzük meg az előbbi lekérdezést, aminek feltétel része a következő volt:

```
WHERE
   LastName LIKE 'B%'
```

E szint logikájának megfelelően a szervernek le kell zárolnia az összes olyan lehetséges index irányokat, amelyeken keresztül B betűvel kezdődő nevű sorokat be lehetne szúrni a táblába. Milyen zárolások generálódnak ennek érdekében? *(az objid oszlopot nyomdai okokból kihagytam)*

spid	dbid	IndId	Type	Resource	Mode
54	6	0	DB		S
54	6	0	TAB		IS
54	6	1	PAG	1:99	IS
54	6	2	PAG	1:97	IS
54	6	2	KEY	(7901573565c0)	RangeS-S
54	6	255	PAG	1:225	IS
54	6	255	RID	1:225:12	S
54	6	1	KEY	(0500d1d065e9)	S
54	6	255	RID	1:225:11	S
54	6	2	KEY	(6c01b4c53be8)	RangeS-S

Látható, hogy a két RangeS-S *(Shared Key-Range and Shared Resource)* zárolás lezárta a LastName-re definiált Nonclustered index két végét *(A és C betűvel kezdődő sorok közötti rész)*, így oda nem lehet új sorokat beszúrni.
A szintek tárgyalásánál nem szóltam az sp_lock kimenetéből az utolsó oszlopról. Abban látható, hogy a zárolást megkapta-e a kérő, vagy csak vár rá. Az összes példámban a mező értéke GRANT volt, azaz a kérő megkapta a zárolását. A READ COMITTED szintnél az UPDATE tranzakció blokkolja az olvasni kívánó tranzakciót, ilyenkor az utolsó oszlopban WAIT olvasható, azaz vár arra, hogy a másik tranzakció feloldja az általa foglalt zárolást.

Locking hints

Többször hivatkoztam arra, hogy az SQL Servert lehet befolyásolni abban, hogy milyen típusú, és milyen finomságú zárolásokat helyez el a tranzakciók során érintett adatokon. Most jött el az ideje, hogy áttekintsük ezeket.
A SELECT, UPDATE, DELETE és INSERT utasításokat ki lehet egészíteni egy WITH *(hint)* záradékkal, amely segítségével az SQL Servert el lehet téríteni az általa választott működéstől, és így megszabhatjuk, hogy milyen index-et, zárolást satöbbi használjon a táblák elérése során. Mi itt, most csak a zárolásokat befolyásoló hint-ekkel foglalkozunk.
Az első csoport a zárolás finomságára vonatkozik. A ROWLOCK arra utasítja a szervert, hogy a zárolandó sorok számától függetlenül *(még ha az egész táblára is vonatkozik)* ne használjon nagyobb kiterjedésű zárolást, mint sor szintűt. Hasonlóan a PAGLOCK, TABLOCK lap illetve tábla szintű zárolás használatára kéri a szervert.

Példa:

```
SELECT * FROM Orders WITH (PAGLOCK)
WHERE OrderID = 1213
```

Az előbbi módosítók a zárolás finomságát állították. A következők a zárolás típusát szabályozzák.

Az UPDLOCK segítségével a SELECT az alapértelmezetten használt Shared lock helyett Update lock-ot helyez el az olvasott táblán. Ennek előnye, hogy a már olvasott sorokon a tranzakció végéig megmarad az Update lock, így mások olvashatják az általunk kiolvasott sorokat, de nem módosíthatják azokat. *(Az Update és a Shared lock között annyi a különbség, hogy a már fennálló Shared lock-ra kiadható egy Update lock, de egy Update-re egy másik Update már nem.)*

Az XLOCK Exclusive lock-ot helyez el az adott utasítás által érintett sorokon. Azaz például egy ilyen módon átidomított SELECT képes exkluzívan zárolni egy egész lapot vagy táblát.

A NOLOCK és a READUNCOMMITTED ugyanazt jelenti, azaz mindenféle zárolástól függetlenül felolvassa a kért adatokat. Ezt kiadva a tranzakció összes utasítására ugyanazt érjük el, mint ha a tranzakció elszigeteltségi szintjét az elején READ UNCOMMITTED-re állítottuk volna. Gyakori felhasználás statisztikákban:

```
SELECT OrderID, SUM(Amount*UnitPrice)
FROM [Order Details] WITH (NOLOCK)
GROUP BY OrderID
```

Azaz az Order Details táblán működő egyéb adatmódosító tranzakcióktól függetlenül, mindenféle zárolást kikerülve olvasunk adatokat.

A HOLDLOCK és a SERIALIZABLE lock hint-ek hatására az SQL Server úgy kezeli az érintett táblákon a zárolásokat, mintha a tranzakció SERIALIZABLE módban lenne, azaz a Shared lock-okat nemcsak az olvasás idejére, hanem az egész tranzakció idejére fenntartja a már olvasott sorokon *(innen a HOLDlock név)*.

A READCOMMITTED hint a READ COMMITTED elszigeteltségi szint párja. Mivel ez az alapértelmezett szint, ritkán van szükség rá, hogy explicit kiírjuk.

Hasonlóan a REPEATABLEREAD az azonos nevű izolációs szint párja.

Az utolsó hint egy kicsit más, mint az előzőek. A READPAST azt mondja egy SELECT utasításnak, hogy egyszerűen ugorja át azokat a sorokat, amelyeket más tranzakció zárolt, és olvassa fel a nem zárolt sorokat. Ez csak READ COMMITTED elszigeteltségi szintű tranzakciókban működik, és csak a sor szintű zárolásokat tudja átlépni. Egy adatbázis elmélettel foglalkozó embernek ettől égnek áll a haja, de a való életben vannak olyan helyzetek, amikor hasznos lehet ez a szolgáltatás.

Azt írtam, hogy ezeket a hint-eket mind a négy alaputasítással lehet használni. Természetesen ez csak korlátozottan igaz, hiszen például a READPAST-nak nincs értelme az adatmódosító utasításoknál, azaz csak SELECT-el használható. Mindegyik hint-nek megvan a maga logikája, és csak azokon a helyeken működik, ahol van értelme.

Application lock-ok

SQL Server 2000 újdonság az application lock-ok megjelenése. Segítségükkel létrehozhatunk saját zárolási mechanizmusokat a szerver lock manager-ének felhasználásával. Láttuk a zárolások finomságának tárgyalásánál, hogy a lock manager az igazából nem tud róla, hogy ő milyen objektumon végez zárolást *(a nevét tudja, de a belső struktúrájáról semmit nem tud)*, csak van neki egy táblázata, amely alapján eldönti, hogy az ütköző zárolás kérések esetén továbbengedheti-e az igénylőt, vagy várakoztatnia kell, amíg elfogynak a konkurens zárolások. Most megkaptuk ezt a logikát, amely segítségével más programnyelveken megszokott kritikus szekciókat illetve szemaforokat valósíthatunk meg az alkalmazásainkban.

Saját zárolás létrehozása nagyon egyszerű. Az sp_getapplock tárolt eljárás meghívásával kérünk egy általunk megállmodott zárolási típust, egyedi néven. Elindítjuk a védendő, zárolandó eljárásunkat. Az eljárásunk lefutása után az sp_releaseapplock eljárással szabadíthatjuk fel a zárolást. Gyakori feladat például az, hogy egy tárolt eljárást egyszerre csak egy felhasználó futtathat. Application lock-ok felhasználásával ezt nagyon egyszerűen megoldhatjuk:

```
EXEC sp_getapplock 'SociLock', 'Exclusive',
'Session'

EXEC VédendőTároltEljárás

EXEC sp_releaseapplock 'SociLock', 'Session'
```

Az sp_getapplock első paramétere a zárolás egyedi neve. A második paraméter a zárolás típusa, amit mi most Exclusive-ra állítottunk, mert azt akarjuk, hogy miután valakinek sikerült túljutni a záráson csak ő futtathassa a VédendőTároltEljárás-t, egészen addig, míg az sp_releaseapplock-al el nem engedjük a zárolást. A 'Session' azt jelenti, hogy ugyanarról a felhasználói kapcsolatról nem hatásos a zárolás, csak különböző kapcsolatok között. Ez azt is jelenti, hogy ugyanaz a felhasználó többször is lefuttathatja a védett eljárást, mert a lock saját magára hatástalan. Ha azt akarjuk, hogy még ugyanaz a felhasználó se futtathassa többször a közbenső eljárást, akkor a 'Session' helyett 'Transaction'-t kell írni, és a három eljáráshívást tranzakcióba (BEGIN TRAN, COMMIT TRAN) kell foglalni. Ekkor a zárolás tranzakciószintű lesz, így még egyazon felhasználói kapcsolaton futó párhuzamos tranzakciók is zárolják egymást, megakadályozva a párhuzamos futtatást.

A Shared és az Exclusive és a többi zárolási típus variálásával kialakíthatunk más jellegű zárolási sémákat is, amelyek megfelelően támogatják az alkalmazásunk logikáját.

Zárszó

Cikksorozatunk eddigi legnehezebb része volt a zárolások témaköre. Ezután már általában könnyebb, gyakorlatiasabb részek jönnek. Úgyhogy az a kedves olvasó, akinek volt türelme végigolvasni és értelmezni a cikket (abban már csak reménykedni merek, hogy az esetleges kérdőjeles részekhez előkerült a Books Online is), már megtette az első lépést abban az irányba, ami a professzionális adatbázis tervezés felé vezet. Az adatbázis zárolási eljárásának ismerete nélkül tranzakciókat és adatbázisokat tervezni vakrepülés, amely előbb-utóbb egy sziklafalon végződik.

A következő cikkbe szorult át a dead-lock-ok elmélete és gyakorlata, amely azonban csak a zárolások ismeretében érthető meg. Visszavárom Önöket a halálos ölelések szigetén, a következő számban!

Soczó Zsolt MCSE, MCSD, MCDBA
Zsolt.Soczo@netacademia.net

ASP suli 3. Bábeli zűrzavar

Rengeteg levelet kaptunk, hogy az előző számban, a Session objektum ismertetése során méltatlanul kevés helyet szenteltünk a CodePage és LCID jellemzőknek. A meglátás jogos, ezért most igyekszem bepótolni a kódtáblákkal, lokalizálással kapcsolatos információkat. Mint mindig, az aktuális példaprogramok természetesen letölthetők a **[1]** címről.

Volt egyszer egy ASCII

Az Internet használatának első éveiben, az aktuális feladatok megoldására a 7 bites ASCII kódolás tökéletesen elegendő volt. A 7 bitbe *(127 karakter)* belefért a teljes angol ABC, a számok és még néhány jel is. Azután a hálózat kezdte átlépni a határokat, és egyre inkább szükség volt az angoltól különböző nyelvek betűinek megjelenítésére is.

Eközben a hétbites rendszerekről lassan megkezdődött az áttérés a nyolc bitre *(így lett az ASCII-ből ANSI)*, ezután kézenfekvő volt, hogy a speciális karaktereket a megjelenő 128 üres helyre lehet bekódolni. Ahány ház, annyi szokás: ki így, ki úgy töltötte ki ezt a felső tartományt. A különböző megoldásoknak köszönhetően megszülettek a kódtáblák *(codepage)*, és elkezdődött a káosz. Még messze a DOS-os időkben járunk, amikor a táblázatrajzoló karakterek helyett néha ékezetes karakterek jelennek meg a rossz beállításoknak köszönhetően. A nyugat-európai *(western)* kódtáblák tartalmaznak ugyan ékezetes betűket, de a magyar hosszú ő és ű már nem fért bele. Sebaj, van hasonló: *(sajnos)* valószínűleg mindannyian találkoztunk már a kalapos ű és hullámos ő betűkkel. Akkoriban ez a kis csúsztatás még elviselhetőnek tűnt, manapság viszont már inkább kínos hibának tűnik a megjelenésük – teszem hozzá, jogosan.

A világ ugyanis azóta sokat fejlődött. A szerteágazó kódtáblákat szabványokká fogták össze, a dokumentumokba beépítették a karaktertáblákat azonosító adatokat, így egy HTML oldal forrásából, vagy egy e-mailből azonnal kiderül, hogy azt a küldője milyen karaktertáblával írta. Bizony, így van ez akkor is, ha a *(főleg nem Windows platformon futó)* levelezőprogramok zöme erről nem hajlandó tudomást venni. Mindenekelőtt két fontos kódtáblára hívnám fel a figyelmet: az iso-8859-1, más néven Western kódtábla a nyugat-európai karaktereket *(és a hullámos/kalapos ő-t és ű-t)* tartalmazza, míg az iso-8859-2 alias Central European nevéhez méltón a számunkra oly kedves magyar változatot. Hasonló hatást lehet elérni a Windows-1250 nevű kódlappal, ez azonban, mint az a nevéből is kitalálható, nem kifejezetten elterjedt Un*x és Macintosh körökben :-). Magyar szövegben, amikor csak tehetjük, használjuk az iso-8859-2-t!

Speciális karakterek a HTML kódban

Térjünk vissza kicsit a HTML és ASP mezsgyéjéhez. A HTML dokumentumokban a különleges karaktereket speciális módon, úgynevezett entity-k segítségével írták le. A *(kalapos)* hosszú ű kódja például û a *(hullámos)* hosszú ő pedig õ. Sokan a mai napig használják ezt a kódolást, pedig egyrészt hosszú, kényelmetlen, másrészt pedig felesleges. A böngészők már régóta képesek feldolgozni a különféle kódtáblákban írt HTML dokumentumokat – azokba pedig egyszerűen csak bele kell írni a betűket.

Itt jegyezném meg, hogy bizonyos jeleket még mindig érdemes entity-k segítségével leírni. Ilyen például a ™ *(™)*, a © *(©)*,az ® *(®)*, illetve a matematikai jelek, és más speciális karakterek *(€ ¥ ½ §)*. A **[2]** címen, a HTML 4 szabvány leírásán belül megtalálható a szabványos entity-k teljes listája, de ezzel már bánjunk nagyon óvatosan, mert a böngészők eléggé hadilábon állnak az igazán speciális karakterekkel.

Ha egy böngészőben megkeressük az Encoding vagy Character Set menüpontot, láthatjuk, hogy mely kódtáblákat képes felismerni és használni. A HTML oldal kódjában pedig a készítő megadhatja a használt kódtábla azonosítóját, az oldal nyelvét, de ha ez elmarad, akkor is van rá esély, hogy a böngésző helyesen ismeri fel azt.

Árvíztűrő tükörfúrógép

Tegyünk hát egy próbát! A fenti mondat tartalmazza az összes speciális magyar betűt, *(izetlen tréfa következik, de nem tudom kihagyni: szegény tisza-menti üvegeseknek már biztosan van ilyen...)*, amit az *arviz1.htm* oldal kódjába nemes egyszerűséggel, kódolatlanul beleírtunk:

```
<html>
  <head></head>
  <body>
   árvíztűrő tükörfúrógép - ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP
  </body>
</html>
```

Ha ezt az oldalt megjelenítjük, a böngésző a kódtábla megadása híján megpróbálja felismerni a használt változatot. Ha úgy dönt, hogy nyugat-európai kódolást választ, jönnek a kalapos ékezetek. *(Az éppen használt kódtáblát az Encoding menüben láthatjuk kiválasztva. Ha ezt kézzel módosítjuk, a kódolás helyreáll.)* A találgatások elkerülése érdekében a böngészőt kifejezetten utasíthatjuk egy adott kódtábla használatára, emígyen *(arviz2.htm)*:

```
<html>
  <head>
    <meta http-equiv="Content-Type"
    content="text/html; charset=iso-8859-2">
  </head>
  <body>
   árvíztűrő tükörfúrógép - ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP
  </body>
</html>
```

A lényeg az aláhúzott sorban, az úgynevezett Content-Type fejlécben van, ahol nemcsak az oldal HTML mibenlétét határozzuk meg, hanem a használt kódtáblát is. Akinek ismerős ez a sor, jól téved: néhány hónappal ezelőtt, a **Response.Charset** jellemző leírásánál már megemlítettük. Ha azt használjuk, ugyanezt a hatást érjük el, miközben az ASP oldalon belül elkerülhetjük a META elem használatát *(arviz3.asp)*:

```
<% Response.Charset = "iso-8859-2" %>
<html>
  <head></head>
  <body>
   árvíztűrő tükörfúrógép - ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP
  </body>
</html>
```

További META elemek

Ha már itt tartunk, felsorolnék néhány, eddig még nem említett hasznos metainformációt, amit a weboldalakba ágyazva különféle hatást érhetünk el.

```
<META name="author" content="Fülöp Miklós">
<META name="date" content="03/07/01">
<META name="description" content="ASP Suli 3">
<META name="keywords" content="ASP, learning">
<META name="robots" content="noindex, nofollow">
<META http-equiv="Content-Language" content="hu">
```

A fenti információk nagy része egyelőre igazán csak a webes keresők számára érdekes. Az author *(szerző)*, date, keywords, description mezők tartalmát a jobb keresők feldolgozzák, keresés során a keywords mezőben megjelenő kulcsszavak például nagyobb súllyal bírnak, mint a dokumentum szövegéből kivonatolt minta. A description mező értékét a keresésre adott válaszban szokás megjeleníteni. Ha oldalunk tartalmaz ilyen mezőt, akkor a rövid leírás nem a dokumentum első néhány sora lesz, hanem a description-ként megadott szöveg.

A robots metaelem a keresők felderítő-egységeinek *(ezek a robotok)* működését szabályozza, a noindex érték arra utasítja a robotot, hogy az oldalt ne indexelje, ne tárolja az adatbázisba, a nofollow pedig azt jelenti, hogy az oldalon található hivatkozásokat nem kell követni. A legfontosabb mégis a Content-Language, ami nem is metainformáció, hanem HTTP fejléc. Értéke a dokumentum nyelvére utaló rövidítés, magyar esetén természetesen a „hu" *(a további kódok megtalálhatók például itt:* **[4]***)*

Vissza a kódtáblákhoz

A különféle kódtáblák használata azonban nem oldott meg minden gondot. Nem nagyon használhatunk egy dokumentumon belül több kódtáblát *(a HTML codepage fejléc az egész dokumentumra vonatkozik)*. Ennél is nagyobb gond az, hogy a legtöbb távol-keleti nyelv többezer különböző karaktert tartalmaz, ezt pedig nehezen lehet leírni egy 8 bites azonosítóval. Ezeken a területeken természetesen 16 bites kódtáblák alakultak ki, ahol egy jelet két bájt határoz meg. Persze ilyen kódtáblából is több fajta létezik, ezért kézenfekvő volt, hogy a kódtáblák közötti káoszt szabályozni kell.

Több vezető számítástechnikai, informatikai cég, nyelvészek, könyvtárosok szervezetei, és még sokan mások ezért megalapították a Unicode Consortiumot. A szervezet célja az volt, hogy olyan szabályrendszert dolgozzanak ki, amivel végre egységesen le lehet írni a világon beszélt *(és már vagy még nem használt)* összes nyelv minden karakterét, grafikai szimbólumokat, nyelvi elemeket. A Unicode szerint is minden karaktert két bájt azonosít, így 65535 különböző jel leírására van lehetőség. A táblázat még napjainkban is frissül, sőt, van olyan területe is, ahova mi magunk definiálhatunk karaktereket *(a 0xE000-tól 0xF8FF-ig)*. Ha van kéznél egy Windows 2000, tessék csak bepötyögni a Run ablakba: eudcedit.exe!

A Unicode azonban nem kódolási szabvány. A karakterek kétbájtos értékét többféleképpen is felhasználhatjuk a dokumentumokban. Többféle kódolási mód terjedt el, természetesen mindegyik alapja maga a Unicode táblázat. Az egyik módszer szerint a dokumentum minden egyes karakterét két bájton ábrázolják, ezzel természetesen megduplázva a méretét. Az alternatív kódolási módok közül az UTF-7 és az UTF-8 terjedt el, ezek közül az UTF-8 lett a gyakoribb.

Az UTF-8 kódolás

Kódolás, és nem kódtábla, hiszen itt már szó sincs tábláról. Kódtáblaként maga a Unicode funkcionál. Lássuk, mit tud az UTF-8. Ebben a kódban egy karakter kódjának hossza 1 és 3 bájt között mozog. Az ASCII karaktereket, tehát az első 127 jelet hagyományosan, egy bájton kezeljük. Ennek köszönhető, hogy az UTF-8 dokumentumok, ha nem tartalmaznak túl sok extra karaktert, emberi szemmel még jól olvashatók.

A 128 feletti értékek már nem férnek el egy bájton, ezért azok esetén már hosszabb leíróra van szükség.

De álljunk meg egy pillanatra! Miért nem fér bele az első bájtba 255 karakter? Hát azért, mert akkor nem tudnánk megkülönböztetni a 0x6161-es kódot a 0x61 0x61-től. Ezért aztán, ha egy UTF-8 bájt értéke:

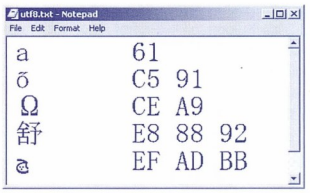
- 0x00-0x7F: az egybájtos ASCII karakter kódja
- 0x80-0xBF: több-bájtos érték további bájtjai
- 0xC2-0xDF: kétbájtos érték első bájtja
- 0xE0-0xEF: hárombájtos érték első bájtja

Tehát minél „messzebb" van a Unicode táblában egy karakter, annál hosszabban kell leírni *(de legfeljebb három bájton)*. Ha egy karakter kódja

- 0x0001-0x007F, akkor 1 bájt *(0x01-0x7F)*
- 0x0080-0x07FF, akkor 2 bájt *(0xC280-0xDFBF)*
- 0x0800 felett, akkor 3 bájt *(0xE0A080-0xEFBFBF)*

A távol-keleten tehát az UTF-8 kódolás kicsit pocsékoló, hiszen ott a legtöbb használt karakter az utolsó tartományba esik. A világ nagy részén viszont, ahol az írás alapját valahol mégiscsak a latin betűk képezik, az UTF-8 sok helyet megspórol.

Mutatok egy példát, kép formájában, mert ez már nem biztos, hogy túlélné az amúgy is rapszodikus nyomdai konverziókat :-). Íme:



Néhány karakter és UTF-8 kódja

Talán feltűnt, hogy milyen speciális alkalmazással hoztam létre ezt a dokumentumot. Nem csalás, nem ámítás, a Windows 2000 Notepad nevű csodaalkalmazása úgy nyeli a Unicode karaktereket, mint kacsa a nokedlit. Kínaiul akarunk írni? Tessék! UTF-8 formátumban szeretnénk menteni? Tessék! *(Vessünk egy pillantást a mentés menüre, ott van az Encoding mezőt!)*. Teheti mindezt azért, mert a Windows 2000 belül Unicode. *(Itt kérnék elnézést, ha az utolsó két sor valamelyikében véletlenül egy japán vagy arab káromkodást idéztem volna, esküszöm, nem volt tudatos :-))*
A kódtábla-hegyek használatához mindössze egy dolgunk van: a Control Panel / Regional Settings dialógus General ablakának alján pipálgassuk be a megcélzott területeket, és némi telepítgetés után birtokunkba vehetjük a teljes Unicode univerzumot. *(Ha szeretnénk kipróbálni a többnyelvű ASP példákat, akkor most tegyük is meg!)*.

Lokalizálás

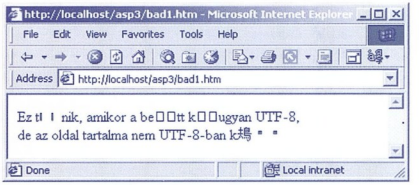
Lokalizálás alatt nem csak az adott ország vagy nyelv betűkészletének használatát értjük, hanem sok minden mást is. Néhány példa, a teljesség igénye nélkül: dátumformátum, hónapok, napok nevei, 12/24 órás időszámítás, AM/PM helyi elnevezése, a dátum és idő elemeit elválasztó karakterek, az elemek sorrendje, a fizetőeszköz jele és írásának módja, a számok írásának módja, a számjegyek csoportosítása, a csoportosításra szolgáló jel, a tizedesjel, a hét első napja, a használt naptár, a sorbarendezés alapja, satöbbi.
Minderre fel kell készülnünk, amikor lokalizált alkalmazást készítünk, és nincs ez másképp az ASP alkalmazások esetén sem. A Windows teljes mértékben támogatja ezeket a lokalizációs megoldásokat, ezért tulajdonképpen nincsen nehéz dolgunk. Maga a Windows is a Regional Settings lokalizációs beállításai alapján működik, így jelennek meg a dátumok, időpontok, számok, de még a numerikus billentyűzet „." gombjának jelentése is *(magyar beállítás esetén ugyanis – helyesen – nem tizedespontot, hanem tizedesvesszőt ír)*.
A VBScript is számos hasznos funkciót tartalmaz **[3]**. A GetLocale() függvény például meghívása után visszaadja a számítógépen használt alapértelmezett beállítás kódját *(a kód értelmezését lásd itt:* **[4]***)*, a SetLocale() segítségével pedig beállíthatunk egy más értéket. Ha az adott nyelv támogatása nincs telepítve, a SetLocale() meghívása hibát okoz. A FormatCurrency(), FormatDateTime(), FormatNumber(), FormatPercent() függvények pedig az éppen érvényes beállításnak megfelelően készítik el a fizetőeszköz, dátum és idő, szám és százalékértékeket *(szöveges formában, persze)*.
A locale.asp példaprogram segítségével kicsit játszadozhatunk a beállításokkal.

Weblapok UTF-8-ban

Miután kigyönyörködtük magunkat a lokalizált adatok nézegetése során, vegyük sorra, mi kell ahhoz, hogy valódi nemzetközi oldalakat hozhassunk létre.
Mindenekelőtt, a böngészőt utasítani kellene, hogy használjon valami értelmes kódtáblát. Természetesen minden nyelvhez megvan a saját kódtábla, de most univerzális megoldást keresünk, sőt, a végén egy oldalon belül több nyelv is megjelenik majd, ezért a választásunk természetesen az UTF-8-ra esik. Helyezzük el a megfelelő META elemet az oldal fejlécében:

```
<meta http-equiv="Content-Type"
 content="text/html; charset=utf-8">
```

Ezután már csak arra kell ügyelnünk, hogy az oldal forráskódját is UTF-8, ban mentsük el, különben a böngésző el fogja nyelni az ékezetes karakterek után következő betűket *(hiszen két- vagy hárombájtos értéket vár)*, valahogy így:



„Ez történik, amikor a beállított kód ugyan UTF-8, de az oldal tartalma nem UTF-8-ban készült."

Ha a fenti példa megtekintése esetén a böngészőben kiválasztjuk a Central European dekódolást, meglátjuk, hogy a dokumentum valójában nem UTF-8-ban készült, de a böngésző engedelmeskedik a benne található META parancsnak.
UTF-8 kódolású oldalt többféleképpen is előállíthatunk: mindenekelőtt, ha a webszerkesztőben a HTML Encoding értékét Multilingual *(UTF-8)*-ra állítjuk. Egy alternatív lehetőség, hogy elkészítjük a komplett oldalt, ahogy nekünk jól esik, majd a Windows 2000 Notepad-jával megnyitjuk, és visszamentjük UTF-8 formátumban.

Az ASP által használt kódtábla beállítása

Az rendben van, hogy mi már tudunk UTF-8-ban írni, de az ASP-nek is meg kellene valahogy magyarázni, hogy mit szeretnénk, különben a rendszer alapértelmezett kódtábláját használja. Erre rögtön két lehetőségünk is adódik, egyrészt, használhatjuk a Session.CodePage jellemzőt, másrészt pedig a @CODEPAGE ASP direktívát. Az UTF-8 kódolás „kódtáblájának" azonosítója 65001. Az alábbi két példa egyenrangú, de a Session.CodePage használata felülbírálja a direktívában meghatározott értéket:

```
<%@ CODEPAGE=65001 %>
<%
  Session.CodePage = 65001
%>
```

A kódtáblához hasonlóan természetesen a lokalizációs beállításokat is tudjuk módosítani. Mint a kódtáblánál, az ASP a számítógép alapértelmezése szerint lokalizál, de ez is megváltoztatható. Itt is két lehetőségünk adódik, a direktíva és a jellemző beállítása:

```
<%@ LCID=1038 %>
<%
  Session.LCID = 1038 ' Hungary
%>
```

Ne feledjük, az ASP direktíváknak az ASP oldal tetején kell elhelyezkedniük, egynél több direktíva esetén egy sorban, egymás után, így:

```
<%@ CODEPAGE=65001 LCID=1038 %>
```

Mint azt már említettem, nemlétező, vagy nem támogatott kódtábla és lokalizációs beállítások használata hibát okoz. A hiba akkor lép fel, amikor a Session.CodePage illetve Session.LCID rossz értéket kap. Mi azonban nem szeretnénk, hogy az oldal végrehajtása hibával befejeződjön, sokkal jobb lenne valami intelligens hibakezelési megoldás.

Hibakezelés az ASP oldalban

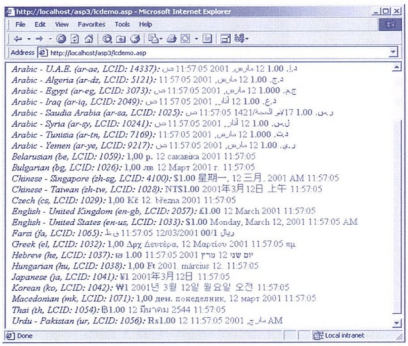
A hibakezelés már nem az ASP környezet, hanem a scriptmotor feladata, ezért ahány scriptkörnyezet, annyi megoldás létezik. Míg a Jscriptben a C programozóknak ismerős try/catch párost használhatjuk a hibák elfogására és kezelésére, addig VBScriptben a Visual Basic hibakezelési megoldása képezi az alapokat. A VBScript annyiban különbözik a VB-től, hogy itt nem definiálhatunk hibakezelő szubrutint. Az egyetlen, amit megtehetünk, az, hogy utasítjuk a scriptmotort, hogy hiba esetén ne álljon meg, hanem ugorjon a következő sorra, és majd mi kezeljük a hibát – ha akarjuk. Ezt a parancsot a következőképpen adhatjuk ki:

```
<%
    On Error Resume Next
    ... ' csináljunk valamit, ami hibát okozhat
    If Err.Number <> 0 Then ' hiba történt
      Response.Write("HIBA! Kód:" & Err.Number)
      Response.Write("Leírás:" & Err.Description)
      Err.Clear
    End If
%>
```

A program futása során bármikor ellenőrizhetjük az úgynevezett Err objektumot. Ha az **Err.Number** értéke 0, nem történt hiba, ha attól különböző, akkor a hibakódot tartalmazza. Ilyenkor az **Err.Description** a hiba leírását adja vissza. Miután lekezeltük a hibát, töröljük azt az **Err.Clear()** metódus segítségével, különben a következő ellenőrzéskor is elkapjuk majd. Az IIS5 ennél fejlettebb hibakezelési eszközöket is tartalmaz *(ott van például az ASPError objektum)*, de erről majd csak a következő számunkban ejtünk szót.

Bábel tornya

Fogjuk tehát az összes elérhető lokalizációs variációt, és írjuk ki egy oldalra az aktuális dátumot, időpontot és pénznemet! Az *lcid.asp* kódja a leírtak alapján teljesen érthető kell, hogy legyen. Az *lcdemo.asp* funkcionalitásában megegyezik az lcid.asp-vel, csak kevesebb példát hoz.



Bábeli zűrzavar, de oly kedves a szívemnek. Hol van már az ASCII karakterkészlet? Érdemes megfigyelni, hogy az arab nyelveknél még az írás iránya is megváltozott!

Valami elkezdődött

Most mondhatná azt az olvasó, hogy könnyű a Microsoftnak. Könnyű a saját webkiszolgálót és böngészőt összehangolni. De nem erről van szó: az előállított oldal igenis szabványos, csak a szabványok megvalósítására kellene egy kicsit odafigyelni. Próbaképpen ránéztem a fenti oldalra alternatív böngészőkkel is *(A képernyőképek letölthetők a* **[5]** *címről.)* Az UTF-8 kódolást mindegyik böngésző felismerte, ahol tudta, megjelenítette az ékezetes karaktereket, és nem okozott gondot, hogy néhány karaktert több bájton ábrázoltunk. A versenyt az Opera 5.01 vesztette el, a megjelenített oldal tele van kérdőjelekkel – az arab, a cirill, a görög és a távol-keleti karakterek egyike sem látszik. A Netscape 4.61 már jobb eredményt produkált: csak néhány távol-keleti karakter helyett láthatunk csinos kis négyszögeket. Kellemes meglepetés, hogy az új Netscape 6.01 Gecko motorja ötös alát kapott: csak azért nem kitűnőt, mert nem vett tudomást az arab szöveg írásirányának megváltozásáról *(a speciális vezérlőkarakterek pedig ott voltak a szövegben)*.

Fontos!

A helyes működés érdekében az IIS kiszolgálóra és az ügyfelek számítógépére is telepíteni kell a megfelelő nyelvi támogatást!

Összefoglalva azt mondhatjuk tehát: itt az ideje, hogy elbúcsúzzunk a hullámos és kalapos ű és ő betűktől. A modern böngészők segítségével gyakorolhatjuk, hogy írják a kuszkuszt eredetileg, saját nyelvén kívánhatunk boldog születésnapot thai barátnőnknek, Pandacsöki Boborján pedig immáron áttérhet az urdu nyelv írásos vetületére. :-)

Fülöp Miklós
mick@netacademia.net

A cikkben szereplő URL-ek:

[1] http://technet.netacademia.net/feladatok/asp/3
[2] http://www.w3.org/TR/html40/sgml/entities.html
[3] http://msdn.microsoft.com/scripting/vbscript/doc/vbstoc.htm
[4] http://msdn.microsoft.com/scripting/vbscript/doc/vsmsclcid.htm
[5] http://technet.netacademia.net/feladatok/asp/3/pic

ASP objektummodell-referencia a weben:

http://msdn.microsoft.com/library/psdk/iisref/vbob74bw.htm

XMLgessünk

Bevezetés

A cikk célja, hogy megadja a kezdő lökést azoknak, akik már ismerik az XML nyelv alapjait, hallottak a hagyományos többrétegű alkalmazásfejlesztésről, de még nem használták ki alkalmazásaikban az XML-ben rejlő lehetőségeket.
Az első részben végigfutunk az XML nyelv alapjain, a további részekben pedig konkrét gyakorlati példákon keresztül megnézzük, hogyan lehet kihasználni az XML előnyeit WEB fejlesztésekben.

XML alapok – turbó sebességgel

Hogy a gyakorlati részben pontosan lehessen követni a programok és az XML kapcsolatát, megpróbálom tömören, de precízen definiálni az XML technológiában alkalmazott fogalmakat.
Az XML nyelv egy olyan leíró nyelv, amely segítségével adatokat reprezentálunk. Úgy van megalkotva, hogy használhatjuk akár önleíró módon, akár egy előre meghatározott szabálysrendszerre *(séma)* épülve. Ez utóbbi azt jelenti, hogy az adatok ábrázolásának struktúráját megköthetem előre, és csak azokat az XML dokumentumokat fogadom el megfelelőnek, amelyek az általam definiált szerkezet, séma alapján vannak megszerkesztve.
Az XML nyelv alapelemei a tagok *(tag)*. A tag egy „kacsacsőrök" <> közé zárt szöveg. Például a <malac> egy tag. Az XML tagok érzékenyek a kis-nagybetűre, <egér> és <Egér> nem ugyanaz!
Az adatokat egy kezdő és egy záró tag közé kell elhelyezni, és az így nyert formációt XML elemnek *(element)* hívjuk. A záró tag ugyanolyan, mint a kezdőtag, csak egy / (per) jel van a tagban található szöveg előtt. Kezdőtag: <malac>, záró tag: </malac>. Így egy teljesen szabályos XML elem:
<malac>Mazsola</malac>
A tagok között levő adatokat az elem értékeként *(value)* kezeljük. Az XML tagoknak tetszőleges számú attribútumokat *(attribute)* is adhatunk. Ezek név="érték" típusú értékadások. Az idézőjelek kötelezőek. Például, minden malachoz írjuk oda a legjobb barátját egy attribútumként:
<malac barát="Manócska">Mazsola</malac>
Az XML dokumentum egy olyan XML elem, amelyben lehetnek beágyazott XML elemek. Tehát egy szem XML elem is már egy XML dokumentum, de a tipikus XML dokumentum több szinten is egymásba ágyazott XML elemekből áll. Például:

```
<mesék>
  <mese>
    <cim>Mazsola</cim>
    <szereplo>Mazsola</szereplo>
    <szereplo>Manócska</szereplo>
    <szereplo>Tádé</szereplo>
  </mese>
  <mese>
    <cim>Futrinka utca</cim>
    <szereplo>Buksi</szereplo>
    <szereplo>Morzsi</szereplo>
    <szereplo>Cicamica</szereplo>
  </mese>
</mesék>
```

Az iménti példa egy jól megformázott *(well-formed)* XML dokumentum. Hogy miért? Egy jól megformázott XML dokumentumnak teljesíteni kell a következő kritériumokat:

- A kezdő és a záró tagoknak egyezniük kell
- Az elemek nem lapolhatják át egymást
- A problémás karaktereket kódolni kell *(<,&,>,",')*
- Kell lennie egy és csakis egy gyökér elemnek

Tulajdonképpen egy jól megformázott XML dokumentum nem más, mint egy olyan dokumentum, ami illeszkedik az XML szabvány által specifikált szabályokhoz.
Mitől lesz egy XML dokumentum érvényes *(valid)*? Attól, hogy nemcsak jól megformázott, hanem egy előre definiált sémának megfelelően van felépítve. Másképpen fogalmazva megkötöm, formálisan leírom az általam használni kívánt XML dokumentum nyelvét, és megnézem, hogy ennek a nyelvnek megfelelő-e a kapott dokumentum. Ha igen, akkor érvényesnek könyvelhetem el.

Névterek

Vágjunk bele egy kicsit bonyolultabb dologba, egy példán keresztül. Mi van akkor, ha több ország iskoláiból kapunk tanulmányi eredményeket *(természetesen XML formátumban)*, és ezeket szeretnénk egy nagy XML dokumentumba foglalni? Azonban az egyik országban az 1-esnek örülnek a gyerekek, míg nálunk *(legalábbis a legtöbb)* az 5-ösnek. Ha egyszerűen összefűznénk az adatokat, akkor lehetetlen lenne összehasonlítani a különböző eredményeket. Valahogyan definiálni kellene az összesített XML dokumentumban a számok kétféle értelmezését. Ezután minden tételnél megjelöljük, hogy melyik tétel melyik értelmezéshez kapcsolódik, így később egyértelműen kibányászhatók az érdemjegyek. Az ilyen jellegű problémák feloldására vezették be a névtér *(namespace)* fogalmát az XML szabványban. A dokumentum elején meghatározzuk a leírni kívánt típusokat, és a tagoknál megjelöljük, hogy azok melyikhez tartoznak. Névtér használata nélkül így nézne ki az összevont érdemjegy táblázat:

```
<Osztályzatok>
  <oszt>4</oszt>
  <oszt>2</oszt>
</Osztályzatok>
```

Melyiknek örülne egy gyerek? Az attól függ, hogy honnan kerültek bele az adatok. Tegyük egyértelművé az adatok értelmezését:

```
<Osztályzatok
xmlns:AzÖtösaJó="http://iskola.hu/Ot"
```

```
xmlns:AzEgyesaJó="http://iskola.hu/Egy" >
  <AzÖtösaJó:oszt>4</AzÖtösaJó:oszt>
  <AzÖtösaJó:oszt>5</AzÖtösaJó:oszt>
  <AzEgyesaJó:oszt>2</AzEgyesaJó:oszt>
</Osztályzatok>
```

Mit láthattunk itt? Készítettünk két névteret. Az egyiknek „AzÖtösaJó", a másiknak „AzEgyesaJó" a neve. A nevek tetszőlegesen választhatók, lehetne Jancsi és Juliska is, de nem árt, ha a választott névtér neve utal a funkciójára. Az xmlns:névtérnév="valami egyedi" segítségével definiálhatunk névteret egy taghoz. A „valami egyedi"-nek az egész világon egyedinek kell lenni, így elvileg a világ összes XML dokumentumát konfliktus nélkül össze lehetne fűzni, mert a „valami egyedi" egyértelműen azonosítja a forrást, így nem lehetnek névütközések. A névtérnév egy álnév (alias) a hosszú, egyedi névtér névre. Így a tagokban könnyebb rájuk hivatkozni.

A névtér öröklődik, így a gyermek tagok *(mint az oszt)* is használhatják a szülőben *(Osztályzatok)*, nagyszülőben, dédszülőben, satöbbi definiált névteret.

Mit jelent az a bűvös URL a „valami egyedi" helyén? Ha beírom az Internet Explorerembe, akkor mi fog ott bejönni? Kakukkos óra? Nem, általában semmi! A névtér miről is szól? Arról, hogy valamilyen módon egyedi neveket kell definiálni. Erre két módszer adódik kézenfekvően: használjunk valamilyen URL-t, vagy alkalmazzuk a Windows-os GUID-ot *(Globally Unique Identifier)*. Mivel a W3C konzorcium, az XML gazdája nem csak a Microsoft által befolyásolt szervezet, ezért a legtöbb névtér deklarációban URL-t látunk, és nem GUID-ot, mint egyedi névtér azonosítót.

Amikor egy program feldolgozza az így összesített adatokat, akkor minden egyes tag feldolgozása során megnézi, hogy az adott tag melyik névtérhez tartozik. A névtereket tudnia kell előre ahhoz, hogy tudja a dokumentumban található számok, mint osztályzatok értelmét. Mivel a névtér garantáltan egyedi, azért a programunk értelmezni tudja a bejövő XML adatokat. Látni fogjuk, hogy konkrét termékek, például az SQL Server 2000 vagy az XML Parser XSL konvertere csak akkor működik helyesen, csak akkor végzi el a kért műveletet, ha a névteret az előírtnak megfelelően, karakterről-karakterre pontosan adjuk meg.

Ha az adatok zöme egyféle értelmezésű, így csak egy kis százaléka kellene, hogy valami más névtérbe tartozzon, akkor érdemes kihasználni az alapértelmezett névtér *(default namespace)* lehetőségét, hogy rövidebb dokumentumot kapjunk. Ha az előbbi példánk bejövő adathalmazában a hazai *(az ötös a legjobb jegy)* értelmezés a leggyakoribb, akkor tegyük azt az alapértelmezett névtérré, így csak azokat az adatokat kell megjelölni, amelyeket nem így kell értelmezni. Az alapértelmezett névteret úgy deklaráljuk, hogy nem adunk álnevet a kívánt névtérazonosítónak:

```
<Osztályzatok
xmlns="http://iskola.hu/Ot"
xmlns:AzEgyesaJó="http://iskola.hu/egy" >
  <oszt>4</oszt>
  <oszt>5</oszt>
  <AzEgyesaJó:oszt>2</AzEgyesaJó:oszt>
</Osztályzatok>
```

Kivételek persze mindig vannak. A névtéröröklődési láncot megszakíthatjuk bármely ponton úgy, hogy egy üres névtérdefiníciót alkalmazunk a gyermekelemre, és attól a szinttől lefelé már nem lesz érvényes az alapértelmezett névtér:

```
<Osztályzatok
xmlns=http://iskola.hu/Ot >
  <oszt xmlns="">
   <o1/>
    <o2/>
  </oszt>
  <oszt>5</oszt>
</Osztályzatok>
```

A példában a "http://iskola.hu/Ot" alapértelmezett névteret kapcsoljuk ki az első <oszt> elemre, és az ő 2 gyermekére (<o1/> és <o2/>).

Fontos, hogy az alapértelmezett névtér nem vonatkozik az attribútumokra, csak a tagokra, azaz az <oszt megj="Okos gyerek"> esetén a megj attribútum az nem tartozik az alapértelmezett névtérhez *(semmilyen névtérhez sem tartozik)*. Ha azt szeretnénk, hogy ahhoz tartozzon, akkor minden egyes attribútum elé is ki kell írni a névtér előtagot: <oszt AzEgyesaJó:megj="Okos gyerek">. Ezzel még sokszor fogunk találkozni az adatbázis-XML kapcsolatok leírásánál.

A fegyvertárunk

Cikksorozatunk további részeiben elmélyedünk a sémák, XSL transzformációk és az XPath rejtelmeibe. Ahhoz, hogy e kalandozásaink kézzelfoghatóak, kipróbálhatóak legyenek, nézzük végig milyen programok, eszközök állnak a rendelkezésünkre, ha XML alapú fejlesztésekbe szeretnénk belevágni. A jövőbeli XML alapú fejlesztések teljes támogatására a Microsoft Visual Studio.NET lesz a legkényelmesebb integrált eszköz. Amíg azonban nincs (legalább fél év), addig is kell valamivel dolgoznunk. Milyen eszközökkel tehetjük ezt meg?

Microsoft XML Parser 3.0 [1]

ő mindennek a lelke, a mozgatórugója. Az XML parser egy COM komponens, amelybe számtalan funkciót belezsúfoltak. Mivel COM komponens, minden COM-ot támogató nyelvből és eszközből használható, VB-ből, VBScript-ből *(ASP)*, JavaScript-ből *(ASP-ben, böngészőben)*, Visual C++-ban, vagy akár Windows Scripting programokban. *(Azt, hogy Office termékekből is használható nem említettem, de természetesen azokból is működik)*. A .NET framework alatt található XML Parser még nem teljes *(ami nem is csoda, hisz még Beta 1 szinten van)*, de a .NET nyelvekből egyszerűen meghívhatók a COM komponensek, így egyelőre abban is az XML Parser 3.0-t érdemes használni.

A parser-ben található szolgáltatásokat két nagy csoportra oszthatjuk: az egyik az XML DOM *(Document Object Model)*, a másik a SAX API *(Simple API for XML)*.

Az XML DOM célja, hogy egy XML dokumentumot beolvasson, értelmezzen, és lehetővé tegye a dokumentumban *(XML fában)* való mozgást és módosítást programozható felületeken keresztül. Azaz miután beolvasta XML csodánkat, képesek leszünk programból új tagokat beleilleszteni, attribútumokat módosítani stb. Ugyanezzel a modullal lehet XSL fájlokat is beolvasni, majd egy XML dokumentumra végrehajthatjuk az XSL-ben leírt transzformációkat.

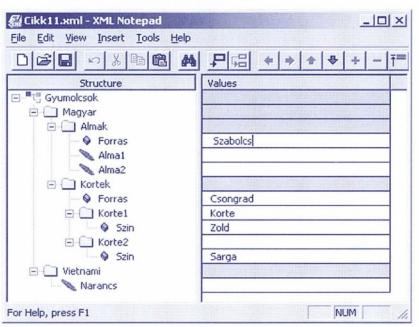
A DOM nagyon kényelmes kis XML állományok feldolgozásához, azonban több megabájtnyi XML fájlok feldolgozására nem megfelelő, mert beolvassa az egész XML fájlt a memóriába, és csak utána értelmezi. Ez különösen kiszolgálóoldali felhasználásnál okoz nagyon hamar problémákat, ahol sok felhasználó egyszerre akarna nagy XML-eket értelmeztetni.
Kiszolgálóoldalon, webalkalmazásokban hasznos lehet az alkalmazás indulásakor felolvasni és értelmezni a sokszor használt XML dokumentumokat, sémákat és XSL-eket. Kifejezetten erre a célra találunk a komponensben olyan Free-Threaded DOM-ot, ami elviseli azt, hogy egyszerre sok weblap egyszerre használja.
Az előbb említett memóriaproblémákon tud segíteni a SAX. A DOM-mal ellentétben a SAX egymás után olvassa fel az XML forrás sorait, és előre meghatározott feltételek esetén egy eseménnyel jelzi a hívó program felé, hogy felolvasott és értelmezett egy adag XML tagot. Ilyenkor a hívó program feldolgozza a kapott adatköteget, majd visszaadja a vezérlést az értelmezőnek. Ezzel a módszerrel tetszőleges méretű XML állományokat is feldolgozhatunk minimális memória felhasználásával, mert mindig csak az éppen feldolgozás alatti dokumentumrész van a memóriában. Egy webalkalmazásban ez nagyon kifizetődő tud lenni.
Emellett a csomagban találunk egy HTTP-n keresztüli XML kommunikációra alkalmas objektumot is, melynek segítségével HTTP protokollra ültetve *(ezt szinte minden tűzfal szereti!)* tudunk XML dokumentumokat küldeni a fogadó alkalmazásoknak. Ez már a Webszervizek lehetőségét villantja fel!
Az XML Parser type library-jét Microsoft XML, v3.0 néven találhatjuk meg a fejlesztőeszközökben. Részletes dokumentációt az MSDN library-ban találhatunk, illetve weben a **[2]** címen.

XML fejlesztőeszközök, segédprogramok

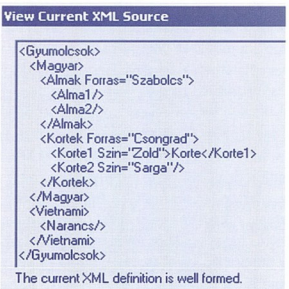
Az XML parser, mint alapkomponens nélkülözhetetlen, de sok egyéb eszközünk is van, amelyeket kihasználhatunk a fejlesztésekben. Lássuk ezeket!

XML Notepad [3]

Az XML nélküli testvéréhez hasonló bonyolultságú eszköz, a fejlesztőeszközök csúcsa, mellyel grafikus felületen szerkeszthetjük meg az XML fánkat. Első ránézésre elég félrevezető a felhasználói felülete, azonban némi gyakorlattal jól használható kis program. Íme egy kép róla, működés közben:



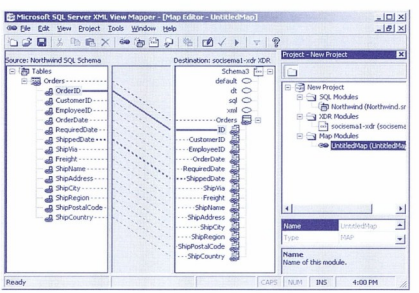
A baloldalon található az XML fa, a jobboldalon pedig annak tartalma síkba kiterítve. Tehát egy-egy tag, attribútum értékét a jobb oldalon írhatjuk be, míg a baloldalon jobb klikk, helyi menükkel hozhatjuk létre a gyerekelemeket, attribútumokat satöbbi. Az elkészült mű forrását megtekinthetjük View/Source... menüpont alatt:



Amellett, hogy láthatjuk az XML forrásunkat, még az is kiderül, hogy sikerült-e jól formázott dokumentumot összehoznunk.

SQL Server XML View Mapper [4]

Az októberi Tech.net különszámban már írtam az XML nézetekről, melyek nem mások, mint az SQL Server 2000 és az IIS együttműködésével előállt transzformációk, melyek a háttér SQL adatbázis adatait szolgáltatják tetszőleges felépítésű XML formátumban. Ehhez egy XML formátumú fájlt kell készítenünk, ami az adatbázismezők és a kimeneti XML dokumentum elemei között teremt kapcsolatot. Ez az XDR annotált séma.
Az XML nézetek segítségével el lehet rejteni a forrásadatbázis szerkezetét, és tetszőleges XML formátumú kimenő adatokat tudunk generálni az SQL Server táblákból. A tipikus feladat, hogy kapunk egy sémát, ami leírja a kívánt XML adatok formátumát, és nekünk össze kell párosítani az SQL Server mezőit-tábláit a kívánt XML dokumentum elemeivel. Ez nagyon mechanikus munka, és ebben segít az XML View Mapper.
A program felhasználói felületének jobb oldalán láthatjuk az összeállítandó XML nézetünket, a baloldalon pedig egy létező SQL adatbázis tábláit és nézeteit. A két oldal mezői között fogd, és vidd módon lehet összerendeléseket teremteni. Az XML nézetben lehet logikai JOIN-okat is létrehozni, azaz olyan táblákat JOIN-olni, amelyek adatbázisszinten nincsenek összekötve. Ennek is elég bonyolult a formátuma, azonban az ehhez szükséges kódot is legenerálja a program.



XML Lint [5]

Ez egy nagyon egyszerű kis parancssori program, melynek segítségével ellenőrizhető, hogy egy XML dokumentum jól megformázott-e, valamint, hogy megfelel-e egy paraméterként megadott DTD-nek vagy XDR-nek.

MSXSL [6]

Parancssori segédprogram, melynek segítségével XSLT transzformációkat tesztelhetünk, illetve sok állományra végrehajtathatunk. Azaz veszi a forrás XML dokumentumokat, végrehajtja rajtuk az XSLT fájlban kijelölt transzformációt, és generál egy eredmény XML fájlt.
Például, ha a forrásfájl, a.xml tartalma:

```
<?xml version="1.0"?>
<xslTutorial >
<title>XSL</title>
<author>John Smith</author>
</xslTutorial>
```

Az transzformálófájl, a.xsl tartalma:

```
<xsl:stylesheet version='1.0'
xmlns:xsl='http://www.w3.org/TR/WD-xsl'>
<xsl:template match="/">
  <H1><xsl:value-of select="//title"/></H1>
  <H2><xsl:value-of select="//author"/></H2>
</xsl:template>
</xsl:stylesheet>
```

MSXSL-lel a transzformáció:

```
msxsl a.xml a.xsl -o k.xml
```

És a kimenet, a k.xml tartalma:

```
<H1>XSL</H1>
<H2>John Smith</H2>
```

XML for SQL Server 2000 [7]

Az SQL Server 2000 XML támogatásának fejlesztésekor egy fontos cél kimaradt a végleges termékből - időhiány miatt. Így történhetett az meg, hogy az SQL Serverből jól kidolgozott, kényelmes módon lehet elérni a tárolt adatokat XML formátumban XML nézetekkel és XML sablonokkal, addig visszafelé, az adatok módosítása még csak SQL Scriptből, az OPENXML segítségével lehetséges. Azaz közvetlenül XML formátumú leírással nem lehet adatokat beletuszkolni a szerverbe, XML nézeteken keresztül. Ezt a funkciót az XML for SQL Server 2000 csomag segítségével felokosított SQL Serveren lehet elérni. A bűvszó az updategram, amely lehetővé teszi a közvetlen, XML adatmódosító parancsok végrehajtását. Csak a teljesség kedvéért megmutatok egy updategram-on keresztüli SQL tábla módosítást, a részletes tárgyalásra külön cikket szánunk.

```
<ROOT xmlns:updg="urn:schemas-microsoft-com:
↳  xml-updategram">
<updg:header>
  <updg:param name="EmployeeID"/>
  <updg:param name="LastName" />
</updg:header>
  <updg:sync >
    <updg:before>
      <Employees  EmployeeID="$EmployeeID" />
    </updg:before>
    <updg:after>
      <Employees  LastName="$LastName" />
    </updg:after>
  </updg:sync>
</ROOT>
```

Az összes, ebben a cikkben felsorolt telepítőcsomag és program letölthető a NetAcademia ftp szerveréről is a **[8]** címen.

Zárszó

Sorozatunk első részében belekóstoltunk az XML technológia alapjaiba. A következő részben megnézzük, mind ügyfél, mind kiszolgálóoldalon az XML dokumentumok feldolgozásának lehetőségét az XML DOM-mal. Addig is a **[9]** címen található példaalkalmazások segítségével bátran lehet tesztelni a jól formázott XML dokumentumokat, valamint ki lelhet próbálni saját, illetve előre elkészített példa XSL transzformációkat. A jövőben is követjük ezt a jó hagyományt, és minden XML cikkünkhöz itt lesznek elérhetőek a példaalkalmazások és tesztprogramok.

Soczó Zsolt MCSE, MCSD, MCDBA
NetAcademia Kft.

A cikkben szereplő URL-ek:

[1] MSXML Parser 3.0 Release http://download.microsoft.com/download/xml/Install/3.0/WIN98Me/EN-US/msxml3.exe
[2] XML Parser referencia http://msdn.microsoft.com/library/psdk/xmlsdk/xml_9yg5.htm
[3] XML Notepad http://msdn.microsoft.com/xml/NOTEPAD/download.asp
[4] SQL Server XML View Mapper 1.0 http://download.microsoft.com/download/SQLSVR2000/Install/1.0/NT5/EN-US/setup.exe
[5] XML Lint http://msdn.microsoft.com/downloads/tools/xmlint/xmlint.zip
[6] MSXSL http://msdn.microsoft.com/msdn-files/027/001/485/XSLTCommandLine.exe
[7] XML for SQL Server 2000 Web Release 1 http://download.microsoft.com/download/SQLSVR2000/Install/1.0/W9X2KMe/EN-US/XML%20for%20SQL.EXE
[8] A fenti eszközök tükrözése a NetAcademia ftp szerverén ftp://ftp.netacademia.net/tools/xml
[9] NetAcademia XML oldalak http://technet.netacademia.net/feladatok/xml

K: *Egy SCSI merevlemezen üldögélő Windows 2000-t átmásoltam egy IDE lemezre. Norton Ghostot használtam, tehát a boot szektor, az MBR és minden más átvitelre került, az IDE partíció is aktív. Az új helyen azonban még sem indul el, kék képernyőzik. A hibaüzenet: INACCESSIBLE_BOOT_DEVICE. Mintha nem ismerné fel a lemezt. Már mindent kijavítottam, a BOOT.INI-ben is átírtam Multi()-ra a menüpontot, mégsem találja. Mi lehet az oka?*

V: Az ok egyszerű. Ha egy konfigurációban nincs IDE lemez, akkor a Windows 2000 nem indítja el feleslegesen annak meghajtóját *(ATAPI.SYS)* sem. Másolás előtt „meg kellett volna mutatni" neki az IDE lemezt, detektálta volna, s az ATAPI.SYS is elindult volna. De még most sem késő, van egy megoldási lehetőség: a telepítő CD-ről indított Recovery Console segítségével képesek vagyunk a halott NT-n módosításokat végezni, többek között eszközmeghajtókat beindítani. Ha nincs telepítve a Recovery Console, bootoljunk Windows 2000 CD-ről, majd telepítés helyett válasszuk a „Repair" üzemmódot, abból pedig a Recovery Console-t. Egy csodálatos, karakteres „operációs rendszerhez" jutunk, amely ismeri a legtöbb DOS parancsot *(dir, copy stb.)*, és – korlátozottan ugyan, de - képes registrymatatásra is. Adjuk ki az

```
ENABLE ATAPI SERVICE_BOOT_START
```

Parancsot, ami átbillenti az ATAPI.SYS indítási értékét a regisztrációs adatbázisban, s az a következő rendszerindításkor elindul, így a Windows 2000 fel fogja ismerni a lemezt. Egyébként mindenkinek melegen ajánlom, hogy túlélőkészletébe vegye fel a Windows 2000 telepítőlemezt, mert a Recovery Console segítségével nemcsak halott Windows 2000 boncolható, hanem Windows NT 4.0 is! Mi több, a Recovery Console jó előre feltelepíthető a gépekre, hogy probléma esetén azonnal, és gyorsan hozzáférhessünk. Telepítése:

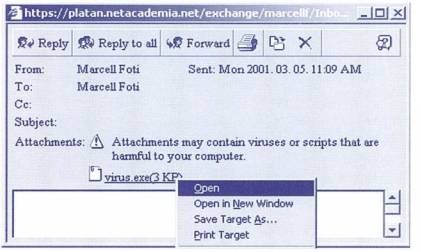
```
I386\WINNT32 /CMDCONS
```

Ennek hatására lefut egy minisetup, és feltelepíti a mintegy 7 megabájtos „operációs rendszert", s felveszi a rendszerindító menübe, a BOOT.INI-be. Katasztrófa esetén tehát egyszerűen csak kiválasztjuk a boot menüből, és már mi is!
Forrás: NetAcademia Windows 2000 lista

K: *Feltettük a levélbombák ellen védő Security patchet a vállalat számítógépein futó Outlook 98-akra. Roppant elégedett vagyok a védelemmel, talán túl jól is sikerült: azóta EXE csatoláshoz semmilyen módon nem férek hozzá. De hogyan lehetne kivételesen mégis leszedni az attachmentként érkező tiltott fájlokat? Vagyis egy konkrét levél vagy személy esetén hogy lehet letiltani a védelmet?*

V: Outlookkal sehogy. Ráadásul ezt a patchet le sem lehet venni. De van kerülőmegoldás:

1. Az Outlook Patch semmilyen hatással nincs az Oultook Web Accessre. Menj be böngészővel a levelesládádba, és töltsd le a kívánatos fájlt *(http://tegéped/exchange)*.



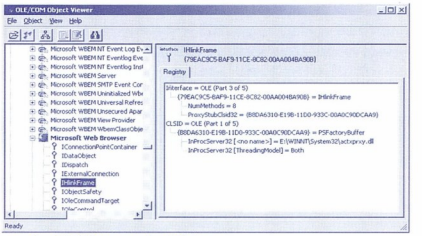
👁 ***Futtassunk vírust OWA-val! :)***

2. Meg lehet kísérelni a patch átverését. Küldesd el magadnak ismét az EXE-t, de a feladó nevezze át valami semleges kiterjesztésre, például semmi.aaa. Ez letölthető, és visszanevezhető.

Forrás: NetAcademia Exchange 2000 lista

K: *Honnan lehet „kitalálni", hogy egy ActiveX objektumnak milyen függvényei vannak? Az Outlook, Outlook Express és az Excel érdekelne.*

V: Ezek az információk általában megtalálhatók az msdn.microsoft.com címen, de kézzel is kibányászhatók. Minden ActiveX objektumnak van úgynevezett Type Library-ja, ami némi információt nyújt az objektumról. Ezeket mindenféle ravasz API hívásokkal le lehet kérdezni. Kódolás nélkül pedig a konyhakész OLEView eszköz használatával jutunk ezekhez az adatokhoz. Az OLEView a platform SDK-ban és a Visual Studioban is megtalálható.



👁 ***Az OLEView munka közben***
Forrás: NetAcademia Msdev lista

K: *Mit tehetünk, ha az SQL Server beépített aggregátumfüggvényei (Sum, Avg stb.) helyett másféle aggregátumot szeretnénk készíteni? Hogyan lehet szorzatfüggvényt használni? Az SQL 2000 új felhasználói függvényei jók lennének, de SQL7 adatbázisban kell megoldani a feladatot. Mit lehet ilyenkor tenni?*

V: Egyik listatagunk állt elő ezzel a megoldással, amely szorzásra valóban jó, azonban az SQL 2000 felhasználói függvényeinek rugalmasságát természetesen nem lehet pótolni natív trükközéssel. Ha van egy táblánk, amelyben a „result" mező tartalmát kell összeszorozni, használhatjuk az alábbi megoldást:

```
select id, exp(sum(log(result)))
   from tabla
      group by id
```

Forrás: NetAcademia SQL2000 lista

K: *Hogyan lehet megtudni egy távoli gép MAC Addressét?*
V: Pl. pingeléssel. A Ping előtt ugyanis lefut egy ARP kérdés, amely a kezdeményező gépre bekéri a távoli masina MAC Addressét. Tehát így:

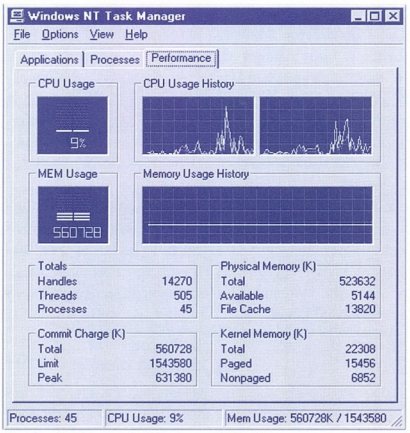
```
PING gep
ARP -A
```

Vigyázat! Ha a távoli gép router túloldalán van, akkor ez a módszer a router MAC Addressét adja meg!
Forrás: NetAcademia Windows 2000 lista

K: *Egy dual processzoros alaplapon jelenleg egy processzorral fut az NT4.0 . Most beletennénk a második procit, de attól félek, hogy a HAL cseréje miatt újra kell majd telepíteni az NT-t. Igaz ez?*
V: Nem igaz. A Windows NT Resource Kiten található az UPTOMP.EXE eszköz, amely elvégzi a konverziót. Mellesleg összesen öt fájlt cserél le, amit kézzel is meg lehet tenni **[2].** Az eljárás menete:

1. Második processzor bedugaszolása
2. Boot. Ilyenkor a két proci közül csak az elsőt látja az NT
3. UPTOMP.EXE
4. Reboot

A műtét eredményességéről például a Task Managerrel győződhetünk meg, ahol a Performance fülön innentől két ablakocska látható majd a processzor terheltségéről: a bal, meg a jobb.



Forrás: NetAcademia Windows 2000 lista

K: *Windows 2000 tartományom alatt a DNS Serverben megpróbáltam „rendet rakni" azonban úgy tűnik, sikerült szétvernem a Windows 2000 nélkülözhetetlen SRV rekordjait. Hogyan tudnék visszatérni az eredeti, hibátlan állapothoz? Zónafájlmentésem nincs!*
V: Ezt a feladatot bízzuk a Windows 2000-re! Induljunk tiszta lappal, töröljük ki a zónafájlból a még megmaradt roncsokat, majd a

```
IPCONFIG /REGISTERDNS
```

parancs segítségével vetessük vissza a hostrekordokat, végül a

```
NET STOP NETLOGON
NET START NETLOGON
```

páros beregisztrálja a _MSDCS, _GC és a többi kívánatos bejegyzéseket.
Forrás: NetAcademia Windows 2000 lista

K: *Ügyesen frissítgetünk Windows 2000-re, már az összes tartományvezérlőnk átállt. Most szeretnénk átállni Mixed módról Natív módra, de félünk, hogy ezzel elveszítjük a régi, Win9X ügyfeleket. Mi lesz a PDC emulátorral, a NetBIOS támogatással, a WINS-sel? Lesz-e NTLM hitelesítés?*
V: A Natív és Mixed mód közötti különbségek egyike sem vonatkozik a kiszolgálók külső hálózati kapcsolataira. Gondoljuk végig: a kompatibilitás jegyében mi mindent tettek a redmondi fejlesztők. Még a DOS-os kliensek is használhatók! A Natív módra váltással ezek a szolgáltatások NEM tűnnek el! Az egyetlen érezhető korlátozás az lesz, hogy többé nem lehet a tartományba NT4-es Backup Domain Controllert felvenni.
Forrás: NetAcademia Windows 2000 lista

K: *A Windows 2000 Professional-ban hiába állítom be a magyar Regional Settings-et, a naptárban a hét szerdával kezdődik.*
V: Egy registry beállításon múlik az egész. Az **[1]** címről letölthető hetfo.reg fájl az alábbi módosításokat tartalmazza:

```
[HKEY_CURRENT_USER\Control Panel\International]
"iFirstDayOfWeek"="0"

[HKEY_USERS\.DEFAULT\Control Panel\International]
"iFirstDayOfWeek"="0"
```

Az első sor az aktuálisan bejelentkezett felhasználó naptárát állítja helyre, a második pedig az alapértelmezést. Ezután minden újonnan létrehozott felhasználónak hétfőn kezdődik a hét *(pedig milyen jó volt még két napot otthon lustálkodni ;-))*
Forrás: NetAcademia Windows 2000 lista

A cikkben szereplő URL-ek:
[1] ftp://ftp.netacademia.net/patch/w2k/hetfo.reg
[2] http://support.microsoft.com/support/kb/articles/Q156/3/58.asp

Háló a falon

Gary Starkweather, a Microsoft egyik vezető kutatója szerint eljön még az a nap, amikor képernyő „tapéta" borítja a szobák falát, és a látvány bőségesen kárpótol majd mindenkit azért a kis nyűgért, amit ennek a „felragasztása" jelent.

Olcsó húsnak nem mindig híg a leve

Mit nem adnék érte, ha a szobám egyik falát így díszíthetném! Hát bizony több millió forintot nem. Százezernél már talán nem hervadna le a mosoly az arcomról, de a legnagyobb sajnálatomra eddig egyetlen cég sem rukkolt elő elfogadható árú lapos képernyővel. És nyugodt lélekkel állíthatom, hogy nem a cinizmus szólt belőlem, hanem az átlagos fogyasztó.

Sajnos, van itt egy-két probléma. A fali képernyő energiaellátása legalább akkora kihívás, mint a megépítése. A képek megjelenítéséhez és mozgatásához szükséges sávszélesség sem éppen szokványos tervezési eljárást kíván. Egy 108 x 72 hüvelyk *(kb. 274 x 183 cm)* nagyságú, 300 dpi-s faliképernyő felbontása 32400 x 21600 képpont, így a képátviteli igény meghaladná az 50 gigaképpontot másodpercenként. Figyelemreméltó számok. A kijelző elképesztően sok fényvisszaverő vagy -kibocsátó alkotóelemből állna, ráadásul ezek összekapcsolása és energiaellátása sem magától értetődő feladat. Hiába bővelkedünk a műszaki megoldásokban, mindegyiknek megvan a maga baja. Az elölről vetítő kijelzők képére árnyékot vethet bárki, aki használja; a hátulról vetítőknek meglehetősen nagy a területigénye, abból pedig soha sincs elég; az LCD klassz, de előállítása drága. Ezeknek a problémáknak a legtöbbjét megvizsgálta a Microsoft egyik kutatócsoportja, de a gyártási költségekkel ezidáig nem tudtak megbirkózni.

Starkweather barátunk azonban nem az a visszariadós típus. *(Szerencsére hiába mondták neki annak idején, hogy a lézernyomtató senkinek sem kell majd.)* Mike Sinclair kutatótársával most éppen olyan miniatűr kijelzőképpontok kifejlesztésén dolgozik, amelyekből tetszőleges nagyságú képernyők állíthatók össze. Nevezzük nevén a gyereket: a MEMS *(Micro-ElectroMechanical Systems)* technológiáról van szó.

Trükkök a tükrökkel

A MEMS technológiával olyan, mikroméretű eszközök készíthetők, amelyek elektromos és mechanikus összetevőket ötvöznek. Sinclair szerénysége bámulatos: állítása szerint még csak tanulja a MEMS-t, de már bemutatta saját készítésű, az emberi hajszálnál alig nagyobb vezérlő- és kapcsolóelemeit. Most 40000 apró tükörből álló, hüvelykujj nagyságú kijelző elkészítését tervezi. Mindegyik tükörnek saját motorja és memória lesz. Ha a memória tartalma 0, a tükör az egyik irányba térül, ha 1, a másik irányba fordul. A tükrök nagyon gyorsan váltanak irányt – hasonlóan a versenypályák eredményjelző tábláinak apró lemezeihez, – így a fénytörés elhanyagolható mértékű. Spékeljük meg mindezt egy kis piros, kék és zöld lézerrel, és már el is készült a hihetelenül nagy felbontású kijelző, ami ráadásul olcsó, nem rongálja a szemünket, és jó nagy képernyőt lehet belőlük összeállítani.

A valóság sem mindig keserű

„Ha elkészíted, meg is veszik!" Ez a szlogen minden bizonnyal nem egy céget sodort már a csőd szélére, de legalábbis nagy bajba. Mary Czerwinski kijelentésével, miszerint a Microsoft a felhasználók érdekeinek szem előtt tartásában az egyik legjobb, valószínűleg sokan vitába szállnának. *(Ebből rögtön ki is található, hogy hol kutató ő.)* Ha netán elragadna valakit a kritikai érzéke, gondoljon talán más cégek termékeire is. Az előzetes piacelemzés semmi esetre sem érdem, ezért nem fogjuk megdicsérni a redmondi fenegyerekeket. De másért igen. Miközben a hardveres csapat egyre jobb megoldásokkal rukkol elő, Czerwinski a szélesvásznú vetítők felhasználóit tanulmányozza: Mit kezdenek ezzel a nagy felülettel a 17" monitorok után? Használják-e majd a szélét, és ha igen, akkor miért nem, de legfőképpen hogyan?

Emellett egy másik projektben, a tevékenységek megszakításának a munkafolyamatokra gyakorolt hatását is vizsgálja. Ez a párosítás első hallásra kissé talán elvontnak tűnik, de semmi esetre sem a véletlen műve: elég ehhez azt végiggondolni – mint ahogy ezt a cikk elején meg is tettük –, hogyan ingázunk a különféle alkalmazások ablakai között.

Czerwinski szerint a nagy képernyőket kivetítő kis dobozok nemcsak egyszerűen ebben a minőségben foglalják majd a helyet az asztalunkon. A legtöbben térben tájékozódnak a legjobban, így az alkalmazások és dokumentumok háromdimenziós környezetbe helyezett „ikonjait" valószínűleg előbb megtalálják, mint kétdimenziós megfelelőjüket egy hierarchikus felépítésű mapparendszerben *(not my favorites)*.

Ezek a gondolatok a műszaki fejlődésben élen járó országokban is igazi víziónak számítanak, nálunk pedig még a 14" monitorok kora sem áldozott le. Ne essünk azonban abba a hibába, hogy nem vesszük észre az elkerülhetetlent. A mai fiatalok *(mondom ezt 29 éves fejjel!)* képesek párhuzamosan akár több csevegőablakban is értelmesen beszélgetni, miközben szól a rádió, és a holtidőben a házi feladatuk is elkészül.

A jövőbelátás képességével megáldott Czerwinski azt állítja, hogy a nagy képernyőknek nagy jövője lesz. Én mindenesetre üresen hagyom a lakásomban az amúgy is üres, legnagyobb falat – egyelőre. És ha egyszer netán ilyen „tapéta" lesz rajta, ígérem, hogy többet nem nézem meg a Vissza a jövőbe című filmet.

Zacco

Microsoft®

TechNet szemináriumok a Microsoft Magyarország szervezésében!

Helyszín: BUDAPEST, Lurdy Ház, Hollywood Multiplex Mozi (1097 Budapest, Könyves Kálmán krt. 12-14.)

A Windows 2000 alapú hálózatok biztonsági kérdései

Időpont: 2001. 04. 04.

Ezen az előadáson a Windows 2000 operációs rendszer nyújtotta biztonsági szolgáltatásokkal foglalkozunk. Egy egygépes rendszerből kiindulva a nap során fokozatosan eljutunk egy nagyvállalati rendszer címtár segítségével felügyelt, PKI infrastruktúrán alapuló rendszerig. Sok egyéb mellett lépésről-lépésre bemutatjuk, hogyan lehet engedélyezni az intelligens kártya alapú felhasználói bejelentkezést, hogyan kommunikálhatnak titkosítva a kiszolgáló farm elemei egymással, és hogyan küldhetünk egymásnak titkosított vagy digitálisan aláírt levelet.

Ismeretkezelő és csoportmunka-megoldások Microsoft-platformon

Időpont: 2001. 04. 18.

A szemináriumon egy tipikus Windows NT/2000 fájl- és nyomtatókiszolgálókat üzemeltető, Office-t és e-mailt használó vállalati környezetből kiindulva fokozatosan építünk fel egy fejlett csoportmunka-, ismeretkezelő és kommunikációs megoldásokat alkalmazó rendszert.

Microsoft Üzemeltetői Konferencia

Időpont: 2001. 04. 25.

A konferencia elsősorban üzemeltetési vezetőknek és rendszer üzemeltetőknek szól. Segítséget nyújt a nagy- és középvállalatok informatikusainak és rendszergazdáinak, hogy a már meglévő Microsoft technológiákra épülő rendszereiket még megbízhatóbban és egyszerűbben üzemeltethessék. Szeretnénk bemutatni és átadni mindazt a tudást, ami a Microsoftnál és partnereinél az üzemeltetési és terméktámogatási gyakorlat során felhalmozódott.

Rendszerfelügyelet Windows 2000 környezetben

Időpont: 2001. 05. 02.

Az előadás a Windows 2000 alapú vállalati környezetek rendszerfelügyeleti és üzemeltetési kérdéseit tárgyalja. A gyakorlatias megközelítésű előadáson egy kiépített Windows 2000 Active Directory, Systems Management Server (SMS) 2.0 és Microsoft Operations Manager környezetben mutatjuk be az asztali gépek és alkalmazások, illetve a kiszolgálók felügyeletét és a napi üzemeltetési teendőket.

Üzleti Intelligencia: adatraktárak létrehozása és az adatok elemzése

Időpont: 2001. 05. 16.

Minden vállalat használ adatbáziskezelőt, de vajon kihasználja-e a felhalmozott üzleti célú adatokban rejlő információkat? Az előadások az adattárház kezelés folyamatának legfontosabb lépéseit tekintik át. Ismertetett technológiák: Data Transformation Services, OLAP kiszolgáló *(Analysis Services)*, MDX, ADO DM.

NetAcademia Kft. H-1105 Budapest, Ihász utca 13. Tel.:36 (1) 263-2732 Fax: 36 (1) 261-7145 E-mail: info@netacademia.net

NetAcademia nyári konferencia!

A NetAcademia **2001. május 18-19-én**, kétnapos konferenciát tervez a **Balatonparton**, **Kenesén**. A konferencián sajnos csak 150 fő vehet részt, így előzetesen szeretnénk felmérni, elég lesz-e ez a hely az érdeklődőknek, avagy valami nagyobb helységet kell-e keresnünk.

Minden érdeklődő előzetes jelentkezését várjuk a következő e-mail címre: **tabor@netacademia.net** Az előzetes jelentkezés nem számít megrendelésnek, egyszerű felmérést szeretnénk végezni, vajon mennyi résztvevőre számíthatunk.

A tervezett program:

Május 18. péntek

10:00-12:00	Érkezés a konferenciaközpontba, ebéd
13:30	A konferencia megnyitója, szekcióra való feloszlás. Esetleges problémák megbeszélése, szimulálása. *(A szekció itt a különböző-levelezési listákat, a különbözőlevelezési listák témakörei szerinti feloszlást jelenti! Levelezési listáinkat megtalálja a* ***http://lyris.netacademia.net*** *címen)*
16:30-18:30	Szórakozás - azaz informális szekciózás
19.00	Vacsora, az elsőnap tapasztalatainak összefoglalása a szervezők szemszögéből, esetleg a szekciók 1-1 kiragadott - érdekesebb - esetének a többiekkel való megosztása.

Május 19. szombat

10.00	Szekciózás, előadások
12.00-13.00	Ebéd
13.30-15.00	Még egy utolsó szekciózás
15.00	A konferencia zárása
16.00	**VÉGE**

A szekciózásokra igyekszünk megnyerni illusztris vendégeket, érdekes témákkal, de magánjellegű felvetéseknek, akár kiselőadásoknak is helyet kapnak majd! A konferencia részvételi díja, mely tartalmazza a szállás és az étkezés költségét körülbelül 20-22 ezer forint *(+ ÁFA sajnos...)*

Kérjük, amennyiben felkeltettük érdeklődését, a mellékelt jelenzkezési lapot faxolja el kitöltve a 261-7145-ös faxszámra. Mivel csak előzetes felmérésről van szó, faxát nem tekintjük megrendelésnek. A konkrét megrendelés ügyében kollégáink a későbbiekben veszik fel a jelentkezőkkel a kapcsolatot.

NetAcademia Kft. H-1105 Budapest, Ihász utca 13. Tel.:36 (1) 263-2732 Fax: 36 (1) 261-7145 E-mail: info@netacademia.net

NetAcademia nyári konferencia 2001.05.18-19.

Előzetes megrendelés

Kérjük az alábbi megrendelőt kitöltve küldjék vissza a fent megadott faxszámra.

Megrendelő cég neve:
Résztvevő(k) neve:
Számlázási cím:
Telefon / Faxszám:
E-mail cím:

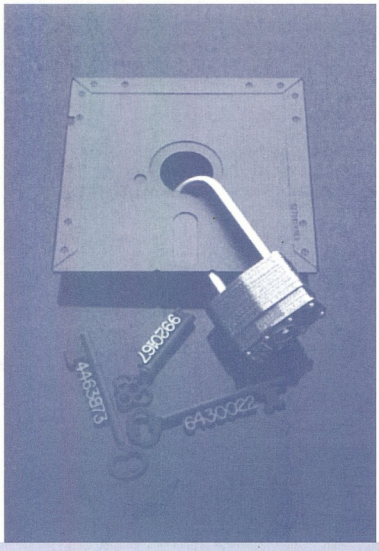
Dátum:

Pecsét

.....

Cégszerű aláírás

Megjegyzés:



Hallott már Ön selejtmentes gyártásról? Rendszer-felügyeletmentes hálózatról (ZAK, ZAW)? Feltörhetetlen hálózatról (C2)?

Elérhetetlen, de megközelíthető célok. Az első kettő elérésében sajnos nem segíthetünk. De a harmadik cél megközelítésében sokat tehetünk Önért. Világszínvonalú, egyedi Security tanfolyamainkon megtudhatja, mitől döglik a hacker!

1. Biztonsági technológiák

rendszergazdáknak, 3 nap

- Titkosítási algoritmusok *(DES, RSA, MD5 stb.)* elmélete és gyakorlata
- Az authentikáció biztonsága *(Kerberos, NTLM, NTLMv2)*
- A nyílt kulcsú titkosítási technológia nagyvállalati, gyakorlati alkalmazása *(Certificate Server, SmartCard, biometrikus azonosítás)*
- Biztonságot növelő technológiák bevezetése *(IPSec, VPN, L2TP, Encrypting File System stb.)*

2. Biztonságos levelezőrendszer kialakítása MS Exchange alapokon

gyakorlott Exchange endszergazdák számára, 3 nap

- Titkosítás és digitális aláírás az elektronikus levelezésben (S/MIME)
- Az Exchange Server védelme, adminisztratív szerepek, jogosultságok, mentés, visszaállítás, journaling
- tűzfal vs. Exchange Server, SMTP relay beállítások, spam védelem, SSL, TSL, nyílt kulcsú titkosítás felhasználása, MAPI, POP3, IMAP, OWA, LDAP protokollok veszélyei.

3. Hálózatunk védelme a támadásokkal szemben

biztonsággal foglalkozó szakembereknek, 3 nap

- Kaló-eszközkészlet *(Denial Of Service, Buffer Overrun, hálózatelemzés, trójai falovak, vírusok, jelszófeltörés, hátsó ajtók)*
- A Windows 2000 biztonsági elemei *(fájlrendszerek, regisztrációs adatbázis, hálózati adatforgalom)*
- Védekezés: tűzfalak, proxyk
- Aktív védekezés: Intrusion Detection

4. Programozzunk biztonságosan!

programozók számára, 3 nap

- Elmélet: védett mód, kernel, memóriakezelés
- Programozási gyakorlatok, rosszul és jól megírt kódok
- A Buffer overrun hatásai user és *(app es service)*+kernel módban
- RFC implementation errors
- JAVA, ActriveX, homokozó, scriptek
- Worm vírusok elemzése *(Iloveyou)*
- CryptoAPI, Smart Card API

Biztonsága érdekében keressen minket!

www.netacademia.net

A legjobbakat tanítjuk.

1105 Budapest, Ihász utca 13. • Tel.:263-2732

NETACADEMIA