
Széchenyi István Egyetem
Műszaki Tudományi Kar

# Informatikai Diákköri Kutatások Szemináriumi Füzetek

1. évfolyam 1. szám
2004.

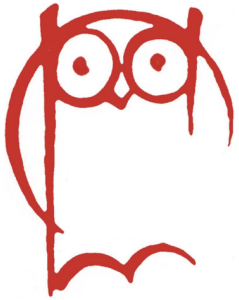
Készült a Széchenyi István Egyetem Műszaki Tudományi Kar Informatikai és Villamosmérnöki Intézete informatikai kutatási szemináriumot vezető oktatójának, dr. RAFFAI Máriának a kezdeményezésére,

dr. CZINEGE Imre rektor,
dr. SCHARLE Péter tudományos rektorhelyettes,
dr. BUKOVECZKY György TDT-elnök,
dr. MOLNÁRKA Győző IVI-igazgató, valamint
az egyetem Tudományos Tanácsának

a támogatásával.

Felelős szerkesztő:
dr. RAFFAI Mária

Kiadja a Széchenyi István Egyetem
Felelős Kiadó az Egyetem rektora
2004. február



# Az Informatikai Diákköri Kutatások Szemináriumi Füzetek jelen kötetében szereplő kutatási témákat

## dr. Raffai Mária

## konzultálta

# Tartalomjegyzék

# Az informatikus hallgatók tevékenysége a Széchenyi István Egyetemen

Intézményünkben az informatikai képzés már több mint egy évtizedes múltra tekint vissza. A szak felfutása időben egybeesik azzal a folyamattal, amelynek során a viszonylag kicsi és erősen specializált főiskolából egy országos összehasonlításban is nagynak számító universitas lett. Ebben a folyamatban meghatározó szerepet játszott az informatika szakok (főiskolai szintű műszaki informatika és az egyetemi szintű gazdasági informatika szakok) intézményi indítása is.

A különböző informatika szakok létrehozásával az alapítók olyan szakemberek képzését kívánják megvalósítani, akik a matematikai, természet- és társadalomtudományi, valamint a szervezési/vezetési ismeretek mellett magas szintű informatikai felkészültséggel, mérnöki készségekkel rendelkezve képesek

– egy szakterület problémáinak a feltárására és elemzésére,
– korszerű alkalmazásfejlesztési elvek, módszerek és technikák készségszintű alkalmazásával a felhasználói elvárásokat kielégítő minőségi szoftvertermék előállítására,
– meglévő alkalmazások korszerűsítésére és integrálására, valamint
– az informatikai rendszerek menedzselésére.

Az oktatási tevékenység mellett az informatikai szakképzésért felelős tanszékek munkatársai egyénileg is és tanszéki kutatócsoportokban is sokrétű tudományos munkát végeznek, amely a szakmai profilnak megfelelően interdiszciplináris területeket érint.

A legjellemzőbb kutatási témák:

– az oktatási tevékenység hatékonyságának vizsgálata, a végzett hallgatók képzésről alkotott véleményének az elemzése (folyamatosan végzett, rendszeres felmérések alapján),
– hatékony oktatási módszerek kifejlesztése, a hallgatócentrikus képzés megvalósításának feltételi rendszere és módozatai,
– információrendszer-fejlesztési módszertanok és technikák hatékony megoldásai, korszerű diszciplínák: objektumorientált elemzési/tervezési megoldások,
– a szoftverfejlesztési folyamat technológiája,
– egységesített módszertanok (RUP), szabványos modellezési technikák a fejlesztésben (UML),
– ősrendszerek értékeinek megőrzése, az informatikai vagyon védelme, a rendszerintegráció jelentősége és megoldásai (MDA),
– a számítógépes hálózatok technológiája, a hálózati alkalmazások tervezésének kérdései,
– informatikai rendszerek biztonsági problémái és megoldásuk,
– biztonságkritikus rendszerek speciális tervezési/tesztelési kérdései,
– számítógéppel támogatott szimulációs rendszerek,
– logisztikai rendszerek fejlesztése,
– közlekedési hálózatok vizsgálata és tervezése,
– számítástudomány, formális nyelvek és automaták elmélete,
– a párhuzamos programozás kérdései,
– döntéselőkészítési megoldások, operációkutatás, döntéstámogatás, valamint
– komputeralgebra és fraktálgeometria.

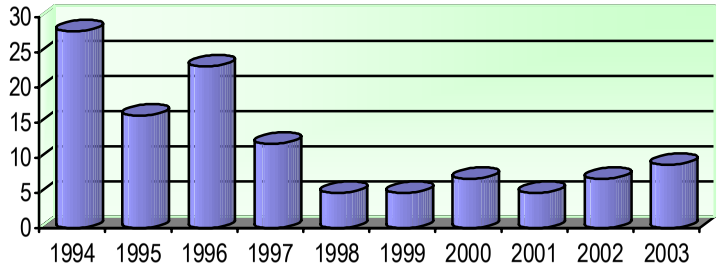
A fenti témákat a kollégák hazai és nemzetközi kutatási projektekben végzik, tudományos eredményeiket többnyire különböző bel- és külföldi konferenciákon ismertetik, szaklapokban és szakkönyvekben publikálják, de számos értékes konferenciaanyag és szakcikk megtalálható az internethálózaton is.

## Tudományos diákköri munka a szakon

Az oktatási és a kutatási tevékenységhez kapcsolódóan az informatika szakokért (műszaki és gazdasági informatika) felelős vezetőoktatók nagy hangsúlyt helyeznek arra, hogy a tanulmányok során kiemelkedő teljesítményt nyújtó hallgatókkal a kötelező tanórán kívül is foglalkozzanak. Ezeket a hallgatókat a kollégák bevonják a kutatómunkába, ösztönzik őket arra, hogy eredményeiket különböző formában ismertté tegyék (előadások, publikációk), és Tudományos Diákköri Konferenciákon mutassák be.

A Műszaki Informatika Szak 1991. évi indítása után a TDK konferenciák *első informatikaszekcióját 1994 tavaszán* rendeztük meg. Ettől kezdve az informatika szakterületen dolgozó oktatók évente átlagosan mintegy 20 tudományos munkát végző hallgatóval foglalkoztak. Az első két évben a tudományos kutatómunkát végző hallgatók magas száma miatt külön rendeztünk Informatikai és külön Számítástechnikai szekciókat, később azonban, részben a kutatási szeminárium hallgatói létszámának a csökkenése, részben pedig a megváltozott témakörök miatt már csak Informatika Szekciót indítottunk. Meg kell jegyezni, hogy néhány hallgatónk, éppen a kutatás eltérő tárgya és irányultsága miatt, más szekciókban (Közgazdasági, Közlekedési, Építési, Környezetvédelmi stb.) mutatta be dolgozatát.

Az elmúlt tíz évben több mint 200 hallgatóval konzultáltunk (összesen 212 fő). A közös munka eredménye 112, Tudományos Diákköri Konferencián is bemutatott dolgozat volt. Elemezve az elmúlt tíz év munkáját megállapíthatjuk, hogy a műszaki informatika szakos hallgatók tudományos tevékenységének lendületes indítását követően a résztvevők létszámában látszólagos visszaesés mutatkozik (lásd 1. ábra), amit azonban nem megtorpanásaként kell minősítenünk, hanem inkább *a tudományosság, az elemző/feltáró kutatások irányába történő előrelépésként.*




1. ábra. A TDK-kon résztvevő hallgatók számának alakulása

A hallgatókkal való foglalkozást, a *tehetséggondozást* a szaktanszékek oktatói *kiemelt feladatnak* tekintik, hiszen olyan fiatalok tudományos érdeklődését keltik fel, akik, mint azt az eredmények is mutatják, méltók arra, hogy nagyobb odafigyelést, több segítséget, ösztönzést és támogatást kapjanak. A kutatási témák természetesen szorosan kapcsolódnak a szaktanszékek munkatársai által művelt tudományterületekhez, vagyis a hallgatók alapvetően információelméleti és –technológiai megoldásokkal foglalkoznak, szoftverfejlesztési munkákat

végeznek, vagy felmérési eredményeket dolgoznak fel és elemeznek. Tevékenységükkel nemcsak saját látókörüket szélesítik, de nagymértékben segítik az egyetem tudományos munkáját is.

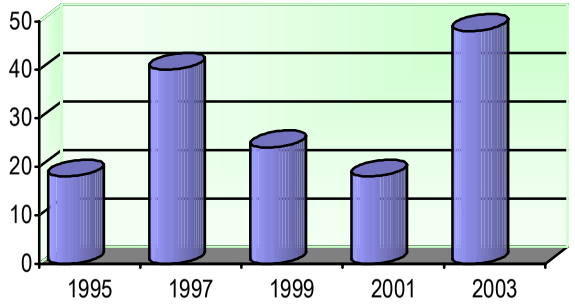
A 2. ábra. a különböző kutatási témacsoportokat és a gyakorisági mutatókat tartalmazza. Jól látható, hogy a dolgozatok közel egynegyede (20,5%) alkalmazásfejlesztési munka volt. Ha mélyebben elemezzük a különböző évek teljesítéseit, akkor láthatjuk, hogy az első két-három év nagy számú rendszerfejlesztési munkáit követően a TDK-tevékenység a tényleges kutatómunka irányába mozdult el [TDK-dok, 93-03]. Az elméleti és módszertani kutatások, a feltáró elemzések, az új megoldások keresése témákban készített dolgozatok sikeresek voltak, és az országos elismerések számának ugrásszerű növekedéséhez vezettek.

| | dolgozatok | |
|---|---|---|
| | száma | aránya |
| Alkalmazások fejlesztése | 24 | 20,5% |
| Elemző, feltáró kutatások | 15 | 12,8% |
| Intelligens megoldások/alkalmazások | 13 | 11,1% |
| Elektronikus üzleti megoldások | 14 | 11,9% |
| Rendszerfejlesztési technológiák | 10 | 8,6% |
| Vizuális és grafikai megoldások | 10 | 8,6% |
| Programozás-technológia | 9 | 7,7% |
| Számítógép-architektúra, hálózati technológiák | 8 | 6,8% |
| Informatikai biztonság és tervezése | 10 | 8,6% |
| Oktatás informatikai technológiák alkalmazásával | 4 | 3,4% |
| Összesen: | 117 | |

2. ábra. A tudományos diákköri munkák témák szerinti megoszlása 1994-2003 között

## Országos Tudományos Diákköri eredmények

A tudományos tevékenység eredményességét mi sem igazolja jobban, mint az a 21 OTDK-díj, amit hallgatóink az 1995. évi országos konferenciától kezdődően folyamatosan szereztek. Az elmúlt tíz év *OTDK-n helyezett pályamunkái* az összes dolgozatok 17,9 %-át teszik ki, de ha a kezdeti útkeresést figyelmen kívül hagyjuk, akkor ez az arány túlhaladja az *egynegyedet* (lásd 3. ábra.). Mivel a dolgozatok egyes konferenciákon elért eredményei nem mutatnak szignifikáns eltérést a díjazott hallgatók arányszámaitól, ezért ezt nem tartjuk fontosnak külön megjeleníteni.




3. ábra. Az OTDK-n eredményesen szereplő hallgatók

Az Országos Tudományos Diákköri Konferencián díjazott pályamunkákat a hallgatók elsősorban az Informatika Szekcióban mutatták be, de néhány dolgozattal, mint ahogyan az egyes OTDK-eredmények értékeléséből is látszik, más szekciókban is képviselték intézményünket. A díjakat és a helyezéseket akkor tudjuk igazán értékelni és minősíteni, ha megismerjük a témákat és az elért eredményeket, vala-

mint az intézménynek is dicsőséget szerzett hallgatókat és konzulenseiket, akik sokat tettek és tesznek azért, hogy az arra érdemes hallgatók megszerezzék a kutatáshoz szükséges ismereteket, és elsajátítsák a készségeket [Raffai et al., 2003].

## Kutatási Szeminárium, eredmények publikálása

A 2003/2004 tanév első félévében új alapokra helyeztük a tehetséggondozási munkát. Összegezve az intézményi és az országos konferencián szerzett tapasztalatokat szükségesnek tartottuk, hogy a hallgatókkal az egyéni konzultációkon túlmenően is foglalkozzunk. A heti rendszerességgel szervezett kutatási szemináriumok alapvető célja, hogy növeljük a hallgatók érvelési és vitakészségét, hogy megtanítsuk őket állításaik helyességének igazolására, mások előtt történő megvédésére, az egyéni és a csoportmunkára, a tudományos igényességű kutatási tevékenységre, valamint eredményeik bemutatására és publikálására.

Tekintettel arra, hogy a kutatási folyamat köztes eredményei nem eléggé érettek arra, hogy országos megmérettetésben vegyenek rész, vagy hogy szakmai folyóiratokban jelenjenek meg, ezért úgy gondoltuk, hogy alapítunk egy olyan publikációs lehetőséget, amelyben a hallgatók a tudományos elvárásoknak megfelelő cikkeiket nyilvánosan megjelentethetik. Útjára bocsátjuk tehát azt a *„Szemináriumi Füzetek”* sorozatot, amely terveink szerint minden félévben számot ad az informatika szakos hallgatók tudományos tevékenységéről, és amelyből egyetemünk oktatói és a hallgatók egyaránt tájékozódhatnak a leendő fiatal szakemberek/kutatók munkáiról.

Ezúton is szeretnénk kifejezni köszönetünket az Egyetemi Tanácsnak és az egyetem felsővezetőinek, akik fontosnak tartják a tudományos diákköri tevékenységet, elismerik a fiatal tehetségekért fáradozó oktatók/kutatók áldozatos munkáját, valamint az egyetemi Tudományos Tanácsnak, amelynek programjában a TDK-tevékenység kiemelt szerepet kap.

Bízunk abban, hogy törekvéseink nemes célt szolgálnak, és munkánkkal hozzájárulunk a fiatalok tudományos igényű felkészítéséhez.

Raffai Mária Ph.D.
a kutatási szeminárium vezetője,
a Szemináriumi Füzetek felelős szerkesztője

## Referenciák

[Raffai et al., 2003] Raffai Mária – Marton László – Kallós Gábor: A Tudományos Diákköri Tevékenység a Széchenyi István Egyetem Informatika Szakán – A TDK-munka tíz éve – Kiadó: Széchenyi István Egyetem, 2003.

[TDKdok, 93-03] Tudományos Diákköri dolgozatok az intézményi és az országos TDK-konferenciákon

**Az Informatikai Diákköri Kutatások**
**Szemináriumi Füzetek**
**jelen kötetében szereplő kutatási témákat**
**dr. Raffai Mária**
**konzultálta**

**Vojácsek Pál**

# Multimédiás engine létrehozása sebességoptimalizálási lehetőségek kisszámítógépes környezetben

voji@mailbox.hu

## Bevezetés

A ’90-es években robbanásszerű információtechnológiai fejlődés ment végbe, amely egyrészt lehetővé tette, hogy a felhasználói programok többsége grafikus környezetben fusson, másrészt pedig, hogy az asztali számítógépek képesek legyenek a háromdimenziós grafikák valós idejű megjelenítésére. A hardver jelenleg is folyamatosan fejlődik, egyre újabb és újabb grafikus kártyák és processzorok kerülnek a piacra. A fejlődés egyik fő inspirálója a szórakoztatóipar.

A multimédiás, valamint a háromdimenziós grafikát használó programok azonban egyre nagyobb kapacitásokat, komolyabb erőforrásokat igényelnek. Amíg a kezdeti időkben egy program általában alacsony poligonszámú, 640x480-as felbontású modellekkel dolgozott, addig mára már általánossá vált az 1024x768-as felbontás és az akár többezer poligonból álló modellek használata. A fejlődés kezdeti szintjén alkalmazott DOS-os programok fejlesztése nem volt könnyű feladat, hiszen az egyes hardvereszközök használatát, esetleges speciális funkcióit a programozónak külön-külön kellett lekezelnie. Egy háromdimenziós program DOS-os környezetben való elkészítését az is nehezítette, hogy a programozónak olyan egyszerű problémákkal is foglalkoznia kellett, mint a láthatóság, a textúrázás stb. A grafikus felületű operációs rendszerek megjelenése nagy előrelépés volt, komoly könnyebbséget jelentett a programfejlesztési munka lényeges egyszerűsödése. Amíg azonban az egyre inkább háttérbe szoruló DOS-os programok optimálisan ki tudták használni a számítógép lehetőségeit, addig Windows-os környezetben a programok futását több tényező is lassítja. Gyakori például az engine-ek licenszelése, a nagy szoftverfejlesztő cégek ugyanis szeretnék a programjaikat minél hamarabb piacra dobni, ezért egy adott program grafikus motorját (engine-jét) más hasonló programok készítéséhez is felhasználják.

Az operációs rendszerek és a grafikus felületek fejlődését vizsgálva megállapíthatjuk, hogy még a Microsoft Windows első verziói is alkalmatlanok voltak a nagy teljesítményt igénylő programok futtatására. Mivel ezekben a rendszerekben a multimédiás és a grafikus alkalmazások futását lassította, hogy a rendszer nem adott lehetőséget a hardver közvetlen elérésére, ezért a Microsoft kifejlesztette a DirectX-et. A DirectX egy programozói interfész (API – Application Programming Interface), melynek segítségével a programozó Microsoft Windows környezetben is hozzáfér a multimédiás hardverekhez [Budai, 1999]. Ez azt is jelenti, hogy a videokártyákhoz adott driver-eknek is támogatniuk kell a DirectX-et. A gyorsaság érdekében azonban a DirectX meglehetősen alacsonyszintű hardverhez történő hozzáférést tesz lehetővé, ezért a felhasználónak magának kell megoldania egy sor magasabb szintű funkciót.

Tudományos diákköri munkánk alapvető célja egy olyan engine létrehozása volt, amely eleget tesz a mai elvárásoknak, amely az általános programozási készséggel rendelkező emberek számára könnyen érthető, és amely objektumorientált környezetben egyszerűen programozható. Az általunk kifejlesztett engine használatával hatékonyabbá

tehető komplex alkalmazások fejlesztése, mivel így a megvalósításnál csak az adott problémára kell összpontosítani, és nem kell foglalkozni a háromdimenziós megjelenítés nehézségeivel. A kutatáshoz és a fejlesztéshez a Microsoft Visual C++ fejlesztési környezetet alkalmaztuk, és a DirectX 9 programozói interfészt használtuk fel.

## Kivitelezés, megvalósítás

A grafikus engine alapvető feladata a háromdimenziós alkalmazások megbízható és gyors kezelése. Fontos sajátosság, hogy az engine funkcióit a megfelelő dokumentáció birtokában más programok készítéséhez is fel lehet használni, így ugyanaz az engine más, hasonló jellegű programok írásához is alkalmazható.

Az engine megvalósításának első lépése a DirectX programozási dokumentációjának és az ahhoz adott példaprogramoknak a tanulmányozása volt. Erre azért volt szükség, hogy a DirectX működését olyan szinten lássuk át, hogy a továbbiakban már ne kelljen a példaprogramok forráskód részleteire támaszkodni, és lehetőségünk legyen a működési folyamatok tanulmányozására, az esetleges sebességoptimalizálási lehetőségek helyeinek, az optimalizálás módjának implementálási munkát megelőző meghatározására.

## Az engine szerkezete, a működés folyamata

A háromdimenziós programok magja egy önmagát ismétlő ciklus, ami az aktuális jelenetben szereplő modelleket és az egyéb objektumokat jeleníti meg. A ciklus másodpercenként minimum 25 alkalommal kell, hogy lefusson, mivel ez már elegendő ahhoz, hogy az emberi szem folyamatos és összefüggő mozgóképet érzékeljen [Budai, 1999]. A program inputja lehet billentyűzet, egér vagy más vezérlő eszköz is, mint például a botkormány. A program az input kezelését eseményorientáltan vezérli. Bár az inputok lekérdezése ciklusonként történik, a program mégis csak akkor reagál, ha valami változás áll be a működésben, például egy egérklikk. A ciklus közben a képernyőn mindig az előző képkocka látható, a megjelenítés egy ún. backbuffer-ben történik, ami a ciklus végén a frame buffer-be másolódva jelenik meg a képernyőn.

## Az objektumorientált szemlélet előnye

Az objektumorientált szemlélet a programozók körében mára már széles körben elterjedt, a programozók szívesen vallják magukénak. Az objektumorientáltság növeli a program átláthatóságát, a forráskódok ismételt felhasználásának, valamint a polimorfikus sajátosságnak köszönhetően pedig csökkenti a programozási munkára fordított időt és a program méretét. Szem előtt kell azonban tartani, hogy az objektumorientáltságnak vannak hátrányai is. Ezek közül az egyik, hogy a fordítóprogramok többsége az objektumorientált programkódok forrásfájljaiból csak kevésbé hatékony kódot tud generálni. A futás tovább lassul, ha többszörös öröklődést is meg kell valósítani. Mivel egy grafikus engine-nel szemben a legfőbb elvárás a gyorsaság, ezért, akár néhány objektumorientált elvet is feladva, inkább a futási sebesség növelésére kell törekedni. Ennek érdekében az egyes függvényeket inline módon érdemes deklarálni [Bodor, 2000], az objektumok egyes tagjait pedig a *Set* és *Get* metódusok helyett közvetlen hozzáféréssel kell változtatni.

## Objektumok


Az általunk tervezett főprogram-osztály, az inputok és a hálózat kezeléséért felelős osztályok csak egyszer példányosíthatók. Ez azt jelenti, hogy nincs szükség két billentyűzet-lekérdező osztályra. A fejlesztett alkalmazás többféle objektumot képes megjeleníteni, amelyeknek a kezelése az objektumnak megfelelő osztállyal valósítható meg. A kétdimenziós objektumok (egyes nyomógombok, egérkurzor, ablakok), a háromdimenziós objektumok, valamint a szövegcímkék kezelésére külön osztályt tartottunk szükségesnek létrehozni. A könnyű kezelhetőség érdekében az egyes osztályokban a műveletneveket a lehetőségekhez mérten érdemes egységesítettünk, így minden objektumot a *Move()*-val mozgatunk annak ellenére, hogy egy ablak esetében ez a művelet teljesen más eljárást jelenthet, mint egy háromdimenziós objektumnál (lásd polimorfizmus érvényesülése [Raffai, 2001]).

## Az objektumok kommunikációja

Az objektumok közötti kommunikáció az objektumorientált programozás lényegi eleme. Az engine által használt objektumok legtöbbje nem tudja elérni közvetlenül a DirectX komponenseket, vagy ha igen, akkor sem módosíthatja a főprogram futási paramétereit. Előfordulhat azonban, hogy az egyes objektumok állapotában olyan változás áll be, ami a főprogram módosítását követeli meg. Ilyen esemény lehet például a képfelbontás megváltoztatása. Ezt a műveletet a főprogramnak kell lekezelnie. Mivel azonban a főprogram nem érhető el az objektumok számára, ezért annak állapota sem módosítható közvetlenül. Az egyes osztályok az *ExecuteCommand* művelet segítségével üzennek a főprogramnak. Az *ExecuteCommand* a főprogram felé irányuló kérést egy közös adatbázisba menti, amit mind az objektumok, mind pedig a főprogram el tud érni. A kérések feldolgozása egymásután történik, akkor, amikor a vezérlés visszakerül főprogramhoz. Miután az összes kérés végrehajtódott, az adatbázis tartalma törlésre kerül.

# A sebességoptimalizálás jelentősége

A sebességoptimalizálásra több ok miatt is szükség van. Az első, és talán legfontosabb, hogy a megjelenítő hardver lehetőségei korlátozottak, és bizonyos határ után fizikailag sem képes a nagy, összetett háromdimenziós képek megalkotására. A sebességoptimalizáló eljárások többségének a hiánya az engine teljesítményének a csökkenéséhez vezet, ami azt eredményezi, hogy a kisebb erőforrással rendelkező gépeken a program futása nem lesz folyamatos, és az ilyen hardverrel rendelkező felhasználók elégedetlenek lesznek.

Az engine implementálási munkája során nagy hangsúlyt fektettünk bizonyos kódolási elvekre, amelyek közül legfontosabbnak tartjuk a kód kiváló teljesítményét. Fontos követendő elv volt, hogy az objektumokban szereplő függvények paraméterei ne lokális változókat, hanem pointereket használjanak, mert ezáltal a függvényhíváskor az átadott paramétereknek nem kell egy külön memóriaterületre másolódniuk [Bodor, 2000]. A másik fontos szempont volt, hogy minden olyan beállítás, amely megváltoztatja a hardverelemek beállításait, vagy amely sebességcsökkenéshez vezető számolással jár, nem a meghívás pillanatában történik, hanem azelőtt, amikor annak a hatása ténylegesen megjelenik a képernyőn. Az ilyen jellegű változók beállításánál körültekintően jártunk el, ha ugyanis az előző beállított érték egyezik az újonnan beállítani kívánt értékkel, akkor a beállítás nem történik meg ténylegesen. Erre azért van szükség, mert tényleges változást nem okoz, viszont jelentős erőforrást von el.

## Sebességoptimalizálás 3D-s megjelenítésben

### ViewDistance eljárás

Ez az egyszerű eljárás azon az elven alapszik, hogy nem adjuk át megjelenítésre a Microsoft DirectX Direct3D interfésze részére azokat a háromdimenziós objektumokat, amelyek annyira távol vannak a nézőponttól, hogy már egyáltalán nem láthatóak (kívül esnek a kamera látóterén), esetleg csak nagyon kis pontként jelennek meg a képernyőn. A teljesítménynövelő hatás azonban nagymértékben függ a jelenetben található modellektől, és előfordulhat, hogy az eljárás a megnövekedett számolási műveletek miatt a legkevésbé optimális esetben akár lassíthatja is az alkalmazás futását.

A ViewDistance eljárás során egy másik optimalizáció is végrehajtható, ha a megjelenítésre kerülő, nem áttetsző objektumokat távolságuk szerint növekvő sorba rendezzük, és így jelenítjük meg őket. Ez azért jó, mert a Microsoft DirectX a ZBuffer algoritmussal meg tudja állapítani, hogy egy megjelenítésre kerülő poligon takarásban van-e egy másik objektum által, és ha igen, akkor azzal nem foglalkozik tovább [Budai, 1999]. Ez azonban úgy a leghatékonyabb, hogy a megjelenítést mindig a legközelebbi objektummal kezdjük. Ez a módszer azonban nem használható az áttetszőséggel (alphablend) rendelkező objektumokra, mert ezek megjelenítéséhez a Microsoft Direct3D-nek ismernie kell azt a hátteret, amelyhez képest áttetszőséget tud számolni. Ezért a megoldásban az áttetszőséggel rendelkező objektumokat fordított sorrendben kell megjeleníteni: először mindig a legtávolabbit és utána folyamatosan a többi, a kamerához egyre közelebb lévő objektumot. Ez azonban sajnos nem gyors eljárás. Ha nem kell tartani attól, hogy az egyes objektumok takarják egymást, akkor az áttetsző objektumok megjelenítése alatt a ZBuffer-t ki lehet kapcsolni.

### Cull eljárás

Culling-nak hívják azt a műveletet, amikor egy poligonnak a megjelenítésből történő kiiktatása nem gyakorol hatást az adott nézőpontból megjelenő képre. A Microsoft DirectX rendelkezik egy olyan, ún. backface-culling lehetőséggel, amelynél a poligonok „hátsó” felületét nem adja át megjelenítésre. Ez előnyös sajátosság, hiszen ezekre a poligonokra nem kell komplex megvilágítási algoritmusokat számolni, és így jelentős teljesítménynövekedés érhető el. Az eljárást azonban ki kell kapcsolni, ha olyan poligonokkal dolgozunk, amelyeknek bizonyos esetekben mindkét oldala látható.

A másik eljárás szerint azokat a modelleket is ki kell hagyni a megjelenítésből, melyek kívül esnek a látótérből. A Microsoft DirectX-ben nem implementált eljárás során a vetítési és a nézőponti mátrixból először inverz transzformációval kiszámítjuk a látótér pontjait, majd a síkjait, és megnézzük, hogy az objektum a látótéren belül van-e. A tesztelés gyorsítása érdekében nem azt kell keresni, hogy az objektum minden poligonja a látótérben van-e, hanem célszerűbb az objektum köré írható gömböt, vagy a négyzetalapú hasábot (object bounding box) tesztelni, mivel ezek tartalmazzák az objektum összes poligonját. A Microsoft Direct3D lehetőséget biztosít a bounding box számolására, de mivel ez egy saját algoritmussal is egyszerűen számítható, ezért a bounding box-ot csak akkor érdemes használni, ha az objektum pozíciója, mérete, iránya megváltozik. Az eljárás főleg akkor praktikus, ha nagyon sok objektum van az adott jelenetben, és ha ezek az objektumok sok poligonból épülnek fel.

## Objektumok típus és távolság szerinti sorba rendezése

Az objektumokat típus és távolság szerint kell sorba rendezni, mert ez a megjelenítési eljárás során fontos. Az áttetsző objektumok esetén előbb a hátteret kell előállítani, mert különben nincs mihez képest áttetszővé tenni az objektumot. A csak kétdimenziós képeket, amelyeket a háromdimenziós képek előtt szeretnénk látni, előbb kell megjeleníteni.

Ez az eljárás használja ki a DirectX azon optimalizálási lehetőségét, miszerint, ha valami teljes egészében kitakarásra kerül, az nem kerül bele a megjelenítésbe. Ezt viszont a DirectX csak akkor tudja eldönteni, ha az objektumokat megfelelő sorrendben adjuk át neki megjelenítés céljából.

## Modellszintű optimalizációk

A gyorsabb megjelenítés érdekében a háromdimenziós modellekre is alkalmazhatóak bizonyos optimalizációs eljárások. Célszerű, ha minél kevesebb poligonszámú modelleket használunk. Optimalizáció programszinten is lehetséges, ezek közül egyes eljárásokat már a DirectX is támogat. Az egyik ilyen eljárás alapelve, hogy az egyes modellekből több példányt is eltárol, mindegyiket egyre kisebb poligonszámmal. Ha a modell a nézőponttól már messze van, akkor nem vehető észre rajta kisebb változások, ezért meg lehet jeleníteni a gyengébb minőségű modellt. Nagyon nagy távolságban már egy egészen torz modell is helyettesítheti az eredetit, mert éppen csak azt tudjuk megállapítani, hogy van ott a térben valamilyen alakzat. Ha viszont a modell újra közel kerül a nézőponthoz, akkor folyamatosan vissza kell váltani az eredeti modellre. Ezt az eljárást Progressive Mesh néven a Microsoft DirectX is támogatja. A megoldás előnye, hogy a poligonredukció futásidő alatt végezhető, hátránya viszont, hogy ehhez a művelethez jelenleg nincs hatékony működtetést támogató hardveres megoldás. De a Microsoft DirectX rendelkezik olyan modelloptimalizáló funkcióval is, amely a modellt a hatékonyabb megjelenítés érdekében többféleképpen tudja átalakítani [DirectX, 2002].

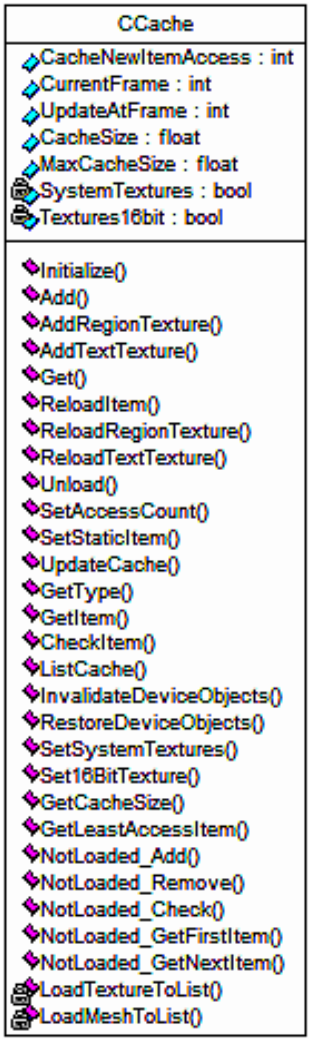
## Programszintű optimalizációk

A program működése során a futást többféle módszerrel lehet gyorsítani. Sebességnövekedést érhetünk el, ha a kétdimenziós objektumokat jelenítjük meg legelőször, mert az ezek mögött levő poligonok nem mennek át a ZBuffer teszten, így nem kerülnek megjelenítésre. Az átlátszósággal rendelkező objektumok megjelenítése a modellekhez hasonlóan a legkésőbb kell, hogy megtörténjen, de ebben az esetben

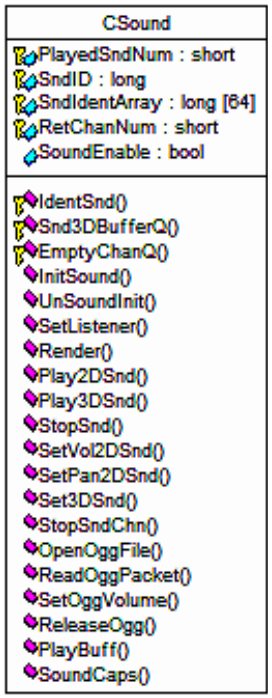
az átlátszó kétdimenziós objektumok előtt meg kell jeleníteni a háromdimenziós objektumokat,. Az újabb Intel-processzorok egy olyan Hyperthreading technológiával rendelkeznek, ami egyfajta, párhuzamos utasítás-végrehajtást is lehetővé tevő virtuális multiprocesszoros üzemmódot biztosít. A Hyperthreading lehetőséget azonban csak akkor tudjuk kihasználni, ha a programunk is több szálat használ egyszerre. A Microsoft Direct3D támogatja a többszálú futásmódot, használata azonban nem ajánlatos, mivel bekapcsolása esetén a teljesítmény jelentősen csökken. Az általunk kifejlesztett program szálait matematikai algoritmusok futtatására, mesterséges intelligencia kezelésére, hálózati kommunikáció lebonyolítására, különféle scriptprogramok értelmezésére, futtatására használhatjuk fel. Figyelni kell azonban arra, hogy a párhuzamosan futó szálak ne érjék el a Microsoft Direct3D eszközt, mert a definiálatlan (undefined) működéshez vezethet. Fontos tudni azt is, hogy a fordítóprogram megfelelő konfigurálásával további optimalizáció érhető el.

A fejlesztés során felhasználtuk a már meglévő, és a Microsoft DirectX által alapból támogatott sebességoptimalizálási eljárásokat is. Ezen eljárások közé sorolhatóak a már említett Progressive Mesh eljárás, valamint a Texture Mip Map eljárás. Ez utóbbi lényege, hogy az objektumra feszítendő képből előállítunk több, adott számú kisebb méretű képet, és ezeket az objektum-nézőponttól lévő távolság függvényében változtatjuk. Bár a messze lévő objektumokra ráfeszített képek így nem olyan részletesek, de a textúrákkal végzendő műveletek (a megjelenítés is ilyen) jelentősen felgyorsulnak.

## A hangrendszer optimalizációs lehetőségei




Mivel a hangrendszer használata is a megjelenítési teljesítmény csökkenéséhez vezethet, ezért ennek optimalizálásával is foglalkoznunk kellett. A hangrendszer megalkotásához a Microsoft által nyújtott DirectSound-interfész legalacsonyabb szintű komponensét választottuk, mert így lehetőség volt saját sebességoptimalizációs eljárások használatára. A CSound-objektum a CCache-objektumon keresztül éri el a hangfájlokat, így azokhoz a lejátszásnál közvetlenül a memóriából férhet hozzá. A hangfájlok esetén a CCache-objektum a betöltésnél elmenti az adott hangfájlra jellemző tulajdonságokat (frekvencia, csatornák száma, stb.), így azt már nem kell lejátszás előtt kiolvasni a fájlból. A hang lejátszására szolgáló csatornákat előre hozzuk létre, de a felhasználás folyamán a jellemzőiket dinamikusan változtatják attól függően, hogy azokon két dimenzióban vagy térben kívánunk hangot lejátszani. Hanglejátszásnál a lejátszott hanghoz egy egyedi azonosító (handle) rendelődik, ami minden lejátszott hang esetében különböző. A hangra történő hivatkozás ezzel a generált számmal történik.




A hangok kezelését maga a CSound-osztály végzi, így az alkalmazásnak nem kell lekérdezést végezni a hang éppen aktuális állapotáról. A CSound-osztály az adott hangra csak az adott lejátszófüggvény által visszaadott számmal hivatkozik, és közli a CSound-objektummal az általa kívánt módosításokat. A CSound-objektum ekkor eldönti, hogy az adott hang lejátszás alatt van-e még, és hogy a módosításokat végre lehet-e rajta hajtani. Lejátszás alatt természetesen lehetőség van a hang pozíciójának a módosítására (például, hogy kétdimenziós esetben a bal vagy jobb hangfalon szóljon, térbeli hanglejátszás esetén megadható egy háromdimenziós pont stb.), a hangok frekvenciájának átállítására, vagy esetleg automatikusan ismételt folyamatos lejátszásra.

A térben elhelyezett hangok paramétereinek tényleges módosítása a képkockák megjelenítéséhez hasonlóan kötegelten történik, így optimalizálni lehetett a háromdimenzióban hallatszó hangok lejátszásának az engine teljesítményére gyakorolt hatását.

## Fájlelérés sebességoptimalizálása

A fájlrendszer elérésének optimalizálása nagyon fontos feladat. Amíg a számítógép a merevlemezen lemezműveleteket végez, addig csökken a teljesítménye, és ez élvezhetetlenné teheti az engine által prezentált jelenetet (a kép akadozhat, a hang lejátszása időben késve érkezik stb.). Ma a számítógépek általában elegendő operatívtár-kapacitással rendelkeznek ahhoz, hogy egy általunk írt program minden szükséges eleme a memóriában legyen, és hogy az igényelt adatokat szükség esetén a memóriából tudjuk elérni, ne kelljen a merevlemezhez fordulni. Nem feltételezhetjük azonban, hogy minden felhasználónak van elegendő memóriakapacitása egy komplex alkalmazás teljes adatigényének a befogadására, így azt, hogy mi, mikor kerüljön be a memóriába, és mikor kerüljön eltávolításra onnan, kezelni kell tudni. Ha a CCache-objektumban tárolt elemek mérete meghaladja a megengedett értéket, vagyis a CCache megtelik, akkor bizonyos fájlok tartalma kikerül a CCache-objektumból. Az ürítés során a LifeTime-technológiát alkalmazzuk, ami abból áll, hogy minden bekerülő fájl kap egy életciklus-számot. Ha az adott erőforrást nem használjuk, akkor csökkentjük ezt a számot, biztosítva ezzel, hogy a legrégebben használt (legszükségtelenebb) memóriaterületet a memóriából felszabadítjuk. Lehetőség van virtuális fájlrendszer használatára is, ami a felhasználók elől elrejti a fájlokat,

# Eredmények, következtetések

Az általunk fejlesztett rendszer törekszik arra, hogy maximálisan kihasználja a hardver-erőforrások által nyújtott lehetőségeket. A megfelelő teljesítmény-kihasználás szükségszerű, mert a ma elterjedt kisszámítógépes rendszerek túlnyomó többsége nem elég fejlett ahhoz, hogy megfelelő szintű optimalizáció nélkül gyorsan lehetővé tegye a bonyolult, összetett háromdimenziós képek megjelenítését. A fejlesztési és implementálási munka alatt szembesültünk az időben párhuzamosan történő programfejlesztés nehézségeivel, és láttuk, hogy azt csak jól megtervezett munkamenettel és az objektumorientált szemlélet követésével tudjuk áthidalni.

A fejlesztési munka során az *iteratív fejlesztési elvet* követtük. Először a Microsoft DirectX által nyújtott példaprogramokat tanulmányoztuk, majd megértettük az álltaluk megvalósított eljárások működését, azok létrehozásának szükségességét és módját. Ezek után a lehetséges sebességoptimalizálási eljárások kidolgozása következett, majd a saját objektumaink megalkotása. Az objektumok definiálásához, az objektumkapcsolatok megvalósításához és optimalizálásához, egyszerűsítéséhez, az engine felépítésének, összetételének átlátásához a Reverse Engineering technológia által nyújtott lehetőségeket használtuk ki. A megvalósítás során az *inkrementális programfejlesztést* alkalmaztuk. Elsődlegesen a keretprogramot, majd néhány alacsony funkciójú objektumot alkottunk meg, később magasabb szintű funkciókat ellátó komplexebb objektumokat hoztunk létre, a régebben implementált objektumokat szükség szerint új, vagy átalakított, javított funkciókkal láttuk el. A program fejlesztése jelenleg is folyamatban van.

A közeli jövőbe mutató célkitűzéseink között az engine fejlesztési munkáinak a befejezése szerepel, majd egy, az engine szolgáltatásaira épülő gyakorlati alkalmazás megfelelő színvonalú kivitelezését és a szabadon elérhető változat terjesztését tervezzük megvalósítani, hiszen az engine segítségével mások által igényelt funkciók implementálásához kívánunk hozzájárulni. Az ehhez szükséges technikai segítségnyújtás és tanácsadás bevezetése is a céljaink között szerepel.

## Hivatkozások

[Budai, 1999] Budai Attila: Számítógépes Grafika – LSI Oktatóközpont
[DirectX, 2002] Microsoft DirectX SDK –http://www.msdn.com/directx/
[Bodor, 2000] Bodor László, Bérci Norbert: C/C++ programozás – LSI
[Raffai, 2001] Raffai Mária: Objektumok az üzleti modellezésben – Az objektumorientált fejlesztés elvei és módszerei – Novadat Kiadó

**Kardos Lajos - Nagy Károly**

# Otthon-irányítás, avagy intelligens házak

karoly119@freemail.hu

## Bevezetés

A mai rohanó világban már lassan teljesen természetes, hogy az emberek a fotelben ülve, a számítógép előtt, kényelmesen intézhetik el különböző ügyeiket. Gondolhatunk itt a banki fizetésen át az on-line vásárlásra, vagy akár olyan hétköznapi dolgokra is, mint a televízió távirányítója vagy a szórakoztató elektronikai berendezések irányítása. Mindezek már teljesen megszokott eszközök, és ugyanúgy részét képezik életünknek, mint akár a napi bevásárlás.

## Otthonunk irányítása Miért ne?

Felmerülhet a kérdés, hogyan tudnánk még inkább megkönnyíteni életünket? Ha távirányítóval vezérelhetünk szinte mindent a házban, akkor vajon miért ne irányíthatnánk úgymond magát a házat is? A ház irányítása, természetesen átvitt értelemben, az otthonunkban található elektronikus berendezések vezérlését jelenti.

Ha a megvalósításon kezdünk el gondolkodni, akkor első lépésként sorra kell vennünk, hogy mit is akarunk pontosan végrehajtani, és ahhoz milyen technikai háttérre lesz szükségünk. A technika mai ro-

hamos fejlődése oda vezetett, hogy az árak folyamatosan csökkennek, a fejlődés új tendenciái szinte beláthatatlanok, és hogy most már szinte minden átlagos család rendelkezik az otthonában számítógéppel..

Manapság azonban nemcsak a számítógépek, de a távvezérelhető eszközök piacán is nagyon nagy a választék, ezért a meglévő technikai háttérre építve, némi programozási és elektronikai ismerettel és rengeteg fantáziával valójában már nincsenek határok! Egy új tudásalapú társadalom bontakozik ki, amelyben az elektronikai eszközök egyre jobban terjednek, használatuk elsajátítása egyre kevesebb erőfeszítést igényel, a zuhanó áraik miatt pedig elvileg bárki számára elérhetőek.

Meggyőződésünk, hogy néhány év múlva olyan alacsony áron lehet majd hozzájutni otthon-irányító rendszerekhez, hogy az a háztartások megszokott, általános része lesz, olyan, mint amilyen manapság például egy automata mosógép. Fejlesztésünk célja, hogy kikísérletezzünk, létrehozzunk és általánosan forgalmazhatóvá tegyünk egy olyan rendszert, amelynek végrehajtásához nem is szükséges, hogy a számítógép csúcskategóriás csodamasina legyen, a korábban beszerzett, jelenleg is használt gépek kiválóan megfelelnek a célnak, legyen az akár csak egy „ősrégi” Pentium I-es gép.

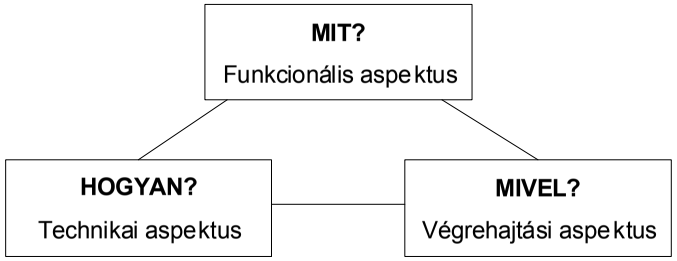
## Fejlesztési aspektusok

Egy rendszer fejlesztése során akkor járunk el helyesen, ha vizsgálódásainkat különböző megközelítésben végezzük [Raffai, 2003/1]. A leggyakoribb vizsgálati aspektusok a funkcionalitás, avagy működési jellemzők, valamint a technikai, vagyis a megvalósítás eszköz vonatkozásai.

A rendszerünk tervezésekor az imént felsorolt három megközelítés alapján indultunk el. Mindenekelőtt kell egy pontos meghatározást kellett tennünk arra nézve, hogy MIT is akarunk készíteni, milyen funkciókat lásson el a fejlesztendő rendszer, és hogyan tudjuk ezt kezelni. Ennek vizsgálatakor nem foglalkozunk a megvalósítás technikai és végrehajtási részével, azt fekete-doboznak tekintjük olyan értelemben, hogy csak azt tudjuk, mik a bemenetek, és mik lesznek a kimenetek.

Ha a MIT? kérdésre pontos választ adtunk, akkor vizsgálhatjuk a következő nézőpontot, hogy mindezt HOGYAN? hajtsuk végre. Meg kell gondolnunk, milyen technikai háttérre lesz szükségünk a feladat megoldásához. A megoldás megválasztásánál nagyon fontos figyelembe vennünk a költségvonzatokat, hiszen ha valóban piacképes terméket szeretnénk készíteni, akkor a végső árat alacsonyan kell tartani. Olyan, lehetőleg olcsó, de a funkcióját mindenképpen ellátni tudó elemekre van szükség, amelyek megbízhatóan tudnak működni, és elfogadható áron szerezhetők be.

Önmagában viszont nem elegendő meghatározni a megoldás módját, hiszen még mindig ott van a MIVEL? kérdésre adandó válasz. Ez azt jelenti, hogy meg kell oldanunk a végrehajtás problémáját, és csak ezek után tudjuk a konkrét munkát elkezdeni. Ezen belül főként a vezérlőprogramokon (portvezérlés stb.) és a vezérlő programok futását biztosító operációs rendszeren van a hangsúly. Fontos, hogy platformfüggetlen megoldást valósítsunk meg, hiszen csak így biztosítható a fejlesztett alkalmazásunk széleskörű elterjedése. Munkánk során meg kell oldanunk a már imént említett portvezérlést, készíteni kell egy olyan felületet, amin keresztül a rendszert irányítani tudjuk, és meg kell oldani a távoli hozzáférés kérdését is. A fejlesztés dimenzióit a 4. ábra szemlélteti.




4. ábra Fejlesztési aspektusok

# A fejlesztendő rendszer kialakítása

## Célmeghatározás

Fejlesztésünk célja az otthonunk elektronikai berendezéseinek a lehető legegyszerűbb interfészekkel történő irányításának a megoldása. A megvalósítás során eltekintünk annak vizsgálatától, hogy a felhasználó milyen rendszeren keresztül juttatja el az információt a kapcsolómodulhoz, ehhez ugyanis olyan, kész megoldásokat alkalmazunk, mint például a WEB, WAP, SMS vagy a hangfelismerő rendszerek, az érintő-képernyő stb. A felsorolt megoldások mindegyike megfelel igényeinknek, nekünk és a felhasználónak tehát csak választani kell közülük. A választás történhet anyagi, kényelmi, megvalósíthatósági stb. szempontok alapján. A házba már beszerelt, működő, „régi” kapcsolókat természetesen nem hagyhatjuk ki a rendszerből, vagyis a hagyományos megoldásoknak is meg kell maradniuk. Ez nem okoz problémát, hiszen lehetséges a megoldások együttes használata is, vagyis a már megszokott rendszer megtartható. Ez gyakorlatilag azt is jelenti, hogy már használt működő elemeket tételezünk fel, és ezekből alkotunk valami újat.

A *házon belüli* otthon-irányítás vezérléséhez célszerű lokális, vagyis teljesen helyi, hálózatot kerülő megoldást választani. Ajánlott és praktikus megoldás a touch-screen, azaz az érintőképernyő használata, vagy esetleg hangfelismerő rendszer alkalmazása. Mindkét megoldás kényelmes, a használata pedig egyszerű. A hang alapján történő vezérlés talán valamivel kényelmesebb, viszont minden felhasználóhoz személyre kell szabni, és jelenleg még nem általánosan elterjedt. Az érintőképernyő ezzel szemben drágább megoldás, ugyanakkor egyszerűbb, és talán látványosabb is. A *távoli, lakáson kívülről* történő irányítás esetén meglévő hálózatokra célszerű támaszkodnunk, és ennek szolgáltatásait igénybe venni. Kézenfekvő az Internet lehetőségeinek a kihasználása, de felhasználhatjuk a WAP, az SMS, a telefonhálózat szolgáltatásait is. A felsorolt megoldások mindegyike képes adatokat továbbítani, és nekünk éppen erre van szükségünk, ugyanis a hálózathoz kapcsolt berendezések aktuális állapotát kell lekérni, illetve annak módosítása esetén az új állapotot kell tudatni a vezérelt berendezéssel. Mivel a célhoz csak kis mennyiségű információt kell eljuttatnunk, ezért a WAP- és SMS-megoldások is teljes mértékben képesek a rendszerigények kiszolgálására.

## Az irányítási funkció kialakítása

Miután kiválasztottuk, hogy mi lesz az a kliens oldali berendezés, amivel irányítani tudjuk a rendszert (ez természetesen több eszköz együttes használatát is jelentheti), meg kell terveznünk annak konkrét kialakítását. Könnyű, gyors irányítást kell biztosítani, valamint felhasználóbarát interfészt kell kialakítani. Gondolni kell olyan megrendelőkre is,

akik kevésbé, vagy egyáltalán nem jártasak az informatikában, mégis szeretnének otthonra ilyen rendszert. Maximális szabadságot kell biztosítani a működtetés, a kezelés tekintetében, de a megoldásnak egyszerűnek kell lenni, nem szabad hagyni, hogy a felhasználó elakadjon a rendszer irányításában. A többletfunkciók elérése közben fontos feladat a biztonság szem előtt tartása is, különösen távoli hozzáférés esetén kell erre nagy hangsúlyt fektetni. Mivel itt alapvetően már meglévő és kipróbált rendszerekre támaszkodunk (dinamikus weboldalak, szerver oldali szkriptek), ezért kis odafigyeléssel nagy biztonságú rendszert tudunk létrehozni.

## Technikai aspektus

Az eszközök kiválasztásakor leginkább az ár/teljesítménymutatót tartottuk szem előtt. Ebben a vonatkozásban fontos döntés volt, hogy az irányítást számítógéppel vagy mikrokontroller alkalmazásával valósítjuk meg. Számítógéppel a feladat egyszerűen elvégezhető. Ezzel a megoldással megbízható, jó rendszert tudunk kiépíteni, de számolni kell azzal a hátránnyal, hogy így egy, a vezérlést végző komplett számítógépet kell szünetmentesen üzemeltetnünk, ami rejtve is maradhat, de egy szekrény alján, vagy akár a padláson is lehet. A mikrokontroller előnye, hogy bár a vezérlés szempontjából tökéletesen ugyanazt a funkciót látja el mint a PC, lényegesen kisebb helyet foglal el.

Mivel a vezérlés és a vezérelt eszközök szempontjából egyaránt széleskörűen, általánosan alkalmazható megoldást szeretnénk kidolgozni, ezért törekednünk kell platformfüggetlen rendszer kidolgozására, többféleképpen is vezérelhető eszköz alkalmazására. Univerzális rendszer kiépítése természetesen nehéz feladat. Elképzelésünk, vagyis otthonunk összes elektronikus berendezésének az irányítása úgy oldható meg a legegyszerűbben, ha a meglévő elektromos hálózatot kihasználva egy nagy kapcsoló szerepét vesszük át. Így a hálózatunkra csatlakozó berendezések valójában „nem is veszik észre”, hogy milyen rendszer vezérli őket, vagyis azt, hogy hagyományos villanykapcsolóval vagy egy otthon-irányító rendszerrel működtetjük-e. Amit észlelnek, az csupán annyi, hogy kapnak-e adott esetben feszültséget vagy sem.

## Megvalósítási aspektus

A vizuális felület kialakításához a Flash Mx szoftvert választottuk [Flash, 2001]. Ez a választás már önmagában is egy lépés a platformfüggetlenség felé, hiszen a flash-alap sok lehetőséget rejt magában, lejátszása pedig a honlapba ágyazva többféle operációs rendszeren is lehetséges. A port vezérlése Visual Basic 6.0 programnyelven készült el [VB, 1999]. Mivel a cél és a feladat elsősorban dinamikus website létrehozása, ezért szükség van egy szerveroldali szkriptnyelvre is. A rendelkezésre állók közül a Perl, a Cold Fusion (CFM), az Active Server Pages (ASP) és a PHP jöhet szóba [Schweiindiman, 2001]. Választásunk a PHP-re esett.

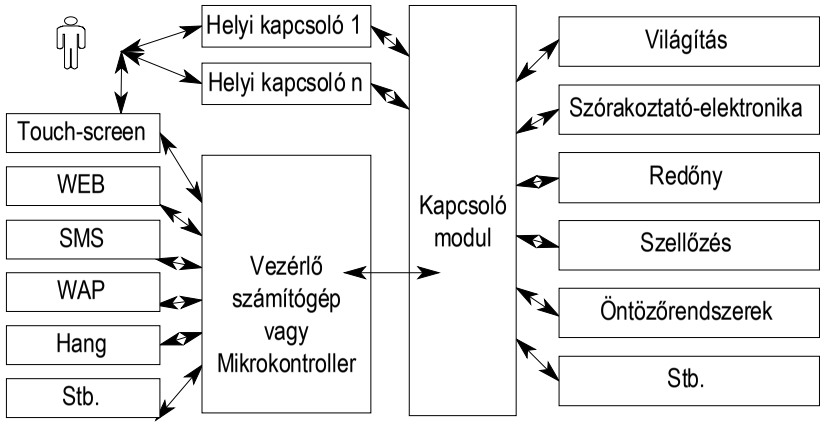
A PHP, amelynek hivatalos neve Hypertext Preprocessor, egy HTML-be ágyazott, C-hez, Perlhez és Javahoz hasonló programozási nyelv, amelyet annak érdekében alkottak, hogy a fejlesztők gyorsan hozhassanak létre dinamikus webalkalmazásokat [László, 2002]. A PHP-prepocesszor a HTML-forrásban a <? php és a ?> címkék közé ágyazott kódot futtatja le, és az utasítások eredményét szöveges formában adja vissza. Szintén fontos megjegyeznünk, hogy a kód a webszerveren, és nem a kliens böngészőjén fut. Ez azt jelenti, hogy a böngésző nem is tud a PHP-alkalmazás tényéről. A kliens böngészője egy HTML-folyamot kap, mintha egy statikus HTML-lapot kapna. Ez kihívást jelent a hagyományos alkalmazások létrehozásához hozzá-

szokott fejlesztők számára, hiszen a hagyományos nyelveknél a megjelenítő kódok, illetve a felhasználói interfész kódjai ugyanazon a gépen futnak, amelyen a logikai kód. A PHP tehát gyakorlatilag egy HTML kódba ágyazható szerveroldali szkriptnyelv. Első nyilvános változata 1995. táján látott napvilágot, de még csak néhány egyszerűbb feladatra volt használható, így számlálót, vendégkönyvet tartalmazott. A programozók szabad és ingyenes részvétele a PHP fejlesztésében nagymértékben hozzájárult a nyelv fejlődéséhez, sokrétűségéhez, és biztosította máig ingyenes elérhetőségét.

## Rendszerünk gyakorlati megvalósítása

Miután problémánkat több nézőpontból is megközelítettük, nézzük meg, hogyan valósítható meg a gyakorlatban. A fejlesztéshez pontos céljaink vannak, tudjuk, hogy mit kell szem előtt tartanunk a konzol tervezésénél és a hardvereszközök kiválasztásakor. Döntöttünk a programozási feladatok ellátásához, a felület készítéséhez, a port vezérléshez szükséges szoftverekről, és van szerveroldali szkriptnyelvünk is.

A rendszer komplett felépítését egy diagramon szemléltetjük (lásd 5. ábra). Az 5. ábra bal oldalán a már korábban említett kliens oldali lehetőségek egy része van felsorolva, ezek az úgynevezett vezérlő számítógéphez vagy mikrokontrollerhez kapcsolódnak attól függően, hogy mivel szeretnénk megvalósítani az otthonunk irányítását. Ezek között a modulok között a kommunikáció kétirányú, hiszen le kell tudni kérni az aktuális adatokat, illetve továbbítani kell az új, kívánt állapotot is. A vezérlő számítógép vagy mikrokontroller egy úgynevezett kapcsolómodullal áll összeköttetésben, ami szintén kétirányú forgalmat biztosít. A kapcsolómodul feladata, hogy az általunk kiadott utasításokat továbbítsa a rá kapcsolt vezérelt berendezéseknek. Ezt a modult természetesen a hagyományos és jól megszokott villanykapcsolókkal is lehet vezérelni. Az ábra jobb oldalán a vezérelhető berendezések sora látható, példaként, a teljesség igénye nélkül.




5. ábra. A rendszer felépítése

A feladat gyakorlati megvalósítása a tervezési munkát követően vált lehetővé. Első lépésként specifikáltuk az alkalmazási környezetet, meghatároztuk a szükséges eszközöket és ezek egymáshoz kapcsolódásának a módját, majd elkészítettük a vezérlő funkciókat ellátó programrendszert.

### Alkalmazási környezet

A programokat IBM PC-n vagy azzal kompatíbilis számítógépen lehet futtatni, gyakorlatilag bármilyen környezetben, legyen az akár Windows vagy Linux operációs rendszer. Az alábbiakban megadjuk a program fejlesztéséhez minimálisan igényelt hardver- és szoftvererőforrásokat.

A rendszer hardverigénye[1]:

– AMD Athlon(tm) XP 1600+ processzor,
– MSI KT4 Ultra (VIA KT400) alaplap,
– 256MB DDR SDRAM (333MHz FSB) memória,
– Maxtor DM+ 60GB ATA100 7200rpm,
– NVIDIA GeForce4 MX440 AGP videokártya,
– LG CD-RW (40x12x40x) CD-író,
– Samsung SyncMaster 765MB típusú monitor,

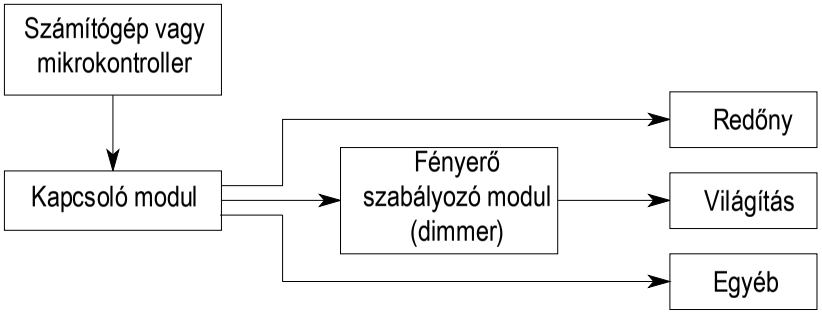
A fejlesztéshez felhasznált szoftverek:

– Microsoft Windows XP magyar nyelvű operációs rendszer
– Microsorft Internet Explorer 6 böngésző
– Apache 2.0.44. verzió számú webszerver
– PHP 4.3.3. fejlesztőeszköz
– Microsoft Visual Basic 6.0
– Flash Mx 2003.

## A konkrét megvalósítás

A feladat konkrét megvalósítását a 6. ábra szemlélteti, amelyből jól látható, hogyan kapcsolódnak össze az egyes részegységek. A számítógépen fut a kapcsolómodult vezérlő program, távoli hozzáférés esetén ezen a gépen egy webszerver működik, helyi elérés esetén pedig a Visual Basic-ben implementált flash plugin. A rendszert a Visual Basic-ben megírt *otthoniranyitas.exe* fájl működteti. A felhasználó a helyi és a távoli elérés esetén egyaránt a Flash-ben írt programot éri el.

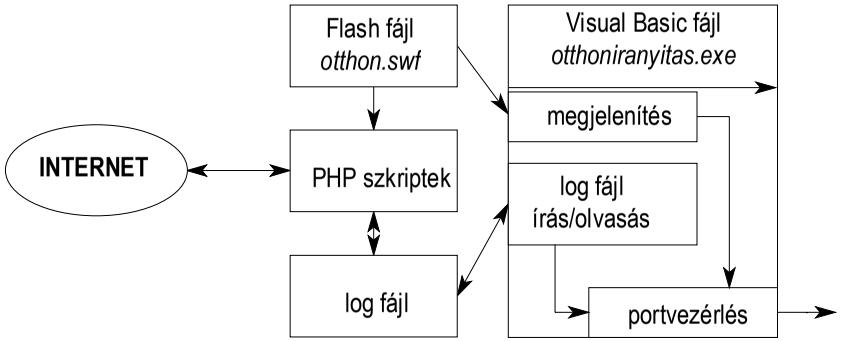
[1] Megjegyezni kívánjuk, hogy a program futtatásához, mint azt a korábbiakban már említettük, egy jóval kisebb kapacitású számítógép is elegendő.




6. ábra Megvalósítási vázlat

## A futó alkalmazások együttműködése

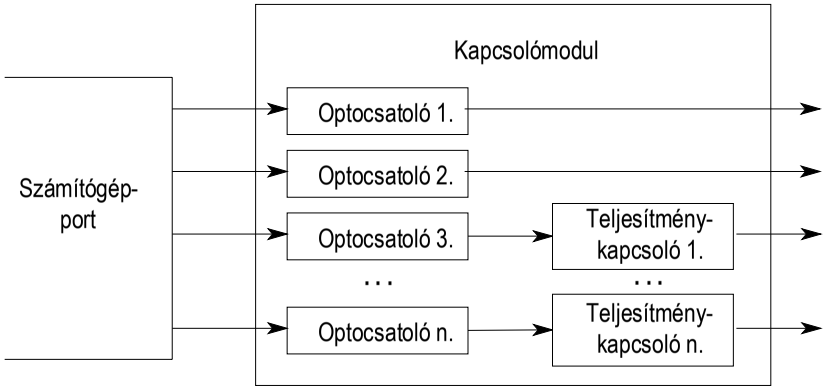
Távoli hozzáférés esetén a felhasználó a böngészőn keresztül először a webszerverhez kapcsolódik, ami lefuttatja a PHP-programot, majd ezen keresztül éri el flash–állományt. Helyi vezérlés esetén a felhasználó ugyanezt az állományt éri el, csak a portvezérlést is végző Visaul Basic-ben készített program segítségével. A könnyebb megértéshez nézzük a 7. ábra diagramját! A bemeneti információt mindkét esetben a felhasználó adja, de ha a bemeneti jelek előre programozottak, akkor az *otthoniranyitas.exe* a beérkező inputok alapján a feszültségszintekkel vezérli a kapcsolómodult.



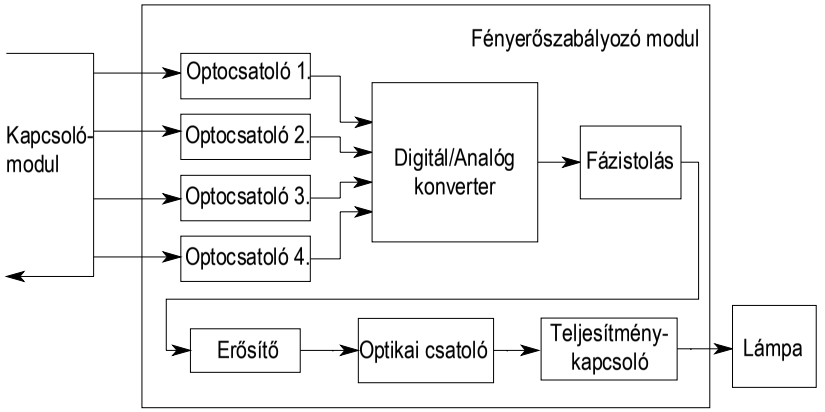

7. ábra A számítógépen futó alkalmazások együttműködése

## A kapcsolómodul felépítése

A kapcsolómodul az inputként kapott feszültségszintek alapján egy optocsatolós leválasztást végez, amely a további fokozatokat működteti. Felépítését az 8. ábra szemlélteti. A modulban nagyteljesítménykapcsoló fokozatok is helyet kaptak [Mocsáry, 1996]. Mivel az optocsatoló elektronikus függetlenséget biztosít a számítógép részére, és bármilyen külső probléma esetén megvédi a portot, ezért szükségesnek tartottuk, hogy beépítsük abba a kapcsolómodulba, amelynek négy kivezetését a fényerőszabályzó (dimmermodul) vezérlésére használjuk fel. A dimmermodul a fázishasítás elvén alapul, azaz a szinusz félhullámok megfelelő fáziskésésű (ß) bekapcsolásával működik [Reincz, 1997]. A szinusz félhullám minél későbbi bekapcsolásától egyre kisebb teljesítmény jut a fogyasztóra, ezáltal kisebb lesz a fényerő. A fázishasítás elvét a 8. ábra, a dimmermodul felépítését pedig a 9. ábra szemlélteti.




8. ábra Kapcsolómodul felépítése




9. ábra A fényerőszabályzó modul felépítése

Az optocsatolók a kapcsolómodultól kapják a vezérlőjelet, ez függetlenséget biztosít, ezért a fényerőszabályzót önmagában is lehet használni, mint modult. Ezután a digitális jelekből a fáziseltoláshoz szükséges analóg jeleket képezünk. Mivel ezek a jelek csak kisebb feszültségen működnek, ezért a fáziseltolás után a jelet felerősítjük, majd egy optikai csatolással a lámpát vezérlő teljesítménykapcsolóhoz vezetjük.

## Üzemeltetés

A konkrét megvalósítás után a rendszer működtethető, a felhasználó által adott jellel vezérelni tudjuk egy lámpa működését. A működtetés menete a következő:

– A felhasználó egy tetszőleges, az Internetre kapcsolódó számítógépről bejelentkezik az otthoni oldalára. A távoli gépről érkező kéréseket a webszerver fogadja, majd a PHP-fordító által generált oldalt elküldi a kliensnek.
– A megjelenő felületen a felhasználó beállítja a vezérelt eszközök kívánt állapotát, majd a kérését visszaküldi a webszervernek.
– A PHP a kéréseket egy logfájlba menti.
– Az *otthoniranyitas.exe* ciklikusan olvassa a log-fájlt, és ha változtatást észlel, akkor annak megfelelően vezérli a számítógép portját.
– A portról érkező jelek eljutnak a kapcsolómodulhoz, amely a további fokozatokat vezérli. Ez lehet egy fényerő-szabályozott lámpa, lehet a redőny, vagy bármi más, amit a felhasználó vezérelni akart.
– A kliens kap egy visszajelzést a változtatásokról.

## Jövőbetekintés

A fejlesztéssel az volt a célunk, hogy megvalósítsunk egy olyan rendszert, amellyel egy otthon berendezései akár automatikusan, akár távolról kézi beavatkozással igény szerint működtethetők, vezérelhetők. A fejlesztést természetesen még nem tekintjük befejezettnek, cikkünkben csak a célokat és a megoldás egy részét tudtuk bemutatni, de elmondhatjuk, hogy a kezdeti elképzeléseink egy részét már ezzel is sikeresen megvalósítottuk, de újabb ötleteinkkel a rendszert tovább tökéletesítjük.

## Hivatkozások

[Flash, 2001] Flash Mx Studio – Pera-Unió, Budapest
[VB, 1999] Programozás Visual Basic 6 nyelven – Kék könyv, Kiskapu
[László, 2002] László József – Dinamikus weboldalak, CGI programozás Windows és Linux rendszereken ComputerBooks
[Schwendiman, 2001] Schwendiman, Blake: PHP 4 Fejlesztők kézikönyve – ComputerBooks
[Raffai, 2003/1] Raffai, Mária: Információrendszerek fejlesztése és menedzselése – Novadat Kiadó
[Reincz, 1997] Reincz, Béla: Megvilágítás-szabályzó – Rádióvilág Kft., Rádió-technika évkönyve
[Mocsáry, 1996] Mocsáry, Gábor: Egyszerű hobbi-áramkörök – Triakos hálózati kapcsoló – InfoGroup Rt., Ezermester hobbi 1996/6.

**Tóth János**

# Adattárolók napjainkban

tjani@freemail.hu

## Bevezetés

Napjainkban többféle számítástechnikai szakkönyv jelenik meg, de az adattárolókról meglehetősen nehéz igazán jó szakirodalmat találni. A legtöbb egy adott eszközt vagy eszközcsaládot tárgyal részletesen, nincs igazán olyan naprakész anyag, amely összegezve értékelné, minősítené ezeket. Célom egy olyan egységes anyag készítése, amelyben összefoglalom a jelenleg használatos adattárolók legfontosabb típusait és azok tulajdonságait.

Munkámban alapvetően azokkal a háttértárolókkal foglalkozom, amelyek a legtöbb PC-t használó számára elérhetőek. A cikk elsősorban alapinformációkat ad közre az eszközök működéséről, főbb jellemzőikről, és egyfajta összehasonlítást tesz a képességeikről és a kapacitásukról.

## A tárolási technológiákról általában

A számítógépes adattárolókat sokféle szempont szerint csoportosíthatjuk, így például a tárolás fizikai megvalósításának a módja, a kapacitás, az írási/olvasási sebesség stb. alapján. Ha a tárolás tartóssága, a megőrzés tápfeszültségtől való függősége szerint csoportosítunk, akkor alapvetően két fő tárolótípust különböztetünk meg: memória és háttértár kategóriákat. Szokták ezeket elsődleges és másodlagos tárolóknak is nevezni. A központi memória a végrehajtás alatt álló programokat és a működés során felhasznált adatokat tárolja. Elérési ideje nagyon rövid, legfeljebb néhányszor tíz nanoszekundum, vagy még annál is kevesebb.

A háttértárak elérési ideje ezzel ellentétben sokkal hosszabb, és függ az olvasni vagy írni kívánt adatok jellegétől, a tárolás szervezési módjától, helyétől. A háttértárak viszont nagyobb kapacitásúak, és mivel a tápfeszültség folyamatos biztosítása nélkül, hosszú időn át képesek megőrizni a rájuk felvitt adatokat, ezért kiválóan alkalmasak adatok, programok tartós megőrzésére, tárolására. Nagy előnyük, hogy cserélhetőek és hordozhatóak, ezáltal az adatok továbbítására is alkalmasak. A PC-s környezetben leggyakrabban használt adattárolók a mágnesszalagok, a mágneslemezek és az optikai lemezek.

Az írási/olvasási technológia szempontjából az adattárolók alapvetően két nagy csoportra bonthatók: mágneses és optikai elven működőkre. A többi, széleskörűen elterjedt technológia alapvetően ezeknek az elveknek a továbbfejlesztése, illetve módosítása. Néhány egyéb, jelenleg fejlesztés alatt álló technika is létezik, amelyek eltérő eljárást használnak.

A technológiák működési alapjait a bináris számrendszer szolgáltatja. A digitális információ kódolási alapegysége a *bit,* amely egy 0 vagy egy 1 lehet. Alapvetően mindegyik adattároló eszköz tárolási technológiája ezen kettős állapot megvalósítására épül, valaminek a meglétét vagy hiányát valósítja meg, és ebből következtet a tárolt adatokra, így például mágnesesség van/nincs, optikai elváltozás van/nincs stb.

## Mágneses elvű adattárolók

A mágneses adattárolás technológiája a mágnesezhető felületek, mágnesezett, illetve nem mágnesezett állapotának elvén alapul. A mágneses adathordozókra az adatfelvitel a felület mágnesezésével történik, oly módon, hogy ’1’ tárolása esetén a felületet mágnesezik, ’0’ esetén pedig nem mágnesezett a felület. Több különböző mágneses tárolófelületet fejlesztettek ki, amelyek hordozófelületben, kapacitásban, sebességben és hozzáférési módban is eltérhetnek egymástól [Raffai, 2003/2]. A mágneses adattárolók két csoportra bonthatóak, a rendszerben állandóan jelen lévő és a cserélhető adathordozókra. Állandóak például a merevlemezek, cserélhetőek a szalagos adathordozók vagy a floppy-k.

### Mágneslemezek

Egy mágneslemez-egység egy vagy több, mágnesezhető bevonattal ellátott alumínium korongból áll (lásd 10. ábra) Eredetileg ezek a korongok mintegy 50 cm átmérőjűek voltak, manapság azonban 3 és 12 cm közöttieket is használunk, a noteszgépekhez, digitális multimédiás eszközökhöz alkalmazott lemezek pedig már 3 cm alatt vannak, és további méretcsökkenéssel kell számolnunk. Az írás/olvasási műveletek végrehajtási technológiája szerint a lemezek felett egy indukciós tekercset tartalmazó fej lebeg, amit a lemeztől csak egy vékony levegőréteg tart távol (kivéve a hajlékonylemezeket, ahol a fej hozzáér a felülethez). Ha pozitív vagy negatív áram folyik az indukciós tekercsben, akkor a fej alatt a lemez felmágneseződik, az áram polaritásától függően a mágneses részecskék balra vagy jobbra állnak be. Amikor a fej egy felmágnesezett terület felett halad át, akkor pozitív vagy negatív áram indukálódik benne, így a korábban eltárolt biteket vissza lehet olvasni. Tehát ahogy a korong forog a fej alatt, bitsorozatokat lehet felírni, illetve azokat vissza lehet olvasni.

10. ábra A mágneslemez mechanikai része

Egy teljes körülfordulás alatt felírt bitsorozat a *sáv*. Minden sáv rögzített méretű, *szektorokra* van osztva, amit a fej írási és olvasási műveleteinek a szinkronizációjához szükséges *fejléc* előz meg. Az adatok után hibajavító kód található, az egyes szektorok között pedig keskeny *szektorrés* (gap) van [Tannenbaum, 2001].

A lemezegység *teljesítménye* sok tényezőtől függ. Egy szektor beolvasásához vagy kiírásához a fejet először a megfelelő sugárirányú pozícióba kell állítani. Ezt a műveletet keresésnek (seek) hívják. Az átlagos keresési idők 5 és 15 ms között vannak, de ez a paraméter erősen függ a sávok közötti távolságtól is. A fej sugárirányú pozícióra történő beállását követően egy kis szünetre van szükség, amíg a keresett szektor befordul a fej alá. A legtöbb lemez 3 600, 5 400 vagy 7 200 fordulatot tesz meg percenként, de vannak 10 000, 15 000 fordu-

lat/perc sebességű lemezek is. A nagy fordulat miatt persze nagyobb melegedés is tapasztalható, ami károsan hat a lemezfelületre (például kitágul), és csökkenti az írási/olvasási művelet hatékonyságát.

Minden lemezhez tartozik egy lemezvezérlő, amely a meghajtót vezérli. Ennek a feladata az írási, az olvasási és a formázási feladatok végrehajtása. A vezérlési technológia fejlődése meglehetősen gyors volt, a külön kártyán elhelyezett vezérlőktől kiindulva ma már meghajtóba integrált vezérlőkkel dolgozhatunk.

A ’80-as évek közepén fejlesztették ki az IDE-típusú (Integrated Drive Electronics) beépített eszközelektronika meghajtókat, amelyeket a más címzési módot (LBA: Logical Block Addressing, logikai blokkcímzés) támogató EIDE-meghajtók követtek (Extended IDE, kiterjesztett beépített eszközelektronika). Az EIDE előnye, hogy lehetővé tette négy meghajtó egyidejű rendszerhez csatlakoztatását, és lényegesen nagyobb adatátviteli sebességet biztosít. A jelenlegi IDE vezérlésű merevlemezek kapacitása rohamos léptekben nő, forgalomban leginkább 60-200 Gbájt kapacitású, 7 200 fordulat/perc sebességű,. 2-8 Mbájt pufferrel kiegészített lemezek kaphatók

## Microdrive

A mikrodrive-ok szerkezete és működése lényegében azonos a PC-k winchesterével, de méretük sokkal kisebb. Az adatátviteli sebesség valamivel alacsonyabb (2,6-4,2 Mbyte/s), mint a memóriachipes kártyáké, motorja és mozgó alkatrészei miatt pedig több energiát fogyaszt, jobban melegszik, mint amazok. Elterjedten alkalmazzák digitális fényképezőgépek tárolóközegeként. A mikrodrive-ok fotóját a 11. ábra tartalmazza.


11. ábra IBM-mikrodrive-ok

Jelenleg a memóriakártyák egyre növekvő kapacitása miatt a nehézkes Microdrive-ok jelentősége csökken, de a nagyobb (6 GB-os) változatok megjelenésével a helyzet várhatóan ismét megváltozik. A jelenlegi Microdirve-ok FAT fájlrendszert használnak. Míg a korábbi Windows-ok a formázáskor megtartották a lemez eredeti formátumát, a Windows XP alapértelmezésben FAT32-re formázza őket. Ez a digitális fényképezőgépeknél történő alkalmazás esetén jelentős hátrány, mert ilyenkor a lemezt FAT-fájlrendszerűre újra kell formázni. A Microdrive-ok jellemző kapacitása 340 MB – 4 GB [IBM, 2003].

## A mágnesszalagos adathordozók

A japán cégek régóta híresek az általuk gyártott mágneses, magnetooptikai és optikai adathordozók kiváló minőségéről, mindig figyeltek arra, hogy gyáraikból csak kifogástalan minőségű termékek kerüljenek piacra, különösen, ha olyan fontos csoportról van szó, mint az adatok hosszútávú tárolására alkalmas adathordozók. A mágnesszalagos adathordozók előnye elsősorban a nagy kapacitás, amelyre a többezer rekordot tartalmazó, többszáz gigabájtnyi adatállományok biztonsági mentéseinél és archiválásánál van szükség. A különböző

szabványok és technológiák lehetővé teszik, hogy mindenki az igényeinek megfelelő sebességű és tárolókapacitású szalagos archiváló rendszert válasszon. Bár a mágneses részecskék a szalag fontos részeit képezik, a szalag kötőanyaga, alapfilmje és hátoldali bevonata szintén meghatározó szerepet játszik a szalag rögzítési és tartóssági jellemzőinél. Néhány figyelemre méltó sajátosság:

– *Kiváló hőállóság:* A hő kihat a fémrészecskék mágneses tulajdonságaira. A mágnesszalagoknál ultravékony kerámiabevonat védi a fémrészecskéket a nem kívánatos hő kihatásaitól, így a részecskék a kiváló mágneses tulajdonságaikat szinte minden környezetben képesek megtartani.
– *Kiváló antioxidációs jellemzők:* Amennyiben valamely fémrészecske felszíne oxidálódik, annak mágneses energiája lecsökken. Stabil keramikus bevonattal ellátva azonban a részecskék védelmet kapnak az oxidációval szemben, és így azok megtarthatják nagy mágneses energiájukat.
– *Kiváló tartósság:* A szalagok tekercselése, végigkeresése nagyon nagy sebesség mellett történik. A masszív kerámiabevonat (Ceramic Armour) biztonsággal védi meg a fémrészecskéket attól a károsodástól, amit a gyorstekercselés okozhat, és így jelentős mértékben hozzájárul a tartósság fokozásához [Maxell, 2003].

# Optikai adattárolók

Az optikai rendszerekben az írás és az olvasás optikai eljárással (fényvisszaverődés, polarizáció, fényszórás, fénytörés), lézersugár alkalmazásával történik Az optikai elven működő tárolók analóg és digitális jelek rögzítésére is alkalmasak. A digitális rögzítés, elsősorban az audió- és a számítógépes technológiák fejlődésének köszönhetően sokkal szélesebb körben elterjedt.

## A CD mint adathordozó

A CD (Compact Disc) anyaga polikarbonát, amelynek felületére minden gyártó saját fejlesztésű festékvegyületet visz fel, majd ennek tetejét, a jobb fényvisszaverő tulajdonság elérése érdekében vékony alumíniumréteggel (vagy aranyréteggel) vonja be. A tükröző felületet lakkréteg védi a sérüléstől, melyet néhány további védőréteg borít.

A lemez közepét és szélét információtárolásra nem használják, a lemezből csak az 50-116 mm átmérő közötti tartományra kerül adat. Az információt spirális felületen rögzítik, a spirálvonal a lemez közepén kezdődik, és a széle felé halad. A spirálok közti távolság 1,6 μm. A lemez felülete három tartományra osztódik. A bevezetőrész (lead-in) 46 mm-től 50 mm-ig tart, itt csak alcsatorna-információkat tárolnak, vagyis a tartalomjegyzék-táblát (TOC: Table of Content). A programtartomány a lemez 50-116 mm-es átmérői között helyezkedik el, a kivezetőtartományban (lead-out, 116-117 mm-es átmérők között) szintén nincs adat. A külső átmérő pontos mérete a felvett anyag hosszától függ.

A fényvisszaverő felület spirális sávján mikroszkopikus lyukak (pitek) és ép felületek (landek) találhatók. A bináris információ a lyukak hosszában és a lyukak közötti szünetek hosszában van kódolva. A lyuk szélessége 0,6 μm, mélysége 0,12 μm.

### CD-írási technológiák

Az egyszer írható optikai lemezek többféle technológiával készíthetők. A leggyakrabban alkalmazott megoldások az alábbiak:

– *Lyuktechnológia:* a tükröző rétegbe kb. 10 mW teljesítményű lézersugárral lyukat égetnek.

– *Buboréktechnológia*: a felvételréteg az írólézer hatására elpárolog, és a tükröző műanyag rétegben kis buborékot hoz létre, amelyről az olvasólézer fénye nem a fejbe verődik vissza.

– *Mintázatváltás-technológia:* a hordozóra felvitt fémtükör felülete szórt fényt ver vissza. Ha a fémréteget lézersugárral felmelegítjük, akkor a felület kisimul, és jó tükör lesz belőle.

– *Festékpolimer-technológia:* a felvételréteg olyan fény- és hőérzékeny szerves vegyületet tartalmaz, amely az író lézersugár hullámhosszán elnyelő képességgel rendelkezik. Az elnyelt lézersugár energiájának hatására a felvételréteg körülbelül 250 °C-ra felmelegszik, megolvad, és nyomást fejt ki a felette lévő tükröző rétegre. A tükröző réteg a nyomástól eltorzul, és rosszabbul veri vissza az olvasólézer fényét, mint az ép felület. Ez a technológia a legelterjedtebb, továbbfejlesztett változatát a CD-RW (ReWriteable: újraírható) lemezek írásánál használják.

– *Fázisváltozós technológia:* az adatok tárolásához néhány fém kristályos (jól tükröző) és amorf állapota közötti különbséget használja fel. Az állapotváltozást az írólézer teljesítményével érjük el. Ezt a technikát szintén az újraírható lemezeknél használják [Ila, 2001].

**CD-R**

Az egyszer írható lemezt 12 vagy 8 cm átmérővel gyártják. Az üres lemez gyártásakor előre elkészítik a sávok nyomvonalát, amelyek üresek, a felhasználó tölti meg tartalommal. A tartalomjegyzék-tábla (TOC) az adatok felírásával együtt kerül fel a lemezre. A bevezetőszektor 9 Mb-ot vagy 1 percet, a kivezetés pedig 13 Mb-ot vagyis 90 másodpercet foglal el a lemezterületből. A programterületen legfeljebb 99 sáv tárolható, amelyek között 2 másodperc szünet van. A CD-R (wRiteable) írók képesek arra, hogy a korábban felírt szekciók után a lemezre új szekciót írjanak, a szekciók között pedig megteremtsék a kapcsolatot, csatolást hozzanak létre. Az ilyen többszekciós lemezeket hibridlemezeknek is nevezik. Minden alkalommal, amikor új szekciót írunk fel a lemezre, elveszítünk 13,5 Mbájt-nyi lemezterületet, ugyanis ennyit foglal el a sáv bevezetése és kivezetése. A lemez korábban már megírt részei nem írhatók újra, ha hibás rész van, akkor azt nem lehet javítani.

**CD-RW**

A CD-RW előírásokat a Philips és további kilenc vezető cég hozta létre. A lemez többször törölhető és újraírható, korábbi neve CD-E volt, ami a törölhető (Erasable) szóból származik. A CD-R és CD-RW közötti egyik lényeges különbség, hogy a CD-R lemezek a lézersugár mintegy 70%-át tükrözik vissza az olvasófejbe, míg a CD-RW lemezről mindössze csak kb. 20% kerül vissza. Ezért az olvasókba, amelyek a CD-RW-t is olvasni tudják egy automatikus, az olvasó érzékenységét szabályozó erősítésszabályozót építenek be, amelyet a gyártók AGC-technológiának (Auto Gain Control) neveznek.

**CD-MO**

Az MO (Magneto Optical) magneto-optikai lemez írható, törölhető és újraírható. A CD-MO felvételi technológia a mágneses jelrögzítés írási és törlési előnyeit egyesíti a lézeroptika nagy írássűrűségével.

A hagyományos optikai tárolók a lemezen lévő adatok olvasásához csak a lézerfény visszaverődési és interferencia tulajdonságait hasznosítják, míg a magneto-optikai lemezek készítésénél felhasználják a fény térbeli mágneses viselkedését is. A CD-MO adathordozó többrétegű, szendvics szerkezetű. A lemez vastagságát a polikarbonát-hor-

dozó határozza meg. A mágneses tárolórétegeket szigetelő (dielektrikum) fogja közre, melynek feladata részben a lézersugár kettős törésének a kompenzálása, részben pedig az oxidációra hajlamos mágnesréteg megvédése. A felső szigetelőt tükröződő arany- vagy alumíniumréteg fedi, ez veri vissza olvasáskor a lézersugarat.

## DVD

A mozgóképes alkalmazásokhoz, a mozifilmek tárolásához a CD kapacitása is kicsi, ezért 1994-ben a videoipar a VHS-formátumnál jobb minőségű és hosszabb játékidejű eszközt keresett. A fejlesztés több ágon indult el (MMCD - MultiMedia CD, hdCD, SDCD), míg végül a fejlesztő cégek 1995-ben konzorciumot hoztak létre, és megállapodtak a műszaki paraméterekben, valamint a tároló nevében. Ez lett a DVD. Bár a DVD nem rövidítés, hanem fantázianév, mégis két jelentést tulajdonítottak neki. Kezdetben a Digital Video Disc (digitális videolemez), később pedig a Digital Versatile Disc (sokoldalú digitális lemez) értelmezés terjedt el.

A DVD lemez kapacitásának nagymértékű növekedése a hagyományos CD több műszaki jellemzőjének a megváltozásából ered. A lemezek közötti alapvető fizikai különbség, hogy a DVD-lemez mindig két 0,6 mm vastagságú lemez összeragasztásával készül, és mindkét oldalon képes adatokat tárolni. A technológiai fejlődésnek köszönhetően ezenkívül a DVD-lemez mindkét oldalán két felvételi réteg alakítható ki. A két oldal és a két réteg bevezetése mellett csökkent a lyukak mérete, megnőtt a spirális sávok sűrűsége és a hasznos lemezfelület. A lézer finomabban fókuszálható, a lemez pedig merevebb. Két réteg, két oldal, 17 Gbájt kapacitás, jelenleg ez a csúcs.

A DVD-RAM lemezek (DVD-RW) használatánál, a CD-RW lemezekhez hasonlóan, fázisváltós technológiát használnak, ahol az aktív réteg az eltérő teljesítményű lézerek hatására a kristályos és amorf állapotok között váltakozik. A DVD-RAM-on kívül léteznek egyéb, újraírható DVD-technológiák is, mint például a DWD+RW, DVD-RW.

# A holografikus adattárolók jellemzői

A jövő útja a holografikus adattárolás. A Bell Laboratoriumhoz tartozó InPhase Technologies már évek óta fáradozik holografikus adattárolótechnológiák kifejlesztésén. Olyan videó-felvevőt készítettek, amely az adatokat három dimenzióban, holografikus eljárással tárolja.

A Tapestry-technológiát használó, egyszer írható, CD-méretű lemezekre 1,3 Mbyte-os hologramcsoportokban akár 100 Gbyte-nyi adat is elfér. Ez kézzelfoghatóbban azt jelenti, hogy 21 DVD-film, vagy félórányi nagyfelbontású, tömörítés nélküli film tárolható egyetlen ilyen lemezen. A főként professzionális, videópiacra szánt készülékek lemezeinek élettartamát több mint 30 évre becsülik, de a bonyolult, többszintű adattárolás miatt jelenleg nem lehet őket másolni. Az InPhase-nek azonban teljesen kész az új holografikus tárolást alkalmazó készüléke is, és tervezik a technológia kisebb, hordozható eszközökben való felhasználását is.

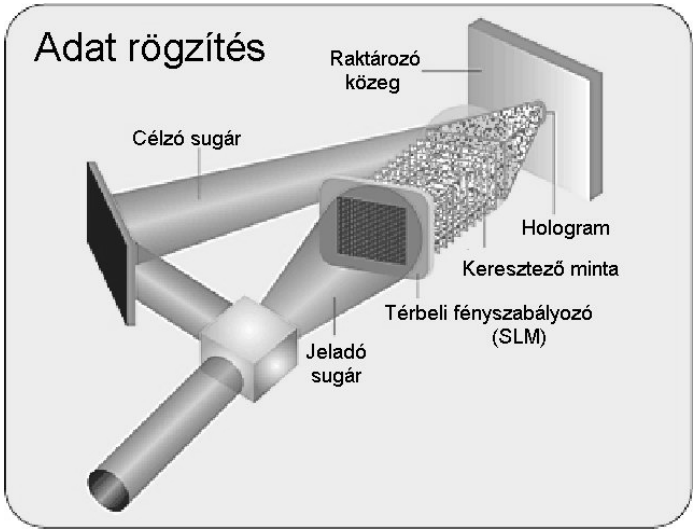
A holografikus adattárolás területén az egyik legnagyobb kihívást az jelenti, hogy számos követelményt kell kielégíteni, így mindenekelőtt megbízható tárolóanyagot kellett kifejleszteni, hiszen az anyagok viselkedését a magas fényérzékenység, a méretállandóság, az optikai tisztaság és az egyenletesség, a nyomnélküli kiolvasás, a millimétervastagság és persze a hőmérsékleti változások is jelentősen befolyásolják.

Az InPhase a Tapestry-anyagot két kémiai fotóműanyag egyesítésével egy magas fényérzékenységű, millimétervastagságú, optikailag sík formájú anyagként tervezte meg (lásd 12. ábra). Ez az anyag bizonyítottan a legjobb a holografikus anyagok között, és jelenleg egyike azon kevés tárolóknak, amelyek a holografikus adattárolókhoz tartoznak.

12. ábra Holografikus tárolóeszközök

## Az adatok rögzítésének a módja

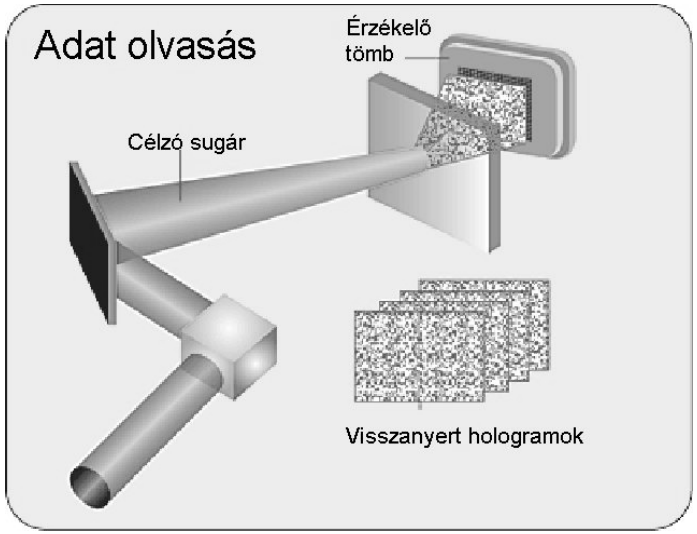
A holografikus adattárolásban egy összefüggő lézer fénynyalábot kétféle sugárrá bontunk: adatokat hordozó jelsugárrá és célzó sugárrá. A digitális anyag, amikor a jelsugarak áthaladnak a térbeli fényszabályozón (SLM Modulátoron), átkódolódik. Az adatok vagy bitsorok először oldalakba vagy nagy tömbökbe rendeződnek. A 0-ákat és az 1-eseket a térbeli fénymodulátor az oldalakról pixelekre fordítja, amelyek tömbökké, vagy további sugarakká alakulnak. A jelsugár fénye, mivel áthalad a térbeli fénymodulátoron, már kódolt állapotban kelül továbbításra. Ez a kódolt sugár találkozik a célzó sugárral, majd behatol a fényérzékeny közegbe, és így tárolja a digitális adatokat. (lásd 13. ábra).




13. ábra Az adatok rögzítése

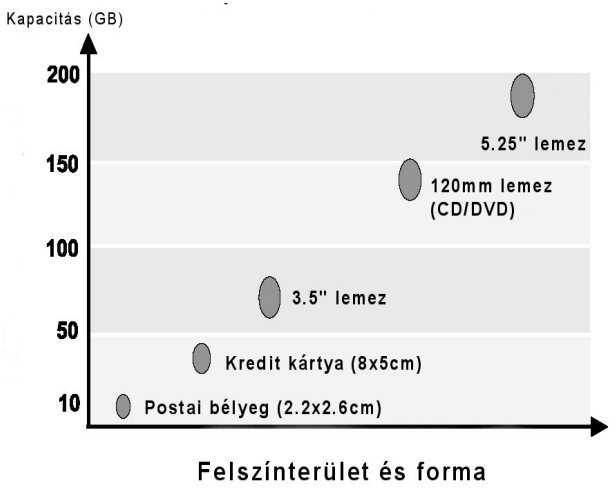
## Az adatok kiolvasása

A célzósugár a felvett rácsok kibontásával, felépíti a tárolt tömb bitjeit, így kiolvasásra is használható. Az újrakészített tömb egy, az adatokat egyidejűleg kiolvasó pixeles érzékelőre vetítődik ki. A holografikus adatok párhuzamos kiolvasása biztosítja a gyors adatátvitelt (10-100 MByte/sec). Az adatok kiolvasása érzékenyen függ a célzósugár jellemzőitől. A célzósugár változhat, így más lehet a beesési szöge vagy a hullámhossza. Egy adott tárolóanyagra ugyanazzal a rögzítéskor használt sugárral sok különböző adatcsomag vehető fel és olvasható (lásd 14. ábra).




14. ábra Az adatok olvasása

A holografikus tárolási folyamat hatalmas adatraktár-kapacitást tesz lehetővé [Inphase, 2002], melynek összefüggéseit a 15. ábra szemlélteti. Az ábrából jól látható, hogy a holografikus tárolásnál kis felületen nagy adatsűrűség biztosítható.

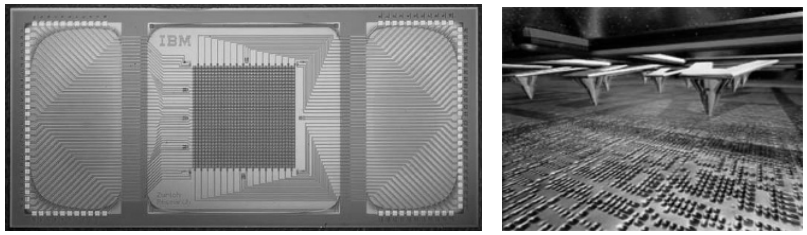



15. ábra A kapacitás- és a felszínnövekedés összefüggése

## Az IBM nanotechnológiás tárolója

Az IBM kutatói 2002 júniusában egy olyan vadonatúj adattárolási technológiáról számoltak be, amelynek a segítségével egy mindössze bélyeg méretű parányi eszközben akár egy billió bit, vagyis mintegy 25 millió nyomtatott oldalon elférő adat is tárolható. A cég elkészítette a vadonatúj adattároló eszköz kísérleti prototípusát, mely több mint ezer felmelegített, rendkívül apró tűt tartalmaz, amelyek segítségével egy

vékony polimer filmrétegben 10 nm-es bemélyedéseket lehet létrehozni, illetve a mélyedések révén adatokat rögzíteni, illetve kiolvasni (lásd 16. ábra).




16. ábra A Millipede adattároló rendszer

A fejlesztők elmondása szerint ez azt jelenti, hogy ha a következő generációs mobiltelefonokat egy-egy postai bélyeg méretű, vagy annál kisebb IBM-típusú adattároló-chippel látják el, akkor akár 10 GB-nyi adat tárolására is képesek lehetnek. Az persze már más kérdés, hogy egy mobiltelefon esetében vajon mit lehet majd kezdeni ekkora adattároló-kapacitással.

A Millipede névre keresztelt adattároló-chip végleges, kereskedelmi forgalomban kapható változata várhatóan nem kerül majd csillagászati összegekbe, sőt az ára valószínűleg rendkívül kedvező lesz. A speciális chip gyártásához ugyanis lényegében nincs szükség semmilyen különleges technológiára, vagyis a jelenleg széles körben használatos technológiák tökéletesen meg fognak felelni erre a célra. Az IBM kutatói elmondták, hogy a Millipede fejlesztési munkálatai rendkívül jó ütemben haladnak, vagyis, ha minden jól megy, akkor körülbelül egy év múlva bemutatják az eszköz mintegy négyezer apró tűt tartalmazó új változatát, és 2005 körül már a kereskedelmi forgalomban is feltűnnek az első Millipede-chipek [IBM, 2002].

# Tapasztalatok, következtetések

## Optikai versus mágneses tárolási technológiák

Az optikai tárolók alapvetően négy olyan tulajdonsággal rendelkeznek, melyek a mágneses technológia fölé helyezik őket:

1. Az optikai tárolók hihetetlenül *nagy tárolási sűrűségének* az alapja, hogy a fény sokkal kisebb felületre fókuszálható, mint a mágneses tárolók elemi tárolófelülete. A jelenleg elérhető sűrűség kb. tízszerese a merevlemezének, a várható fejlesztések pedig 100-szoros sűrűséget prognosztizálnak (lásd a holografikus adattárolóknál írtak).
2. A másik előnyös tulajdonság a *hosszú élettartam*. Az optikai tárolók élettartamát évtizedekben mérik, a gyorsított öregedésvizsgálatok alapján a CD-k élettartama 10-100 év közé tehető.
3. A harmadik az *adathordozó ára.* Bár a merevlemezek kapacitás/ár aránya kezd közelíteni az optikai lemezekéhez, a hajlékonylemezekhez hasonlítva az optikai lemezek ára a kapacitást tekintve mindenképpen kedvezőbb.
4. A negyedik az optikai adathordozó *cserélhetősége, szállíthatósága.* A cserélhetőség nagyobb adatbiztonságot jelent, hiszen a használaton kívüli lemezt a rendszertől akár távol, biztonságos, zárt helyen tárolhatjuk. A merevlemezhez képest az optikai lemezek a külső mechanikai hatásokra kevésbé érzékenyek, de vegyi anyagokkal és fizikai behatással mégis tönkretehetők.

A számos előnyös tulajdonságot látva felmerül a kérdés, vajon miért nem szorítja ki az optikai adattároló a mágnesest? Ennek több oka is van, amelyek közül két fontos tényezőt említenék. Az egyik az adatok

elérési sebessége: még a leggyorsabb CD-k átviteli sebessége is elmarad a több fejjel olvasó, gyorsabban forgó merevlemezétől. A másik, hogy egy mágneslemezcsomag esetében még jelenleg is nagyobb kapacitással lehet számolni, mint egy optikai lemeznél.

### Téves adatok, hiányos szakirodalom

Kutatásom alatt sok mindent tapasztaltam. Az egyik, hogy az általam választott témáról nagyon kevés magyar nyelvű irodalom áll rendelkezésre. A legtöbb információt az eszközök gyártóitól lehet beszerezni, általában idegen nyelven, de ezekkel vigyázni kell, mivel többnyire reklám célból készülnek, és így nem teljesen korrektek. Az egyes technikákról/technológiákról ugyan jelennek meg cikkek, de mivel ezek forrása bizonytalan, hitelessége kérdéses, tartalmuk pontatlan, ezért meglehetősen nehéz volt összeszedni azokat a korrekt információkat, amelyeket a téma feldolgozásához használni tudtam.

## Hivatkozások

[Tannenbaum, 2001] Andrew S. Tannenbaum – Számítógép-Architektúák – Panem Könyvkiadó Kft
[IBM, 2003] http://www.storage.ibm.com/hdd/micro
[Maxell, 2003] http://www.maxell.com (Hitachi Maxell, Ltd , Japan)
[Raffai, 2003/2] Raffai, Mária: Adatbázis-tervezés – Novadat Kiadó
[Ila, 2001] Ila László – CD-DVD - Panem Könyvkiadó Kft.
[Inphase, 2002] http://www.inphase-technologies.com – Inphase T.
[IBM, 2002] http://www-3.ibm.com/chips

**Nagy Boldizsár - Pánczél Zoltán**

# A Linux, mint szoftveralternatíva kisvállalati szegmensben –hálózati biztonságtechnika bemutatása–

boldi01@freemail.hu

## Célkitűzés

A 2003/2004 tanévben indított tudományos diákköri kutatásunk alapvető célja, hogy átfogó képet nyújtson a vállalkozói szféra leginkább költségérzékeny szegmensébe tartozó kisvállalatai számára, és hogy megismertetessen egy költségtakarékos, ám mégis rendkívül hatékony szoftveralternatívával, a Linuxszal. Munkánkban bemutatjuk a Magyarországon elérhető Linux-disztribúciókat, illetve az egyes alternatívákat, amelyek bevezetése nagyságrendekkel csökkentheti az informatikai kiadásokat. Tekintettel azonban arra, hogy a számítógépes alkalmazások többsége hálózati környezetben működik, ezért nem csak szoftver szinten, hanem a dolgozók oktatásának a vonatkozásában is külön kitérünk a hálózati biztonságtechnika alkalmazásának a fontosságára.

Alaposan tanulmányoztuk a Linux kialakulását, fejlődését, majd egy kisáruház informatikai rendszerének a felépítését modelleztük. Feltételezzük, hogy a hálózat most kerül létrehozásra, és javaslatot teszünk a meglevő szoftver-, illetve hardvereszközök optimális felhasználására is. A kutatásunkról szóló cikkben azonban nem csupán az egyes biztonsági problémák/támadások elleni védekezést mutatjuk be, hanem a

többnyire általunk fejlesztett forráskódokkal érzékeltetjük, hogy egy adott rendszerben mekkora sebezhetőséget jelentenek az egyes puffer-túlcsordulási hibák, és hogy milyen egyszerű ezeket a hibákat kihasználva betörni egy rendszerbe. Alapvető célunk egy Linux-alapú hálózat megvalósítása, de mivel az átállás nem történhet egyik napról a másikra, ezért a vírusok, férgek bemutatása, az aktív védekezés szintén elengedhetetlen témája munkánknak. Ebben a részben mutatjuk be azt is, hogyan építjük fel a vállalkozás informatikai hálózatát. Meghatározzuk az egyes konfigurációk erőforrásigényét, és javaslatot teszünk a meglevő eszközök felhasználására.

Az alkalmazott szoftverek bemutatását munkánk egyik legfontosabb elemének tekintjük. Külön tárgyaljuk a szerver oldalon alkalmazandó és külön a kliens oldalon felhasznált programokat, röviden jellemezzük azokat, és megadjuk az elérhetőségüket. A költségelemzés részben a bemutatott szoftvereket a Microsoft Windows környezetre elérhető, azonos funkciót megvalósító alternatívákkal hasonlítjuk össze. Jelen cikkünkben a két legjelentősebb kliens és szerveroldali alkalmazást ismertetjük, majd dokumentálva az egyes lépések szükségességét, kitérünk arra, milyen szintű biztonsági elemeket alkalmazunk.

A következtetések megfogalmazásánál a rendszerbe integrálandó programokat és azokat a részeket elemezzük, amiket a későbbiekben még részletesebben szándékozunk kidolgozni, hiszen tudatában vagyunk annak, hogy 100%-os biztonság sajnos nem létezik, de folyamatos fejlesztéssel és odafigyeléssel, megfelelő megoldások alkalmazásával hálózatunk védettségét mindig magas szinten tarthatjuk.

## A Linux bemutatása

A Linux a UNIX operációs rendszernek a ’90-es évek elején PC-re kifejlesztett változata. A Linux főleg az egyetemi oktatók és kutatók körében tett szert nagy népszerűségre, mivel az otthoni PC-n megírt programjaikat könnyedén tudták az egyetemi mainframe-eken vagy a szuperszámítógépeken futtatni. A Linus Torvalds által kezdetektől fogva 32 bitesnek, több felhasználósnak, több feladatosnak fejlesztett Linux operációs rendszer stabilitása kezdettől fogva sokkal jobb volt, mint az akkori PC-s rendszereké (lásd DOS, Windows 3.x, Windows 95).

Miután Lius Torvalds nyílttá tette a Linux forráskódját, hamarosan programozók ezrei írtak kiegészítéseket hozzá. Mivel a profitorientált vállalatoktól teljesen független Linux-fejlesztést nem befolyásolták pénzügyi érdekek, ezért a Unix operációs rendszertől eltérően a GNU General Public License (GPL) hatálya alatt álló Linux teljesen ingyenesen terjeszthető. Az eltelt 13 év alatt a Linux teljes értékű operációs rendszerré nőtte ki magát, és mára már nemcsak a szerverpiacon ért el komoly részesedést, és szerzett olyan jeles támogatókat, mint az IBM, az Oracle, a SUN Microsystems vagy a Novell, hanem már az asztali munkaállomások piacán is megállja a helyét, sőt a beágyazott eszközökben is találkozhatunk ezzel, a bármely architektúrára rugalmasan portolható rendszerrel. A szerveroldali, szintén nyílt forráskódú és ingyenesen terjeszthető kitűnő web-, ftp-, mail-, DNS- és proxyszerver-programok mellett a Linux olyan hatékony munkaasztalkezelő programokat is kínál, amelyek a modern munkaasztalkezelő szoftverektől elvárt funkcióikkal lehetővé teszik a grafikus, ablakozó megjelenítést. A két legkomolyabb projekt a KDE: Kommon Desktop Enviroment (www.kde.org), valamint a GNOME (GNU Network Model Object Enviroment http://www.gnome.org) [Peterson, 2001].

## Hálózati biztonságtechnika

A 20. század végére az információ valós hatalommá vált, aki megfelelő időben birtokában van a megfelelő információnak, az hatékonyan tudja befolyásolni az eseményeket. Az egyre erősödő piaci verseny azonban ipari kémkedések sorozatát válthatja ki. Sokszor nem is tudunk róla, és versenytársaink vagy a rosszindulatú felhasználók ellopják titkos fejlesztéseinket. Ebben a környezetben a védelem, a biztonság kialakítása minden vállalat számára elengedhetetlenül fontos.

Mivel sokat lehet hallani a különböző vállalatok kárára elkövetett számítógépes betörésekről, ezért egyre nagyobb hangsúlyt kell fektetni a számítógépes rendszerek biztonságos működtetésére, a rendszer és adatbiztonság tudatos tervezésére [Raffai, 1999]. A vállalatok többsége bízik a nagyon drága betörésfigyelő rendszerekben, a tűzfalakban, amelyek megvédhetnek a támadásoktól, de ha nem fektetünk kellő hangsúlyt a dolgozók oktatására, ha nem mondjuk el, hogy mire figyeljenek oda, akkor a sok ügyeskedőt, akik dolgozónak, szerelőnek, felhasználónak, rendszergazdának adják ki magukat, nem tudjuk megfékezni [Mitnick-Simon, 2003].

### Biztonsági megoldások Linux alatt

A különböző Unix rendszerek forráskódja, köztük a Linuxé is, szabadon felhasználható, vagyis a biztonsági problémákat, megoldásokat bárki tesztelheti. Egy rendszer nyíltsága sokkal inkább az erőssége, mint a gyengesége, hiszen így a világon számos fejlesztő dolgozhat azon, hogy a felfedezett hibákat kijavítsa. A biztonság érdekében végrehajtott javítások általában napokon belül elkészülnek, de volt már arra is példa, hogy a hiba bejelentése után pár órával letölthető volt a biztonsági frissítés.

A Linux biztonsági rendszere a különböző fájl- és könyvtáreléréseken alapul, amelyeknek alapvetően három csoportját különböztetjük meg: írás, olvasás és futtatás/belépés.

A különböző műveletek végzéséhez kapcsolódó jogokat egy fájl vagy egy mappa esetében szintén három csoportba oszthatjuk:

– tulajdonosi jogok,
– csoportjogok, valamint
– egyéb felhasználók jogai.

A géphez közvetlenül hozzáférő felhasználókat csak a mindennapi munkájukhoz szükséges jogokkal ruházzuk fel. A legpontosabb beállítások azonban sajnos semmit sem érnek, ha a rendszeren túl sok felesleges szolgáltatás fut, vagy ha egyes programok túl nagy jogosultságszabadsággal futnak. Azokat a szolgáltatásokat tehát, amelyeket nem használunk, érdemes leállítani vagy eltávolítani, mert potenciális biztonsági rést nyithatnak meg. Kutatómunkánk során számos biztonsági tesztet végeztünk, és sajnos számos olyan szerverrel találkoztunk, amelyen, bár nem használtak C-fordítót, mégis telepítve volt. Mivel bármelyik felhasználónak joga volt azt futtatni, így óriási biztonsági lyuk keletkezett a rendszeren.

A rendszerünket azonban nemcsak a távoli támadásoktól, hanem a helyi felhasználóktól is meg kell védeni, célszerű például a jelszavas védelem (bootmanager jelszó, BIOS jelszó) használata. Ne feledkezzünk meg a gép fizikai védelméről sem, a legfontosabb rendszerelemeket elzárt terembe telepítsük.

## Biztonsági auditok

Egyre több vállalat bíz meg külső céget biztonsági rendszerének a kidolgozására, tesztelésére. Ekkor a megbízott cég munkatársai megvizsgálják az adott hálózatot, az informatikai rendszert a programok és az alkalmazás-beállítások biztonsági hibáit kutatva leteszteli, és többek között vizsgálják a dolgozók munkavégzését is. Ez utóbbit úgy hajtják végre, hogy például telefonon a cég egyik belső munkatársának adják ki magukat, és az alkalmazottak jelszavai után érdeklődnek. Természetesen mindezt úgy, hogy a dolgozó azt higgye, arra jogosult személynek adja az információkat. Az audit eredményét, vagyis a tesztelés alatt készült, a rendszer hibáit tartalmazó dokumentációt a vizsgálatot végzők átadják, a megbízónak.

## Overflow típusú hibák

Az overflow típusú hiba eredete tipikusan azoknak a programozóknak a lustaságából ered, akik a programozási munka során nem megfelelően kezelik a memóriaterületeket. Így például a karakterek hosszát nem vizsgáló függvények {*strcpy() strcat() sprintf() vsprintf()*} használata biztonsági hibákhoz vezethet.

A *Stack overflow* a legrégebben felfedezett verem-alapú túlcsordulás, amelynek elterjedése körülbelül 1995-re tehető. A *Heap overflow* néven emlegetett, a stack overflow-nál bonyolultabb túlcsordulás-kezelési áttörés 1998 körül volt. A Heap overflow lényege, hogy a memóriacímeket futás közben kell megváltoztatni. A legveszélyesebb azonban a 2000-ben publikált *Format string overflow*, amely a formázó karakterek hiányát használja fel tetszőleges kódok futtatására.

Bár az overflow-technika az idők folyamán nem sokat változott, csak újabb változatait találták k, a programokban mégis többféle biztonsági hiba van. A bemutatásra kerülő szkripttel például egy adott gépen könnyedén szerezhetünk adminisztrátori jogosultságot. A 17. ábra például egy általunk készített offset-tesztelő szkriptet mutat be. Ezzel végignéztünk egy olyan tartományt, amely szerint a program futása megáll, ha megfelelő offset-et talál, majd az *exit* parancs hatására tovább folytatódik egészen addig, amíg meg nem állítjuk, vagy amíg a tartomány végére nem ér [Flickenger, 2003].

```
[+]Az aktualis offset: 5
Trying system() at 0x8048407
system()'s address = 0x8048400
before overflow: funcptr points to 0x8048628
after overflow: funcptr points to 0x8048407
./brute: line 6:  368 Segmentation fault      ./hexpl $i
[+]Az aktualis offset: 6
Trying system() at 0x8048406
system()'s address = 0x8048400
before overflow: funcptr points to 0x8048628
after overflow: funcptr points to 0x8048406
sh-2.03# id
uid=0(root) gid=0(root) groups=0(root)
```

17. ábra Offset-tesztelő szkript

# Javaslat egy működőképes alkalmazásra

Kutatási munkánk során a megvalósítandó hálózathoz több, azonos feladatot ellátó szoftvert vizsgáltunk meg annak érdekében, hogy a leginkább megfelelőt telepítsük a rendszerre.

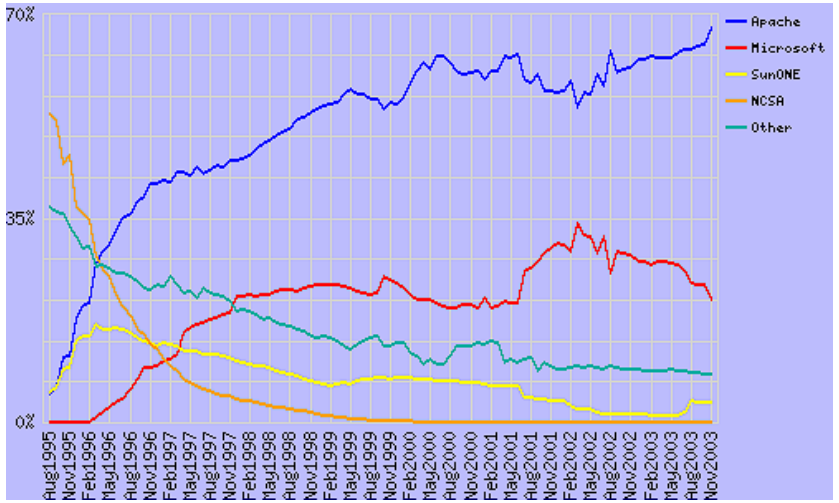
## Alkalmazott szoftvermegoldások

### Apache webszerver

Az Apache egy jól megírt webszerver, amely moduláris felépítésének köszönhetően nagyon biztonságos. Ha ugyanis az egyik moduljában hibát találnak, akkor elég a hibás modult kikapcsolni, a többi szolgáltatás pedig zavartalanul működhet tovább. Az Apache a föld egyik legelterjedtebb webszervere, ami a nyílt forráskódú szoftverek megbízhatóságát és elvárt minőségét bizonyítja. A 18. ábra egy 2003. októberében készült felmérést szemléltet, amelyben 43 700 759 szervert vizsgáltak meg. Ebből jól látható, milyen változásokat jelentett az Apache megjelenése a webszerverek piacán.

### OpenOffice.org

A Microsoft Office programcsomag kiváltására az OpenOffice.org programot találtuk a legmegfelelőbb szoftvernek. A többmillió kódsorból álló OpenOffice.org a nyílt forráskódú közösség legnagyobb projektje. Képességeit tekintve fontos kiemelni, hogy meg tudja nyitni az összes Microsoft Office formátumot, illetve menteni is tud ezekbe a formátumokba.



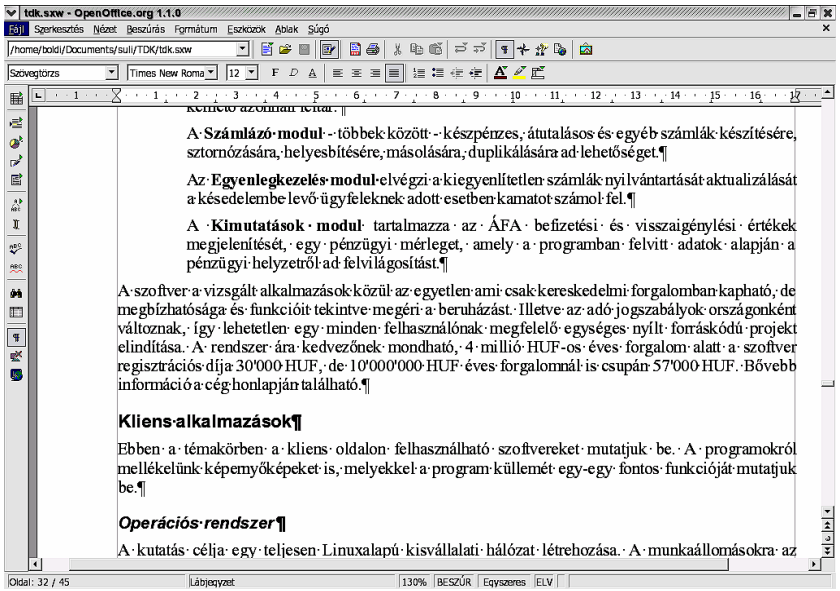

Forrás: http://www.netcraft.com

18. ábra Webszerverek használata 1995-2003 között

A Microsoft Office-szal összevetve meg kell állapítanunk, hogy amíg annak képességeit a felhasználók általában mindössze 10%-ban használják ki, mégis kénytelenek a teljes programcsomagot több mint 100 000 Ft-ért megvásárolni. Az OpenOffice.org ugyanakkor teljesen ingyenes, és tartalmaz minden általánosan használt programot. Megtalálható benne a szövegszerkesztő, a számolótábla, a prezentációkészítő, a rajzoló és az egyenletszerkesztő program is, illetve képes különféle adatbázisokhoz csatlakozni. Ennek a segítségével az e-mailek, körlevelek címzettjeinek a meghatározásához a Mozilla címjegyzéke, illetve a központi LDAP címtár is használható.

A 19. ábra az OpenOffice.org-gal készített dokumentum képernyőképét mutatja. Az eszköztárakon jól látható a program képessége. A dokumentumon minden szükséges formázást elvégezhetünk, legyen az a betű tulajdonságainak módosítása, egyenletszerkesztés, diagramkészítés vagy akár animált, több pontból megvilágított 3D-s felirat beszúrása.

Az OpenOffice.org-gal prezentációt is készíthetünk, amivel leendő ügyfeleinknek tarthatunk előadást, vagy akár a vállalton belüli oktatást az így elkészített prezentációkra építhetjük. A dokumentumok közreadása sem okoz problémát hiszen az OpenOffice.org képes a munkát rögtön .pdf-formátumba exportálni, így az Microsoftos és Linuxos környezetben egyaránt olvasható.




19. ábra OpenOffice.org-gal készített dokumentum

Az OpenOffice.org jelenlegi legfrissebb változata az 1.1 szintén elérhető a projekt magyar honlapján a http://office.fsf.hu címen. Ebbe a lokalizált változatba a Hunspell magyar helyesírás-ellenőrzőt és a szinonímaszótárat is beintegrálták.

## Költségelemzés

A szoftverek képességeit vizsgálva fontosnak tartottuk, hogy az alkalmazás pénzügyi vonatkozásait is elemezzük. Az elemzés alapját a Windows-rendszer megfelelő programjával/funkciójával való összehasonlítás képezte.

A vizsgálat eredményeként megállapítottuk, hogy a vállalkozás esetében egy Microsoft Windows alapú szoftverekből álló informatikai rendszer felépítése több mint 24-szer többe kerül, mint tisztán nyílt forráskódú szoftverekből. Érdemes tehát megfontolni a nyílt forráskódú technológiára való váltást, mert a megtakarított több mint egymillió forintból más fontos beruházásokat végezhet a cég, így például a fennmaradó összegből bőven lehet finanszírozni a dolgozók oktatását, képzését is.

## A rendszer általános biztonsági beállításai

A vállalatok számára legfontosabb a cég presztízsének, a nem publikus információknak a védelme. Sok vállalat azonban nem méri fel, hogy milyen veszély leselkedik az informatikai rendszerekre, hogy a vállalat megbízhatóságára nagyban rányomja bélyegét szervereinek a gyakori feltörése.

A mi kis vállalatunknál megpróbáltunk maximális biztonságot nyújtani úgy, hogy emellett a munka zavartalan legyen. A költségekkel úgy takarékoskodtunk, hogy az ne menjen a megbízhatóság rovására. A megoldás annyira egyszerű, hogy az itt bemutatott biztonsági beállításokat egy közepes képességű rendszergazda pár nap alatt el tudja végezni.

Az általunk megtervezett hálózat három fő részből épül fel :

– tűzfal, gateway (hálózati tartománya: 1.2.3.4)
– DMZ (hálózati tartománya: 192.168.1.0)
– belső hálózat (hálózati tartománya: 192.168.2.0)

Létrehoztunk egy DMZ-t (Demilitarizált Zóna), amelyben az ftp-, a web-, a mail-, az adatbázis- és a log-szerverek helyezkednek el. Ennek a zónának az a szerepe, hogy ha esetlegesen betörnek a hálózatba, akkor innen ne tudjanak az irodai alkalmazásokat futtató belső hálózat felé eljutni. A hálózatunk leggyengébb láncszeme egyértelműen a tűzfal-gép, tehát ezt kell a lehető legkörültekintőbben konfigurálni [Garfinkel, 2003].

## Következtetések

Egy kutatómunka, egy fejlesztés valójában soha sem ér véget, de munkánk során tanulunk, tapasztalatokat gyűjtünk, ezekből következtetéseket vonunk le, és egy-egy állomásnál megjelöljük azt az irányt, amelyen a kutatást folytatni kívánjuk.

A tesztelés során megállapítottuk, hogy a nyílt forráskódú programok tekintetében egy adott feladatra számos ígéretes alternatíva kínálkozik. A világszerte beszerezhető Linux-disztribúciók száma több tízre tehető, de nincs ez másként a levelezőszerver-megvalósítások terén sem. Az egyes alkalmazási feladatokra rengeteg kitűnő megoldás létezik, ezért, a számunkra legmegfelelőbb lehetőség kiválasztása sokszor nagyon nehéz volt.

Mivel 100%-os biztonság nem létezik, ezért fontos, hogy naprakészen ismerjük a legújabb biztonsági behatolási módszereket, és azok kivédésének a módját. Későbbi fejlesztéseink során egy olyan, központilag menedzselhető terhelésfigyelő rendszer integrálását tervezzük, amivel az egyes kritikus szervereket bárhonnan kezelni tudjuk, ellenőrizhetjük a szerverszoba hőmérsékletét, illetve riasztást kapunk, ha ezek meghaladnak egy kritikus értéket. Megfelelően konfigurált alkalmazásszűrő proxykkal a továbbított csomagok tartalmának vírusellenőrzésére ugyanúgy lehetőség nyílik, mint a tunneling alagút-technikával megvalósított kapcsolatok aktív szűrésére, amire egyébként a csomagszűrő szoftverek képtelenek. Már ezekre a célokra is léteznek ingyenes és nyílt forráskódú programok, így például ezekre a feladatokra tökéletesen megfelel a Nagios hálózat-felügyeleti szoftver vagy a Zorp professzionális, magyar fejlesztésű alkalmazásszűrő tűzfal-implementáció (http://www.nagios.org, http://www.balabit.hu).

Fontosnak tartjuk egy átállási ütemterv meghatározását is, ahol részletesen végigmennénk a migráció minden lényeges mozzanatán, ezzel is segítve a fejleszteni szándékozó vállalkozásokat a Linuxra történő zökkenőmentesebb átállásban.

Úgy gondoljuk, hogy mivel a Linux-alapú operációs rendszerek és a nyílt forráskódú programok eddigi eredményei (több nagy cég sikeresen állt át erre az alternatívára, az amerikai hadsereg kritikus környezetben alkalmazza, jelentősen növekszik az otthoni felhasználók köre stb.) igazolják a nyílt forrású alkalmazások létjogosultságát, ezért ezeket a fejlesztéseknél hatékony megoldásként kell figyelembe venni, különösen olyan informatikai beruházások esetében, amelyeket kisebb tőkével rendelkező vállalkozások hajtanak végre.

## Hivatkozások

[Flickenger, 2003] Flickenger, Rob: Linux Server Hacks – O'Reilly
[Gagné 2002] Gagné, Marcel: Linux rendszerfelügyelet – Kiskapu Kft.
[Garfinkel, 2003] Garfinkel: Practical Unix & Internet Security – 3. kiadás, O'Reilly
[Mitnick-Simon, 2003] Mitnick, K. D. – Simon, W. L.: A Legendás Hacker – Perfact-Pro Kft.
[Petersen, 2001] Petersen, Richard: LINUX Teljes referencia – Panem
[Raffai, 1999] Raffai, Mária: BCP üzletmenet-folytonosság biztosítása - Megelőzési felkészülési és helyreállítási terv – Novadat Kiadó