Informatika 1.

Az operációs rendszer szerepe, szolgáltatásai /Windows és Linux/

Dr. h.c. Dr. Szepes , András

Created by XMLmind XSL-FO Converter.

Informatika 1. : Az operációs rendszer szerepe, szolgáltatásai /Windows és Linux/

Dr. h.c. Dr. Szepes , András
Lektor : Cseri , Tamás

Ez a modul a TÁMOP - 4.1.2-08/1/A-2009-0027 „Tananyagfejlesztéssel a GEO-ért” projekt keretében készült. A projektet az Európai Unió és a Magyar Állam 44 706 488 Ft összegben támogatta.

v 1.0

Publication date 2010
Szerzői jog © 2010 Nyugat-magyarországi Egyetem

Kivonat

Az informatika egyik legösszetettebb területe a szoftverekkel foglalkozó szakterület. Igen szerteágazó részei vannak, melyek gyakran nincsenek is kapcsolatban egymással. Legalább is látszólag!

Jelen szellemi terméket a szerzői jogról szóló 1999. évi LXXVI. törvény védi. Egészének vagy részeinek másolása, felhasználás kizárólag a szerző írásos engedélyével lehetséges.

Created by XMLmind XSL-FO Converter.

Tartalom

Created by XMLmind XSL-FO Converter.

1. fejezet - Az operációs rendszer szerepe, szolgáltatásai /Windows és Linux/

1. 1.1. Bevezetés

Az informatika egyik legösszetettebb területe a szoftverekkel foglalkozó szakterület. Igen szerteágazó részei vannak, melyek gyakran nincsenek is kapcsolatban egymással. Legalább is látszólag!

Ön a következőkben meg fog ismerkedni:

- a szoftverek csoportosításával,
- az operációs rendszerek fogalmával,
- a DOS és a Windows világ,
- valamint a Linux

alapvető ismereteivel.

A fejezet elolvasása után képes lesz

- értelmezni az operációs rendszerek alapfogalmait,
- alkalmazni az operációs rendszerek alapszolgáltatásait,
- választani céljának megfelelő szoftvert.

Ha valamely alfejezet túl kevés lenne Önnek, akkor nézzen utána az Irodalomjegyzékben jelzett könyvekben, vagy a könyvtárban, ahol van még jó néhány, a témába vágó szakkönyv, melyeket nyugodtan felhasználhat. Ez a lista csak segítséget akar nyújtani, de nem tesz kötelezővé semmit.

2. 1.2. A SZOFTVER

A feladatmegoldás gépi részén túl elhelyezkedő, szellemi termékként megjelenő fontos része a szoftver (software=puha árú). Tág értelemben a szoftver magában foglalja a működtető programokat, a kezelési útmutatókat, és a dokumentációkat. A szoftverek alapeleme az operációs rendszer (ld. következő fejezet). További tagozódásának egy lehetséges vázlata:

Created by XMLmind XSL-FO Converter.

1-1. ábra Programok osztályozása

A számítógépek egyetlen nyelven értenek, ez a gépi kód. Az egyes eszközök gépi nyelve, kódja eltérő, de egyben azonban hasonlítanak mégis: számjegy-kombinációk sorozatából állnak. A gépi kód teljes részletességgel lebontva adja meg az utasításokat. A gépi kódú programozás igen alapos hardver és szoftver ismereteket igényel.

A felhasználók, de még mindig inkább a professzionális fejlesztők számára dolgozták ki az assembly nyelvet. Ez még alacsony szintű, szimbolikus nyelv, de már bizonyos könnyítéseket tartalmaz a gépi kódhoz képest. Alkalmazása a rendszerszoftverek fejlesztésekor és a tárkihasználás támogatásakor indokolt elsődlegesen.

A felhasználók számára szükséges volt olyan nyelveket létrehozni, melyek közelebb állnak az emberi nyelvhez és alkalmasak lehetnek a különböző gépek közötti program-átvitelre.

Ilyen célra születtek az emeltszintű, vagy más néven magas szintű programozási nyelvek. Ezek igen széles skálán mozognak, különböző rendeltetésűek és hatékonyságúak. Néhányat közülük már a Bevezetőben említettünk is.

Hangsúlyozni kell, hogy az alkalmazott nyelv csak eszköz a számítógép irányításában! Nem lehet meghatározó, hogy milyen nyelven oldjuk meg feladatainkat. Mindig a célnak megfelelőt, az adott eszközön alkalmazhatót kell választani. Döntőbb kérdés az alapfeladat jó megfogalmazása, a leghatékonyabb megoldás (a legjobb algoritmus) megtalálása.

3. 1.3. AZ OPERÁCIÓS RENDSZER

Az operációs rendszer (operating system=OS) gondoskodik a számítógép alapvető működéséről, kezeli a rendszerbe kapcsolt erőforrásokat, gondoskodik az esetleges megszakítások figyeléséről. Fejlettségi szintjétől függően az előbbieken túl még egyéb szolgáltatásokkal is rendelkezhet. A különböző számítógépek eltérő operációs rendszereket alkalmaznak.

Az 1980-as évek kiskapacitású számítógépeinek operációs rendszerei szinte rejtve voltak az átlagos felhasználók elől. Itt leggyakrabban közvetlenül egy emeltszintű nyelvvel, kezdetben Fortran, Cobol, Algol nyelvekkel, majd a széles körben elterjedt BASIC-kel, és Pascal nyelvvel találkoztak az alkalmazók. Ezek közvetlen gépi hozzáférése nehézkes, és nem eredményezi azok igazi előnyeit.

Created by XMLmind XSL-FO Converter.

Ki kell emelni azonban ebben a kategóriában a széleskörűen elterjedt CP/M programot(Control Program for Microcomputer = mikroszámítógépek vezérlőprogramja). Ez az Intel8080 és a Z80 jelű mikroprocesszorra épülő számítógépek hajlékony mágneslemezes operációs rendszere.

A mikroszámítógépek másik nagy csoportjába a professzionális személyi számítógépek tartoznak. Az operációs rendszerük már részben vagy egészben mágneslemezen helyezkedik el. Innen származik az elnevezésük is : DOS (disk operating system). A későbbiekben ezek közül az IBM számítógépekben alkalmazott operációs rendszerrel foglalkozunk részletesebben, mivel ezek elterjedése ezt különösen indokolja.

Nagyobb eszközök esetében már felmerül a többfelhasználós alkalmazás, a multiprogramozás igénye. Ekkor az operációs rendszer gondoskodik a végrehajtás során az időosztásos működésről. Az **időosztásos, time-sharing vagy multitasking rendszerek** közvetlen kommunikációt biztosítanak a felhasználó és programja, valamint az operációs rendszer között. Bár a CPU egyszerre több párhuzamosan dolgozó felhasználó között meg van osztva, azok úgy dolgozhatnak a számítógépen, mintha az kizárólag hozzájuk tartozna. Fontos elvárás, hogy a rendszer válaszideje egy megadott tűréshatáron belül legyen. A háttérben futó batch rendszer biztosítja azt, hogy a feldolgozás szüneteiben se legyen tétlen a gép. Ezek jellegzetes megvalósításai a UNIX, a VMS, az OS/390, az OS/400, a Windows NT, stb. operációs rendszerek. Legfőbb jellemzője a hierarchikus állományrendszer, mely kiegészül "felszerelhető" (mountable) kötetekkel. Ez annyit jelent, hogy cserélhető lemezeken levő állományok hozzákapcsolhatók az alaprendszerhez. Az I/O rendszere egységes az állományok, a fizikai berendezések és a folyamatok között.

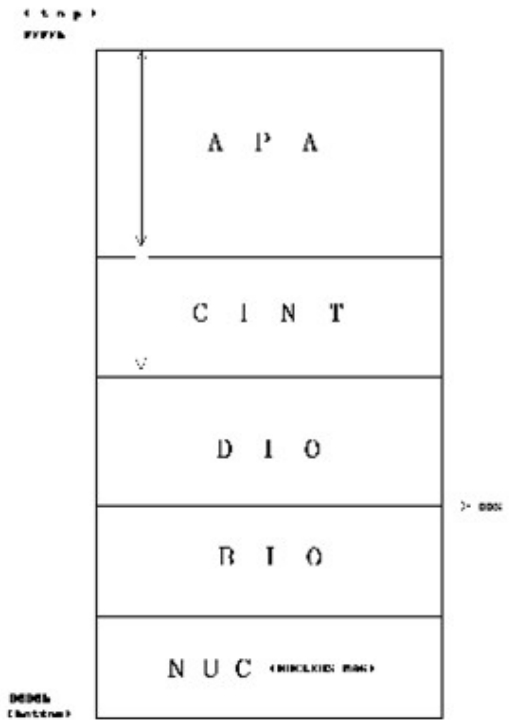
4. 1.4. A DOS általános felépítése

Az operációs rendszerek felépítését egy DOS általános tanulmányozásával szemléltetjük. Ehhez vegyünk egy közepes teljesítményű mikroszámítógépet alapul. Maga a DOS több szegmensből áll. Felépítése szerint vagy az operatív tár felső részén (top), vagy ellenkezőleg az alján (bottom) helyezkedik el (ld. 2-1. ábra).

Központi szerepet játszik a mag, amely az erőforrások elosztását és az I/O eszközökhöz való hozzáférést biztosítja. Amikor az alkalmazói programnak perifériára van szüksége, a NUC-hoz kell fordulnia. A periféria elérést a lehető legközvetlenebbé és leggyorsabbá, de periféria függetlenné igyekeznek tenni, ezért többnyire egyetlen rendszer-szubrutint alkalmaznak erre a célra. Ezt nevezhetjük rendszer erőforráskérésnek (system resource requist = SRR). Az SRR a tár egy alacsony címén helyezkedik el. Hívásakor meg kell adni egy attribútumot, amely kifejezi, hogy melyik szolgáltatást kívánjuk igénybevenni. Az attribútumot egy CPU regiszterben kell hívás előtt elhelyezni.

A következő DOS szegmens a Byte-orientált be/kiviteli rész (byte-oriented input-output = BIO). Ez gondoskodik valamennyi byte-orientált periféria - konzol, nyomtató, mágnesszalag - kezeléséről. Ha a NUC-hoz kiszolgáláskérés érkezik, az az adott kódtáblában szereplő funkciókód alapján átadja a vezérlést valamelyik alacsonyabb szintű szegmensnek. A BIO elején egy sor ún. ugrásvektor van, melyek a BIO belső rutinjaira mutatnak.

Created by XMLmind XSL-FO Converter.



1-2. ábra A DOS elhelyezkedése a memóriában

Hasonló feladatokat lát el, de sokkal bonyolultabb a lemezes be/kiviteli szegmens (disk input/output = DIO). Ez kezeli a lemezegység(ek) mechanikáját és szervezi az egységhez való hozzáférést.

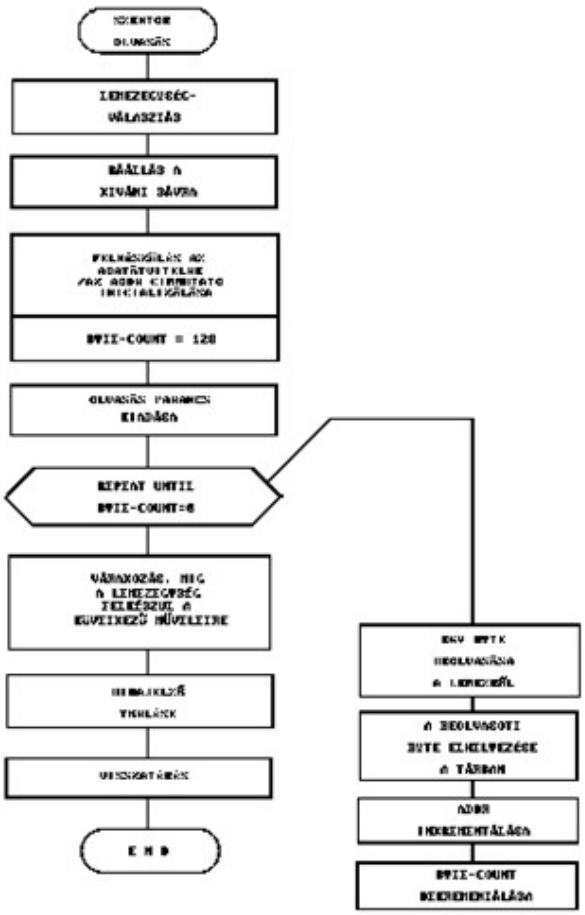
A mágneslemezeken az adatokat szektorokba (sector) és sávokba (track) szervezve rögzítik. A lemezegységtől és az alkalmazott lemez minőségétől függően több vagy kevesebb sávot képeznek ki koncentrikus formában. A sávokon belül képződnek szektorok. Ezek lehetnek sávonként azonos számúak, de gyakran befelé haladva csökkenő számú szektort képeznek ki.

A DIO két legfontosabb funkciója a szektorolvasás és a szektorírás. A szektorolvasás általános szubrutinjának folyamatábráját a 1-3. sz. ábra tartalmazza. A folyamat elején meg kell határozni, hogy melyik egység melyik sávjáról kell olvasni (drive = track selection).

Az eredmény tárolása után hardveres művelet következik, az olvasó fej pozicionálása. A betöltendő adat tárcímének beállítása után következhet az olvasás byte-onként. Ha az olvasás eredményeként nem jelentkezett hibakód, a DIO visszaadja a vezérlést a főprogramnak. Az írás folyamata is ehhez hasonlóan megy végbe.

Az írás és olvasás funkció hatékonysága nagymértékben függ a szektorok szervezésétől. A művelet végrehajtásakor fellépő probléma abból adódik, hogy mire a szoftver felkészül a következő szektorban végzendő tevékenységre, a lemez már túlfordul a szektorhatáron. Ilyenkor egy teljes körülfordulást kell várni a műveletvégzéshez, ami lassítja a végrehajtást. A probléma megoldása az, ha a logikailag egymást követő szektorok fizikailag nem egymásután helyezkednek el. Általában háromszektoros ugrásokat alkalmaznak. A felosztásról szektor-térképet készítenek (sector mapping). Lényeges ismerni a lemezen levő szabad szektorok helyét a gyors és biztonságos felhasználáshoz. Erről szabadszektor-nyilvántartást kell vezetni. Ez rajta van a lemezen, és felhasználás előtt be töltésre kerül a tárba. Itt NUC vezeti a változásokat és szervezi a felhasználást.

Created by XMLmind XSL-FO Converter.



1-3. ábra A szektorolvasó szubrutin folyamatábrája

A parancsértelmező szegmens (command interpreter = CINT) már alkalmazói programnak is tekinthető, működésekor a NUC által rendelkezésre bocsátott erőforrásokat használja. Az 5. sz. ábrán is látható, hogy a CINT területét más szegmens is használhatja. Ennek oka az, hogy alkalmazói programok futása közben nincs szükség a CINT-re. A CINT feladata, hogy a DOS működését irányítsa és ellenőrizze, tegye lehetővé a felhasználó számára a rendszer vezérlését.

Az utolsó szegmens az alkalmazói programokat tartalmazza (application program area = APA). A CINT és az APA nem memória rezidens, azaz nincsenek állandóan a tárban.

A számítógép bekapcsolása és alaphelyzetbe állítása után be kell tölteni az operációs rendszert. Ehhez már a tárban kell lennie egy erre alkalmas betöltő programnak. Ezt kezdeti programbetöltésnek (initial program load =IPL) nevezzük. Szakzsargonban boot-programnak is nevezik. Ez a BIOS része, amely a különböző hardver-elemek eltérő tulajdonságait úgy hidalja át, hogy az operációs rendszer felé egységes felületet mutat. A BIOS hardverfüggő és egy flash memóriát tartalmazó chipben található az alaplapon.

5. 1.5. A Windows operációsrendszer család

A karakteres üzemmódú operációs rendszerek idejében gyorsan kialakult az igény az olyan segédprogramok iránt, melyek megkönnyítenék a felhasználók munkáját. Készült olyan program, melyben csak rá kellett mutatni az állományok nevére, majd ezt követően a műveletek jelére, és ezzel végre is lehetett hajtani a parancsokat. (Pl. Norton Commander, stb.) Más gépcsaládok alakuló grafikus felülete mellett elkészült a Windows család őse is. Ez még nem volt operációs rendszer, bár átvette annak funkcióit. Nevezték „kvázi operációs rendszernek” is. A Windows 3.1 mellett megjelent a Workgroup 3.11 is, mely már lehetővé tette a számítógépek hálózatba kapcsolt kezelését is.

A nagy váltás a Windows NT (New Technology) megjelenése volt. Ez már valódi operációs rendszer lett. Lehetővé tette mind az egyenrangú, mind pedig a szerver-kliens üzemmódú hálózatok kialakítását. Igen nagy

Created by XMLmind XSL-FO Converter.

gondot fordított a biztonsági kérdésekre. Már valós módon támogatta a többfeladatúságot is (multitasking). A gondot csak az igen nagy erőforrás igénye jelentett. Ezért várható volt, hogy az otthoni felhasználók számára is ki fognak alakítani egy külön programot. Erre az igényre a Windows 95 megjelenése válaszolt, majd a továbbfejlesztett változata, a Windows 98. Ezzel párhuzamosan jelent meg a Windows NT 4.0 a vállalati felhasználók számára.

Az NT rendszer valójában 2 különböző szoftvert jelent; a szerver (server) és a munkaállomás (workstation) programokat. Az alapvető különbség közöttük, hagy a szerver program támogatja a hálózati kezelés mellett annak magasfokú biztonsági szolgáltatásait. A hálózatok kialakítása során a Windows maximálisan támogatja a „tartomány" (domain) szervezést. Ennek az a lényege, hogy 1-1 szerver „köré" építjük ki az egymással logikai kapcsolatba sorolható számítógépeket, melyeknek hálózati szolgáltatásait a szerver fogja biztosítani. Ez esetben a szerver programot már 3 módon installálhatjuk:

- domain controller (tartomány vezérlő).
- backup domain controller (a tartomány vezérlő biztonsági tartalékja).
- standalone server (olyan munkaállomás, mely alkalmas szerver üzemmódra is).

A két – az otthoni és a vállalati - rendszer egyre jobban közeledett egymáshoz. Nem csak a külső megjelenés lett közel azonos, de sok szolgáltatás is „átvándorolt" közöttük. A legközelebb a Windows ME (Milleneum Edition) és a Windows 2000 került egymáshoz. A Windows XP volt talán az utóbbi idők legsikeresebb terméke, melyet nem tudott leváltani a Vista, míg napjainkban kezdi történetét a Windows7 rendsze.

Alapvetően fontos jellemzőik:

- egységes felhasználói felület,
- egységes periféria kezelés,
- szolgáltatások átadása,
- objektum szemlélet.

5.1. 1.5.1. A felhasználói felület

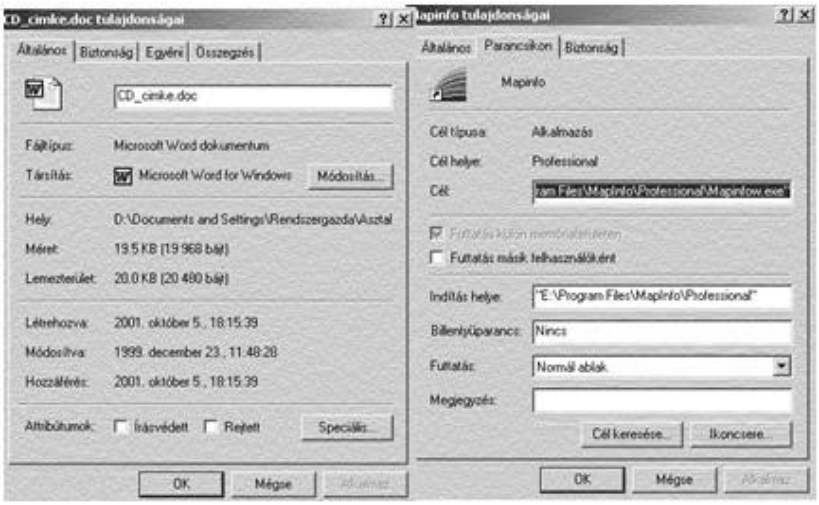
A rendszer a nevéhez illően az egyes önálló alkalmazásokat külön-külön ablakban jeleníti meg. Ez kezelői oldalon jelent előnyt. A „belső működésben" ugyanez ma már azt is jelenti, hogy az egyes alkalmazások önálló memória-felhasználással futnak. Ez biztosítja azt, hogy valamelyik feladat (task) sérülése esetén a többi még gondtalanul tud működni.

A belépés utáni bejelentkező képen az Asztal (desktop) fogadja a felhasználót (1-4. ábra). Az Asztal arra szolgál, hogy azon helyezzük el a legfontosabb alkalmazások és szerkesztett dokumentumaink elérését szolgáló ún. parancsikonokat. Az Asztal alsó része az ún. Tálca (taskbar). Itt a jobb oldalon az idő és a nyelv mellett változóan megjelennek a legfontosabb beállítási lehetőségek. A bal oldalon van a Start gomb, melyhez tartozó hagyományos menü felépítése a 1-4. ábrán jobb oldalán látható. Mellette az általunk elhelyezett ikonok és az elindított programok jelzőgombjai találhatók

Created by XMLmind XSL-FO Converter.

1-4. ábra

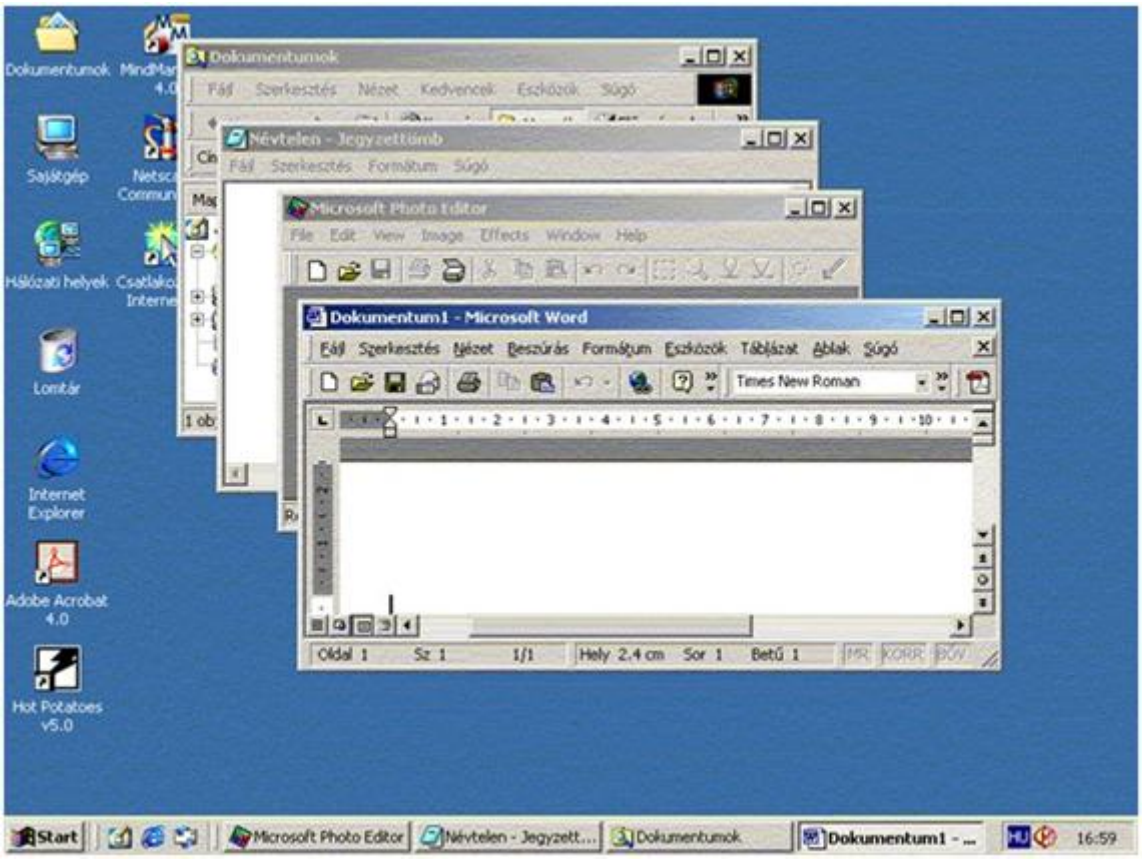
Az ikon önmagában pusztán csak egy grafika. A lényeg a hozzá rendelt hivatkozás. Ha megtekintjük egy ikon Tulajdonság leírását, akkor ott megtaláljuk hozzátartozó elemeket. (1-5. ábra)



1-5. ábra Tulajdonság lapok

Visszatérve az ablakokra nézzük azok felépítését!

Created by XMLmind XSL-FO Converter.



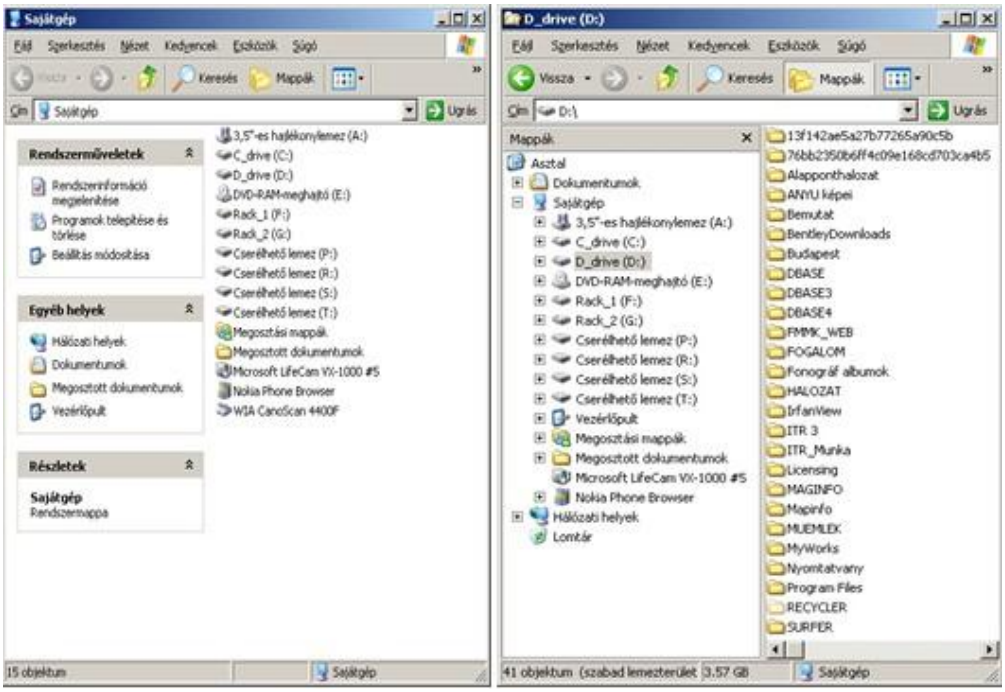
1-6. ábra Ablakok

Láthatjuk, hogy bizonyos szabályosság mutatkozik az ablakok elrendezésében. Minden ablak jobb felső sarkában 3 gombot találunk, melyek az ablakok átméretezésére szolgálnak (sorban minimalizálás [tálcára helyezés], előző méret/maximális és bezárás). A menüsorban általában az első az állományok kezelésére vonatkozó Fájl (Létrehozás, Megnyitás, Mentés, Nyomtatás, ...), az utolsó pedig a segítséget nyújtó Súgó. A legtöbb alkalmazás esetében hagyományosan a Fájl mellett a Szerkesztés található.

A rendszerhez alapértelmezetten tartozik egy fájlkezelő, az Intéző (Windows Explorer), illetve annak alrendszereként a Sajátgép. Az Intéző felépítésében az alapelv az, hogy a teljes rendszer egy fastruktúrába szervezhető. A fa ágai számítógép alapvető elemeire mutatnak (1-7. ábra jobb oldali kép). Az egyes egységeket mappákba (Folder) szervezve találjuk. A mappák jó közelítéssel hasonlítanak a hagyományos értelemben vett könyvtárra, de annál szélesebb körben használja a rendszer. A fa kiindulópontja az Asztal, melyből kiágaznak a Dokumentumok, a Saját gép, a Hálózati helyek, a Vezérlőpult, a legfontosabb megosztási mappák és a Lomtár (kiegészülhet néhány utoljára használt alkalmazással). A Saját gép megnyitása után ott kizárólag a számítógépünk erőforrásainak jeleit találjuk, kiegészítve a Vezérlőpult elérésével. Ez utóbbi a számítógép alapvető beállításainak helye.

A Lomtár a törölt állományokat és dokumentumokat gyűjti össze, és lehetővé teszi a téves törlések utáni helyreállítást. Ha a törlést véglegesnek szánjuk, úgy lehetőség van a Lomtár kiürítésére. Ez a törlés már nem állítható helyre!

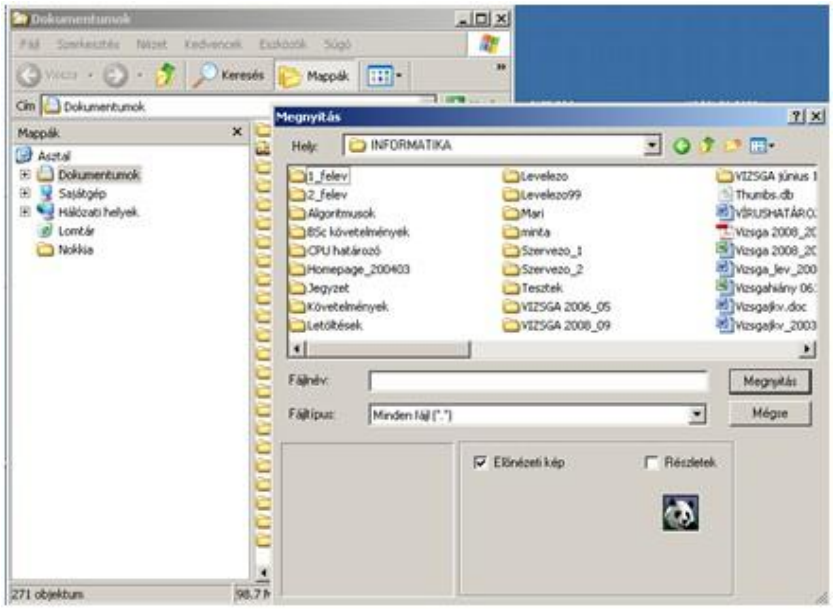
Created by XMLmind XSL-FO Converter.



1-7. ábra Az Intéző (Explorer)

5.2. 1.5.2. Szolgáltatások átadása

Folytatva az egységesség megismerését, nézzük meg egy alkalmazás Fájl menüjének megnyitás parancsának hatását! (1-8. ábra)



1-8. ábra

A képen láthatjuk, hogy az állományokkal kapcsolatos műveletek során az Intézőhöz hasonlatos állománykezelőt kapunk. Ugyanez lenne a helyzet a nyomtatás esetén is. Az abból következik, hogy ezeket a szolgáltatásokat az alaprendszer tartalmazza, s a programozók szabadon felhasználhatják azt. Ezáltal viszonylag könnyű dolga van a felhasználónak, hisz a különböző alkalmazások igénybevételekor nem kell mindig új elveket megtanulnia.

5.3. 1.5.3. Egységes perifériakezelés

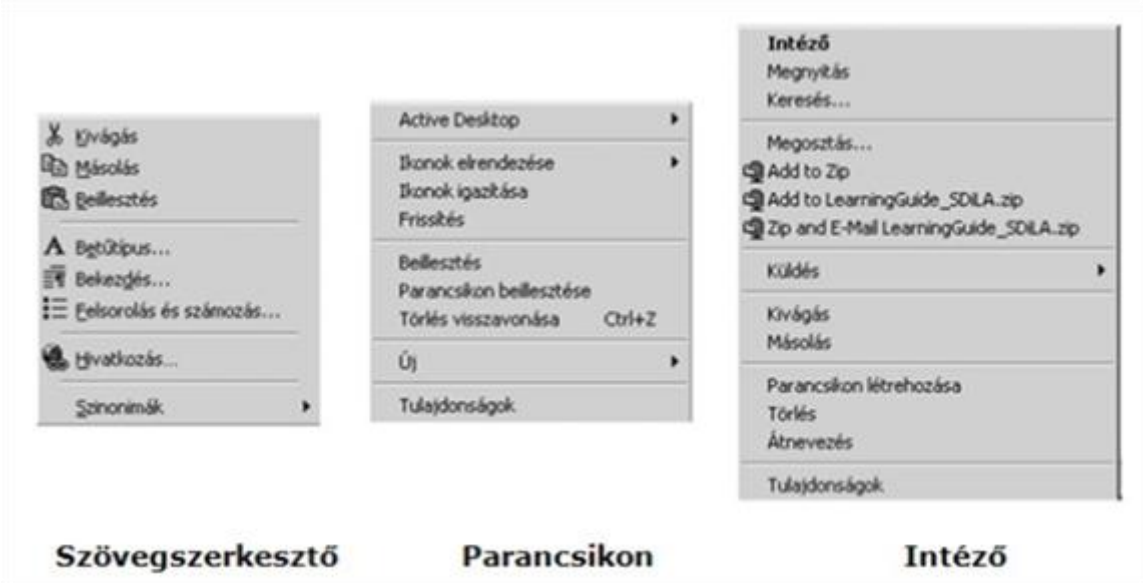
Created by XMLmind XSL-FO Converter.

A Windows rendszer – hasonlóan más korszerű operációs rendszerekhez – saját maga végzi a perifériák kezelését. Azaz a különböző eszközök meghajtó-programjait (driver) az operációs rendszerhez kell illeszteni, és alkalmazások azon keresztül érik el. Ezzel éri el a rendszer az egységes kezelés megoldását.

5.4. 1.5.4. A vágólap

A vágólap működésének megértéséhez előbb az objektum fogalmát tisztázzuk. A rendszer és alkalmazásai minden olyan elemet, mely önálló egységbe foglalható, objektumnak tekintik. Ezekkel az objektumokkal műveleteket végezhetünk oly módon, mintha az egyetlen elem lenne. A végrehajtható alapműveletek egységesek (Másolás, Kivágás, Beillesztés, mozgatás...). Ezen túlmenően természetesen az alkalmazások egyedi műveleteket is tartalmazhatnak az objektumok kezelésére.

Azok az alapvető műveletek, melyek egy feladat adott helyén egy-egy objektumra vonatkozóan végrehajthatók, nemcsak a menürendszerből, hanem még egy módon elérhetők. Ez a lehetőség a helyzetérzékeny menü, melyet az egér jobb gombjának lenyomásával érhetünk el. A művelet végrehajtásakor értelemszerűen mindig más menüsort kapunk.



1-9. ábra

A feladataink végrehajtása során igen gyakori, hogy egy objektumot át szeretnénk helyezni, illetve másolni más környezetbe.

5.5. 1.5.5. A biztonság kérdése

Itt meg kell említeni, hogy a magyar biztonság szó az egyszerre a fordítása az angol safety és security szavaknak, ezért többet jelent, mint amit mi első hallásra mögé értünk (általában betörés vagy hozzáférés biztonságot).

A legtöbb szervezet rendelkezik olyan alkalmazásokkal, amelyeknek egymás között meg kell osztaniuk az információt. Amennyiben alkalmazási területeik között nincs integráció, úgy három probléma merül fel:

- adatok inkonzisztenciája jelentkezik,
- romlik a hatékonyság,
- a felhasználók az egyes alkalmazási csomagok foglyai lesznek

Integráció nélkül az adatokat többszörösen kell tárolni a különböző alkalmazásokban. Az adatoknak ezek az eltérő változatai egymással könnyen inkonzisztenssé válhatnak. Az adat-kettőződés mindenképpen alacsony hatékonyságú. Amennyiben egyik alkalmazásban szereplő adatot újra be kell vinni, vagy át kell alakítani, mielőtt még egy másik alkalmazásban felhasználhatnák, ez csökkenti a hatékonyságot. A felhasználók egy

Created by XMLmind XSL-FO Converter.

meghatározott szállítótól eredő specifikus alkalmazás foglyai lehetnek, mivel az nem vihető át egy alternatív csomagba.

Az adatok integrálásának két különböző megnyilvánulása van: az adatcsere és az adatmegosztás:

- *Adatcsere* az, ha az adatot az alkalmazások között áthelyezik, ami a létező adatok egynél több másolatát eredményezi.
- Az *adatmegosztás* különböző alkalmazások számára többszörös hozzáférést biztosít az adat egyetlen másolatához, központi adatbázishoz folyamodva az adattárolásban.

5.6. 1.5.6. Biztonság, elérés és titkosság

Mindenkor igény volt az adatok biztonságának és hozzáférésének ellenőrzésére, úgyszintén a titkosság szavatolására. Életbevágó nemcsak a tényleges adatokhoz, hanem a metaadatokhoz való hozzáférés ellenőrzése is. A Brit Számítógépes Társaságnak az adatbázis biztonsággal foglalkozó közös munkacsoportja részletes jelentést készített 1990-ben a biztonság, a hozzáférés és a titkosság egész területéről. Ebben meghatároztak számos e téren használatos fogalmat

A belső ellenőrzési rendszer az egész ellenőrzési rendszerre utal, pénzügyi és egyéb jellegű ellenőrzésre, amelyet a vezetés hozott létre azért, hogy a vállalat működését szabályszerűen és eredményesen vigyék végbe, biztosítsák az igazodást a vezetés irányelveihez, megőrizzék a vagyonkészleteket és amennyire lehetséges, biztosítsák a nyilvántartások teljességét és pontosságát. Egy belső ellenőrzési rendszer egyedi komponensei "ellenőrzés(i módok)" néven ismeretesek.

Biztonság

A biztonság a fent jellemzett ellenőrzés egyik módja, s azokat az eljárásokat fedi, amelyek a kockázatok korlátozását szolgálják egy szervezetben a minősítettség, az integritás, a rendelkezésre állás és az auditálás (független vizsgálat) területein.

A minősítettség az adatok, információk és más értékes elemek elzárását jelenti a fel nem hatalmazott személyek elől. Kiterjed azokra az eljárásokra, amelyeket az l985. évi Adatvédelmi Törvény ír elő (Nagy-Britanniában), s amelyekre "Titkosság" kifejezéssel utalnak.

Az integritás az adatok és az adatokból származtatott információk teljességét, helyességét, pontosságát és időszerűségét jelenti. Az adatgazdálkodásnak az adatok integritását minőségi kérdésként kell kezelnie. Az adatok helyességének megkövetelt szintjét célként kell kitűzni: az adatelem vagy egyedszintű elérendő értékek a hasznosak, pl. a téves elemek száma 100 bevitt elemre egy adott időszakban. Az adathibák forrásait azonosítani kell és azok az alábbiak lehetnek:

- manuális bevitel,
- helytelen rendszer-feldolgozás,
- nem megfelelő érvényesítési gyakorlat,
- rossz adatcsere-eljárások a rendszerek között,
- adat-meghatározási hibák (pl. közösen elfogadott meghatározások hiánya).

Miután lefektették a minőségi célokat és meghatározták a hibaforrásokat, az adatgazdálkodásnak a tulajdonosokkal együtt kell működnie, hogy kialakítsák a minőségi követelmények betartásához szükséges stratégiát.

A rendelkezésre állás foglalkozik azokkal az eljárásokkal, amelyek biztosítják, hogy a szolgáltatások és az adatok a felhatalmazott személyek rendelkezésére álljanak akkor, amikor azokat előreláthatólag igénylik. A rendelkezésre állás egyensúlyozó hatás az adatok biztonsága és titkossága között. Az adat-tulajdonosoknak meg kell fontolniuk, hogy az adatelérést mennyire kell korlátozni, mivel nagy a kísértés, hogy túlzottan is korlátozóak legyenek, míg az adatgazdálkodás hangsúlyozott elve, hogy az adatok mindazok rendelkezésére álljanak az adott szervezetben, akiknek azokra szükségük van. A titkosság és a rendelkezésre állás közötti helyes egyensúlyt csupán az adatgazdálkodás összehangoló szerepével lehet megvalósítani.

Created by XMLmind XSL-FO Converter.

Az auditálás (független vizsgálat) úgy határozható meg, mint egy tevékenység vizsgálata és véleménynyilvánítás a tevékenység megvalósításának minőségéről olyan személyek által, akik függetlenek a tevékenység megvalósításáért és felügyeletéért felelős személyzettől. A tevékenységek rutinszerű ellenőrzése és nyomon követése az egyéni kötelezettségek szokásos menetében nem auditálás, hanem ellenőrzés. Az adatgazdálkodással kapcsolatos auditálást az adatgazdálkodási eljárásokra, valamint az adatok minőségére és terjedelmére alkalmazzák.

Miután a technológia közelebb került a felhasználókhoz, jelentősen megnőtt a nem engedélyezett hozzáférések lehetősége. Az információ-rendszerek méretének csökkentésére jelenleg irányuló törekvések (down-sizing) azt jelentik, hogy azok a kérdések, amelyekre a belső ellenőrzési rendszer kiterjed, szorosabb irányítást és kezelést igényelnek.

Elérés

Az adatok elérését háromféle jogosultság engedélyezésével lehet ellenőrzés alatt tartani:

- olvasási jog, vagyis az a lehetőség, hogy belenézzenek az adatokba és/vagy kinyomtathassák azokat,
- olvasási és írási jog, vagyis a fentiek mellett még az adatok módosításának joga,
- adatok létrehozásának és törlésének joga, vagyis fenti a két jogon túl az a lehetőség, hogy új rekordokat hozzanak létre, és előfordulásokat töröljenek.

Egyre inkább szoftver szintű ellenintézkedéseket alkalmaznak a hozzáférési korlátozások kikényszerítésére. Valamennyi adatbázis csomag tartalmaz ilyen jellegű lehetőségeket. E mellett a CASE-eszközök, az alkalmazás-generátorok, lekérdező nyelvek és a jelentés-generáló eszközök tartalmaznak a hozzáférés ellenőrzését szolgáló lehetőségeket, hogy csupán néhányat említsünk meg. Következésképpen lényeges, hogy kialakítsák azt az eljárásmódot, amely pontosítja, milyen eszközöket alkalmaznak a felhasználói hozzáférés ellenőrzésére. Célszerű egy kockázat-elemzési és kezelési módszerhez folyamodni, amilyen pl. a CRAMM (Kockázatelemzési és Kezelési Módszertan), hogy segítsenek meghatározni a kockázatokat és megválasztani a megfelelő ellenintézkedéseket. Egyes esetekben nem elegendő csupán az adatokhoz való hozzáférést korlátozni, mivel az egyedek létének ismerete is korlátozott. E problémának két lehetséges megoldása van:

- korlátozzák ezen egyedek adat-meghatározásaihoz való hozzáférést az adatszótárban,
- több adatszótárra történő logikai szétválasztáshoz folyamodnak.

A második alternatíva a kielégítőbb megoldás, miután a bizalmas elemeket tartalmazó adatszótár valószínűleg eleve a felhatalmazott használók szűkebb körére korlátozott.

Titkosság

A titkosság a hozzáférés ellensúlya. Míg az adatgazdálkodás egyik fő célja az, hogy az adatokat hozzáférhe-tővé tegye mindazok számára, akik használni akarják azokat, a hozzáférhetőséget titkossági megfontolásokból korlátozni kell (különösen a személyi adatok tekintetében). A titkosság két megnyilvánulása merül fel:

- törvényes kötelezettségek,
- nem engedélyezett hozzáférés.

A személyes adatok védelméről szóló Adatvédelmi Törvény rendelkezései vonatkoznak az adatgyűjtés alanyaként megjelenő személy azon jogaira, hogy személyes adataihoz arra fel nem hatalmazott személyek ne férjenek hozzá. Miután egyre több adat válik hozzáférhetővé hálózatok igénybevételével és olyan új eljárások révén, mint a dokumentumok képi feldolgozása, a titkosságra vonatkozó további törvények közeli példái a Levéltári Törvény és az Európai Unió irányelve az adatbázis szerzői jogáról.

Az adattulajdonosok törvényes kötelezettségeiken túl is ellenőrzésük alatt akarják tartani az adatok titkosságára vonatkozó szabályokat. A tulajdonosok-nak együtt kell működniük az adatgazdálkodással, ha bizonyosak akarnak lenni afelől, hogy adataikhoz nem lehet hozzáférni, s következésképpen a titkosságot megtörni olyan rendszerek vagy rendszerszoftverek révén, például lekérdező nyelvekkel, amelyeknek nincs oka a hozzáférésre.

Created by XMLmind XSL-FO Converter.

Az adatszótárt, mint a metaadatok központi tárát, léte alkalmas hellyé teszi az Adatvédelmi Törvényben előírt sajátos kötelezettségek rögzítésére egyes adatok vonatkozásában. A hozzáférési jogokat, a titkosságra gondolva, különösen ellenőrizni kell.

5.7. 1.5.7. A hozzáférési jogok megvalósítása

A különböző operációsrendszerek, hálózatkezelő szoftverek illetve adatbázis-kezelők részletesen foglalkoznak a biztonsági kérdések megvalósításával. A kérdés megoldását több szinten valósítják meg:

- belépési jogok,
- könyvtár/állomány hozzáférési jogok,
- alkalmazás szintű jogok, pl. adatbázis használati jogok.

Belépés a rendszerbe

Bármilyen fejlettebb operációs rendszert használunk, mindegyik biztonsági gátat alkalmaz. Ennek első jele, hogy a belépéskor azonosítania kell magát a felhasználónak. A rendszer belépő nevet (Login name, Username) és jelszót (Password) kér. Ennek azonosítása révén már meghatározott, hogy az újonnan beléptetett felhasználónak milyen jogosultságai vannak. Ezt a rendszerek egy szigorúan védett belső adatbázisban tárolják, melynek beállítására, módosítására, kezelésére csak a rendszergazdának (Administrator, Sysop, Root) van jogosultsága. Az, hogy kinek milyen jogot adjon ki, nem az ő hatásköre, hanem a szervezet vezetőjének. A rendszergazda csak végrehajtója a feladatnak!

A belépő név általában nem változtatható meg, legfeljebb új felhasználó vehető fel a rendszerbe. A jelszót az esetek többségében az első belépéskor a felhasználónak meg kell változtatni. Ezzel érhető el, hogy ezt követően már csak ő tudja használni a saját jogait. Kezd kialakulni már a hazai gyakorlatban, hogy a bejelentkezés során a környéken dolgozók udvariasan elforduljanak, ezzel is biztosítva a belépés titkosságát.

Nagyon fontos, hogy a munka hosszabb megszakítása, a rendszer magára hagyása esetén a bejelentkezett felhasználó függessze fel a folyamatot (Logoff, Logout). Ezzel lehet elkerülni, hogy távollétében valaki az ő jogosultságait használva férjen a rendszerhez („Tiszta asztal, tiszta képernyő” elv)..

Milyen jogok kötődhetnek a bejelentkezés után a felhasználóhoz?

A **UNIX** rendszer alapvetően háromféle jogot ismer:

- R (Read) olvasási jog,
- W (Write) írási jog,
- X (eXecute) végrehajtás (futási) jog.

Ez kiadható mappára (~könyvtár) és állományra is. A jogok mellett fontos, ki az adott joghordozó tulajdonosa. Létezik még ugyanis egy 4. lehetőség is, a S (Set-userID), a felhasználói azonosító állítása. Ezzel lehetőséget kap a felhasználó, hogy a tulajdonos jogaival használja az állományt.

A **Novell Netware** rendszerben nyolcféle jog állítható be három szinten:

- könyvtárakra vonatkozó kezelői jogok (trustee rights),
- könyvtárakra vonatkozó elérési jogok (directory rights),
- az előző kettő eredőjeként létrejövő effektív jogok (effective rights).

Az első két esetben beállítható jogok:

- R (Read) csak olvasási lehetőség,
- W (Write) olvasási és írási jog,
- O (Open) megnyitási jog,

Created by XMLmind XSL-FO Converter.

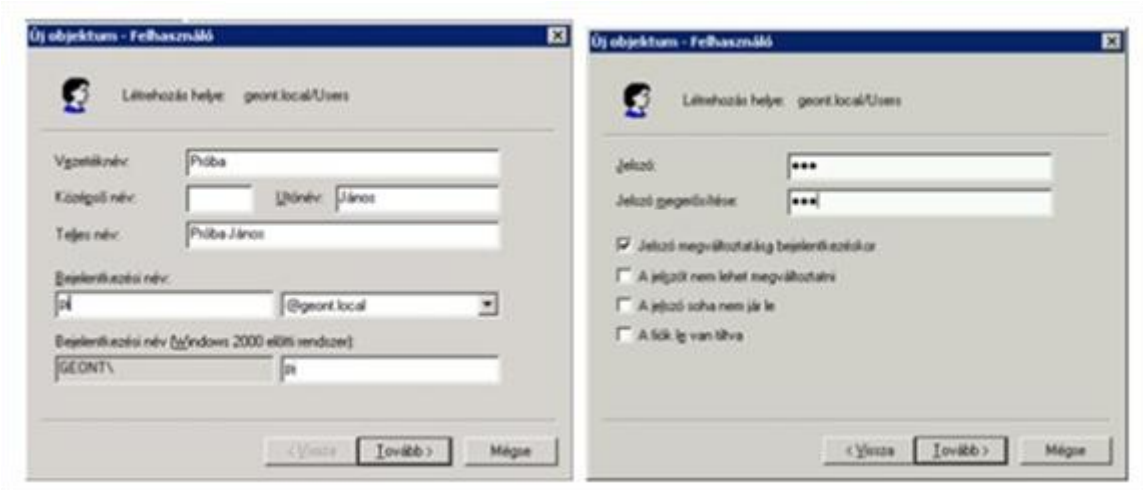
- C (Create) új állomány létrehozási joga,
- D (Delete) törlési jog,
- P (Parental) kezelői jogok átadási joga,
- M (Modify) az attribútumok megváltoztatási joga.

Az egyedileg kialakított jogokat a védelmi egyenlőség elvén (security equivalences) hozzárendelhetjük más felhasználókhoz vagy csoportokhoz is.

Az effektív jogok a könyvtár elérési jogok és kezelői jogok összevetéséből alakulnak ki. Mindkét jogcsoport kifejezhető egy-egy 8 elemű vektorként [RWOCDPM]. Természetes konkrét esetekben az egyes elemek lehetnek üresek is (pl. [R−OC− − −]). A két ‘vektor’ elemei között ‘és’ (and) kapcsolatot kell kialakítani, s abból származik az effektív jog.

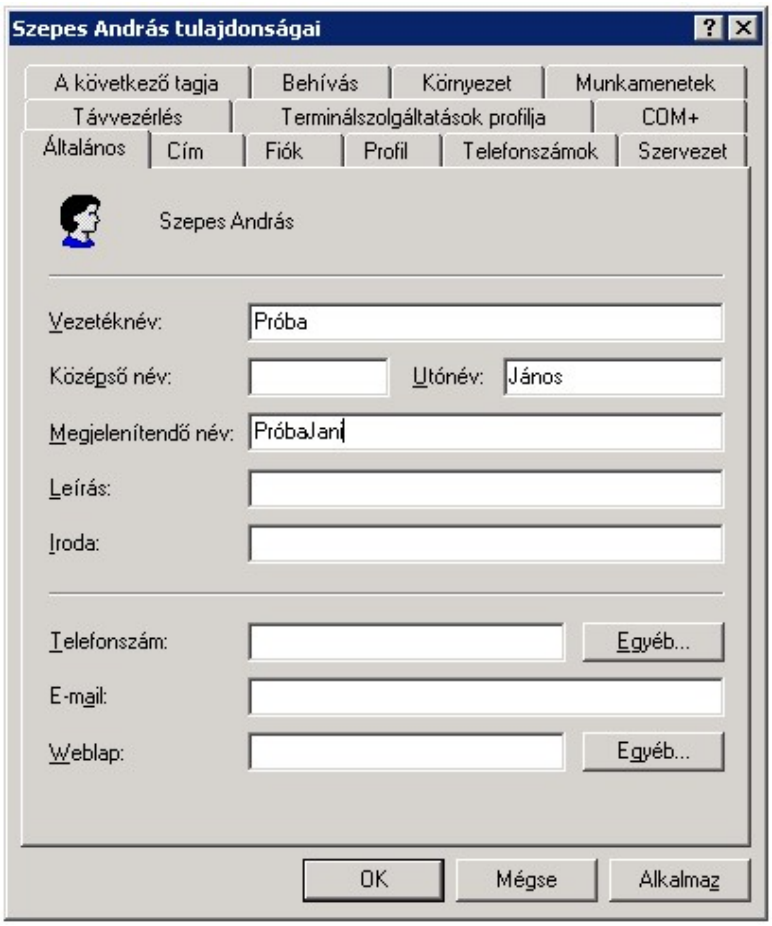
A Windows rendszerben hasonlóképpen lehet személyekhez, könyvtárakhoz és állományokhoz jogokat kötni (egyes esetekben meghajtókhoz – drive-okhoz – is). Az új felhasználó létrehozásának menete a 1-10. ábrán látható. Itt adjuk meg a felhasználói nevet és jelszót, és intézkedünk a jelszó hatályáról.

A Felhasználó tulajdonságlapján (kartonon) megadhatjuk a felhasználó működési környezetének paramétereit, a csoporttagságokat, és az egyéb személyi adatokat.



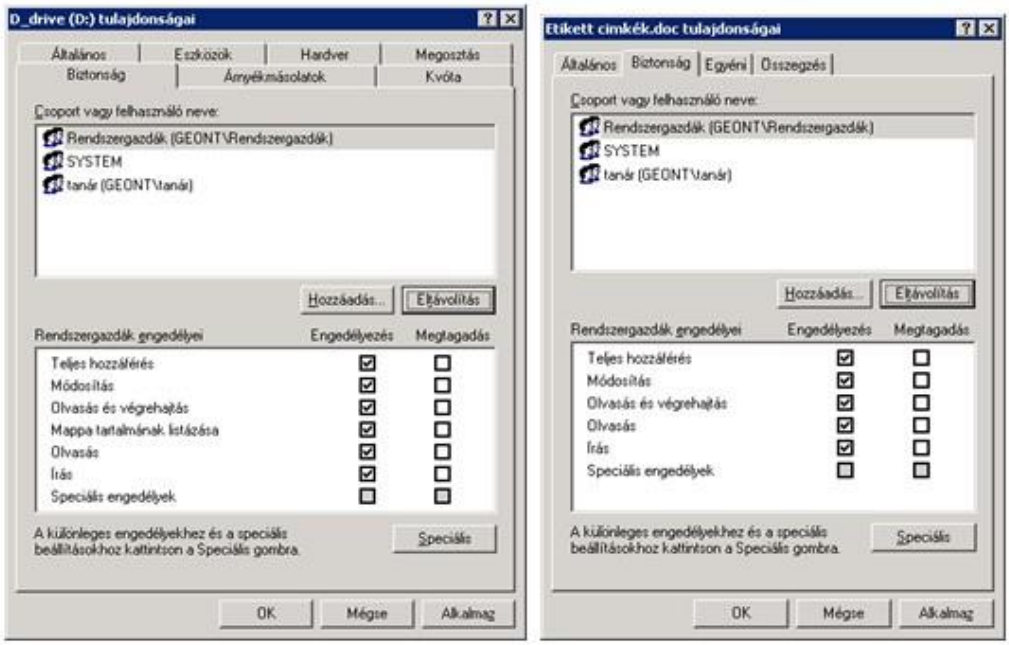
1-10. ábra Személyi lap

Created by XMLmind XSL-FO Converter.



1-11. ábra Felhasználó tulajdonságai

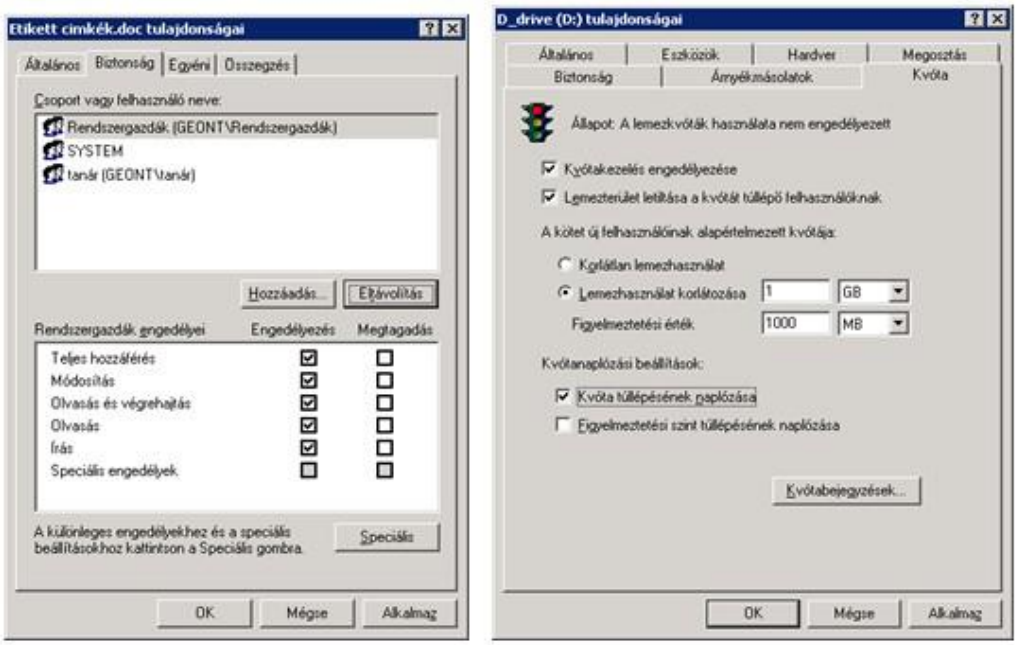
A könyvtárakhoz és állományokhoz való hozzáférést külön állíthatjuk be (1-12. ábra).



1 12. ábra Mappa és Állomány beállítása

Külső felhasználó a hálózatról csak akkor érheti el valamely meghajtó (drive) vagy mappa (könyvtár) tartalmát, ha azt előzőleg felkínáltuk megosztott felhasználásra (Sharing). Itt is megadható, hogy ki és milyen joggal éri el az állományainkat. (1-13. ábra)

Created by XMLmind XSL-FO Converter.



1 13. ábra Megosztások

Külön kategóriát jelentek egyes felhasználói rendszerek, melyekhez külön elérési szabályokat rendelnek. Ez különösen az adatbázisokat kezelő rendszerekre érvényes, így védik azok adattartalmát. Az ugyanis, hogy beléptünk egy hálózati rendszerbe, még nem jelenti az ott található alkalmazások, adatbázisok szabad felhasználhatóságát. Ezt a rendszerek, illetve az adatbázis-kezelők (DBMS – Database Management System) külön is szabályozhatják, illetve szabályozzák is. Itt is létezik egy beléptetési procedúra, ahol szintén felhasználói névvel és jelszóval kell azonosítani magunkat. A két név és jelszó nem feltétlenül egyezik egymással, sőt célszerű eltérő neveket használni.

1-14. ábra Belépés egy adatbázisba

Az 1-14. ábra egy honlap-rendszer egyik adatbázisába való bejelentkezést mutatja. Hiába jogosult felhasználó valaki a rendszer elérésekor, ezt a területet csak arra felhatalmazottak érhetik el szerkesztői joggal.

De sorolhatnánk a példákat pl. a földhivatali TAKAROS rendszer, vagy a eMagyarország stb. esetében.

Created by XMLmind XSL-FO Converter.

6. 1.6 Röviden a Linux rendszerről

6.1. 1.6.1. Előzmények

A Linux operációs rendszer helyének és szerepének megértéséhez érdemes áttekinteni kialakulásának történetét.

Az 1960-as években az AT&T Bell Labs a General Electric (GE) és az MIT közösen vezetett egy projektet, hogy az egyre gyorsuló számítógépekre kidolgozzanak egy több felhasználó egyidejű hozzáférését biztosító, időosztásos operációs rendszert. A projectet angol elnevezése után (Multiplexed Information and Computing Service) MULTICS néven emlegették. 1969-ben a Bell megvonta a project támogatását a közvetlenül felhasználható eredmények hiányára, és a használt GE számítógép magas üzemeltetési költségeire hivatkozva. Ekkor az MIT kutatói számítógép nélkül maradtak nem csak a kutatási témájuk tekintetében, de a GE-645 gépre írt Space Travel [1] nevű játékukat sem volt hol futtatni. A játék programozója Ken Thompson végül talált egy kevéssé használt PDP-7 számítógépet a Bell laboratóriumban, és elkezdték több lépésben átírni a játékot erre a jóval kisebb teljesítményű gépre. A játék igényeinek kiszolgálására, de a MULTICS project tapasztalatai alapján, operációs rendszert is írtak a PDP-7-re. 1970 nyarán jóváhagytak a projekt számára egy PDP-11 beszerzését ($65.000-ért), amely gép az operációs rendszernek 16kB, a felhasználói programoknak 8kB memóriát biztosított. A gépen csak egy felhasználó dolgozhatott így az eredeti project nevét kissé gúnyolva Uniplexed Information and Computing Service-nek (UNICS) nevezték. Később kapta a UNIX nevet, amikor felruházták több-felhasználós funkciókkal is a rendszert. Ez az eredeti név rövidült formája, az egyes betűinek nincs az eredeti névtől eltérő jelentése.

A UNIX hasznosságát bizonyítandó különböző szövegszerkesztési funkciókkal kezdték ellátni. A PDP-7-ről portolt roff nevű szöveg formázó alkalmazásnak nagy sikere lett a Bell szabadalmi osztályán. Ez tovább erősítette a projekt támogatását, egymás után születtek az újabb verziók. 1975-ben a hatodik verziónak már a jelentős része C nyelven íródott, ami nem is meglepő, hiszen a projekt oszlopos tagja volt Dennis Ritchie, aki korábban, szintén az AT&T-nél, megalkotta ezt a programnyelvet. A V6 volt az első verzió, amely megjelenhetett a Bell-en kívül is. Amikor a rendszer dokumentációját az alkotók nyomdába kívánták adni, és nem találtak megfelelő eszközöket erre, a szöveg manipuláló programok újabb széles körét hozták létre. Ezek jelentőségére visszatérünk ’ A kód újrahasznosítás formái [19] ’ fejezetben.

A nagyon szigorú amerikai tröszt ellenes törtvények miatt a Bell nem árusíthatta pénzért a kutatási eredményekből származó termékeit, ezért a UNIX-ot ingyen bocsátották az amerikai egyetemek rendelkezésére. A C nyelvnek köszönhető egyszerű hordozhatósága, viszonylag kis mérete, áttekinthetősége, és praktikussága miatt hamar népszerű lett a diákok körében. Ahogy a ’80-as évek elejére ezek a diákok fontos pozíciókat töltöttek be nagyvállalatoknál, úgy vitték magukkal a UNIX-ot is újabb és újabb környezetekbe. Az ekkoriban induló hardver gyártók is mind kijöttek a maguk verziójával (Digital – Tru64, HP – HP-UX, Sun – Solaris) és az IBM is beszállt az AIX-al. A kis gépeken az egyik legnépszerűbb Unix variáns a Berkeley Egyetemen fejlesztett BSD lett.

6.2. 1.6.2. Open Source, Free Software

A ’70 években a hardverek gyors fejlődése mellett a szoftverek sok tekintetben kísérleti stádiumban voltak, gyakran egy-egy géptípusra egy adott problémára írt egyedi megoldásokként jelentek meg. A programozók előtt végtelen mennyiségű feladat állt, a ma iskolában tanított algoritmusok és adatstruktúrák még nagyrészt kidolgozásra vártak. A legtöbb program vagy az amerikai egyetemeken készült, vagy velük együttműködésben. Ilyen körülmények között természetes volt, hogy a programozók megosztották egymással nem csak az elkészült programokat, de azok forrás kódját is, így tanulva egymástól.

Szintén ez volt az a korszak, amikor az amerikai közvélemény is kezdett nagy figyelmet fordítani a számítástechnikára, és túlfűtött várakozásokkal tekintettek a mesterséges intelligencia fejlődésére. A jelentős kutatási támogatásoknak köszönhetően az MIT mesterséges intelligencia laborja lett a kor legjelentősebb szoftver kutatóinak a gyűjtő helye. A laborban a legtöbb programot Lisp nyelven írták. Mivel ennek a nyelvnek eltérőek az elméleti alapjai a ma is használatos számítógépek architektúrája által jobban támogatott nyelvekétől (pl. C), ezért a laborban újfajta hardver fejlesztésébe is belefogtak. Ebből született a Lisp Machine, ami hardver

[1] Sok forrás hibásan a Spacewar nevű játékra hivatkozik. A Spacewar szintén ebben az időben az MIT-n írt és a Space Travelnél népszerűbb játék volt. A Spacewar is igen fontos szerepet tölt be a számítástechnika történetében (igényeinek kiszolgálására építették az első képernyőt és trackball-t is), de a UNIX történetében nem volt szerepe. Míg Spacewar-ban két játékos igyekszik a másik űrhajóját kilőni, addig a Space Travel-ben a játékos a naprendszer bolygóit látogathatja végig, és azokon megkísérelhet leszállni.

Created by XMLmind XSL-FO Converter.

szinten tudta gyorsítani ennek a nyelvnek a futtatását. Az említett felfokozott várakozásokkal teli közhangulatban a labor két vezetője, Russell Noftsker és Richard Greenblatt 1979-ben úgy döntött, hogy eredményeiket vállalkozás formájába szervezik ki. A két vezető azonban nem tudott megegyezni a vállalkozás formájáról. Noftsker kockázati tőke bevonásával egy hagyományos piac orientált céget alapított Symbolics néven. Greenblatt az Apple mintájára, az MIT-s közösségi hangulatot megőrizve, tőkebevonás nélkül próbálta elindítani Lisp Machines Inc. néven vállalkozását [2] . Mindkét cég a labor munkatársaiból válogatta alkalmazottait, összesen két embert hagyva az egyetemen. Egyikük volt Richard Stallman, akit mélységesen elkeserített a labor ilyetén szétverése, és ezért Noftskert okolta. Két éven át támogatta Greenblatték cégét azzal, hogy a Symbolics által kihozott új funkciókat a specifikációk alapján újra írta, és átadta a Lisp Machines-nek.

Ezen tapasztalataira alapozva jutott a szabad szoftver fontosságának gondolatára, és 1984-ben kilépve az MIT-től megalapította a GNU Projectet, melynek mai napig tartó célja egy Unix-jellegű teljesen szabad operációs rendszer elkészítése. Itt érdemes megjegyezni, hogy a szabad szofter már a kezdetekben sem jelentette minden esetben az ingyenes szoftvert [3] . Stallman például az általa írt, és első GNU programnak számító Emacs szövegszerkesztőt $150-ért küldte el mágnesszalagon az azt igénylőknek. Igaz a pénzt nem a programért, hanem a terjesztésért kérte, de az azon keletkező haszonból tartotta fenn magát. A project következő jelentős eredménye is Stallman nevéhez fűződik a 'gcc' nevű platform és nyelv független fordítóprogram kidolgozásával.

A GNU project jól haladt a felhasználói programok szabad felhasználású változatainak elkészítésével és összegyűjtésével, azonban a teljes operációs rendszerhez nélkülözhetetlen kernel fejlesztése (Mach) nem a terveknek megfelelően alakult. Így szükség volt egy kernelre...

6.3. 1.6.3. Linus Torvalds

Linus Torvalds finn egyetemista szeretett volna az új 80386-os számítógépéről mint terminálról bejelentkezni az egyetemi Unix rendszerre. Mivel a géptípus még új volt az akkor elérhető Unix jellegű rendszerek egyike sem használta ki a gép adottságait. Úgy döntött, hogy az alapoktól kezdi projectjét, csak a gcc fordítóra hagyatkozva. Csak amikor elkészült ismerte fel, hogy egy teljes kernel alapjait megírta. Ezt 1991 augusztus 25.-én jelentette be a Usenet nevű levelező hálózaton. Az akkoriban használható többi nyílt forráskódú kernel (pl. MINIX vagy BSD) fejlesztése hiába tartott sokkal előrébb, azok vagy a licenszelésük vagy egyéb jogi okokból [4] nem feleltek meg a GNU projectnek. Mivel Torvalds 1992-től (0.99 kernel) a GNU szabad felhasználást támogató licensze (GPL) alatt adta ki a kódot, a Linux kernel lett a GNU szoftverekkel leggyakrabban használt operációs rendszer mag, amiből kialakult a GNU/Linux operációs rendszer. A Linux kernel fejlődése azóta is töretlen. Idő közben a ma használatos hardverek legnagyobb részéhez létezik illesztő programja (driver), ezért óriási előnyre tett szert minden más hasonló fejlesztéshez képest, és a jövőben is megkerülhetetlen szereplője marad a számítástechnikának. A Linux másik kiemelkedő tulajdonsága a hordozhatósága. Ma már létezik Linux gyufás skatulya méretű beágyazott rendszertől a mobil telefonokon át a több száz processzoros szuperszámítógépekig mindenféle architektúrára.

6.4. 1.6.4. A Unix filozófia

A Unix rendszerekre általánosan jellemző fejlesztési alapelvek gyűjteményét Unix filozófia néven szokás emlegetni. Míg ez sohasem volt egy egységes, írásba foglalt irányelv, Mike Gancarz 9 pontját széles körben elfogadják a Unix fejlesztők és felhasználók egyaránt:

1. Ami egyszerű az gyönyörű [5] .
2. Minden program végezzen egy feladatot jól.
3. A prototípus szülessen meg mielőbb.
4. A hordozhatóság a hatékonyság felett áll.

[2] Mivel a bő évtizedes várakozás után az átlagemberek még mindig nem láttak az utcán két lábon járó, beszélő robotokat, elfordultak a mesterséges intelligenciától, és a kutatási pénzek is elapadtak. A nem speciális hardverek is egyre gyorsabbak lettek, és ez mindkét cég gyors bukásához vezetett. Ennek ellenére ez az időszak óriási eredményeket hozott a kereső algoritmusok, a programnyelvek és a kezelői felületek terén is.
[3] Free as freedom, not as free beer. Szabad mint szabadság, nem pedig szabad fogyasztás (ingyen sör).
[4] Ebben az időben az AT&T szabadalmi pereket indított különböző UNIX technológiák használói ellen mivel úgy látták, hogy az általuk közreadott eredményekből mások jelentős piaci haszonra tesznek szert. A per érintette a BSD-t is.
[5] "Small is beautiful", parafrázis az amerikai "Big is beautiful" mondásra.

Created by XMLmind XSL-FO Converter.

5. Az adatok tárolása egyszerű szöveges fájlokban.

6. A szoftver újrahasznosítás kihasználása.

7. A shell script-ek használata az újrahasznosítás és a hordozhatóság érdekében.

8. A rögzített felhasználói felületek kerülése.

9. Minden program legyen egy filter.

Ennek a filozófiának direkt hatásai vannak a Unix rendszerek üzemeltetésére. A következőkben ezen hatások némelyikét vizsgáljuk meg közelebbről, a Unix rendszert szándékosan mint szerver környezetet tekintve.

A [1] elv megjelenik például a hagyományos Unix alkalmazások telepítésében, ami legtöbbször csak egy összecsomagolt állomány (tarball) kibontásából áll. A program minden függőségét, konfigurációs paraméterét és adatát egy könyvtárban tárolja. Így nem csak a napi mentése egyszerű, de ugyanígy lehet egy futó környezetből teszt példányt klóónozni, vagy a mentés helyreállíthatóságát ellenőrizni. Mivel az alkalmazások kisebb részekből állnak [2], az adataikat a rendszergazdák által értelmezhető formában tárolják [5] ezért könnyű őket az adott környezethez igazítani utólag. Például, ha egy éjszakai adatfeldolgozási folyamat rendszeresen hibára fut mert kevés a szabad hely a munka köteten, akkor a rendszergazdáknak lehetőségük van a folyamatot végző részek közé egy shell script-et [7] illeszteni, ami a kritikus rész előtt ellenőrzi a rendelkezésre álló helyet, igény szerint becsatol újabb diszk területet, és csak annak sikeres befejezése után engedi tovább az eredeti kódot. Ilyen szintű módosításhoz nem kell igénybe venni fejlesztői támogatást, ami körülményes és költséges is lehet.

6.5. 1.6.5. A kód újrahasznosítás formái

Egy vállalati csoportban dolgozó programozó termelékenysége, a tesztelt és dokumentált kész szoftver teljes ráfordításából visszaszámolva, 3 sor óránként. Ez az érték nagyjából változatlan a ’60 évek óta. Egy közepes funkcionalitással rendelkező segédprogram 1000-5000 sor kódot tartalmaz, míg egy mai operációs rendszer nem ritkán 50 millió programsor. Ilyen arányok mellett a kód újrahasznosítás nem filozófia kérdése, hanem a versenyben maradás feltétele.

A C nyelv - amely alap fejlesztési nyelve a mai operációs rendszereknek is – tervezésekor is számoltak azzal az igénnyel, hogy programozók egymás megoldásait a működés részleteinek megismerése nélkül fel tudják használni. A C nyelv specifikációja azonban csak forrás kód szintű kapcsolódási lehetőséget tartalmaz, a bináris formát az implementációra bízza [6] . Mivel a lefordított és mindenki számára hozzáférhető kód nem tartalmazza az újrahasznosításhoz szükséges információkat, kialakult az együttműködésnek egy másik módja. Az előző fejezet [9] pontja arra utal, hogy a Unix operációs rendszer minden futó alkalmazás számára biztosítja a standard input és output eszközöket (kicsit leegyszerűsítve a billentyűzetet és a képernyőt) valamint azt a mechanizmust amin keresztül az egyik program kimenete a másik program bemenetével összekapcsolható. Ezt nevezzük pipe-nak, a jele pedig a | (ASCII 124) karakter.

$ cat jelentkezok.txt | wc | mail -s "Mai jelenkezok" tansz@geo.info.hu

A fenti parancssor 3 programot [7] kapcsol össze. Az egyes parancsok jelentése:

cat, a paraméterként kapott fájl tartalmát kiírja a standard outputra;

wc, az inputján kapott szövegben megszámolja a sorokat és az eredményt kiírja a kimenetére;

mail, az inputjára érkező szöveget elküldi az utolsó paraméterként megadott címre, a –s kapcsolóval jelzett tárggyal.

A három parancs kapcsolatából felépülő sor jelentése: vegyed a jelentkezok.txt fájl tartalmát, számold meg benne a sorokat és az eredményt küldjed el e-mail-ben.

[6] Egy fejlesztő készíthet kód könyvtárat (library) amelynek felhasználásához egy másik fejlesztő számára a lefordított library kód mellett szükség van az úgynevezett header fájlra is. A header fájl a C forráskódnak az a része, ami csak a függvények nevét, visszatérési értékét és paraméter listáját tartalmazza, de a megvalósításukat nem. Ez tehát egy interfész definíció a kódkönyvtárhoz. A lefordított könyvtár bináris kódja nem tartalmazza szabványos módon ezeket az interfész leírásokat így azok önmagukban nem használhatóak.
[7] Ebben az esetben a (külső) parancs és a program egymás szinonimái.

Created by XMLmind XSL-FO Converter.

A pipe-ban résztvevő programok egyetlen feltételezéssel élnek az előttük és mögöttük álló tagokról: mindegyikük ASCII karaktereket képes fogadni és küldeni. Így válnak a korábbi fejezetben említett szövegszerkesztést támogató parancsok egy hatékony kód újrahasznosítási mechanizmus alkotóelemeivé.

A bináris kód interfész hiánya régóta foglalkoztatja a programozókat. A probléma első széles körben használt változatát a Microsoft dolgozta ki 1993-ban és Component Object Model (COM) [8] néven vált ismertté. Egy COM objektum (bináris formájában pl. egy dll) képes magával vinni a benne található függvények leírását, és azokat elérhetővé tudja tenni külső, tőle független programok számára. Ha egy COM objektum implementálja a megfelelő interfészeket, akkor tudása elérhetővé válik script nyelvekből is (pl. VBScript vagy JScript). Ezzel elméletben egy hatékony eszközt kínálhat a komponensek új módokon történő felhasználására. Ez azonban nem kevés plusz erőforrást igényel a kód könyvtár írójától, bonyolult objektum regisztrációs mechanizmust az operációs rendszertől, összetett futtatási környezetet, és nem mellékesen egy majdnem programozói képességekkel bíró, az egészet átlátó rendszergazdát.

A különböző programok egymás közti szöveges üzenet küldése egy alacsony hatásfokú megoldás (lásd. [4]), ami nem garantál szoros és megbízható kapcsolatot a komponensek között, a technológiai fejlettség négy évtizeddel ezelőtti szintjét jelenti. Ezzel szemben a COM specifikáció az azóta eltelt idő tapasztalatainak tudományos szintű összesítése, és a problémára adott válasz jól dokumentált megvalósítása.

Mégis, minden elfogultság nélkül, kijelenthetjük, hogy a mindennapi rendszerüzemeltetés során az említett Unix mechanizmusok, a Unix filozófia többi elemének konzekvens betartása mellett, hatékony eszközt biztosítanak a felmerülő problémák széles körének megoldására, míg a Microsoft-os eszközök e téren elbuknak [9] . Valószínűleg ebben a rugalmasságban rejlik a Unix jellegű rendszerek töretlen népszerűsége a világ szervertermeiben.

7. 1.7. Összefoglalás

Ez a fejezet kissé hosszabbra sikerült, mint azt terveztük. Ennek sok magyarázata lehetne, de csak az állja meg a helyét, hogy igen fontos ismereteket akartunk átadni. Felvetheti az Olvasó, hogy a DOS már nem számít korszerű rendszernek, kár azzal foglalkozni. Erre csak annyit mondhatunk, hogy még mindig vannak olyan ismeretek, melyek nélkül nehéz megérteni, megértetni a rájuk épülő korszerű ismereteket. Arról már nem szólva, hogy még elég gyakori az olyan feladat, melyet könnyebb DOS platformról, illetve DOS ablakból megoldani!

Mindenesetre most, hogy már túljutott a fejezeten, próbálja ki ismereteit!

1. Milyen csoportjai vannak a szoftvereknek? (Válasz [1])
2. Mi az operációs rendszerek feladat? (Válasz [2])
3. A Windows OS milyen nagyobb szoftver részeket jelent? (Válasz [6])
4. Mutassa be a Windows felhasználói felületét! (Válasz [6])
5. Mit jelent az egységes felület a szolgáltatások terén? (Válasz [6])
6. Milyen biztonsági szolgáltatásokat találunk egy fejlett operációs rendszernél? (Válasz [10])
7. Milyen hozzáférés jogokat lehet beállítani a védelem érdekében? (Válasz [13])
8. A Linux szoftver rövid jellemzése (Válasz [17])

Irodalomjegyzék

Ádám S. : *Népszerű számítástechnikai kislexikon*, Magánkiadás , Budapest , 1988

[8] Valójában a COM sok szempontból hasonlít az Object Management Group 1991-ben megjelent CORBA specifikációjára.
[9] A teljesség kedvéért meg kell említeni, hogy a grafikus felülettel rendelkező felhasználói (desktop) rendszerek esetében, ahol a Microsoft rendszerek gyökerei vannak, a COM jellegű modern megoldások élveznek előnyt.

Created by XMLmind XSL-FO Converter.

Benkő és tsai. (1995) : *Amit a Windows Xp-ről tudni érdemes!* , BÉDA Books Kiadó Kft. , Budapest , 2001

Csépai J. : *A számítástechnika alapjai* , Műszaki Könyvkiadó , Budapest , 1985

http://miau.gau.hu/szgep/szgep1tj.html (digitális jegyzet, 2010. 03.)

http://www.muszeroldal.hu/assistance/szamitastechnikai_kislexikon.html

Kovács Tibor és tsai : *Mit kell tudni? A PC-ről* , Computer Books , Budapest , 2003

Fritz J. : *Bevezetés az információelméletbe* , Tankönyvkiadó , Budapest , 1971

Rényi A. : *Napló az információelméletről* , Gondolatkiadó , Budapest , 1976

Created by XMLmind XSL-FO Converter.