Móricz Attila

DOS alapismeretek II.

LSI
SZÁMALK

Móricz Attila

DOS Alapismeretek II.

Lektorálta:
Dr. Kiss Jenő
Főiskolai adjunktus

Nyitott rendszerű képzés - távoktatás -

oktatási segédlete * Tankönyv

LSI Oktatóközpont

A könyvben szereplő ismeretek az MS-DOS operációs rendszer minden verziójához alaklmazhatóak.

Nyelvi lektor: Hérincs Judit tanító

ISBN 963 577 089 8 Ő.

963 577 091 X II.

Kiadó: **LSI Oktatóközpont**
Felelõs vezetõ: **Dr. Kovács Magda**
Témafelelõs: **Flier István**

LIGATURA KFT - VÁCI ÁFÉSZ NYOMDA
Munkaszám: 187-94

Tartalomjegyzék

Tartalomjegyzék

Mottó:

A számítógépet saját céljainkra kell használni,
és nem a DOS-parancsok futtatására.
(Morati)

Tisztelt Olvasó!

Ha veszed a fáradságot, és elolvasod az előszómat, akkor megtudhatod, hogy milyen ismeretekkel gazdagodhatsz a könyv elolvasása által.

Ennek a tankönyvnek az első részében már megismerkedhettél a DOS alapvető ismereteivel - könyvtárkezelésekkel, fájlokkal, azok másolgatásával, és a különféle megjelenítési formákkal. Az a tankönyv tartalmazta azokat a minimális ismereteket, melyeket még egy szoftverüzemeltetőnek is tudnia kell, még akkor is, ha csak egy felhasználói programot kezel.

A tankönyv második részében pedig, megismerkedhetsz, a rendszerkonfigurálással, ami azt jelenti, hogy a két legfontosabb konfigurációs fájl szerkesztésén keresztül, közelebb kerülhetsz, a számítógép működésének megértéséhez. A konfigurációs fájlok készítésével jobb hatásfokkal használhatjuk ki a számítógépünket, ami elsősorban a programok gyorsabb végrehajtásában fog megnyilvánulni. Áttekintjük a DOS-ban végezhető feladatokat, a szakirodalmak használatát, és egyéb olyan témát, ami a DOS-ban elvégezhető, és hasznos a felhasználó számára. A további ismeretek szerintem csupán arra jók, hogy a témában nagyobb jártasságot szerezzünk, de a felhasználói programok használatához, a Windows vagy más rendszer használatához nem feltétlen szükségesek.

Ebben a tankönyvben is elsősorban az elvégzendő feladatok tükrében fogjuk megvizsgálni a különféle DOS parancsokat, melyiket, mikor használjuk, és mi lesz az eredménye. Több helyen hivatkozni fogok a megfelelő DOS kézikönyv használatára, melyek listáját a *könyvajánlóban* megtalálod a tankönyv végén. Ha megelégszel csupán e tankönyvből megszerzett ismeretekkel, akkor is boldogulni fogsz más rendszerekben, programokban is, mert nem az összes parancs ismerete a fontos, hanem az átfogó szemlélet, az, hogy mindig tudjuk, hol találunk segítséget. Utána már össze tudjuk szedni a szükséges anyagot az adott témához, mert a számítástechnikát teljes egészében még egy számítógéppel is nehéz lenne megtaníttatni.

A további DOS-ban vagy Windows-ban elvégezhető feladatokat konkrét hétköznapi feladatok tükrében fogjuk megvizsgálni, úgymint *programkészítési feladatok, archiválások, szövegszerkesztések, számítógépes hardverek célirányos alkalmazása, ...stb.* Ezekben a témákban a későbbiekben szintén tankönyveket fogok írni, az LSI Oktatóközpont kiadásában, melyekkel remélem tovább gazdagíthatjuk a használható tankönyvek sorát, a számítástechnika oktatásában.

Kelt.: Győr, 1994. április

Móricz Attila sk.

Számítástechnikai tankönyvíró

1. A számítógép felélesztése

A számítógépünk valójában több összekapcsolt elektronikai áramkör, ami csak akkor képes működni, ha áram alá helyezzük. Ilyenkor egy memória IC-be égetett program indul el, utasítva a processzort különféle dolgokra. Ekkor kezd feléledni a számítógépünk. Az elektronikai alkatrészek halmazából, kezd igazi számítógéppé válni. Hosszú, munkaigényes, esetleg monoton számításokat, percek, vagy másodpercek alatt végrehajt, különféle alapadatok mellett. Egy kiadvány elkészítése esetén pedig, olyan lehetőségek nyílnak meg, amelyekre számítógép nélkül képtelenek lennénk.

Mindezekre csak akkor van lehetőségünk, ha a számítógépünket beprogramozzuk az adott feladat elvégzésére, vagy egy más által megírt programot betöltünk a memóriába. Ekkor a program vezérli a számítógépet, annak minden részét felügyelete alatt tartja. Ahhoz, hogy a program futni tudjon a memóriában, használni tudja a lemezegységeket, el kell indítanunk olyan programokat, amelyek biztosítják azok legjobb működését. Talán úgy lesz a legszembetűnőbb, ha összehasonlítjuk a számítógépünk működését segédprogramokkal, majd azok nélkül. Például, ha lekérjük az egyik lemezegység könyvtárának listáját a **DIR** paranccsal, akkor az első lekérdezéskor várnunk kell addig, amíg ténylegesen beolvassa a lemezegységről az adatokat. Ha utána újra kiadjuk a parancsot, akkor két eset lehetséges:

- Ha fut a háttérben a SMARTDRV.EXE program, akkor a második lekérdezéskor már nem a lemezegységről fogja beolvasni a katalógust, hanem az átmeneti memóriából, ami természetesen sokkal gyorsabb (*lásd a 49. oldalon*).
- Ha nem fut az említett program, akkor a második lekérdezéskor is ugyanolyan lassan fogjuk megkapni a listát, mint az első esetben. Az említett program ismertetésénél még további példákkal találkozhatunk, amelyek jól megmutatják a parancs használatának gyorsító hatását.

A másik probléma, ami a számítógép memóriacímzési rendszerében keresendő, az a számítógépben lévő memória kihasználásának problémája. Ha a számítógépünkben van például 4 MB memória, az nem azt jelenti, hogy abba közel 4 megabájtnyi programot be is tölthetünk a DOS-ban. Segédprogramok használata nélkül, észre fogjuk venni, hogy csak a 640 kB alapmemóriát tudjuk használni. A felette lévő majdnem 3,5 MB memória pedig, teljesen kihasználatlan lesz. Elég nagy pazarlás ez, különösen ha még a memóriaegységek árait is figyelembe vesszük. Segédprogramokkal viszont egész jól hasznosítani tudjuk a felső memóriaterületeket is.

Összefoglalva, ha a legjobban akarjuk kihasználni a számítógépünk kapacitását, akkor nem elég bekapcsolni a számítógépet, és elindítani a programunkat. A számítógép elindulása után be kell töltenünk olyan programokat is, amelyek biztosítani fogják, hogy a lehető legjobban kihasználjuk a gépünk összes erőforrását.

Ahhoz, hogy a különféle számítógépek esetén, mindig tudjuk, hogy milyen programokat használunk, meg kell értenünk a számítógépünk működését, tudnunk kell mindig azt, hogy éppen mi zajlik benne.

1.1. A DOS közelebbi értelmezése, a DOS "magja"

Mint már tudjuk, a DOS "magját" három fájl alkotja - IO.SYS, MSDOS.SYS, COMMAND.COM -, melyek a számítógép indulásakor a memóriába töltődnek. Ez azt jelenti, ha ezek a programok betöltődtek, akkor minden ***belső parancs*** a rendelkezésünkre áll. Ha viszont a DOS külső parancsait is használni akarjuk, akkor szükségünk van egy olyan könyvtárra is, amiben ezek megtalálhatók, például egy "DOS" nevű könyvtárra. Ha ezeket a parancsokat nem akarjuk használni, akkor viszont nincs szükségünk erre a könyvtárra. Ezzel arra akarok rámutatni, hogy ha egy olyan munkaállomást állítunk üzembe, ahol például egy adott programot használó alkalmazott fog dolgozni, akkor neki csak arra van szüksége, hogy a DOS betöltése után elindítsa az általa használt programot. Ezzel a programmal dolgozik egész nap, majd a munka végeztével kikapcsolja a számítógépét.

Nem ad ki semmilyen DOS *külső parancsot* sem, így DOS könyvtárra sincs szüksége. A DOS könyvtár nem szükségszerű része a DOS-nak, csak tartalmaz olyan parancsokat, melyek segítségével elvégezhetünk olyan lemezes vagy egyéb műveleteket, amelyeket a *belső parancsokkal* nem tudnánk megoldani.

Valójában a DOS külső parancsaihoz hasonló programokat magunk is írhatunk valamilyen programozási nyelven, ami ugyanolyan jól - vagy jobban - ellátja az adott feladatot, mint a DOS könyvtárban lévő "társa". Ilyen esetben használhatjuk azt is, mellőzve a DOS külső parancsait. Olyan esetekben azonban, amikor a számítógéphez kis kapacitású merevlemez csatlakozik (például 20 MB-os), akkor sajnos rákényszerülünk, hogy letöröljünk olyan programokat a merevlemezünkről, amelyeket csak ritkán, vagy sohasem használunk. Persze ha szükségünk lesz rá, akkor egy hajlékonylemezről be tudjuk tölteni, amire a törlés előtt kimentettük. (Ezt nevezzük archíválásnak, amiről még említést találsz a tankönyvben, de részletesen az *"Archíválások"* című tankönyvemben olvashatsz róla.)

Így eljutottunk oda, hogy a számítógépünk használatbavételéhez elegendő egy olyan lemez, amiről betöltődik az a bizonyos ***három rendszerfájl***, amiből a COMMAND.COM-ot minden parancs kiadásakor el kell tudnunk érni. Ezután már használhatjuk a saját felhasználói programunkat, ami lehet, hogy egy hajlékonylemezen is elfér. Így, ha számítógépről beszélünk, akkor nem kell azonnal egy nagy merevlemezt is feltételezni hozzá, mert anélkül is nagyon jól lehet használni. Természetesen minden esetben az adott feladat dönti el, hogy mi a ***megfelelő számítógép***.

Most a további alfejezetekben át fogjuk tekinteni, hogy mi szükséges ahhoz, hogy egy újonnan vásárolt - vagy egy meglévő - merevlemezt használatba vegyünk. Iskolákban, vagy olyan helyen, ahol nem akarjuk a programokat tartalmazó merevlemezt "megbolygatni", a *"Partícionálás"* -ról, és a *"Formázás"* -ról szóló fejezetet csak olvas-

Megfelelő számítógép alatt mindig azt kell érteni, ami az adott program futtatásához elengedhetetlenül szükséges. Ez nem a szoftver minimális hardverigényét jelenti, hanem azt a kiépítettségű számítógépet, amelyiken a szoftver elfogadható módon működik. Az fogadható el, amikor alapvető funkciók azonnal végrehajtódnak, és nem kell rá perceket várnunk. Egy grafikai program esetén például a társprocesszor használata elengedhetetlen a rengeteg számítási feladat miatt.

suk el, de ne próbáljuk ki, mert jóvátehetetlen károkat tudunk okozni! Erre még az adott helyen külön is fel fogom hívni a figyelmet.

A *konfigurációs fájlok* (CONFIG.SYS, AUTOEXEC.BAT) készítésekor, tanulásakor , mindig használjunk *saját hajlékonylemezt*, és ezen hozzuk létre az adott fájlokat. A merevlemezen lévő konfigurációs fájlokat csak akkor módosítsuk, ha tartósan szükségünk lesz egy adott szolgáltatásra. Ellenkező esetben használjunk a rendszer indítására inkább hajlékonylemezt. Tarthatunk akár több tucat olyan hajlékonylemezt is, amelyen ugyanaz a DOS rendszer található, de különféle konfigurációs fájlokkal, így mindegyik más és más rendszerindulást eredményez. Az első részben ismertetett könyvtárfelépítések gyakorlására pedig, használjunk *memórialemezt*, aminek ismertetésére ebben a tankönyvben fogunk kitérni. Gyakorlásra a lehető legjobb hely, mert gyorsan elvégezhetünk olyan műveleteket, amelyekre hajlékonylemez esetében sokat kellene várnunk. A számítógép kikapcsolásakor pedig elvész a memórialemez teljes tartalma, így nem kell annak törléséről külön gondoskodnunk.

1.2. Rendszerlemez készítése (FORMAT, COPY)

Rendszerlemez készítése sok esetben szükségessé válhat:

- Olyan esetben, amikor valamilyen egyedi beállítások alapján szeretnénk indítani a számítógépünket, és nem akarjuk a merevlemezen lévő konfigurációs fájlokat megváltoztatni. (Iskolában, vagy olyan helyen, ahol többen is használnak egyszerre egy számítógépet, ott nem is szabad önkényesen megváltoztatni ezeket a fájlokat.)
- Merevlemez nélküli - csak hajlékonylemezzel rendelkező - számítógépek esetén, csak hajlékonylemezről tudjuk a DOS-t betölteni a gépünkbe.
- Ha vírusos a merevlemezünkön lévő rendszer, és az ottani rendszer betöltésével a vírust is aktiváljuk, vagy már jóvátehetetlenül kiirtotta a DOS-t a merevlemezről.
- A tanulás időszakában - amikor a különféle beállításokat próbálgatjuk ki -, ilyenkor könnyen előfordulhat, hogy "nem áll fel a rendszer", vagy betöltéskor "lefagy", mert hibásan adtunk meg egy vagy több utasítást a konfigurációs fájlokban.

Vásároljunk egy darab, vagy egy doboz (idővel úgyis szükségünk lesz rá) floppy lemezt, amit a használathoz először meg kell formáznunk a **FORMAT** paranccsal. Ahhoz, hogy ebből rendszerlemez legyen, használnunk kell a "/s" paramétert is. Ekkor a lemez megformázása után felkerül rá a három rendszerfájl is. Erről a lemezről ekkor már indíthatjuk is a számítógépünket. Mivel sem a CONFIG.SYS, sem az AUTOEXEC.BAT nem található meg rajta, így bekéri alapértelmezés szerint a dátumot és az időt, majd kiteszi az alapértelmezett készenléti jelet (**A>_**), és a kurzor villogtatásával jelzi, hogy várja a további parancsainkat.

Viszont egy "igazi" rendszerlemez még ezzel nem készült el. Ahhoz, hogy ezt a lemezt olyankor is használhassuk, amikor vírusos a rendszerünk, szükségünk lehet a következő állományokra is:

FDISK.COM, FORMAT.COM, SYS.COM, valamilyen víruskereső és -hatástalanító program(ok), esetleg a *konfigurációs fájl*ok, és az ahhoz kapcsolódó fájok, de azok nem szükségszerűek.

Ezeket a fájlokat pedig egy egyszerű **COPY** paranccsal ki tudjuk másolni a hajlékonylemezünkre. Utána - és ez nagyon fontos! - ellenőrizzük le a víruskereső programunkkal, hogy az elkészített rendszerlemezen nincs-e vírus. Sohasem lehet az ember eléggé elővigyázatos! Miután meggyőződtünk arról, hogy a víruskereső programunk nem talált vírust a lemezen, ragasszuk le az írásvédő nyílását (5¼"), vagy toljuk át a lapkát (3½"), és soha ne bántsuk ezt az egy lemezünket. Soha ne oldjuk fel az írásvédelmet, mert akkor lehetőséget adunk a vírusoknak arra, hogy megfertőzzék a rendszerlemezünket. Az pedig, hogy a víruskereső nem talált semmit, az csak azt jelenti, hogy vagy valóban nincs vírusunk, vagy a vírus frissebb keletű, mint a keresője.

1.3. A merevlemez használatbavétele

Ha egy újonnan vásárolt - vagy egy meglévő - merevlemezt használatba akarunk venni, akkor ehhez több munkálatot kell elvégeznünk rajta. Először, ami a legfontosabb, gondoljuk át, mire, és hogyan fogjuk használni a számítógépünket. Az elkövetkező műveletek olyan jellegűek lesznek, hogy azok eredményeit nem lehet később megváltoztatni, így ha mégis meg szeretnénk változtatni valamit, akkor minden programunk elveszik, ami a merevlemezünkön található. Ha nincs rajta semmi, akkor még nem késő újrakezdeni a ***partícionálást***, de utána már véglegesnek tekinthetjük a döntésünket. (*Partícionálás*, főbb területekre osztás, lásd a 9. oldalon)

Valójában a probléma egy kezdő számára nem is olyan nagy, csak akkor merül fel igazán, ha többféle rendszert is használni szeretnénk a számítógépen.

Például előfordulhat, hogy egy Novell hálózatban egy második - vagy sokadik - szervert szeretnénk üzembe állítani, abból a célból, hogy gyakoroljuk rajta a rendszergazdai feladatokat. Olyankor is szükség lehet egy ilyen szerverre, ha rendszergazdaként olyan dolgokat szeretnénk kipróbálni a szerveren, amelyek esetleg a rendszer összeomlásához vezethetnek. Az ilyen próbálkozásokat nem tanácsos egy olyan szerveren kipróbálni, amelyiken a cég napi munkálatai folynak. Tételezzük föl, hogy a merevlemez további területeit megsokszorozzuk az MS-DOS 6 rendszer **Double Space** programjával. A használt merevlemez pedig legyen például 80 MB-os. Ez azt hiszem elegendően kicsi ahhoz, hogy a kedves olvasó ne higgye el, hogy milyen sok feladat ellátására lesz alkalmas ez a merevlemez. Általában akkor szoktunk a segédprogramokhoz fordulni, amikor helyhiány van a merevlemezünkön. (És kérdem én, mikor nincs?) A Novell hálózatról részletesen a "*Novell hálózati alapismeretek I. és II.*" című tankönyveimben olvashattok. Most nézzük végig egy példán keresztül, hogyan is zajlik egy ilyen munka, és milyen előnyökkel jár. (A rendes, jól áttekinthető munka mindig meghozza a gyümölcsét, például akkor is, ha valamit változtatni kell rajta. Fontos, hogy minden megfelelően dokumentálva legyen és akkor nem érhet kár bennünket.)

Ilyenkor a következő felosztást célszerű megtenni:

20 MB: **Novell partícióra** fenntartva.

5 MB: **DOS partíció: C:>_**

100% tömörségű, azaz tömörítetlen merevlemez a rendszerfájloknak, és egyéb célokra fenntartva.

55 MB: **DOS partíció: D:>_**

Ezen: 25 MB: **DOS partíció: E:>_**

ami az MS-DOS 6 -ban található Double Space-szel 1:10 -es becslési arányban tömörítve van, így a becsült kapacitása 250 MB.

és: 30 MB: **DOS partíció: F:>_**

ami az MS-DOS 6 -ban található Double Space-szel 1:3 -as becslési arányban tömörítve van, így a becsült kapacitása 90 MB.

A "D:" meghajtón valójában csak két rejtett, rendszerfájl található, ami a két tömörített meghajtót tartalmazza. Több szabad hely nincs rajta, így ezt a meghajtót ne is tegyük soha aktuálissá. (A legideálisabb az lenne, ha "rejtetté" tennénk, hogy még akkor se tudjuk kiválasztani, ha akarjuk.)

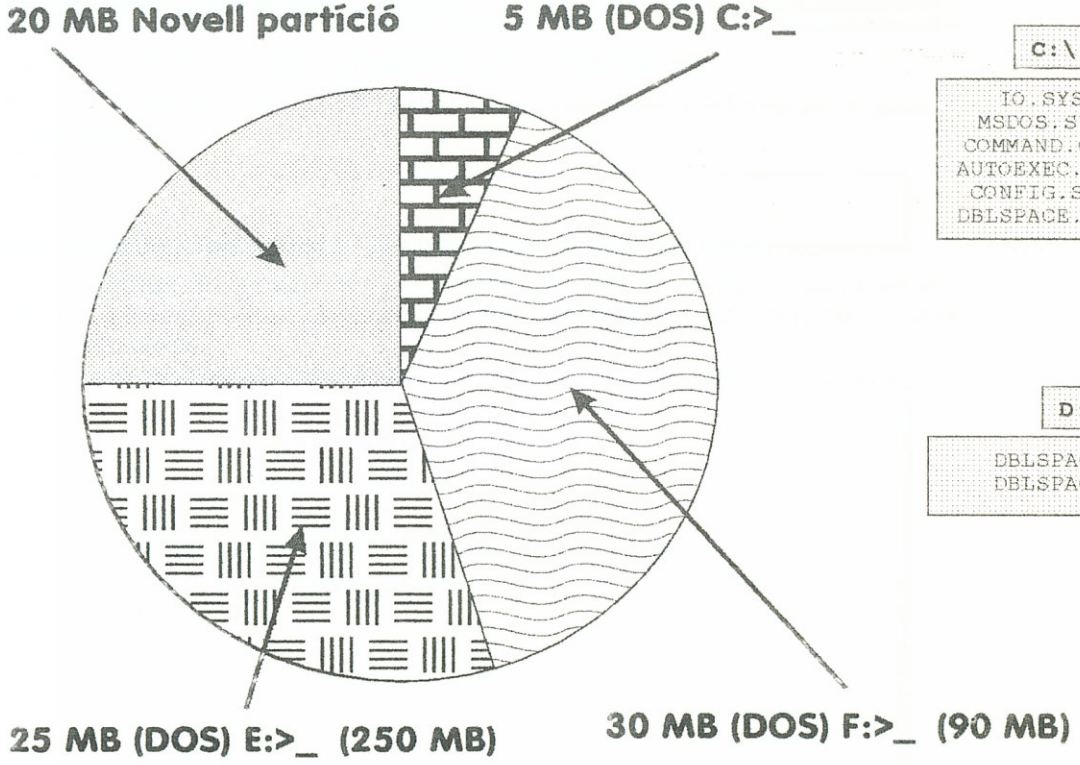
Valójában a következő eredményre jutottunk:

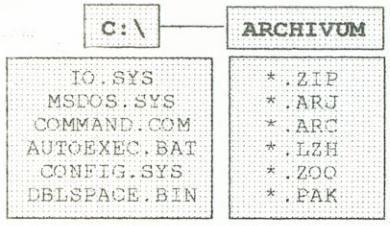
- Van egy 20 MB-os Novell partíciónk, amire felinstallálhatunk egy szervert, aminek a kapacitása csak 20 MB. Ilyenkor nem tudjuk használni a többi 60 MB lemezterületet. (Viszont nincs is rá szükségünk.)
- Van a DOS-ban egy 5 MB-os tömörítetlen meghajtónk, amire a DOS-t telepíthetjük.
- Van egy 250 MB-osra becsült (!) kapacitású meghajtónk, amire olyan fájlokat másolhatunk, amelyek jól tömöríthetők. Ilyenek például a dBase-fájlok (*.DBF), a Windows bittérképes típusú grafikus állományai (*.BMP), ... *stb.* Ha olyan állományokat másolunk a tömörített meghajtóra, ami nem tömöríthető legalább 1:10 -es arányban, akkor csökkenni fog a meghajtónk becsült kapacitása.
- Van egy 90 MB-osra becsült (!) kapacitású meghajtónk, ami hasonlít az előző megoldáshoz, de itt már olyan állományokat tartunk, amelyek csak körül-belül az egyharmadukra tömöríthetők.

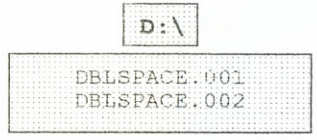
Σ Összesen a DOS-ban 345 MB-nyi programot vehetünk fel a 80 MB-os merevlemezünkre, ha az említett eljárást követjük, és még 20 MB marad a Novell szervernek is, ami a DOS-ból nem érhető el. Ha hasonló módon megtervezzük a leendő merevlemezünk felhasználását, akkor sok kellemetlenségtől, és pluszmunkától kímélhetjük meg magunkat a későbbiek folyamán.

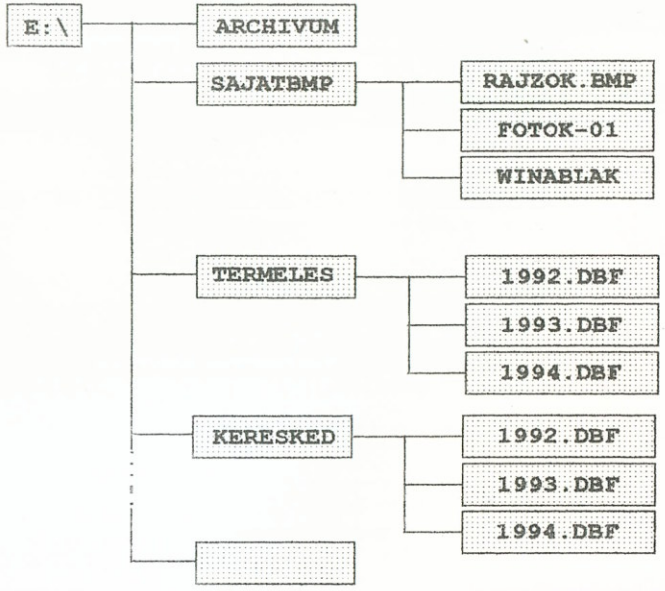
Most nézzük meg hogyan kell kinéznie egy igazi tervnek, ami a merevlemezünk további hasznosítására készült. Soha ne sajnáljuk az időt a tervezésre, és akkor nem kell soha kétszer dolgoznunk egy feladaton. Mindig rajzoljuk le, írjuk le mit akarunk megvalósítani, és vizuálisan sokkal jobban át fogjuk tudni tekinteni a tervet, mintha az egész csak a fejünkben lenne meg.

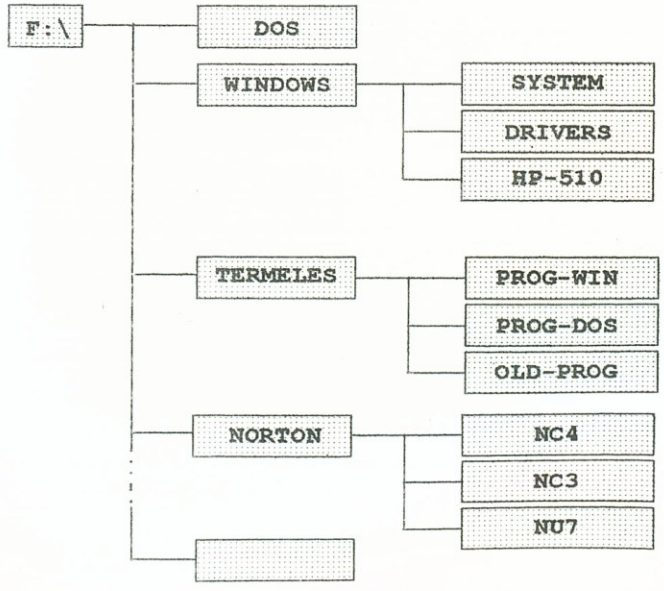
A merevlemez felosztási terve:











	20 MB	Novell partíció
C:	5 MB	DOS rendszerfájlok
		(D: 55 MB tömörített meghajtóknak)
E:	25 MB	(250 MB) tömörített meghajtó
F:	30 MB	(90 MB) tömörített meghajtó
Összesen:	80 MB	

1.3.1. Partícionálás (FDISK)

Ennek a programnak a használatával nagyon kell vigyázni, mert az egész merevlemezünket tönkretehetjük vele. Iskolákban, és olyan helyeken, ahol nem egyedül használunk egy számítógépet, ott kifejezetten tilos a használata! A programot természetesen elindíthatjuk, meg is tekinthetjük a merevlemezünk partícióját, de ezután lépjünk ki a programból, mert a további próbálkozásokkal csak kárt tudunk tenni.

FDISK **Partícionálás**

A merevlemez területekre osztását - ***partíciónálását*** - végezzük vele.

A parancs paraméterek nélkül indul, és folyamatosan tájékoztatást kapunk a szükséges teendőkről.

```
A:\>fdisk
```

A parancs kiadása után kapunk egy új képernyőt, melyen a kínálati lista olvasható. Választani a kiválasztott tétel sorszámával lehet. Mindegyik pont kiválasztása után kapunk egy újabb menüt.

```
                    MS-DOS Version 5.00
                Fixed Disk Setup Program
          (C)Copyright Microsoft Corp. 1983 - 1991
                      FDISK Options
 Current fixed disk drive: 1
 Choose one of the following:
1. Create DOS partition or Logical DOS Drive
2. Set active partition
3. Delete partition or Logical DOS Drive
4. Display partition information
5. Change current drive

Enter choice: [4]



 Press Esc to exit FDISK
```

1. **Create DOS partition or Logical DOS Drive** (DOS partíció vagy logikai DOS meghajtó készítése).
2. **Set active partition** (Az aktív partíció beállítása): A rendszer töltése erről az egységről történik.
3. **Delete partition or Logical DOS Drive** (Partíció vagy logikai DOS meghajtó törlése).
4. **Display partition information** (Partíció információk kiírása): A meglevő területi felosztásokat írja ki a képernyőre.

5. Change current drive (Az aktuális meghajtó váltása): Kettő vagy több merevlemez használata esetén kerül be a listába ez a menüpont.

Az **FDISK** programba belépve célszerű először mindig a negyedik pontot választani . Az a biztos, ha meggyőződünk a jelenlegi partíciós tábla állapotáról. ***Ha nem akarjuk megváltoztatni a partíciós táblát, vagy nem akarunk kárt tenni a merevlemezünkön, akkor csak ezt - és ha van az 5. pontot- használjuk!***

Partíciós információk megtekintése

Válasszuk ki a 4. menüpontot úgy, hogy beírjuk a "4"-es számjegyet, ami megjelenik az "**Enter choice:**" sorban. Utána meg kell nyomni az [ENTER] billentyűt. Ha másik számot szeretnénk kiválasztani, akkor írjuk be azt a számot, és az [ENTER] billentyűt csak akkor nyomjuk le, ha biztosan a megfelelő menüpontot választottuk ki.

```
                    Display Partition Information

Current fixed disk drive: 1

Partition  Status   Type    Volume Label  Mbytes   System   Usage
 C: 1         A    PRI DOS  MORATI_DISK       50  FAT12      63%
    2              Non-DOS                    30             37%

 Total disk space is     80 Mbytes (1 Mbyte = 1048576 bytes)



Press Esc to continue
```

Láthatjuk, hogy a példánkban szereplő merevlemez használt, van partíciós táblája, egy *elsődleges DOS partícióval* (PRI-Primary-elsődleges), és egy *"nem DOS" területtel*. Ha új merevlemezünk van, akkor ezt a menüpontot ugorjuk át. Olyankor itt a "**No partitions definied**" felirat jelenik meg, azaz nincs partíciós tábla definiálva a merevlemezen. Ebben az esetben ugorjunk a "*DOS partíció készítése*" című fejezethez.

A további teendőkről mindig a képernyő alján olvashatunk információkat. "**Press Esc to continue**", azaz nyomj [ESC]-et a folytatáshoz.

Dos partíció vagy logikai DOS meghajtó törlése

```
                Delete DOS Partition or Logical DOS Drive

Current fixed disk drive: 1

Choose one of the following:
1.  Delete Primary DOS Partition
2.  Delete Extended DOS Partition
3.  Delete Logical DOS Drive(s) in the Extended DOS Partition
4.  Delete Non-DOS Partition

 Enter choice: [ 4]




 Press Esc to return to FDISK Options
```

Ezt a menüpontot választva újabb menüt kapunk, ahol mindig a törölni kívánt partíció típusának megfelelő menüpontot kell kiválasztani.

```
                    Delete Non-DOS Partition

Current fixed disk drive: 1

Partition  Status   Type    Volume Label  Mbytes   System   Usage
 C: 1         A    PRI DOS  MORATI_DISK      50   FAT12      63%
    2              Non-DOS                   30              37%

 Total disk space is    80 Mbytes (1 Mbyte = 1048576 bytes)


WARNING! Data in the deleted Non-DOS Partition will be lost.
What Non-DOS partition do you want to delete..? [ 2]

 Do you wish to continue (Y/N).....? [ Y]


Press Esc to return to FDISK Options
```

Válasszuk ki először a "nem DOS" terület törlését, ahol további kérdésekre kell válaszolnunk. Először azonban kapunk egy figyelmeztetést:

(WARNING! Data in the deleted Non-DOS Partition will be lost)

"Figyelem! A törölni kívánt területen minden adatunk el fog veszni."

(What Non-DOS partition do you want to delete..? [2])

"A 'nem DOS' területen, mit akarunk törölni ? [_]"

A szögletes zárójelek között a partíció sorszámát kell megadni. Jelen esetben csak egy ilyen terület van, így csak a ***2 -es*** számot választhatjuk.

(Do you wish to continue (Y/N).....? [Y])

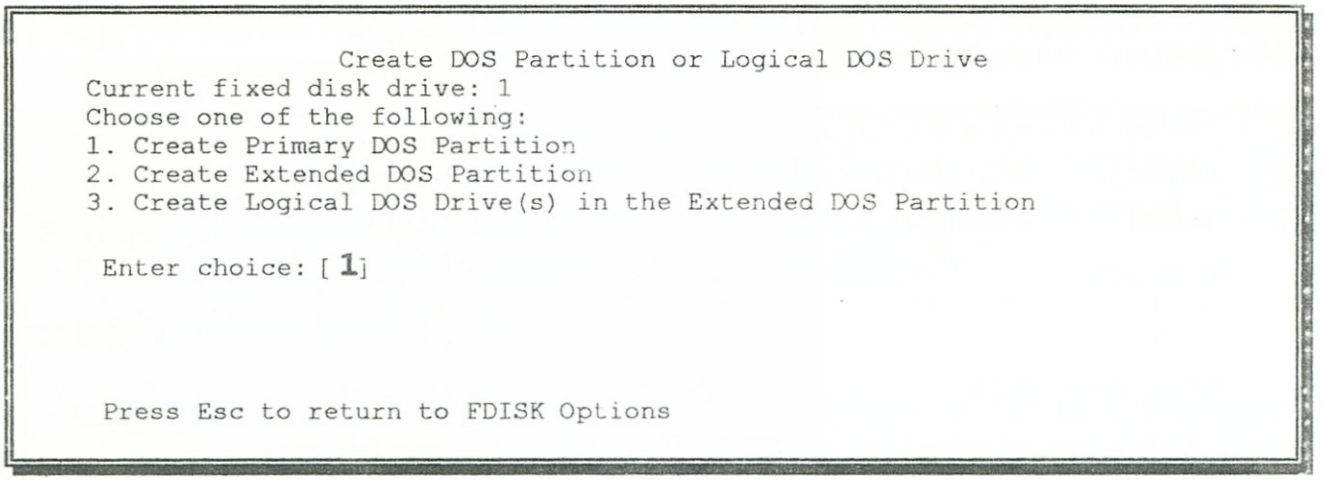
"Biztosan folytatni akarod (Igen vagy Nem)?"

Ha igennel (Y) válaszolunk a kérdésre, csak akkor fogja törölni a partíciót. Még innen is vissza lehet lépni egy nem (N) válasszal, és akkor nem vesznek el az adataink a kiválasztott partíción. ***Ha igennel válaszolunk, akkor azon a partíción vagy DOS meghajtón minden adatunk végérvényesen elveszik***. Jól gondoljuk meg, nehogy véletlenül egy másik meghajtót töröljünk.

Most töröljük az elsődleges DOS partíciót is hasonló módon.

DOS partíció készítése

Miután nincs partíciós tábla a merevlemezünkön, készítsünk egy 5 MB méretű DOS partíciót. Ha új a merevlemezünk, akkor itt kezdjük el az előkészítési munkát a merevlemezen. Ehhez a főmenüből (9. oldalon látható) válasszuk ki az első menüpontot.



1. **Create Primary DOS Partition** (Elsődleges DOS partíció létrehozása): A menüpont kiválasztása után kapunk egy új képernyőt, ahol a jelenlegi partíciós információk olvashatók, és felkínálja a program az egész szabad területet. Nekünk most pont erre van szükségünk, de ne fogadjuk el a teljes területet, mert nekünk most csak egy 5 MB -os DOS partíció kell.

Utána ahhoz, hogy a DOS rendszer innen induljon még aktívvá is kell tenni. Ezt a főmenü 2.pontja teszi meg. A partíció számát (most: 1) kell megadni, és már kész is.

2. **Create Extended DOS Partition** (Kiterjesztett DOS partíció készítése): Itt kell létrehoznunk egy 55 MB-os kiterjesztett partíciót, ami majd a "D" meghajtó-betűjelet fogja kapni.

3. **Create Logical DOS Drive(s) in the Extended DOS Partition** (Logikai DOS meghajtó készítése a kiterjesztett DOS partíción): Mivel a létrehozandó "D" meghajtó nem különbözik fizikailag a "C"-től, ezért nevezzük azt ***logikai meghajtó***-nak, ezért kell ezt a menüpontot is használnunk.

Miután végeztünk a partícionálással, lépjünk ki az [ESC] billentyűvel.

Miután a merevlemez partícionálását elvégeztük, indítsuk újra a számítógépünket az "A:" meghajtóba helyezett rendszerlemezről. Hiába is próbálnánk a "C" meghajtóról indítani, ott már nincs semmi, mert az átpartícionálással minden adatunk elveszett.

Hogy használni tudjuk a merevlemezt, meg kell formáznunk mindegyik DOS partíciót (C: és D: egység).

1.3.2. Formázás (FORMAT)

A merevlemez formázását a már ismert **FORMAT** paranccsal végezhetjük el. A hajlékonylemez formázásától annyiban különbözik, hogy megjelenik egy felirat, ami arra figyelmeztet, hogy a formázással minden adatot el fogunk veszíteni a merevlemezünkről. Nekünk, a példánkban most nincs semmi adat a merevlemezünkön, így azt nyugodtan leformázhatjuk.

```
A:\>format c: /s /v:MORATI_DISK
WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE C: WILL BE LOST!
Proceed with Format (Y/N)? Y__
Checking existing disk format.
Formatting 5.03M
_99 percent complete
System transferred

      5271552 bytes total disk space
      73728 bytes used by system
      5197824 bytes available on disk
         4096 bytes in each allocation unit.
       1269 allocation units available on disk.

  Volume Serial Number is 050C-6619

A:\>c:

C:\>_
```

A formázást gyorsan el fogja végezni, mert csak 5 MB méretű az első merevlemezünk. A formázásnál használjuk a "/S" paramétert is, hogy a rendszerlemezünkről felkerüljön a DOS a merevlemezre is. A "D" meghajtót is külön le kell formáznunk, aminek a kapacitása 55 MB.

Itt látható a másik előnye annak, ha több darabra osztjuk (partícionáljuk) a merevlemezünket. Ha véletlenül leformázzuk az egyik meghajtót, akkor a másikon lévő adatok még sértetlenek maradnak függetlenül attól, hogy egy merevlemezen vannak.

Ebből következik, hogy a **FORMAT** parancs nem egy lemezegységet formáz le, hanem egy meghajtó betűjel által mutatott lemezt. Ez a lemez lehet hajlékonylemez vagy egy merevlemez egy területe, partíciója. (Hálózati meghajtót, memórialemezt nem lehet megformázni.)

1.3.3. Könyvtárfelépítés kialakítása (MD, CD, RD)

Ezeket a munkálatokat már bizonyára mindenki elsajátította a tankönyv első részének olvasásakor, így most a technikai lebonyolításra nem térünk ki részletesen. A terven meghatározott könyvtárfelépítést létre kell hozni, és utána állhatunk neki a fájlok másolgatásának, telepítésének. Vannak olyan programok - ez a gyakoribb -, amelyek telepítéskor létrehozzák a szükséges könyvtárakat, így arra nekünk nincs gondunk. Azokat kell csak létrehoznunk, amelyekben az adatállományaink vannak, illetve a saját levelezésünk.

1.3.4. Konfigurációs fájlok elkészítése (EDIT)

A merevlemezünk ugyan rendelkezik DOS-szal, már el is indíthatjuk, de akkor minden rendszerinduláskor egyesével ki kell adni olyan parancsokat, amelyek a különféle részegységek jobb kihasználtságát biztosítják. Vannak olyan beállítások is, amelyeket nem tudunk a készenléti jel mellett állva beírni. Ezeket az utasításokat már a rendszer indulása közben ki kellene adni a számítógépünknek, de akkor még nem tudunk beavatkozni a rendszer folyamataiba.

Erre szolgál a CONFIG.SYS fájl, amiben különféle eszközmeghajtókat (**Device**) definiálhatunk, és egyéb beállításokat tehetünk meg. Ezeket a beállításokat később már nem tudjuk módosítani, és az eszközmeghajtókat sem tudjuk kitörölni a memóriából. Így ha túl sok eszközmeghajtó programot töltünk be rendszerinduláskor a memóriába, akkor előfordulhat, hogy olyan kevés helyünk marad, hogy a felhasználói programjainkat nem is tudjuk használni a megfelelő hatásfokkal memóriahiány miatt. A CONFIG.SYS fájl betöltését és feldolgozását az IO.SYS fájl végzi el, ha megtalálja *a betöltő lemez gyökérkönyvtárában* a rendszer indulásakor.

A másik konfigurációs fájl az AUTOEXEC.BAT fájl, ami egy olyan speciális -nevű-batch-fájl, amit a *rendszerindításkor* a COMMAND.COM lefuttat, ha megtalálja *a betöltő lemez gyökérkönyvtárában*. A betöltő lemez lehet az "A:" meghajtóban lévő hajlékonylemez, vagy a "C:" meghajtó (merevlemez). Ha van hajlékonylemez a meghajtóban, akkor onnan töltődik a rendszer bekapcsoláskor, vagy újraindításkor, ha nincs akkor a merevlemezről. Így nem mindegy, hogy hová tesszük a konfigurációs fájlokat. Ha saját konfigurációs fájlokkal akarjuk indítani a számítógépünket, akkor készítsünk egy rendszerlemezt (hajlékonylemezen), melyre olyan konfigurációs fájlokat írunk, ami csak tetszik. Utána tegyük a lemezmeghajtóba a rendszerlemezünket, és indítsuk újra a számítógépünket. Ilyenkor nem a merevlemezen lévő konfigurációs fájlok fognak végrehajtódni, hanem a mi hajlékonylemezünkön lévők. Ha valami hiba folytán nem úgy indul a rendszer, ahogyan azt mi szeretnénk, akkor vegyük ki a lemezt a meghajtóból, és indítsuk újra a számítógépet a saját merevlemezéről.

Ismétlem, a számítógép elindul a konfigurációs fájlok nélkül is, a DOS üzemképes, a felhasználói programokat tudjuk futtatni is, de a számítógépünk csak kisebb (~75%-os) hatásfokkal fog üzemelni. Így akár vehetnénk egy 386-os helyett egy 286-os számítógépet is, ha nincs szükségünk a számítógép maximális kapacitására.

Ez - és a másik konfigurációs fájl az AUTOEXEC.BAT - egy tetszőleges

Betöltő lemeznek nevezzük azt a lemezt, amelyikről a rendszer indulásakor a rendszer betöltődik. Ha az "**A:**" meghajtóban van hajlékonylemez olvasható állapotban (vagyis 5¼"-osnál az "ajtó" is csukva van), akkor onnan töltődik a rendszer, ha nem akkor a "**C:**" nevű merevlemezről, ha van. Ha nincs, akkor hibával leáll a rendszer indulása.

A rendszer indulása történik a számítógép bekapcsolásakor, vagy az [ALT]+[CTRL]+[DEL] billentyűk együttes megnyomásakor, vagy a **RESET** gomb megnyomásakor. Ha az említett billentyűkombinációt nyomjuk le, akkor nem hajtódik végre a számítógép részegységeinek tesztje. A konfigurációs fájlok tesztelésekor mindig ezt használjuk !

szövegszerkesztővel létrehozható, de ha ilyen nincs kéznél, akkor használhatjuk az első részben már megismert **COPY CON CONFIG.SYS** parancsot (lásd a **COPY** parancsnál). Ha mindig csak egy-két CONFIG.SYS utasítást akarunk kipróbálni, akkor használjuk csak nyugodtan ezt a formát, felülírva mindig az előző fájlt. Ilyen rövid fájl szerkesztésére nem érdemes szövegszerkesztőt behívni.

Ha meglévő több utasítást tartalmazó konfigurációs fájlt akarunk módosítani, akkor használnunk kell egy szövegszerkesztőt. Egy egyszerű karakteres fájl elkészítésére ne használjunk soha bonyolult, többfunkciós szövegszerkesztőt, mert feleslegesen pazarlunk el annyi időt a szövegszerkesztő és a konfigurációs fájl betöltésére.

A célnak megfelel a DOS **EDIT** szövegszerkesztője (EDIT.COM és QBASIC.EXE fájlok alkotják együtt!), ami egy egyszerű kezelésű, és az alapvető szövegszerkesztői fogások ellátására alkalmas program. Ha valaki használja az **NC**-t, akkor annak a szövegszerkesztője alkalmas a feladat ellátására, sőt, akkor nem is célszerű külön az **EDIT**-et betölteni, mert az még rövidebb, és egyszerűbb.

Most viszont maradjunk az **EDIT** szövegszerkesztőnél, aminek az indítása kétféleképpen történhet. Mint mindig, most is segítséget kérünk a számítógéptől, mert ez mindig a rendelkezésünkre áll:

EDIT **Szövegszerkesztő program**

```
C:\>edit /?
Starts the MS-DOS Editor, which creates and changes ASCII files.
EDIT [[drive:][path]filename] [/B] [/G] [/H] [/NOHI]
  [drive:][path]filename  Specifies the ASCII file to edit.
  /B          Allows use of a monochrome monitor with a color graphics card.
  /G          Provides the fastest update of a CGA screen.
  /H          Displays the maximum number of lines possible for your hardware.
  /NOHI       Allows the use of a monitor without high-intensity support.

C:\>_
```

[[drive:] [path] filename]: A *meghajtó*, *elérési út* és *fájlnév*, a megszokott paraméterhármas, ami a szerkesztendő fájlra utal. Ha létező fájlt akarunk szerkeszteni, akkor mindig célszerű megadni, hogy hol található, és mi a neve. Ha új állományt akarunk létrehozni, akkor is adjuk meg ezt a paramétert. Ekkor az adott helyen nem fogja megtalálni a megadott állományt, így azt új fájlként fogja értelmezni. Ebből következik, ha a megadott állományt nem tölti be indulás után, akkor az a megadott helyen nem létezik.

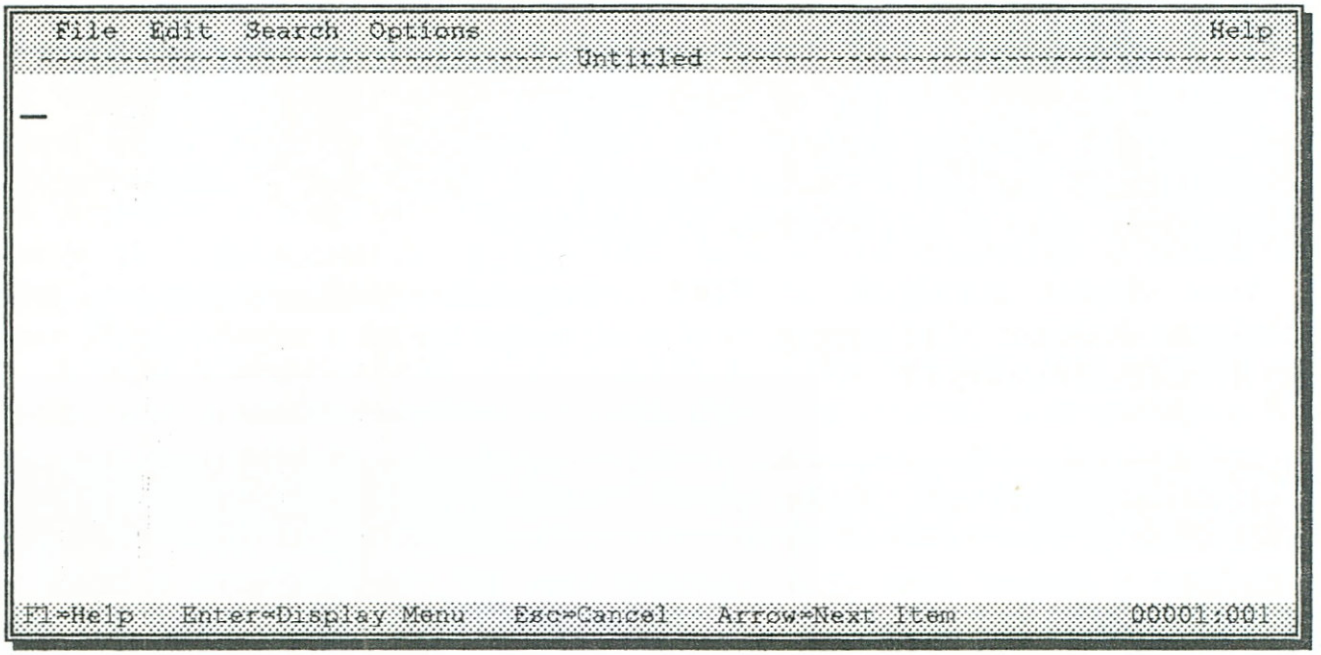
/b: A számítógép monokróm képernyővel rendelkezik, és program e szerint induljon el.

/g: A CGA típusú monitor esetén, a leggyorsabb képernyőfrissítést fogja használni.

/h: A képernyő típusához illeszkedő legnagyobb sorszámban jeleníti meg a szövegszerkesztőt. Ez EGA, VGA, SVGA monitorok esetén 43 ill 50 soros képernyőhasználatot jelent.

/nohi: Lehetővé teszi a nyolcszínű monitorok használatát. Például VGA monitoron ebből annyit lehet látni, hogy a kiemelt színű betűk fehér helyett pirosak lesznek.

Alapértelmezésben ezekre a kapcsolókra semmi szükségünk sincs, sőt paraméterek nélkül is elindíthatjuk a szövegszerkesztőt. Ekkor megjelenik a szövegszerkesztő üdvözlő képernyője, aminél [ESC]-et kell nyomnunk. Ha [ENTER]-t nyomunk, akkor elindul a *help* (segítség) programja, aminek segítségével megismerhetjük a szövegszerkesztő használatát. Ha erre nincs szükségünk, akkor nyomjunk [ESC]-et, és megjelenik egy ablak **"untitled"** azaz *névtelen* fejléccel, és írhatjuk a szöveget.



A szövegszerkesztő a jelenleg forgalomban lévő Microsoft® programokhoz igazodva jelenik meg, az itt megtanult ismeretekkel más Microsoft programokban is találkozni fogunk. Úgy a kezelésében, mint a párbeszédablakok használatában hasonlít más programra.

A szövegszerkesztő részletes ismertetésére, és sokrétű alkalmazására, a "*Szövegszerkesztési ismeretek*" című tankönyvemben fogok kitérni, mert a számítógépes munka egyik alapvető, rendszeresen ismétlődő feladatai közé tartozik. Ha levelet vagy programot írunk, akkor is minden esetben szövegszerkesztőt kell használnunk a feladat megoldására. A szöveg kivitelezését tekintve eltérhetnek egymástól a szövegszerkesztők, de használati módjukban mind megegyeznek.

Az alábbiakban csak arra szorítkozunk, hogy a konfigurációs fájlokat el tudjuk készíteni, így az ismertetés nem lesz teljeskörű.

Most a CONFIG.SYS fájlt kell elkészítenünk, így írjuk be a készenléti jel mellett:

```
A:\>EDIT CONFIG.SYS
```

Fontos, hogy a gyökérkönyvtárban álljunk, vagy adjuk meg a CONFIG.SYS fájl helyét, mert különben egy alkönyvtárban fogja létrehozni, akkor viszont nem fog végrehajtódni rendszerinduláskor. Ekkor a következő formát használjuk:

```
C:\DOS\UTIL>EDIT A:\CONFIG.SYS
```

Mint ismeretes, a példában szereplő megadásnál az "**A:**" megadása is fontos, mert ha nem az "**A:**" meghajtón állunk, akkor rossz helyre kerül a konfigurációs fájl - a "**C:**" meghajtó gyökérkönyvtárába -, így jobb, ha mindig pontosan megadjuk az elérési utat, ha nem akarunk galibát okozni. Ha pedig többen is dolgoznak egy számítógépen, akkor soha ne módosítsuk önhatalmúlag a "**C:**" meghajtón lévő konfigurációs fájlokat, mert úgy csak a többiek munkáját fogjuk megnehezíteni. Az MS-DOS 6-tól felfelé lehetőségünk van menüvezérelt CONFIG.SYS fájlt készíteni, így többféle rendszerindulás is lehetséges. Az 5.0-ás DOS-ban csak azt tudjuk tenni - segédprogramok nélkül - ha másfajta rendszerindulást szeretnénk, hogy hajlékonylemezről indítjuk a számítógépet. Ezen olyan konfigurációs fájlokat helyezhetünk el, amilyet csak akarunk.

A parancs végrehajtásaként megjelenik a szövegszerkesztő, és munkaterületén a CONFIG.SYS fájl - ha már létezett -, ha nem, akkor az üres lesz. A szövegszerkesztő területén a kurzormozgató-billentyűkkel szabadon mozoghatunk, a fájl eleje és a vége között. Így ebből az következik, hogy amennyiben nem tudunk lejjebb menni, akkor ott van a fájl vége. A sorok végén az [ENTER] billentyűvel léphetünk új sorba, ami egészen pontosan azt jelenti, hogy a kurzor pozíciójába beszúr egy sorvégjelet. Ezért vigyáznunk kell arra, hogyha a sor közepén állunk, akkor a sorvégjelet a sor közepére fogja beszúrni, azaz a tőle jobbra lévő szöveget "átdobja" a következő sorba. Ekkor, mint minden más karaktert is, a visszatörlés ([BACKSPACE],[⇦]) billentyűvel törölhetünk vissza. Ekkor ugyan a sor elején állva egy kicsit furcsának tűnhet, mit is törlünk vissza, hisz' látszólag semmi sincs a sor első karaktere előtt. Pedig van, az előző sor sorvégjele, amit letörölve, a kurzortól jobbra lévő szöveget, az előző sor végére mozgatjuk fel. Próbálgassuk ki, gyakorolgassunk egy kicsit, és akkor érthetővé válik, miről is van szó.

Most visszatérve a konfigurációs fájlunkhoz, írjuk be a kipróbálni kívánt utasítást vagy utasításokat - sor végén [ENTER]-t nyomva! -, és ha készen vagyunk, akkor következik a mentés és a kilépés.

A megírt szöveget el kell mentenünk (to Save, Saving), amit a **File** *menü* segítségével tehetünk meg. Nyomjuk le [ALT]+[F] -et, mire megjelenik - *legördül* - egy menü, azaz megjelenik több kiválasztható tétel. Ekkor a kurzormozgató-billentyűkkel lépjünk a **Save** *menüpontra*, és nyomjunk [ENTER]-t. Ekkor elmentettük a készített fájlt úgy, mint amikor a **COPY CON CONFIG.SYS** formát használtuk, és a végén a [CTRL]+[Z] (**^Z**) és [ENTER] billentyűkkel lezártuk a fájlt, mire ott megjelent az "**1 file(s) copied**" üzenet.

Egy menü *legördítése* alatt azt értjük, hogy a *menü nevének* aktivizálásakor megjelenik egy nagyobbacska ablak, amiben egymás alatt szavak - *menüpontok* - helyezkednek el, amelyek logikailag kapcsolódnak a menü nevéhez. Ekkor kiválaszthatjuk az egyik *menüpontot*, aminek hatására aktiválódik egy funkció, vagy további információkat kér tőlünk a számítógép egy felbukkanó párbeszédablakban.

Most már ne módosítsunk semmit, hanem lépjünk ki, amit a **FILE** menü **Exit** menüpontjának kiválasztásával tehetünk meg az előző módon. Ha ilyenkor megjelenne a következő ablak, akkor az azt jelenti, hogy a mentés után - véletlenül, vagy szándékosan - módosítottunk a szövegen.

```
Loaded file is not saved. Save it now?
<Yes>     <No>     <Cancel>     <Help>
```

"A betöltött fájl nincs elmentve. Elmentsem most?"

Mint láthatjuk, az első tétel "< >" jeleivel ki van emelve, ami azt jelenti, hogy [ENTER]-t nyomva ezt a választ hagyjuk jóvá. Válaszolhatunk a kiemelt színű kezdőbetűk leütésével is, de ha ki akarunk térni a válaszadás elől, akkor nyomhatunk [ESC]-et is, de ennek csak akkor van értelme, ha meggondoltuk magunkat, és mégsem szeretnénk kilépni.

Kilépés után visszajutunk a készenléti jel mellé, és ekkor az [ALT]+[CTRL]+[DEL]-lel újraindíthatjuk a számítógépünket. Ekkor ha jó helyre került a konfigurációs fájlunk, akkor az lefut, és ha valamelyik utasítása hibás volt, akkor megjelenik valamilyen üzenet a hiba okára utalva. Ha az leszaladna a képernyőről, akkor a [PAUSE] billentyűvel "fogjuk meg", és olvassuk el, írjuk le a füzetünkbe, fordítsuk le, és próbáljuk meg kikövetkeztetni a hiba okát. Az *"MS-DOS 6 kézikönyv"* vagy az *"MS-DOS 5.0 Felhasználói kézikönyv I.-II."* (*LSI Oktatóközpont*) részletes útmutatást ad, hogy milyen hiba esetén, milyen teendőink vannak.

2. CONFIG.SYS fájl használata

A számítógépünkben lévő részegységek jobb kihasználhatósága érdekében elindíthatunk olyan *rezidens programokat* és *eszközvezérlőket*, amelyek ezeket a feladatokat ellátják. Ezeket a programokat még a rendszer indulásakor el kell indítani, és erre szolgál a CONFIG.SYS fájl.

A CONFIG.SYS fájl tesztelésének csak egyetlen módja van. A fájl elkészítése után újraindítjuk a számítógépünket, például [ALT]+[CTRL]+[DEL]-lel.

Minden esetben azok a konfigurációs fájlok fognak lefutni, amelyek a betöltő lemez gyökérkönyvtárában vannak. Teszteléshez mindig egy rendszerlemezre (**A:**) készítsük el ezeket a konfigurációs fájlokat, és a lemezről indítsuk el a számítógépünket.

2.1. Memóriakezelés

A számítógép használata során ez a részegység az, amit a legrosszabb hatásfokkal kezel a DOS, ha nem használunk segédprogramokat. Hiába van a számítógépünkben akár 4 MB memória, ha nem használunk segédprogramokat, akkor a DOS-ban csak az alsó 640 kB alapmemóriát fogjuk tudni használni. Ennek az az oka, hogy a DOS-nak futnia kell egy 640 kB-os XT számítógépen is, így a DOS-ban egy 486-os számítógép-tulajdonos (4-16 MB RAM-mal) sincs sokkal nagyobb előnyben, mert az nem más a DOS szempontjából, mint egy gyors XT. A számítógépben használatos címzési rendszerekről, a számítógépek közötti különbségekről részletesen a "*Számítógépes hardver*" című tankönyvemben olvashatsz.

A továbbiakban azzal fogunk foglalkozni, hogy a létező memóriatípusok közül melyik van meg a számítógépben, és ez hogyan használható ki a legjobban. A rajzon bemutatott memóriatípusok nem minden számítógépben vannak meg. Erről a következőkben meg is fogunk győződni.

A számítógép processzorának üzemmódjai

Valós (Real) üzemmód: Ebben az üzemmódban a számítógép processzora maximum 1 MB memóriát tud megcímezni, azaz csak ennyit tud elérni. Programokat futtatni viszont csak 640 kB területen tud, mert a DOS csak ennyit engedélyez a számára. A felette lévő 384 kB terület képernyőmemóriának, és egyéb rendszerterületeknek van fenntartva. Valós üzemmódban még egy 486-os számítógép is csak azt az 1 MB-ot tudja elérni, hiába van benne több megabájt memória. Ebben az üzemmódban egy 486-os nem más, mint egy "gyors XT". Az XT típusú számítógép pedig, csak ebben az üzemmódban tud dolgozni.

Kiterjesztett üzemmód: Ebben az üzemmódban már több memóriát képes megcímezni a processzor. A 80286-os 4 MB-ot, a 80386-os (és 80486-os,...) pedig 4 GB-ot. Ekkor a processzort át kell - programból - kapcsolni erre az üzemmódra, és csak ekkor használhatjuk ki az 1 MB feletti memóriaterületeket. A DOS az XT számítógépekkel való kompatibilitás megtartása miatt, még egy 486-os számítógépen is valós üzemmódban fut, és a programok futtatására nem használja ki a kiterjesztett üzemmódban elérhető több megabájt memóriát.

A 80386-os számítógépet és a 80486-os számítógépet ***32 bites számítógépnek*** is szokás nevezni, mert a processzora a memóriacímzést már 32 biten végzi, ami biztosítja a 4 GB memória elérését. A Pentium (80586) processzor már 64 bites számítógép, de arra is érvényes minden, amit a 386-os számítógépekre elmondunk. Így a továbbiakban minden esetben, amikor 386-os vagy 32 bites számítógépet említek, akkor az értelemszerűen az említett magasabb processzorszámú számítógépekre is igaz. A számítógépek és a programok általában ***lefelé kompatibilisek***, ami azt jelenti, hogy az olyan program ami XT számítógépen lefut, az futtatható akár 486-oson is, de ami 386-oson fut, az nem működik 286-oson vagy XT-n.

A Windows-rendszer viszont kihasználja, és 386-os (486-os,...) számítógépek esetén kiterjesztett üzemmódban fut, kihasználva az összes beépített memóriát. Újabban megjelentek már olyan operációs rendszerek is, amelyek szakítottak az XT-kel való kompatibilitással, és csak 32 bites processzorú számítógépeken futnak, ott viszont a kihasználják a processzor kiterjesztett üzemmódjának összes lehetőségét. Ennek eredményeképpen sokkal gyorsabb futást eredményeznek, minek következtében megoldható lett a közkedvelt grafikus felhasználói felület használata.

(A példákban a számítógépünkben lévő memória lekérdezésére a **MEM** parancsot fogjuk használni, aminek az ismertetésére csak később térek ki. A lekérdezéseknél viszont mindig megmutatom azt, hogy éppen mit kell nézni.)

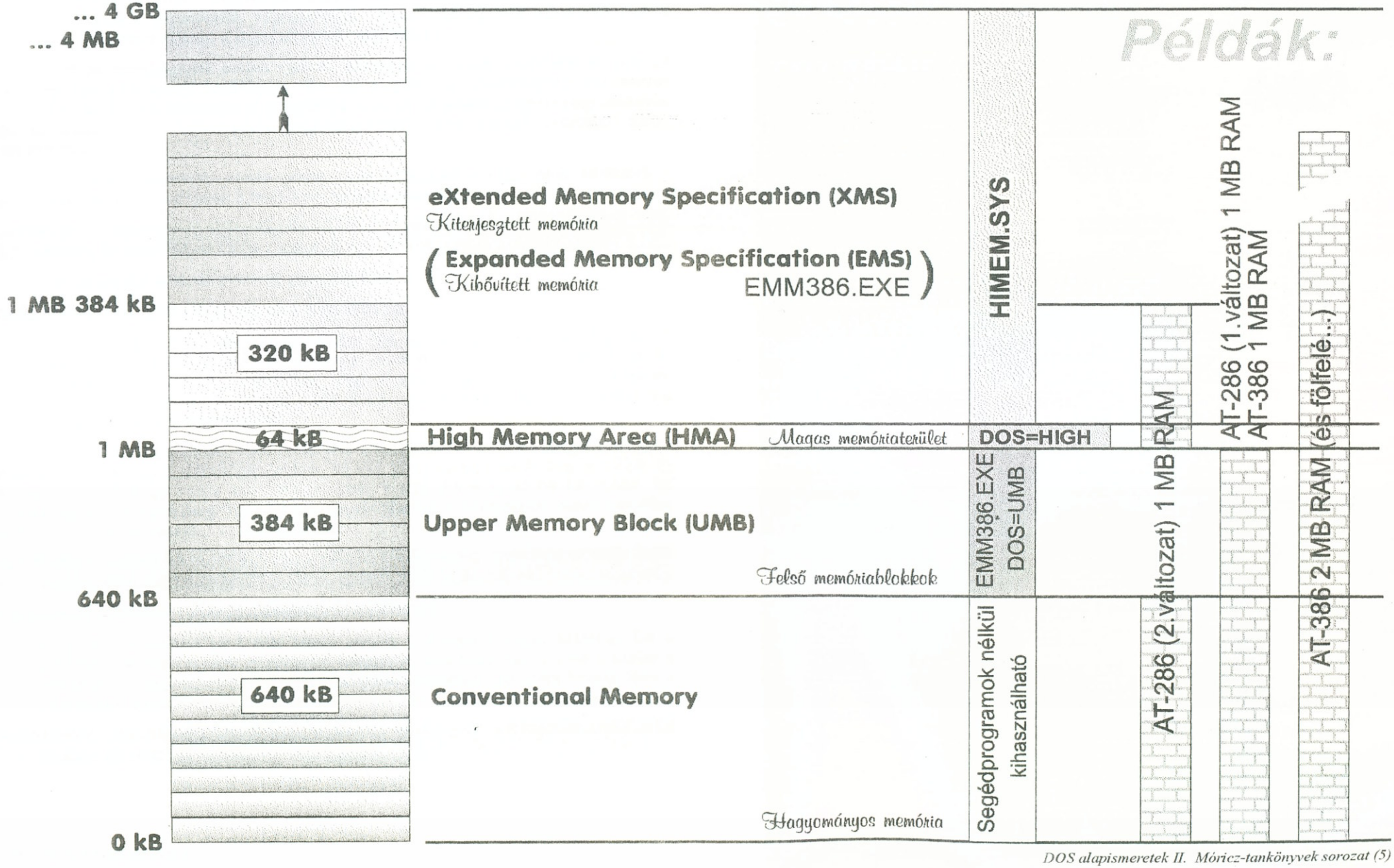
Hagyományos memória (Coventional Memory):

A "00000" memóriacímen kezdődő memóriaterület, ahová alapértelmezésben kerülnek betöltésre a programok. Ez az egyetlen memóriaterület, ami segédprogramok nélkül is elérhető teljes terjedelmében. A maximális mérete 640 kB lehet, de használhatunk kevesebbet is. Például üzemeltethetjük a számítógépünket 256 kB memóriával is, a DOS-nak például elegendő is.

Felső memóriablokkok (Upper Memory Block):

Az a memóriaterület, ami a 640 kB hagyományos memória után következik, és csak segédprogramokkal használható ki szoftveres célokra. Az olyan számítógépekben, ahol van ilyen típusú memória is, ott hardveresen is kihasználható ez a terület. Itt helyezkedhet el a képernyő-memória területe, hálózati meghajtóprogramok pufffere, a ROM BIOS másolata, ... *stb.*

A számítógépben lévő memóriák értelmezése:



Ezeket saját céljainkra csak 386-os számítógépeken tudjuk kihasználni, a processzor kiterjesztett üzemmódjában. Mondhatnánk, hogy akkor minek ez a 384 kB RAM például XT számítógépek esetén? A számítógép egyes részeinek gyorsabb működése érhető el vele. A legfelső részen lehet például a ROM BIOS másolata is. Ennek az az értelme. hogy a RAM memória sokkal gyorsabban elérhető, mint egy ROM memória. Ha a számítógép a működése közben, a BIOS függvényhívásait a RAM-ban elvégezheti, akkor sokkal gyorsabban fognak a parancsaink végrehajtódni. Nincs maximálisan kihasználva ez a felső terület, de gyorsít valamit a számítógépünk működésén.

A 386-os számítógépek esetén viszont, kihasználhatjuk ezt a területet is, például eszközmeghajtó programok betöltésére. Ha ide töltjük - a későbbiekben részletesen ismertetésre kerülő - eszközmeghajtó programjainkat, akkor szinte az egész hagyományos memória megmarad a felhasználói programjainknak, ami sokkal gyorsabb futást eredményezhet.

A Windows használata esetén pedig, a programjaink - vagy azok adatainak - tárolására is használhatjuk ennek a területnek szinte minden bájtját.

Használatához szükséges a CONFIG.SYS fájlban elhelyezett **DEVICE=EMM386.EXE** utasítás, de csak 386-os (&⇧) számítógépen.

(&⇧ - ... és magasabb processzorszámú számítógépen...)

Magas memóriaterület (High Memory Area-**HMA**):

Ez a terület az 1 MB után következő első "64 kB-16 bájt" memória. Hogy miért pont ennyi, erre majd a "*Számítógépes hardver*" című könyvemben térek ki részletesen. Annyit elég tudni róla, hogy a számítógép által használt címzési eljárás eredményeképpen, még az 1 MB felett el tud érni egy ilyen kis területet is a processzor valós üzemmódjában.

A terület lehatárolásának csak szoftveres szempontból van jelentősége, mert ez a terület csak akkor van a számítógépünkben, ha van kiterjesztett memóriánk is. Tehát ez nem úgy értendő, hogy van a számítógépünkben 1 MB RAM plusz 64 kB RAM. Ez technikailag sem oldható meg. Ez a 64 kB memóriaterület a kiterjesztett memóriaterület első 64 kilobájtos darabkája.

Használatához szükséges a CONFIG.SYS fájlban elhelyezett **DEVICE=HIMEM.SYS** utasítás, de csak 286-os (&⇧) számítógépen.

Kiterjesztett memória (Extended Memory System-**XMS**):

Ez a memóriaterület az 1 MB feletti terület, a memória végső határáig. Ezt a területet a processzor csak kiterjesztett üzemmódjában tudja elérni, kivéve azokat a speciális 286-osokat, amelyekben a felső memóriablokkok rovására hozták létre a kiterjesztett memóriaterületet. (*Ezt részletesen lásd később, a 31. oldalon.*)

Használatához szükséges a CONFIG.SYS fájlban elhelyezett **DEVICE=HIMEM.SYS** utasítás, de csak 286-os (&⇧) számítógépen.

Kibővített memória (Expanded Memory Specification)

Ez a memóriatípus régebben külön bővítőkártyán helyezkedett el, ma már ritkán használunk ilyen típust, mert lassú és nehézkes az elérése. Vannak viszont olyan programok, amelyek igénylik az ilyen memória meglétét a gyorsabb futásuk érdekében. Ha nekünk nincs a számítógépünkben memóriabővítő-kártya, akkor a meglévő kiterjesztett memóriában létrehozhatunk kibővített memóriát, de hangsúlyozom, ebben az esetben csak egy szimulált memóriáról beszélünk, ami a kiterjesztett memóriában létrehozható, de nem szükségszerű! Csak 32 bites számítógépek esetén szimulálható.

Ez a memóriatípus úgy működik, hogy az EMS meghajtóprogram egyszerre 64 kB-ot képes *belapozni* a felső memóriablokkok egy 64 kB-os területére. Ezt a területet a felhasználói programok elérik, és így tudják használni. Ha egy másik 64 kB-os szeletkére van szükség, akkor azt fogja *belapozni* erre a területre, így a program mindig megkapja a kívánt adatokat, és nem is tud arról, hogy nem egy összefüggő memóriaterületet használ, hanem egy lapozós memóriát. Éppen emiatt a lapozgatás miatt lassú az ilyen típusú memóriák elérése, így ma már kerüljük a használatát.

Használatához szükséges a CONFIG.SYS fájlban elhelyezett EMS memóriavezérlő program, de csak 286-os (&ű) számítógépen.

Szimulálásához szükséges a CONFIG.SYS fájlban elhelyezett **DEVICE=EMM386.SYS** utasítás, de csak 386-os (&ű) számítógépen.

A megismert memóriatípusok adatainak - kihasználtságának - megtekintésére szolgál a következő parancs, amivel gyakran fogunk találkozni, mert ezzel tudjuk kiíratni, hogy mennyi szabad, és mennyi foglalt memória van a számítógépünkben, és ami a legfontosabb, hogy mire foglalt az adott memóriamennyiség.

MEM — Memóriatérkép

Szokásunkhoz híven forduljunk ismét a számítógéphez egy kis segítségért:

```
C:\DOS>mem /?

Displays the amount of used and free memory in your system.

MEM [ /CLASSIFY | /DEBUG | /FREE | /MODULE modulename] [ /PAGE]

  /CLASSIFY or /C  Classifies programs by memory usage. Lists the size of
                   programs, provides a summary of memory in use, and lists
                   largest memory block available.
  /DEBUG or /D     Displays status of all modules in memory, internal drivers,
                   and other information.
  /FREE or /F      Displays information about the amount of free memory left
                   in both conventional and upper memory.
  /MODULE or /M    Displays a detailed listing of a module's memory use.
                   This option must be followed by the name of a module,
                   optionally separated from /M by a colon.
  /PAGE or /P      Pauses after each screenful of information.

C:\DOS>_
```

/C: Osztályozva (Classify) írja ki a memóriafoglaltságot.

/D: Nyomkövető (Debug) módon írja ki a memóriafoglaltságot. Ez a megjelenítési mód sokkal részletesebb, mint az előző.

/P: Programok (Programs) szerinti listát kapunk, ami nagyon hasonlít a "/D" paraméterrel kapott listára, csak ez kevésbé részletes. Éppen emiatt hagyták ki az MS-DOS 6-ból. Ott a "/P" paraméternek már más lesz a hatása.

MS-DOS 6 után:

/F: A szabad (Free) memóriaterületeket írja ki megnevezésükkel, és a szabad terület méretének feltüntetésével.

/M: A megadott modul (Module, itt program, eszközvezérlő, ...*stb.*) elhelyezkedését írja ki, memóriabeli helyfoglalását.

/P: Oldalanként (Page) várakozással, mint a **DIR** parancsnál már megszokhattuk. Az előző verziókban (MS-DOS 6 előtt) ez nagyon hiányzik, ott a **MEM /c|MORE** formában tudunk képernyőoldalanként tagolni. (Vigyázzunk, ez a kapcsoló még az MS-DOS 5.00-ban más jelentéssel bírt. Ellenőrizzük le használat előtt, hogy melyik verziójú DOS-t használjuk **(VER)**)

Legjobban példákon keresztül tudjuk megtekinteni, hogy melyik kapcsoló használatakor, milyen listát kapunk. Az elkövetkező konfigurációs fájlok tesztelésekor mindig használni fogjuk ezt az utasítást, hogy lássuk mi lett a hatása. Így nagyon fontos, hogy ezzel a paranccsal ismerkedjünk meg először. A parancs sok olyan információt is kiír, amire egy kezdőnek nincs szüksége, amit esetleg egy programozó tud csak értékelni. Azért van a parancsnak több paramétere is, hogy mindenki csak azt használja belőle, ami neki kell.

Azért kezdtem a példát egy **VER** paranccsal, hogy mindenki lássa, melyik DOS verzióban készült a lista. Az MS-DOS 5.00-ás verzióban csak a program-modul neve és helyfoglalásának mérete íródik ki, a részletes ismertetésnél. Az MS-DOS 6-ban már nem ismeretlen az a fogalom, hogy egy adott program a hagyományos (Conventional), és a felső (Upper) memóriaterületen is helyet foglalhat. Ilyenkor külön-külön kiírja, hol mennyi helyet foglal a memóriában.

Mivel többféle memóriatípus is létezik, ezért az 5-ös DOS-tól eltérően, itt már készül egy összegzés is, hogy a programok összesen melyik memóriatípusban, mennyi helyet foglalnak, illetve hagytak szabadon.

A kiírásokban szerepelni fognak a felső memóriaterületek (UMB) kihasználtságára vonatkozó adatok is, melyeket csak 32 bites számítógépek esetén lehet kihasználni szoftveresen. Így ha a Te számítógépeden a listában nem szerepel az "**Upper Memory**" kiírás, és az utána szereplő adatok, akkor annak az lehet az oka, hogy még nincs definiálva (lásd **EMM386.EXE**, **DOS=UMB** *(39. oldalon)*), vagy csak 286-os számítógéped van. Ezekről a későbbiek során még lesz szó, így most csak annyit jegyezz meg a parancs használatával kapcsolatban, hogy milyen jellegű adatokat ír ki a program.

Példa a "/C" paraméter használatára:

```
C:\DOS>VER

MS-DOS Version 6.20

C:\DOS>MEM /C

Modules using memory below 1 MB:

  Name           Total        =   Conventional   +   Upper Memory
  --------  ----------------   ----------------   ----------------
  MSDOS       29 405    (29K)      29 405    (29K)          0     (0K)
  HIMEM        1 168     (1K)       1 168     (1K)          0     (0K)
  EMM386       3 120     (3K)       3 120     (3K)          0     (0K)
  DBLSPACE    47 056    (46K)      33 680    (33K)     13 376    (13K)
  COMMAND      3 776     (4K)       3 776     (4K)          0     (0K)
  NC          13 424    (13K)      13 424    (13K)          0     (0K)
  COMMAND      3 184     (3K)       3 184     (3K)          0     (0K)
  ANSI         4 240     (4K)           0     (0K)      4 240     (4K)
  IMOUSE      10 768    (11K)           0     (0K)     10 768    (11K)
  DISPLAY      8 336     (8K)           0     (0K)      8 336     (8K)
  SMARTDRV    29 024    (28K)           0     (0K)     29 024    (28K)
  SHARE       16 944    (17K)           0     (0K)     16 944    (17K)
  K-KERNEL     1 712     (2K)           0     (0K)      1 712     (2K)
  Free       645 648   (631K)     567 392   (554K)     78 256    (76K)

Memory Summary:

  Type of Memory       Total    =    Used    +    Free
  ----------------  ----------   ----------   ----------
  Conventional         655 360       87 968      567 392
  Upper                162 656       84 400       78 256
  Reserved             262 144      262 144            0
  Extended (XMS)    15 697 056    2 340 000   13 357 056
  ----------------  ----------   ----------   ----------
  Total memory      16 777 216    2 774 512   14 002 704

  Total under 1 MB     818 016      172 368      645 648

  Largest executable program size          567 376    (554K)
  Largest free upper memory block           73 360     (72K)
  MS-DOS is resident in the high memory area.

C:\DOS>_
```

A kiírt adatok bizonyára különbözni fognak a Te számítógépeden, ezért először főbb egységeiben ismertetem, hogy mit is kell nézni a listában. Először a memóriát foglaló modul (program, eszközmeghajtó, ...*stb.*) neve szerepel, majd utána az általa lefoglalt memória mennyisége bájtokban összesen (Total-Összes), illetve zárójelben kilobájtokban. Az utóbbi adat bőven megfelel, ha csak hozzávetőlegesen érdeklődünk a memóriafoglaltságról. Ha konkrétan érdekel, hogy egy modul mennyi helyet foglal pontosan bájtokban, akkor az elsőnek kiírt adatokat nézzük. A következő két-két adat pedig - amint a fejlécben is olvasható - a helyfoglalást részletezi ki, hogy a hagyományos (Conventional) és a felső memóriablokkok területén (UMB) mennyi helyet foglal el az adott program. Ebből jól láthatjuk, hogy melyik programunk került fel az UMB területre, ezáltal is több helyet hagyva a felhasználói programoknak a hagyományos memóriában. Láthatjuk a listában, hogy melyik programokat lehet még feltölteni az UMB területre, mit kell még módosítanunk a konfigurációs fájlokban. Utána kapunk egy összegzést a foglalt és szabad memóriaterületek méreteiről, memóriatípusonként kiválogatva.

Most nézzünk egy-egy példát, a többi paraméter használatára is:

```
C:\DOS>MEM /D /P

Conventional Memory Detail:

  Segment                Total         Name          Type
  -------          ----------------    -----------   --------
   00000                1 039     (1K)               Interrupt Vector
   00040                  271     (0K)               ROM Communication Area
   00050                  527     (1K)               DOS Communication Area
   00070                2 864     (3K)  IO            System Data
                                          CON         System Device Driver
                                          AUX         System Device Driver
                                          PRN         System Device Driver
                                          CLOCK$      System Device Driver
                                          A: - E:     System Device Driver
                                          COM1        System Device Driver
                                          LPT1        System Device Driver
                                          LPT2        System Device Driver
                                          LPT3        System Device Driver
                                          COM2        System Device Driver
                                          COM3        System Device Driver
                                          COM4        System Device Driver
   00123                6 112     (6K)  MSDOS         System Data
   002A1               56 480    (55K)  IO            System Data
                        1 152     (1K)     XMSXXXX0   Installed Device=HIMEM
                        3 104     (3K)     EMMQXXX0   Installed Device=EMM386
                       33 664    (33K)     DBLSYSH$   Installed Device=DBLSPACE
                        4 432     (4K)                FILES=80
                          960     (1K)                FCBS=16
                        7 984     (8K)                BUFFERS=15
                        2 032     (2K)                LASTDRIVE=W
                        3 008     (3K)                STACKS=9,256
   0106B                   80     (0K)  MSDOS         System Program
   01070                2 656     (3K)  COMMAND       Program
   01116                   80     (0K)  COMMAND       Data
   0111B                1 040     (1K)  COMMAND       Environment
   0115C                  256     (0K)  NC            Environment
   0116C               13 168    (13K)  NC            Program
   014A3                  256     (0K)  COMMAND       Data
   014B3                2 656     (3K)  COMMAND       Program
   01559                  272     (0K)  COMMAND       Environment
   0156A                  240     (0K)  MEM           Environment
   01579               88 992    (87K)  MEM           Program
   02B33              478 400   (467K)  MSDOS         -- Free --

Upper Memory Detail:

Press any key to continue . . .
```

Nálam itt telt meg az első képernyőoldal, mert 50 soros kijelzésűre van kapcsolva. (Ezt EGA típustól felfelé tehetjük meg, a **MODE CON: LINES=50** paranccsal, ha a CONFIG.SYS-ben szerepel a **DEVICE=ANSI.SYS** utasítás.) Azért ált meg, és írta ki a **"Press any key to continue . . ."** szöveget, mert használtuk a "/P" (Page-oldal) paramétert is. A "/D" paraméter használata esetén célszerű a "/P"-t is használni, mert olyan sok sort ír ki, hogy lefutna a képernyőről a lista eleje, amit akkor nem tudunk elolvasni. Az MS-DOS 5.00-át használók pedig "**MEM /D |MORE**" formában használják a parancsot, mert ott még nincs más mód, az oldalakra tördelésre.

Most folytassuk a kiírt adatok megtekintését. Nyomjunk meg egy tetszőleges billentyűt, és folytatódik a kiírás.

```
  Segment  Region       Total          Name          Type
  -------  ------  ---------------  -----------  --------
   0C94A       1      2 880    (3K)  MSDOS        -- Free --

   0D101       2      4 224    (4K)  IO           System Data
                      4 192    (4K)     CON       Installed Device=ANSI
   0D209       2     10 752   (11K)  IO           System Data
                     10 720   (10K)     PRIMAX    Installed Device=IMOUSE
   0D4A9       2      8 320    (8K)  IO           System Data
                      8 288    (8K)     CON       Installed Device=DISPLAY
   0D6B1       2     13 376   (13K)  DBLSPACE     Data
   0D9F5       2         48    (0K)  MSDOS        -- Free --
   0D9F8       2     29 024   (28K)  SMARTDRV     Program
   0E10E       2        256    (0K)  MSDOS        -- Free --
   0E11E       2     16 944   (17K)  SHARE        Program
   0E541       2      1 712    (2K)  MSDOS        -- Free --
   0E5AC       2        240    (0K)  K-KERNEL     Environment
   0E5BB       2      1 472    (1K)  K-KERNEL     Program
   0E617       2     73 360   (72K)  MSDOS        -- Free --

Memory Summary:

  Type of Memory       Total   =    Used    +    Free
  ---------------  ----------   ----------   ----------
  Conventional        655 360       87 968      567 392
  Upper               162 656       84 400       78 256
  Reserved            262 144      262 144            0
  Extended (XMS)   15 697 056    2 340 000   13 357 056
  ---------------  ----------   ----------   ----------
  Total memory     16 777 216    2 774 512   14 002 704

  Total under 1 MB    818 016      172 368      645 648

  Memory accessible using Int 15h                    0      (0K)
  Largest executable program size              567 376    (554K)
  Largest free upper memory block               73 360     (72K)
  MS-DOS is resident in the high memory area.

  XMS version  3.00; driver version  3.16

C:\DOS>_
```

Amint láthatjuk, egész tekintélyes mennyiségű információ zúdult ránk, így nem is lesz könnyű kibogozni, hogy ebből nekünk mire van szükségünk. Aki teheti nyomtasson ki egy ilyen listát (**MEM /D >PRN**), hogy össze tudja hasonlítani ezzel, hogy áttekinthetőbb legyen a kapott eredmény.

A kiírt adatok alapvetően négy oszlopba sorolhatók:

MS-DOS 5.00:

1. **Address-Cím:** Hol kezdődik a memóriában a program, melyik címen.
2. **Name - Név:** A helyfoglaló program, eszközvezérlő - modul - neve.
3. **Size - Mérete:** Az adott program mérete, helyfoglalása bájtokban.
4. **Type - Típus:** A helyfoglaló típusa, program, eszközvezérlő, környezeti adatok, *...stb.*

MS-DOS 6:

1. **Segment - Szegmenscím:** Hol kezdődik a memóriában a program, melyik címen.
2. **Total - Összes:** Az adott program mérete, helyfoglalása bájtokban.
3. **Name - Név:** A helyfoglaló program, eszközvezérlő - modul - neve. Az előző, és ez csak fel van cserélve a két verzióban. Így egymás mellé került a két számadatos információ, és a két szöveges a jobb áttekinthetőség kedvéért.
4. **Type - Típus:** A helyfoglaló típusa, program, eszközvezérlő, környezeti adatok, *...stb.*

A részletes memóriatérkép után következnek az összegző adatok, amelyek szintén eltérőek a két legelterjedtebb DOS verzió között. Az MS-DOS 6-ban már az UMB memóriáról is részletes térképet kapunk, a kihasználtság szempontjából. Utána pedig a létező memóriatípusok szerint, soronként megkapjuk azok kihasználtságát.

Az előbbi listák olyan sok információt adtak, hogyha csak szabad memória-területeket keresünk, akkor szinte ki sem igazodunk a listákban. Ennek felderítésére szolgál a "/F" paraméter, ami kigyűjti a memória szabad területeit, és azokról tájékoztat. Erre már csak azoknak van lehetőségük, akik MS-DOS 6-ban dolgoznak.

```
C:\DOS>MEM /F

Free Conventional Memory:

  Segment          Total
  -------    -----------------
   0156A             240     (0K)
   01579          88 992    (87K)
   02B33         478 400   (467K)

  Total Free: 567 632   (554K)

Free Upper Memory:

  Region   Largest Free      Total Free       Total Size
  ------   --------------    --------------   --------------
      1     2 880    (3K)     2 880    (3K)     2 880    (3K)
      2    73 360   (72K)    75 376   (74K)   159 776  (156K)

C:\DOS>_
```

A kapott listában szintén két fő terület különíthető el, a hagyományos memória, és a felső memóriablokkok (UMB) területének ismertetése. Mint láthatjuk a hagyományos memóriában három érték is szerepel. Ha visszalapozunk az előző oldalakon látható listákra, akkor láthatjuk, hogy mindegyik lista egymás után készült, ugyanarról a memóriahelyzetről. Ebben láthatjuk, hogy a **MEM** parancs is foglal két területegységet. Először egy kisebbet a környezeti adatai (Environment) részére, egy nagyobbat pedig maga a program (Program) foglal el. A mostani listában a feltüntetett üres helyek közül az első kettő a **MEM** parancs által foglalt terület, a következő pedig az utána még szabadon lévő szabad hely.

Gondoljuk csak át, a **MEM** parancs a munkája közben azt olvassa ki, hogy saját maga is foglalja a memóriát, de a kiírások végeztével már kilép a memóriából, és nem fogja azt tovább foglalni, ezért jelzi ki a "/F" paraméter használatakor szabadnak, és azért szerepel több sorban. Így következetes a kiírás, összhangba tudjuk állítani az előző képernyőlistákkal, és jobban meg tudjuk érteni, mintha össze lennének adva a szabad értékek.

Ezt a parancsot leginkább arra fogjuk tudni használni, hogy a felső memóriablokkokat minél jobban meg tudjuk "tömni" programokkal, azaz minél több programot tudjunk feltölteni az UMB területére, ezáltal még több helyet felszabadítva a hagyományos memóriában.

Az utolsó példánkban azt nézhetjük meg, hogy egy adott program (modul) helyfoglalását hogyan kérdezhetjük le a számítógéptől. Erre olyankor lehet szükségünk, ha az adott program nem abba a memóriatípusba töltődött, ahová mi szerettük volna.

```
C:\DOS>MEM /M IMOUSE

'IMOUSE is using the following memory:

  Segment  Region       Total         Type
  -------  ------  ---------------  --------
   0D209       2      10720   (10K)  Installed Device=IMOUSE
                    ---------------
  Total Size:         10 720   (10K)

C:\DOS>_
```

Amint láthatjuk, megkaptuk a keresett modulról a kért információt - az előzőekben megismert tagolásban -, hogy hol, mennyi helyet foglal a memóriában. Nézzük meg az előzőekben kapott listákat, és válasszunk olyan modulnevet, ami többször is előfordul a memóriában, mint például "IO" vagy "COMMAND" *...stb.*

A továbbiakban a konfigurációs fájlok ismertetésekor, az adott parancs bemutatása olyan formában fog történni, hogy megmutatom milyen konfigurációs fájlt használtam, és a **MEM /C** parancs mit írt ki eredményül. A jobb érthetőség kedvéért az MS-DOS 5.00-ában használható **MEM** parancs eredményeit mutatom meg, mert ez nem közöl olyan részletes listákat, mint az MS-DOS 6-ban lévő változata. Igaz ugyan, hogy az részletesebb lista, de egy kezdőnek éppen elég lesz ezeket az információkat is feldolgozni. Ahol esetleg szükséges lesz az MS-DOS 6 **MEM** parancsát használni, ott erre külön fel fogom hívni a figyelmet.

2.1.1. XT (640 kB vagy 1 MB RAM)

Ha egy XT számítógépbe 640 kB memória van ténylegesen szerelve, akkor az csak a rajzon látható 640 kB alapmemóriát használhatja ki. Ez azt jelenti, hogy a betöltésre kerülő eszközmeghajtó programok, és rezidens programok a memóriába a "00000"-ás memóriacímtől kezdve kerülnek letárolásra, majd a készenléti jel megjelenésekor szabadon lévő memóriát használhatjuk a továbbiakban a programjaink futtatására. Vigyázni kell arra, hogy ne indítsunk el sok ***rezidens program***ot , mert kevés marad a ***felhasználói program***jainknak.

Az olyan XT számítógépek, amelyekben 1 MB memória van, annyiban különböznek a 640 kB-os kiépítésűektől, hogy a felső 384 kB-ban úgynevezett ***felső memóriablokkok*** helyezkednek el. Ezeket szoftveresen nem tudjuk kihasználni, csak az említett BIOS másolat, ...*stb.* kaphat itt helyet, gyorsítási célból. (Természetesen csodákat, vagy szemmel látható javulást ne várjunk egy XT számítógéptől.)

2.1.2. AT-286 (1-4 MB RAM)

A fejlettebb technológiával (**A**dvanced **T**echnology) készült számítógépek esetén, már lehetőségünk van, akár 4 MB RAM-mal is felszerelni a számítógépünket. Ez a több memória gyorsabb futást eredményezhet a számítógépünkben, de csak akkor, ha a szükséges segédprogramokat elindítjuk.

Most több lépcsőben fogjuk megvizsgálni, hogy milyen lehetőségeink vannak a különböző memóriakiépítések mellett. Ha több, mint 640 kB memória van a számítógépünkben, akkor két eset lehetséges:

- Az alapesetben olyan a számítógépünk hardveres felépítése, hogy folyamatosan helyezkedik el benne a memória. Ez teljesen szokványos, és magától érthetődő, nem is kíván bővebb magyarázatot. A ***640 kB hagyományos memória*** után következnek a ***felső memóriablokkok*** (***384 kB*** méretben), majd az 1 MB felett a ***magas memóriaterület***, és a ***kiterjesztett memória***.
- Vannak azonban olyan számítógépek is, amelyekben a jobb szoftveres kihasználhatóság érdekében olyan hardvermegoldást alkalmaztak - és ez elég gyakori -, hogy a ***640 kB hagyományos memória*** után nem következik folyamatosan a felső memóriablokkos terület, hanem a ***magas és a kiterjesztett memóriában*** folytatódik a memóriák címzése. Ez valójában azt jelenti - ha az ábrát megnéz-

*Rezidens program*nak nevezzük azokat a programokat, amelyek futás után nem szabadítják fel maguk után a teljes memóriát, hanem egy részt folyamatosan lefoglalva tartanak maguknak. A megszakításrendszeren keresztül folyamatosan figyelik a számítógép működését, és ha szükséges, akkor közbelépnek. Ilyenek például a billentyűkombinációra előjövő számológép vagy szótárprogramok.

*Felhasználói program*nak azokat a programokat nevezzük, amelyekkel a konkrét napi termelő vagy adminisztrációs vagy egyéb munkáinkat végezzük, amiért valójában a számítógépet üzembeállítottuk.

zük -, hogy van egy 384 kB-os kihagyás a memóriarajzunkon. Így nincs lehetőségünk kihasználni a felső memóriablokkok adta gyorsítási lehetőségeket, viszont van egy szoftveresen elérhető magas és kiterjesztett memóriaterületünk, amit a rezidens programjaink tárolására felhasználhatunk, így több helyet hagyva a hagyományos memóriaterületen a felhasználói programjainknak.

DEVICE= Eszközmeghajtó betöltése

Használata: **DEVICE=**[*hely*] *{eszközmeghajtó program neve, paraméterei}*

DEVICE=Eszköz

hely: A megadott helyen - ***meghajtó, elérési út*** - keresi az eszközmeghajtó programot. Ha nem adjuk meg, akkor alapértelmezésben a betöltőlemez gyökérkönyvtárában keresi. ***(A betöltőlemez fogalmát gondoljuk át!)***

Az eszközmeghajtó program neve - ami általában SYS vagy EXE kiterjesztésű-, és a megadott helyen megtalálható. ***A paramétereit*** mindig szóközzel elválasztva kell megadni, de azok szintaktikáját már az adott parancsnál kell megnézni. Ugyanis van olyan, amelyiknél több paraméter megadható szóközzel elválasztva, de van olyan, amelyiknél zárójelben kell megadni, vesszővel elválasztva a paramétereket.

Ez nem egy parancs, hanem egy utasítás, ami csak a CONFIG.SYS fájlban használható az eszközmeghajtó programok betöltésére.

DEVICE=HIMEM.SYS Memóriavezérlő

Használata: **DEVICE=**[*hely*] **HIMEM.SYS** [*/TESTMEM:OFF*] *...stb.*

HIMEM= HIGH MEMORY - Magas memória

hely: A HIMEM.SYS fájl helye (*meghajtó: elérési út*).

/TESTMEM:OFF: Az MS-DOS 6.20 használóinak számítógép indulásakor, amikor ez a program betöltődik, alapértelmezésben leteszteli a memóriát. Ezzel a paraméterrel tudjuk letiltani a tesztelést.

(Az MS-DOS 6.20-ban már elég sok paramétere van ennek a programnak, de ezek ismertetését a megfelelő verziójú DOS felhasználói kézikönyvéből keresd ki. Most éppen elég lesz neked, ha a memóriakezelés alapjait megérted, és megfelelően tudod majd hasznosítani.)

Ha a CONFIG.SYS fájlban elhelyezzük ezt az utasítást, és az előbb említett 286-os számítógépünk van -vagy magasabb processzorszámú-, akkor elérhetővé válik számunkra a ***magas*** és a ***kiterjesztett memóriaterület***. Ha a számítógépünk indulásakor azt írja ki, hogy nem sikerült az említett utasítást végrehajtani, az azt jelenti, hogy nincs a számítógépünkben ilyen típusú memória, azaz folyamatosan történik a memória megcímzése (286-os), és nincsen csak maximum 1 MB memória a számítógépünkben.

A most következő példában megnézhetjük, hogy mit ír ki a **MEM** parancs, ha nincs egyetlen konfigurációs fájl sem a betöltő lemezünkön. Ilyenkor a három rendszerfájl töltődik be, és az AUTOEXEC.BAT hiányában bekéri a dátumot és az időt, majd kiírja az alapértelmezett készenléti jelet. Ezt a helyzetet tudjuk szimulálni, ha egy hajlékonylemezt leformázunk "**FORMAT a: /s**" paranccsal, és utána erről indítjuk újra a rendszert, például az [ALT]+[CTRL]+[DEL]-lel úgy, hogy nem vesszük ki a lemezt a meghajtóból:

Az elkövetkező **MEM** parancsok listáiban szerepelni fog egy olyan modul, program, aminek a neve "SNIPPER". Ez volt az a program, amivel a képernyőkről "levadásztam" a parancsok eredményeit, melyek a könyvben valósághűen mutatják egy-egy parancs eredményét. Ezért szerepel nálam ez a modul is a listában, és ezért nem fog nálad megjelenni.

```
C:\DOS>MEM /C

Conventional Memory :

  Name                 Size in Decimal       Size in Hex
-------------       ----------------------   -------------
  MSDOS                57456       ( 56.1K)       E070
  COMMAND               4704       (  4.6K)       1260
  SNIPPER               3728       (  3.6K)        E90
  FREE                    64       (  0.1K)         40
  FREE                    48       (  0.0K)         30
  FREE                589168       (575.4K)      8FD70

Total  FREE :         589280       (575.5K)

Total bytes available to programs :                          589280    (575.5K)
Largest executable program size :                            589168    (575.4K)

  15859712 bytes total contiguous extended memory
  15859712 bytes available contiguous extended memory

C:\DOS>_
```

Most készítsünk erre a lemezünkre egy CONFIG.SYS fájlt, ami csak az újonnan megtanult utasítást tartalmazza. Erre használhatjuk az **EDIT** szövegszerkesztőnket is, de sokkal egyszerűbb, ha a következő eljárást alkalmazzuk:

```
A:\>COPY CON CONFIG.SYS
DEVICE=c:\dos\HIMEM.SYS
^Z
      1 file(s) copied

A:\>_
```

Ebben a használati formában a merevlemezünkön van egy "DOS" nevű könyvtár, amiben feltehetően az összes DOS külső parancs és egyéb fájl megtalálható. Ez a meghajtó nem lehet esetünkben tömörített meghajtó (csak MS-DOS 6-ban), mert annak definiálását mi most nem tesszük meg, anélkül pedig nem használható.

Ha teljesen önálló, bármikor használható rendszerlemezt akarunk készíteni, akkor a következő eljárást kell követnünk minden esetben:

```
C:\DOS>COPY HIMEM.SYS a:\

      1 file(s) copied

C:\DOS>a:

A:\>COPY CON CONFIG.SYS
DEVICE=HIMEM.SYS
^Z
      1 file(s) copied

A:\>_
```

Ki kell másolnunk azt a fájlt is a hajlékonylemezre, amelyiket a konfigurációs fájlokban használni fogjuk. Ilyen esetben nem szükséges, hogy legyen merevlemezünk, ahonnét az adott programot betöltsük.

A továbbiakban nem fogom külön kiemelni, és az előbbi sorokat ismételten leírni, de minden konfigurációs fájlban lévő parancsnak meg kell lennie a hajlékonylemezünkön, vagy meg kell adnunk a helyét a merevlemezen. A példákban használt **MEM** parancsnak vagy a hajlékonylemezen kell lennie, vagy meg kell adnunk a helyét a parancs kiadásakor! Erre nagyon figyeljünk oda, mert a bemutatott példák esetében még nincs AUTOEXEC.BAT fájlunk, amiben megadnánk a keresési utakat, ami biztosítaná a parancs elérését.

```
A:\>c:\dos\MEM /C
     :
     :
     :
A:\>PATH c:\DOS;
PATH C:\DOS;

A:\>MEM /C
     :
     :
     :
A:\>_
```

Az előbbi két példa mutatta, ha nincs a MEM.EXE fájl a hajlékonylemezünkön, akkor hogyan kell elindítani a parancsot. Bár a programok indítása, keresési út, és hasonló fogalmakkal az előző részben kellett megismerkedni. Ha mégis hiányosságaid lennének ez ügyben, akkor lapozd fel bátran a "*DOS alapismeretek I.*" című tankönyvemet, és ismételd át az ott leírtakat.

```
HIMEM: DOS XMS Driver, Version 2.77 - 02/27/91
XMS Specification Version 2.0
Copyright 1988-1991 Microsoft Corp.

Installed A20 handler number 1.
64K High Memory Area is available.
    :
    :
    :
A:\>MEM /C

Conventional Memory :

  Name                 Size in Decimal       Size in Hex
------------       ---------------------   ------------
  MSDOS                57456      ( 56.1K)       E070
  HIMEM                 3200      (  3.1K)        C80
  COMMAND               4704      (  4.6K)       1260
  SNIPPER               3728      (  3.6K)        E90
  FREE                    64      (  0.1K)         40
  FREE                    48      (  0.0K)         30
  FREE                585952      (572.2K)       8F0E0

Total  FREE :         586064      (572.3K)

Total bytes available to programs :                    586064    (572.3K)
Largest executable program size :                      585952    (572.2K)

  15859712 bytes total contiguous extended memory
  15859712 bytes available contiguous extended memory

A:\>_
```

Ha sikeresen lefutott a program, amint az a képernyőrajzon is látható, akkor használhatjuk a kiterjesztett és a magas memóriaterületet. A fenti ablak a számítógép indulása utáni állapotokat tükrözi, és a készenléti jel megjelenéséig lévő kiírások, a HIMEM.SYS szabályos betöltésekor íródik ki. Mint láthatjuk, az MS-DOS a hagyományos memóriába töltődött be 56,1 kB méretben. A HIMEM.SYS is megtalálható utána, ami 3,1 kB-ot foglal csak, és biztosítani fogja, hogy a kiírásban látható további ~15 MB helyet gazdaságosan ki tudjam használni. Utána következik a COMMAND.COM, aminek egy része rezidensen bekerül a memóriába, de a nagyobb része lemezen marad. Ezért kell megadnunk a "COMSPEC" környezeti változóban a parancsfeldolgozó helyét.

A következő parancs, és példa ismertetése után lapozz vissza ide ismét, és nézd meg a **MEM** parancs utolsó néhány sorát. Itt még azt mondja, hogy rendelkezésre áll a teljes kiterjesztett memória, pedig már betöltöttük a HIMEM.SYS-t. Csak ezért nem az a felirat jelent meg, mint a következő példában, mert nem használtuk semmire sem.

DOS=HIGH — DOS helye: Magas memória (HMA)

Használata: **DOS=HIGH** vagy **DOS=LOW**

HIGH=Magas ; LOW=Alacsony.

A számítógép indulásakor a DOS rendszerfájljait nem csak a ***hagyományos memóriába*** fogja tölteni (**DOS=LOW**), hanem a ***magas memóriába*** is (**DOS=HIGH**). Nem a teljes terjedelmében, csak annyit, amennyit tud, de ez is nagy segítség, amint a példa is érzékelteti:

```
A:\>MEM /C

Conventional Memory :

  Name                 Size in Decimal       Size in Hex
------------       ---------------------   -------------
  MSDOS                13120       ( 12.8K)       3340
  HIMEM                 1184       (  1.2K)        4A0
  COMMAND               2624       (  2.6K)        A40
  SNIPPER               3728       (  3.6K)        E90
  FREE                    64       (  0.1K)         40
  FREE                    48       (  0.0K)         30
  FREE                634368       (619.5K)      9AE00

Total  FREE :         634480       (619.6K)

Total bytes available to programs :                    634480    (619.6K)
Largest executable program size :                      634368    (619.5K)

  15859712 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
  15663104 bytes available XMS memory
           MS-DOS resident in High Memory Area

A:\>_
```

Mint láthatjuk az előző példához képest csak egy kicsit változott az eredmény, de ez mégis jelentős a memóriakihasználást tekintve. A **DOS=HIGH** utasítás hatására a DOS egy része felkerült a magas memóriába. Láthatjuk, hogy az előbbi 56,1 kB helyett most csak 12,8 kB-ot foglal el a hagyományos memóriában, és a HIMEM.SYS is kevesebb helyet igényel. A kiírás utolsó soraiban a kiterjesztett memória kihasználását részletezi ki. 15.859.712 bájt a teljes kiterjesztett memória ebben a számítógépben, amiből 0 bájt áll rendelkezésünkre (Available). Rendelkezésünkre áll viszont 15.663.104 bájt XMS memória, ami viszont szoftveresen kihasználható. A különbség 192 kB, amiből 64 kB a HMA terület (az 1 MB feletti első majdnem 64 kB), és 128 kB az 1 MB alatt. Ennek az az oka, hogy ebben a számítógépben a kiterjesztett memória már az 1 MB alatt elkezdődik 128 kB-tal. Így az UMB terület nem 384 kB, hanem csak 256 kB.

Megjelent egy új sor is : "**MS-DOS resident in High Memory Area**", vagyis "**az MS-DOS állandóan a magas memóriában tartózkodik**". Ide került fel a DOS a hagyományos memóriából.

Ha részleteiben meg akarjuk érteni a számítógépünk memóriakezelését, akkor ilyen szintre le kell mennünk, és a kiírt adatokat pontosan leellenőrizni, hogy melyik bájtot melyik program foglalja le, nehogy veszendőbe menjen egyetlen bájt is. Különösen fontos ez akkor, ha csak 1-2 MB memória van a számítógépünkben. Olyankor is kihasználhatjuk a teljes memóriát, a különféle programjainkkal.

Ha már töltöttünk valamilyen programot a magas memóriába, akkor a DOS nem fog tudni betöltődni oda, ilyenkor automatikusan az alacsony (Low) - hagyományos - memóriaterületre fog betöltődni.

Ha nem használjuk - vagy valamiért nem indul ez a HIMEM.SYS -, akkor a következő üzenetet kapjuk:

```
HMA not available : Loading DOS low

A:\>_
```

Ilyenkor a DOS a hagyományos memóriába kerül, azaz magában a **DOS=HIGH** utasítás nem használható.

A *kiterjesztett memóriaterületet* pedig úgy tudjuk kihasználni, hogy az egyes programok futásuk közben használják ezt a területet (például egyes szövegszerkesztők, táblázatkezelők, ...*stb.*) általában adattárolásra, mert a processzor valós módjában csak a hagyományos memóriában futhatnak a programok. Használhatjuk még ezt a területet *memórialemeznek* és *Cache-tárolónak* is (*lásd ott!*). A Windows *standard* üzemmódban futtatható, és ebben az esetben ez a terület is használható programok futtatására.

2.1.3. AT-386 (1 MB -4GB RAM)

A 32 bites számítógépek esetén, már lehetőségünk van a számítógépben lévő memóriaterületek jobb kihasználására. Szoftveres céljainkra is felhasználhatjuk a *felső memóriablokkok* közül még szabadon lévő blokkokat egy segédprogram használatával. Ezeken a számítógépeken már *EMS memóriát* is szimulálhatunk a *kiterjesztett memóriaterületen*. Erre van szükség akkor, ha egy program igényli az EMS memóriát, de nekünk nincs ilyen memóriabővítő-kártyánk.

EMM386.EXE — **UMB vezérlő és EMS szimulátor**

Használata: **DEVICE**=[*meghajtó:*] [*elérési_út*] **EMM386.EXE** [*RAM*] [*NOEMS*] ...*stb.*

EMM=Extended Memory Manager - A kiterjesztett memória menedzsere.

Ez a program a felső memóriablokkok kezelését látja el, és a kibővített memóriát szimulálhatja a kiterjesztett memóriaterületen.

meghajtó: elérési_út: Az EMM386.EXE fájl helyét adjuk meg vele. Akkor hagyható el, ha az a betöltő lemez gyökérkönyvtárában található. Általában ez a parancs a DOS nevű könyvtárunkban van, például a "C:" meghajtón, így a helyes megadása:

DEVICE=C:\DOS\EMM386.EXE

RAM: A paraméter használatával a felső memóriablokkok - az **UMB** - elérését biztosítjuk. A paraméter elhagyása esetén nem tudjuk elérni a felső memóriablokkokat, ez az alapértelmezés (Az MS-DOS 6-tól felfelé nem szükséges a paraméter használata, mert anélkül is eléri az UMB-t). A paraméter használata után használhatjuk a **DEVICEHIGH** utasítást, amivel az eszközmeghajtó programokat az **UMB** területre tölthetjük. Így több hely marad a hagyományos memóriaterületen.

NOEMS: Ne hozzon létre EMS memóriát a kiterjesztett memóriaterületen. Ha nem tiltjuk le - nem adjuk meg a paramétert -, akkor létrehoz egy 256 kB alapértelmezett méretű EMS-t.

(A parancsnak nagyon sok kapcsolója van még, ennek ismertetésére nem térek ki, mert az alapvető ismeretekhez ez is elegendő. A többi paraméter használata, már csak pontosításhoz, aprólékosabb kidolgozáshoz szükséges, ami nem feladata ennek a tankönyvnek.)

Ha az itt látható CONFIG.SYS fájlt lefuttatjuk, akkor a következő eredményre jutunk:

```
device=HIMEM.SYS
DOS=HIGH
device=EMM386.EXE
device=RAMDRIVE.SYS 362 512 64 /e
```

```
MICROSOFT Expanded Memory Manager 386  Version 4.20.06X
(C) Copyright Microsoft Corporation 1986, 1990

EMM386 successfully installed.

  Available expanded memory . . . . . .   256 KB

  LIM/EMS version . . . . . .   4.0
  Total expanded memory pages . . . . . .   40
  Available expanded memory pages . . . . .   16
  Total handles . . . . . .   64
  Active handles  . . . . . .   1
  Page frame segment  . . . . . .  D000 H

EMM386 Active.


Microsoft RAMDrive version 3.06 virtual disk P:
     Disk size: 362k
     Sector size: 512 bytes
     Allocation unit: 1 sectors
     Directory entries: 64

A:\>_
```

Elöljáróban jegyzem meg, hogy a RAMDRIVE.SYS csak azért szerepel a példában, hogy használjuk is valamire a kiterjesztett memóriát. Az új sor a CONFIG.SYS fájlban az EMM386.EXE fájl, amit paraméterek nélkül használva a következőket kapjuk:

Alapértelmezésben kapunk egy 256 kB méretű EMS memóriát (expanded memory), amit az EMM386.EXE utáni **NOEMS** paraméterrel letilthatunk, ha nincs szükségünk EMS memóriára.

```
A:\>MEM /C

Conventional Memory :

  Name                 Size in Decimal       Size in Hex
-------------       ---------------------   -------------
  MSDOS                13200       ( 12.9K)       3390
  HIMEM                 1184       (  1.2K)        4A0
  EMM386                8400       (  8.2K)       20D0
  RAMDRIVE              1184       (  1.2K)        4A0
  COMMAND               2672       (  2.6K)        A70
  SNIPPER               3728       (  3.6K)        E90
  FREE                    64       (  0.1K)         40
  FREE                    48       (  0.0K)         30
  FREE                624608       (610.0K)      987E0

Total  FREE :         624720       (610.1K)

Total bytes available to programs :                    624720    (610.1K)
Largest executable program size :                      624608    (610.0K)

    655360 bytes total EMS memory
    262144 bytes free EMS memory

  15859712 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
  14931968 bytes available XMS memory
           MS-DOS resident in High Memory Area

A:\>_
```

A másik szolgáltatását az EMM386.EXE-jének, a felső memóriablokkokhoz való hozzáférést viszont nem kaptuk meg, mert akkor szerepelne a **MEM** parancs eredményében az "**Upper Memory**"-ra vonatkozó kiírás. Ilyet viszont nem láthatunk a listában. Láthatjuk viszont a 256 kB (262.144 bájt) EMS memóriát, amibe akár be is tölthetnénk valami programot. Ismét számoljunk egy kicsit! A 15.859.712 bájt kiterjesztett memória a következőképpen oszlik meg az adott programok között:

HMA: 64 kB ; DOS: 128 kB ; RAMDRIVE: 362 kB ; EMS: 256 kB ; UMB-re foglalt : 96 kB ; maradék 14.931.968 bájt.

DOS=UMB — A DOS használhatja az UMB-t.

Használata: **DOS=UMB**

UMB=Upper Memory Block - Felső memóriablokkok

Engedélyezi a DOS számára, a felső memóriablokkok használatát. Használni viszont csak akkor tudjuk, ha az **EMM386** paranccsal előkészítettük azokat a következő utasítással:

```
DEVICE=c:\dos\EMM386.EXE RAM
```

Ekkor a **MEM** parancs kiírásában is külön szerepelteti a felső memóriablokkok kihasználtságát. Ha a **MEM** parancs kiírásában ez nem szerepel, akkor nem sikerült a felső memóriablokkokat kihasználnunk (esetleg hibás lehet valahol a CONFIG.SYS fájlunk).

Ekkor ellenőrizzük, hogy fut-e egyáltalán az **EMM386.EXE** program (**MEM /c** kiírja-e a listájában?), vagy esetleg a számítógépünk csak 286-os.

A mostani CONFIG.SYS fájlban használjuk az EMM386.EXE programnál a **RAM** paramétert, ami az UMB-hez való hozzáféréshez szükséges.

```
device=HIMEM.SYS
DOS=HIGH
device=EMM386.EXE RAM
devicehigh=RAMDRIVE.SYS 362 512 64 /a
```

A memórialemez definíciójában pedig a "/a" paraméter is megjelenik, minek hatására az nem az *Extended*, hanem az *ExpAnded* memóriába fog kerülni. Figyeljük meg a méreteket is:

```
  Available expanded memory . . . . . .   256 KB

  LIM/EMS version . . . . . .   4.0
  Total expanded memory pages . . . . . .   40
  Available expanded memory pages . . . . .   16
  Total handles . . . . . .   64
  Active handles . . . . . .   1
  Page frame segment . . . . . .   D000 H

  Total upper memory available . . . . . .   31 KB
  Largest Upper Memory Block available . . .   31 KB
  Upper memory starting address . . . . . .   C800 H

EMM386 Active.


Microsoft RAMDrive version 3.06 virtual disk F:
     Disk size: 256k
     Sector size: 512 bytes
     Allocation unit: 1 sectors
     Directory entries: 64

A:\>_
```

Ha figyelemmel kísértük a rendszer indulását, akkor láthattuk, hogy ismét létrehozta a 256 kB alapértelmezett méretű EMS memóriát, mert nem tiltottuk le a **NOEMS** paraméterrel. A **RAM** paraméter használata következtében megjelent néhány új sor a kiírások között, amiket ki is emeltem. A felső memóriablokkokat alakította ki, amelyre 96 kB áll a rendelkezésére. Ebből 64 kB-ot elfoglal az EMS memória "lapja" és még rendelkezésre áll (Available) 31 kB (1 kB pedig egyéb célokra foglalt). Az EMS "lapja" alatt az értendő, hogy a DOS a processzor valós üzemmódjában nem "lát" az 1 MB fölé, ezért az 1 MB alatti területen - az UMB-ben - van egy 64 kB-os lap (Frame), amit viszont tud kezelni. Ha további területeket szeretne látni a 256 kB-os EMS memóriából, akkor az EMS vezérlő program, újabb 64 kB-ot lapoz be erre a területre, felülírva az előző tartalmát. Így egyszerre az EMS memóriából mindig csak 64 kB-ot lát a DOS, de ezt a "titkot" nem árulja el a programnak, ami az EMS-t használja, mert az mindig megkapja a kért adatokat erről a memóriaterületről.

Ha egy pillantást vetünk a memórialemezünkre, akkor láthatjuk, hogy a mérete igencsak összement. Ennek az az oka, hogy a "/a" paraméter hatására az EMS területre raktuk, ami viszont csak 256 kB méretű. Így a RAMDRIVE.SYS csökkentette a méretét.

Ebben a példában már végre együtt van minden olyan utasítás, paraméter, ami a memória változatos kihasználását elősegíti:

```
device=HIMEM.SYS
DOS=HIGH
device=EMM386.EXE RAM
DOS=UMB
devicehigh=MOUSE.COM
```

Megjelent végre a **DOS=UMB** utasítás, aminek hatására használhatja a DOS az UMB területet. Hiába használjuk az EMM386.EXE-t, hiába alakította ki - mint az előző példában láthattuk - akkor sem tudtuk volna használni. Most engedélyeztük a DOS számára a használatot, és egy eszközmeghajtó programot be is fogunk tölteni oda.

```
A:\>MEM /c

Conventional Memory :

  Name                 Size in Decimal       Size in Hex
------------       ----------------------   ------------
  MSDOS                13136      ( 12.8K)       3350
  HIMEM                 1184      (  1.2K)        4A0
  EMM386                8400      (  8.2K)       20D0
  COMMAND               2672      (  2.6K)       A70
  SNIPPER               3728      (  3.6K)        E90
  FREE                    64      (  0.1K)         40
  FREE                    48      (  0.0K)         30
  FREE                625872      (611.2K)      98CD0

Total  FREE :         625984      (611.3K)

Upper Memory :

  Name                 Size in Decimal       Size in Hex
------------       ----------------------   ------------
  SYSTEM              163840      (160.0K)      28000
  MOUSE                10720      ( 10.5K)       29E0
  FREE                 21984      ( 21.5K)       55E0

Total  FREE :          21984      ( 21.5K)

Total bytes available to programs (Conventional+Upper) :      647968    (632.8K)
Largest executable program size :                             625872    (611.2K)
Largest available upper memory block :                         21984    ( 21.5K)

    655360 bytes total EMS memory
    262144 bytes free EMS memory

  15859712 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
  15269888 bytes available XMS memory
           MS-DOS resident in High Memory Area

A:\>_
```

Mint láthatjuk sokkal bővebb listát kaptunk, mint eddig. Megjelent végre a **MEM** parancs listájában az "**Upper Memory** :" felirat is, ami az UMB foglaltságáról tájékoztat bennünket. Ez arról is tájékoztat, hogy sikerült létrehozni ezt a területet, és újabb memóriaterületeket vontunk be, ahová eszközmeghajtó programokat tölthetünk. Ezzel is több memória marad a felhasználói programjainknak.

Újdonságként jelent meg, hogy az egérmeghajtó programot a felső memóriablokkok területére töltöttük be, ezáltal több helyet hagyva a hagyományos memóriában. (Sok helyünk ugyan nem maradt utána, de akit érdekel a téma, az nézze meg ugyanezt egy olyan számítógépen, ahol fut az MS-DOS 6, és látni fogja, hogy ott sokkal jobb az UMB kihasználtsága.)

Megjelent szintén egy új sor az összegzések között, ami a felső memóriablokkok területén lévő szabad helyekről tájékoztat bennünket. Ide még további eszközmeghajtó programokat tölthetünk úgy, hogy nem a hagyományos memóriából veszünk el helyet.

DEVICEHIGH= Eszközvezérlő helye: UMB

Használata: **DEVICEHIGH=** *{eszközmeghajtó neve, elérési úttal, paraméterekkel}*

DEVICE - eszköz ; HIGH- magas {Az eszközmeghajtó kerüljön a memória magasabb területeire. Nem a **HMA**-ba - ne keverjük össze -, csak az **UMB** területére}

Ez az utasítás magában nem használható. Az egyenlőségjel jobb oldalán megadott eszközmeghajtó programot tölti be a felső memóriablokkokba, ha van hely. Ha nem sikerül, akkor automatikusan a hagyományos memóriába tölti, és ezt egy rövid üzenetben tudatja velünk a ***boot-folyamat*** során (rendszerinduláskor). Például ezért is kell figyelemmel kísérni a betöltési folyamatot, hogy az ilyen jellegű üzeneteket észrevegyük.

Az előző példában láthattuk az egérmeghajtó program betöltésének módját a felső memóriaterületre.

2.2. Lemezek

DBLSPACE.SYS Röptömörítő

Használata: **DEVICEHIGH=**[*hely*] **DBLSPACE.SYS** [*/MOVE*]

DBLSPACE=DUOBLE SPACE - dupla hely ; MOVE - mozgat.

Ezt a sort nem kell nekünk beírni, mert ha használunk tömörített meghajtót, akkor a **DBLSPACE** program automatikusan elhelyezi ezt a sort a CONFIG.SYS fájlban.

Ez az eszközmeghajtó program elérést biztosít a tömörített meghajtókhoz. A számítógép indulásakor a DBLSPACE.**BIN** program alapértelmezésben a hagyományos memóriába kerül, melyet ezzel a paranccsal tudunk az **UMB** területre helyezni.

A program csak az MS-DOS 6-tól felfelé található meg!

/MOVE: A végső helyére mozgatja a DBLSPACE.BIN-t a memóriában.

DBLSPACE.EXE — Tömörített meghajtók kezelése

A program használható parancssorból is felparaméterezve, de hasznosabb paraméterek nélkül elindítani, használni a program menüvezérelt formáját az új meghajtók létrehozásakor, lévén, hogy ez nem olyan gyakori feladat, így jobb ha azt nyugodtan, és körültekintően végezzük el. Most indítsuk el a DBLSPACE.EXE programot, amiben menüvezérelt formában fogjuk létrehozni az új tömörített meghajtóinkat. Lapozz vissza a(z) 5. oldalra, és nézd át újra a merevlemez felosztási tervét. Most aszerint a példa szerint fogom bemutatni, hogyan is kell megvalósítani egy ilyen tervet.

Merevlemez tömörítése

```
C:\DOS>DBLSPACE
```

Megjelenik a **DBLSPACE** képernyője, ami felépítésében hasonlít az **EDIT** szövegszerkesztőnél látottakhoz. Általában minden programnál hasonló felépítéssel találkozunk, mondhatni ez már szabvánnyá alakult. A képernyő legfelső sorában található a *menüsor*, amiben *menüneveket* találunk. A menüneveket kiválasztani kétféleképpen lehet: [ALT]+[KEZDŐBETŰ]-vel - például a **Drive** menüt [ALT]+[D]-vel lehet megnyitni. A másik módszer, amit általában ritkán alkalmazunk, hogy lenyomjuk az [ALT] billentyűt, minek hatására az egyik menünév kiemelt színű lesz, a többinek pedig az egyik betűje - általában a kezdőbetűje - lesz kiemelt színű. Ezt a módszert akkor alkalmazzuk, ha a menüt nem a kezdőbetűjével lehet legördíteni, hanem egy másik betűjével. Miután megjegyeztük melyik billentyűkombinációval kell a menüt aktiválni, használhatjuk az első módszert.

A képernyő közepén lévő ablakban - keretben - láthatók a már meglévő tömörített meghajtók adatai, amik most még nincsenek nekünk, de hamarosan több is lesz. Válasszuk ki a **Compress** menü **Create New...** menüpontját:

```
Drive  Compress  Tools  Help

        Select the drive you want to use. DoubleSpace will convert
        that drive's free space into a new compressed drive.

                          Current          Projected Size
                Drive     Free Space       of New Drive
                C:          2.0 MB           4.0 MB
                D:         55.0 MB         110.0 MB

        To accept the current selection, press ENTER.

        To select a different drive, press the UP ARROW or DOWN
        ARROW key until the drive you want is selected, and then
        press ENTER. If there are more drives than fit in the
        window, you can scroll the list by pressing the UP ARROW
        DOWN ARROW, PAGE UP, or PAGE DOWN key.

ENTER=Continue  F1=Help  ESC=Previous screen
```

Megjelentek a képernyő közepén az ablakban azok a meghajtók, amelyiken jelenleg (Current) van szabad hely (Free Space). Mellette a tervezett (Projected) mérete látható, az alapértelmezett kétszeres tömörséggel számolva.

Select the drive you want to use. DoubleSpace will convert that drive's free space into a new compressed drive.

"Válaszd ki azt a meghajtót, amit használni akarsz. A DoubleSpace átkonvertálja a meghajtó szabad helyeit egy új tömörített meghajtóvá."

Válasszuk ki a "D:" meghajtót, amelyiken két tömörített meghajtót terveztünk létrehozni.

Ennyi szabad helyet fog hagyni a "D:" meghajtón: **2.0 MB**
Becsült tömörségi arány az új meghajtón: **2.0 to 1**
Az új meghajtó betűjele: **E:**

Continue

Ha megváltoztattuk az alapértelmezett beállításokat, akkor [ENTER]-rel folytathatjuk a munkát.

A kiemelt sávot a kurzormozgató-billentyűkkel tudjuk mozgatni. Most válasszuk ki vele az első bejegyzést (2.0 MB), és ott nyomjunk [ENTER]-t. Ennek hatására kapunk egy újabb képernyőt, amiben egy újabb ablakban beállíthatunk egy értéket.

```
Free space to leave on drive E:  [ 30.0_ ] MB
```

Itt lehet beállítani azt, hogy a tömörített meghajtó mennyi szabad helyet hagyjon a kiválasztott meghajtón.

Figyeljünk jól, mert nem a tömörített meghajtó méretét kell megadni! Az a kérdés, ***mennyi szabad hely maradjon rajta***! A többi a - program számára természetesen - mind tömörített terület lesz.

Most a példánknál maradva, adjunk meg itt 30.0 MB-ot, így a meglévő 55 MB szabad helyből 25 MB lesz a tömörített meghajtó. Az [ENTER]-rel visszatérve az előző képernyőre, most válasszuk ki a becsült tömörségi arányt (**2.0 to 1**) [ENTER]-rel, és a felbukkanó új ablakban állítsuk be - a példánknál maradva - **"10.0 to 1"** értékre.

Compression ratio: ⇩ 9.8 to 1
{A tömörségi arány:} ⇧ 9.9 to 1
[ENTER] **10.0 to 1**

Utána visszatérve az előző ablakhoz, megadhatjuk még az új meghajtó betűjelét is:

[ENTER] **E:**
Drive letter for new drive: ⇩ F:
G:
{Az új meghajtó betűjele:} ⇧ H:

Ha ezt is kiválasztottuk, akkor az előző menübe visszatérve a **Continue** menüpontot válasszuk ki [ENTER]-rel. Ezután két választást kínál nekünk:

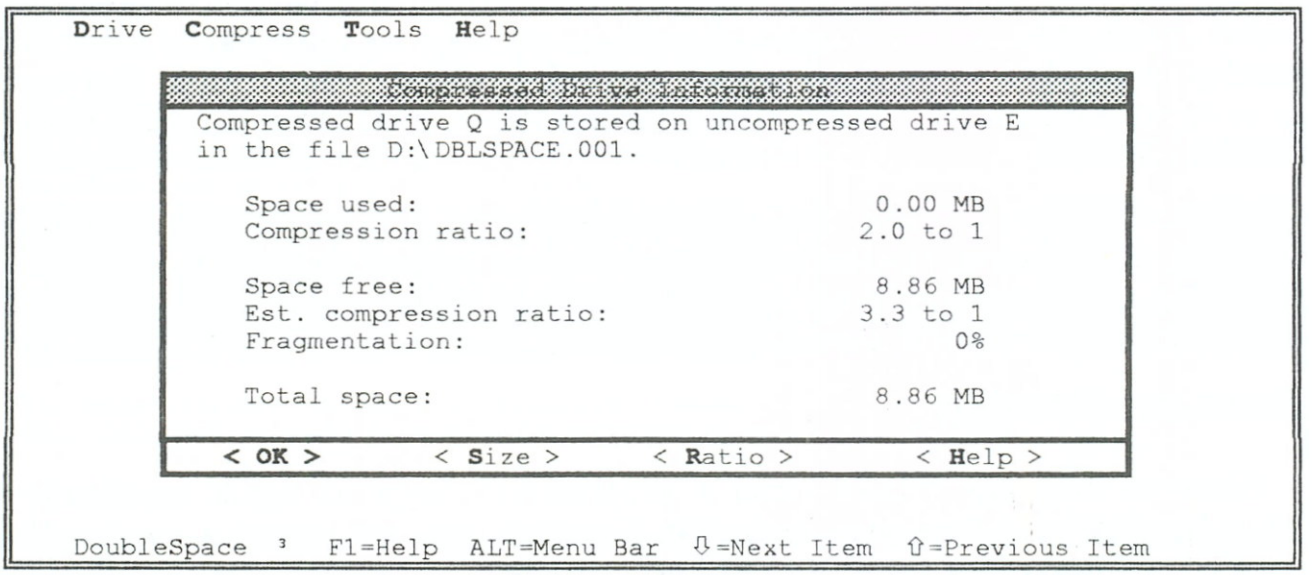
To create the new compressed drive, press C.

"Ha létre akarsz hozni egy új tömörített meghajtót, akkor nyomj 'C'-t."

To return to the previous screen, press ESC.

"Ha vissza akarsz térni az előző képernyőre, akkor nyomj 'ENTER'-t."

Utána már nincs más dolgunk, mint figyelni a képernyőt, mert minden automatikusan fog működni. Végrehajt néhány tesztet, létrehozza a tömörített meghajtót. Ameddig a program dolgozik, nem kell hozzányúlni, ha befejezte, akkor visszakapjuk a kiindulási képernyőt, melyben lesz egyetlen bejegyzés, az új tömörített meghajtó adataival. Mivel már van egy tömörített meghajtónk, kérhetünk róla információt. Válasszuk ki a **Drive** menü **Info...** menüpontját:



Kaptunk egy újabb képernyőt, melynek a közepén van egy ***ablak*** - bekeretezett terület -, melyben az újonnan létrehozott tömörített meghajtó adatait olvashatjuk. Az ablak alsó felén, találunk néhány szót, melyek " < > " jelek között vannak. Ezek a Windows környezetből származnak, és ***parancsgomb***nak, vagy ***nyomógomb***nak szokás nevezni. Az egyik parancsgomb színe mindig kiemelt, a többinek pedig az egyik betűje van csak kiemelve valamilyen színnel (monokróm monitoron a normál, tompa, és kiemelt fényű fehér karakterekkel oldják meg a "színezést"). Váltani a **[TAB]**-bal lehet, vagy visszafelé a **[SHIFT]+[TAB]**-bal. Figyeljük meg közben, hogyan változik a gombok színe, kiemelése!

Amint láthatjuk az "**<OK>**" gomb van kiemelve, ami azt jelenti, hogy ha **[ENTER]**-t nyomunk, akkor ezt a ***nyomógomb***ot fogjuk kiválasztani, olyan mintha az egérrel erre a gombra kattintottunk volna. Ezt az egész ablak-dolgot, egérkezelést hagyjuk a Windows-ra, mert ott jellemző ezek használata. Most egyenlőre a DOS-szal kell megismerkednünk, ami mint mondtam már csak egy környezet a számunkra, a felhasználói programjaink számára, és nem egy felhasználói program.

Ha a "**<Size>**" gombot megnyomjuk (**[ALT]+[S]**), akkor ugyanazt kapjuk, mintha a kiinduló képernyőnkön a **Drive** menü **Change Size...** menüpontját választottuk volna ki. Itt változtathatjuk meg a tömörített meghajtó méretét.

A "**<Ratio>**" gomb megnyomása ([ALT]+[R]) pedig, ugyanazt váltja ki, mintha a kiinduló képernyőnkön a **Drive** menü **Change Ratio...** menüpontját választottuk volna ki. Itt változtathatjuk meg a tömörített meghajtó ***becsült tömörségi arányát***.

Ha az "**<OK>**" gombbal kilépünk ebből az ablakból, akkor visszajutunk a kiindulási képernyőnkhöz. Most Próbáljuk meg létrehozni a terven szereplő másik tömörített meghajtót is. Itt ismét a "D:" meghajtót választjuk ki, aminél nem kell hagyni szabad helyet (csak 0.13 MB-ot, ezt a program kéri), az összes maradék hely a második tömörített meghajtónké lehet. A tömörségi aránynak válasszunk "**3.0 to 1**"-et, és a meghajtóneve legyen "F:". Ha ezt is az előző módon létrehoztuk, akkor a "D:" meghajtónkon nem lesz szabad hely, ezt egy **DIR** paranccsal tudjuk ellenőrizni. Használjuk viszont a "/a" paramétert, mert csak két rejtett rendszerfájl lesz rajta, a DBLSPACE.001 és a DBLSPACE.002, amelyek a tömörített meghajtókat takarják. ***Ezt a meghajtót éppen ezért soha ne tegyük aktuálissá, ide soha ne vegyünk fel programokat*** - ha esetleg lenne rajta valamennyi szabad helyünk -, nehogy véletlenül is kárt tegyünk - vagy az egyik programunk - ezekben az állományokban, mert akkor az egész tömörített meghajtónk elveszett.

A számítógép indulásakor automatikusan betöltődnek a tömörített meghajtóink, és úgy használhatjuk azokat, mintha hagyományos tömörítetlen meghajtók lennének. Azt azonban soha ne felejtsük el, hogy csak olyan állományokat vegyünk fel ide, amelyek valóban annyira tömöríthetők, amennyit mi előírtunk.

Hajlékonylemez tömörítése

Hátravan még egy feladat ezzel a programmal, hogy az egyik hajlékonylemezünkön is kipróbáljuk a tömörítési technikát. Tegyünk egy üres lemezt a meghajtóba, és válasszuk ki a **Compress** menü **Existing Drive...** menüpontját (létező meghajtók). Ekkor megjelenik egy ismert ablak, melyben némi várakozás után megjelennek a létező meghajtók adatai. Az első lesz éppen a hajlékonylemezünk. Válasszuk ki az [ENTER]-rel, majd kapunk egy figyelmeztetést:

Before compressing this drive, you should back up the files it contains.

"Tömörítés előtt, ha lenne a lemezen állomány, akkor azt mentsd el."

Fel is kínál két megoldást, lépj ki (Exit), és használd a **Microsoft Backup** programot, majd utána térj ide vissza, vagy ha minden rendben - valóban üres lemezt tegyél a meghajtóba, akkor folytathatod (Continue) a munkát.

A [C] billentyű leütésével folytatódik a munka, és a merevlemeznél tapasztaltakkal ellentétben, itt nem kínálja fel, hogy állítsuk be a tömörséget, és egyebeket. Ismételten a [C] billentyűvel folytathatjuk a munkát, majd elvégzi a szükséges ellenőrzéseket, és munkákat, majd visszalép a kiindulási képernyőre. Ekkor már kiválaszthatjuk a kiemelt színű sávval, és a **Drive** menü **Change Ratio...** menüpontjával megváltoztathatjuk a becsült tömörségi arányt is (például 16:1-re).

Ekkor kapunk egy olyan hajlékonylemezt, amin akár 22 MB-nyi dBase fájlt is tárolhatunk. Archíválási szempontból nem is utolsó ez a megoldás.

Tömörített hajlékonylemez használata

A programtól kért segítségben kiemelten látható sor az, amelyik megmutatja, hogyan kell parancssorból a tömörített meghajtót betölteni. Ugyanis nem elég a meghajtóba tenni a tömörített lemezt, mert azt még nem veszi észre a DOS. Ki kell adnunk neki egy parancsot, aminek hatására betölti az "A:" meghajtó adatait.

```
C:\>DBLSPACE /?

Creates or configures DoubleSpace compressed drives.

DBLSPACE /AUTOMOUNT=0|1
DBLSPACE /COMPRESS drive: [ /F] [ /NEWDRIVE=drive2:] [ /RESERVE=size]
DBLSPACE /CREATE drive: [ /NEWDRIVE=drive2:] [ /SIZE=size | /RESERVE=size]
DBLSPACE /DEFRAGMENT [ /F] [ drive:]
DBLSPACE /DELETE drive:
DBLSPACE /FORMAT drive:
DBLSPACE /HOST=drive2: drive:
DBLSPACE [ /INFO] drive:
DBLSPACE /LIST
DBLSPACE /MOUNT[ =sss] [ drive ] [ /NEWDRIVE=drive2:]
DBLSPACE /RATIO[ =r.r] [ drive: | /ALL]
DBLSPACE /SIZE[ =size | /RESERVE=size] [ drive:]
DBLSPACE /UNCOMPRESS drive:
DBLSPACE /UNMOUNT [ drive:]

To set up DoubleSpace or use the DoubleSpace program to manage
compressed drives, type DBLSPACE at the command prompt.

For more information about DoubleSpace command-line options, type
HELP DBLSPACE or HELP DBLSPACE /switchname at the command prompt.

C:\>_
```

Tömörített meghajtót betölteni a **Mount** paraméter használatával lehet a következő módon: helyezzük a meghajtóba azt a lemezt, amelyiket már tömörítettük. Ezt egy **DIR a: /a** paranccsal tudjuk ellenőrizni úgy, hogy a listában szerepelni fog egy DBLSPACE.000 nevű rejtett, rendszerfájl. Ekkor a következő parancsot adjuk ki:

```
C:\>DBLSPACE /mount a: /newdrive=Z:

C:\>_
```

A második paramétere el is maradhat, akkor a program keres egy szabad meghajtóbetűjelet, és arra teszi az "A:" meghajtó tömörítetlen formáját, az "A:" meghajtónévvel pedig a tömörített meghajtót fogja kezelni. Így a **DIR A:** parancs a tömörített hajlékonylemezes meghajtó tartalmát fogja mutatni, a **DIR Z:** pedig a tömörítetlen - előzőleg "A:" névre hallgató - hajlékonylemezes meghajtó tartalmát fogja mutatni. Ez a lista továbbra is azt fogja mutatni, amit a példa előtti **DIR a: /a** paranccsal láttunk. Azért célszerű mégis megadni a második paramétert is, hogy biztosan tudjuk, mi lett a neve a tömörítetlen "A:" meghajtónak.

Ha nem kívánjuk a tömörített meghajtót továbbiakban használni, akkor a következő paranccsal letölthetjük azt:

```
C:\>DBLSPACE /unmount a:

C:\>_
```

Hasznos szolgáltatása még a parancsnak, hogy összesített listát lehet a meghajtókról kérni. Ebben a listában szerepel minden meghajtónév, és mellette a típusa, maximális és a szabad kapacitása. Ez a lista így magában elég hosszú, és ha csak néhány meghajtónevet használunk, akkor felesleges kiírásokkal terhel bennünket a számítógép. Ezt természetesen mi azonnal kezelni tudjuk, ugyanis ismerjük a **FIND** parancsot. Ez a bemenetére irányított adatokból kiszűri azt, amire mi kérjük. Ha ismeretlen lenne, hogy miről van szó, akkor lapozd fel légyszíves a "*DOS alapismeretek I.*" című tankönyvemet, és olvasd el a **FIND** parancs használatát. Hogy tudd milyen aggályaim voltak, add ki a **DBLSPACE /LIST** parancsot. A kapott eredményt csak akkor tudod elolvasni, ha 50 sorosra van kapcsolva a VGA monitorod képernyőüzemmódja. Hogy a problémát orvosoljuk, csak azt jelenítsük meg, amire nekünk szükségünk van:

```
C:\DOS>DBLSPACE /LIST |FIND "Ava" /v

Drive  Type                      Total Free  Total Size  CVF Filename
-----  ------------------------  ----------  ----------  ---------------
  A    Removable-media drive     No disk in drive
  C    Local hard drive             2,22 MB     5,00 MB
  D    Local hard drive             0,00 MB    55,00 MB
  E    Compressed hard drive      166,05 MB   250,00 MB  D:\DBLSPACE.001
  F    Compressed hard drive       89,10 MB    90,00 MB  D:\DBLSPACE.002

DoubleGuard safety checking is enabled.
Automounting is disabled for all drives.

C:\DOS>_
```

(A példában az "A:" meghajtó nem volt betöltve, ezért írta ki azt az üzenetet.) Ugye mennyivel áttekinthetőbb ez a lista, mint az előbbi, amelyikben minden meghajtóbetűjel szerepel. A listát tovább szűkíthetjük úgy, hogy azokat is kiszűrjük, amelyekre szintén nem vagyunk kíváncsiak, vagy csak azokat a sorokat engedjük át a szűrőn, amelyekre szükségünk van. Ha például a "D:" meghajtót is ki akarjuk szűrni a listából, mert azon soha nem lesz szabad hely, hiszen ott a két tömörített meghajtó adatai szerepelnek, akkor fűzzük hozzá az előző parancshoz a '**|FIND " D " /v**'.parancsot. Így lehet tovább fűzögetni a szűrőket, egészen 5 mélységig. Ekkor is megtehetjük azt, hogy az utolsó szűrés eredményét fájlba irányítjuk, és egy másik paranccsal tovább rendezgetjük az eredményeket. Ez viszont fárasztó lenne minden alkalommal, ilyen esetekben érdemes batch-fájlt írni, és csak azt elindítani.

Ismételten láthattunk egy példát, hogy a parancsok paramétereit nem bemagolni kell, hanem értelemszerűen használni. Nem kell minden paramétert fejből tudni, hiszen ezt senkitől sem lehet elvárni, de tudjunk a létezéséről, arról, hogy az adott paranccsal milyen jellegű műveleteket tudunk elvégezni. Ezért kell kipróbálni a tankönyvben közölt példákat - már amennyire lehetőségünk van rá -, hogy lássuk legalább egyszer, hogy konkrétan mit kell megadni az adott parancsnak, hogy valóban úgy működjön, ahogyan mi szeretnénk. Utána, ha szükségünk lesz rá, akkor tudni fogjuk, hogy honnan kérjünk segítséget (ami mindig kéznél van, a számítógéptől HELP *parancs*, vagy *parancs /?*).

SMARTDRV.SYS
SMARTDRV.EXE

Cache memória, lemezgyorsító

MS-DOS 5.00: **DEVICEHIGH=[*hely*] SMARTDRV.SYS**

MS-DOS 6 **DEVICEHIGH=[*hely*] SMARTDRV.EXE /DOUBLE_BUFFER**

vagy parancssorban, vagy AUTOEXEC.BAT fájlban:

[*hely*] **SMARTDRV.EXE** [[*meghajtó* [+] | [-]] [*méret*] [*winméret*] [*/s*] ...*stb.*

SMARTDRV=SMART DRIVE - Gyors meghajtó.

Gyorsítja a lemezműveleteket. Ha a lemezekről már egyszer beolvasott adatokat újra kéri a számítógép, akkor nem a lemezről olvassa be, hanem a "gyorsmeghajtó"-ból, ami sokkal gyorsabban elérhető, mint egy lemezegység.

Ugyanis a "gyorsmeghajtó" a memória egy lefoglalt területe, így az újraolvasás a *memória-memória között* zajlik le. A gyorsítás ott is megmutatkozik, hogy a lemezről beolvasott egy bájt helyett, egy egész egységnyit - például egy klaszternyit - olvas be a program, így a többi bájt olvasásához nem kell a lemezegységhez fordulni, mert az már a memóriában megtalálható.

Ha írásra is be van kapcsolva a gyorsítás, akkor a program csoportosítja a kiírandó adatokat, és az adatterületre kerülő adatokat egész klaszterméretben írja ki, majd utána írja ki a módosított FAT-táblát is egy menetben. Szemmel nem látható, mint egy meghajtó, nem kérdezhetjük le, mint egy állománylistát a **DIR**-rel, de ha van legalább 256 kB felesleges szabad memória a számítógépünkben, akkor töltsük be ezt a meghajtót, és figyeljük meg a munkánkat az elkövetkezendő napokban, hetekben. Meg fogjuk látni, hogy bizonyos munkáknál szinte megváltás lesz a használata.

Az MS-DOS 6-ban már EXE formában jelent meg a program, és nem a CONFIG.SYS fájlban használjuk, hanem az AUTOEXEC.BAT fájlban. A CONFIG.SYS fájlban csak akkor használjuk a **/DOUBLE_BUFFER** paraméterrel, ha a program nem tud megfelelően együttműködni egyes merevlemezvezérlőkkel. Az AUTOEXEC.BAT fájlban használatos módját nézzük meg most, de abból is csak a "fontosabb" paramétereket.

meghajtó+ vagy *meghajtó*-

Például:

```
C:\DOS>SMARTDRV A+
```

Bekapcsoljuk az "A:" meghajtón írásra a gyorsítótárat. Ez egy bizonyosfokú késleltetést jelent a lemezre írásban. Először az adatok a gyorsítótárba kerülnek a megfelelő rendezettségben, és ha megtelt, vagy vége lett az írási műveletnek, akkor kerülnek az adatok ténylegesen kiírásra a fizikai lemezegységre. Viszont a DOS parancs megkapja a gyorsítótártól az üzenetet, hogy az adatokat átvette, így az befejezettnek tekinti a munkáját, és a DOS kiteszi a készenléti jelet. Ezzel még korán sincs vége a műveletnek, mert a hajlékonylemezre lehet, hogy még nem került ki egyetlen bájt sem.

Tehát ha mi akkor kikapjuk gyorsan a meghajtóból a lemezt, akkor az hibaüzenetet fog adni, mert még éppen munka közben volt. Ha pedig gyorsan kikapcsoljuk a számítógépet, akkor az adatok nem kerülnek ki a lemezre, és elvesznek.

Ez hajlékonylemez esetében azt jelenti, hogy a lemezre írás még folyamatban van - nagyméretű tár esetén lehet, hogy sokáig is -, de a készenléti jelet megkaptuk, tehát további parancsokat adhatunk ki a DOS-nak. Ezzel felgyorsította a munkánkat, hogy a lemezre írás a háttérben zajlik, miközben mi más jellegű tevékenységet tudunk folytatni.

méret, winméret:

Az első méret a gyorsítótár méretét adja meg, a másik pedig azt a méretet, amit a Windows alatt használjon. Ez általában kisebb szokott lenni, mert a memória általában minden számítógépben kevés, így plusz szolgáltatásokra általában nem szokott annyi hely jutni, mint szeretnénk. Mivel a Windows elég helyigényes a memóriában is, ezért szoktak ilyenkor kisebb tárat foglalni a gyorsítótár részére.

A Windows indulásakor a gyorsítótár mérete *winméret* nagyságúra változik meg. Ez persze lehet nagyobb is, de az ritkán fordul elő, hogy a DOS-ban egy program több helyet igényel, mint maga a Windows rendszer. Ha nem adunk meg értékeket, akkor a program a számítógépünkben lévő memóriához igazodva, alapértelmezett értékek szerint állítja be a méreteket. Ha részletesebben érdekel, akkor kérdezd meg a számítógépedtől (**HELP SMARTDRV**)

A méret értékeket kilobájtban kell megadni, például:

```
C:\DOS>SMARTDRV 1024 512
```

/s: Ez a paraméter a gyorsítótár állapotát (**status**) mutatja meg.

A munkánk közben, bármikor bekapcsolhatjuk, vagy kikapcsolhatjuk a gyorsítótárat egy-egy meghajtón, amit ennek a paraméternek a segítségével tudunk ellenőrizni.

```
C:\>SMARTDRV /S
Microsoft SMARTDrive Disk Cache version 4.0
Copyright 1991,1992 Microsoft Corp.

Room for 256 elements of 8192 bytes each
There have been 43338 cache hits
    and 4262 cache misses

Cache size: 2097152 bytes
Cache size while running Windows: 2097152 bytes

            Disk Caching Status
drive    read cache    write cache    buffering
-----------------------------------------------
  A:         yes            yes           no
  B:         yes            no            no
  C:         yes            yes           no
  D:         yes            yes           no
  E:         yes            yes           no

For help, type "Smartdrv /?".

C:\>_
```

Ismételten kaptunk egy képernyőnyi információt a programtól, amire nem minden esetben van szükségünk. Ennek kiküszöbölésére meg fogunk nézni egy példát a jól ismert **FIND** parancs felhasználásával. Egyszer-egyszer persze szükséges lehet a gyorsítótár állapotára vonatkozó összes adat megtekintése, de általában csak arra vagyunk kíváncsiak, hogy be van-e kapcsolva egy adott meghajtóra a gyorsítótár, vagy sem. A szűrőparancs használatához keressünk egy olyan karaktert a kiírásokban (:), mely szerepel az összes olyan sorban, ami számunkra fontos információt tartalmaz.

```
C:\>SMARTDRV /S |FIND ":"
Cache size: 2097152 bytes
Cache size while running Windows: 2097152 bytes
  A:         yes            yes           no
  B:         yes            no            no
  C:         yes            yes           no
  D:         yes            yes           no
  E:         yes            yes           no

C:\>_
```

Ez a kiírási forma már sokkal inkább megfelel az igényeinknek.

2.2.1. Merevlemez

Az előbb ismertetett parancsok (DBLSPACE, SMARTDRV) merevlemezen alkalmazva szemmel látható gyorsulást hoznak a számítógépünkben. A megduplázott lemezmeghajtó révén, például egy 10 MB-os területen tárolhatunk akár 100-150 MB-nyi jól tömöríthető állományt, ami rendkívül gazdaságossá teszi a számítógépünk kihasználtságát. Még ha csak kisebb tömörítési arányban tudjuk kihasználni, akkor is sokat nyerhetünk vele. Elsősorban archíválási feladatoknál jelent nagy segítséget, de olyankor is hasznát vehetjük, ha hosszú időre visszamenőlegesen a merevlemezünkön akarjuk tartani a vállalati adatokat, amelyek a **Double Space** nélkül hamar megtöltenék a

merevlemezünket. Ennek következménye viszont csak az lehet, hogy az állományokat valamilyen tömörítővel összetömörítjük, így viszont csak kicsomagolás után tekinthetünk bele az adatainkba. A tömörített meghajtó kezelése állítólag az MS-DOS 6-ban még rejtett egy-két programozási hibát, amit az MS-DOS 6.2-ben már kijavítottak a Microsoft programozói. A tömörített meghajtók elérési ideje olyan gyors, hogy észre sem vesszük, hogy a DOS honnan olvassa az adatokat.

A lemezgyorsító **Smart Drive** program egyik előnye szemmel is látható. Adjunk ki egymás után két **DIR** parancsot ugyanarra a könyvtárra. Az első esetben értelemszerűen a lemezről fogja beolvasni az adatokat, így kis várakozás után fogjuk csak megkapni a listát. A második **DIR** parancsnál már a memóriából fogja átvenni az adatokat, minek hatására a lista egy szemvillanás alatt meg fog jelenni a képernyőn.

Ez a program kíméli is a meghajtónkat, mert nem ugrál az olvasófej a FAT-tábla és a lemez egyéb területei között, hanem a FAT-táblát már a második esetben a memóriából olvassa. Így a lemezegység olvasófeje a lemez egy területén minimális elmozdulás mellett, folyamatosan írja ki, vagy olvassa be az adatokat. Ha elegendően nagy a gyorsítótár területe, akkor sokkal gyorsabban és csöndesebben fog dolgozni a merevlemezünk, ami viszont hosszabb élettartamot is biztosít a számára. Ha a lemezművelet közben halkabb a meghajtónk, mint régebben, akkor ezzel tudjuk ellenőrizni, hogy a gyorsítótár be van-e kapcsolva. Ha a fejnek a lemez közepe és a FAT-tábla között kell minden klaszter írása - vagy olvasása - között ugrálnia, akkor ez egy "hangos" zakatoló hangot eredményez (és a merevlemezünk idő előtti tönkremenetelét). Ha csak egyszer-kétszer megy ki a fej a lemez szélére a FAT-táblát megnézni - vagy felírni -, és a többi időben az adatokat írja a lemezre, akkor a fej kevés mozgást végez, így csak a lemez halk forgásából eredő suhogást halljuk, ami sokkal csendesebb az előbbinél.

2.2.2. Hajlékonylemez

Hajlékonylemezek esetén kicsit óvatosabban kell eljárnunk, de itt is alkalmazhatók az előbbi lemezkezelő programok. A lemezterület megsokszorozása itt is lehetséges, viszont a cserélhető lemezes kivitelénél fogva, nem látja azonnal a DOS, ha kicseréljük a lemezt a meghajtóban. Ezért itt el kell végeznünk egy-két műveletet a használatbavétel előtt. Egy **DIR** parancsra csak néhány rejtett fájlt láthatunk a lemezen (**DIR /a**). A **Double Space**-szel kezelt hajlékonylemez esetén, először be kell tölteni (Mount) a lemezen lévő információkat a memóriába, ami egy kis időt vesz igénybe. Utána már a **DIR** parancs azt mutatja, ami a tömörített meghajtón van.

A lemezre íráskor viszont jelentkezhet egy probléma: az adatok folyamatos tömörítése folytán rendkívül lelassulhat a lemezre írás, azaz annyi idő alatt írja ki például az 1 MB-ra tömöríthető 10 MB-nyi adatot, mintha 10 MB-ot írna ki hajlékonylemezre. Még mielőtt bárki is letenné a könyvet, közlöm, hogy ez a kis apró kellemetlenség, miből fakad, és hogyan küszöbölhető ki egyszerűen.

A folyamat a következőképpen zajlik le: *az adat elindul, elkapja a röptömörítő, összetömöríti, és kiírja a lemez egyik szektorába.* Így halad végig a teljes adatállományon, ami egy nagyobb méretűnél rengeteg időt elvesz, mert a lemezre írás rendkívül lassú. Ha ez a része felgyorsítható lenne - ha a tömörítés a memóriában lezajlana, és a hajlékony-

lemezhez csak az adatok kiírása miatt fordulna a számítógép -, akkor hihetetlenül felgyorsulna az egész folyamat.

Ismerjük már a lemezgyorsítót, a **Smart Drive** programot. Alkalmazzuk itt is, és gondoljuk át a folyamatot ismét! *Az adat elindul, elkapja a röptömörítő, összetömöríti, és kiírja - a lemez egyik szektora helyett - a memóriában lévő gyorsítótárba.* Így halad végig a teljes adatállományon, és még egyszer sem fordult a rendkívül lassú hajlékonylemezhez. Mikor megtelt a lemezgyorsító területe, akkor egy menetben kiírja az adatokat a lemezre, **SORFOLYTONOSAN, MINIMÁLIS FEJMOZGÁS MELLETT**, így az adatok gyorsabban kikerülnek a hajlékonylemezre, mint gondolnánk. Az adatok kiírása után kiírja a FAT-táblát is a hajlékonylemezre, így elkerültük a fej felesleges rángatását is, amivel meghosszabbíthatjuk a lemezegység élettartamát is. A művelethez elegendő egy kisméretű - például 256 kB-os - gyorsítótár is, amit igazán biztosítani tudunk már egy 286-os 1 MB-os számítógépen is, a kiterjesztett memóriában.

Ha használjuk a **Smart Drive** programot, és az "A:" meghajtón írásra is bekapcsoltuk, akkor ne felejtsük el, hogy a lemezre írás késleltetett ütemű, így figyelnünk kell a hajlékonylemezes egység LED-jét, hogy világít-e. Ha megállt a meghajtó, és nem világít a LED, akkor vegyük csak ki a lemezt.

2.2.3. Memórialemez

RAMDRIVE.SYS — **Memórialemez készítése**

Használata: **DEVICE=**[*helye*] **RAMDRIVE.SYS** [*Kap Sec Bej*] [*/e* | */a*]

RAM DRIVE=A RAM-ban létrehozott lemezmeghajtó.

A számítógép memóriájában definiálunk egy olyan területet, ami úgy működik, mint egy lemezegység. **FIGYELEM! A RAM területén létrehozott meghajtó tartalma kikapcsoláskor elvész! Ezt soha ne felejtsük el!**

helye: meghajtó: elérési út: A RAMDRIVE.SYS fájl helye. Ha nem adjuk meg, akkor a betöltőlemez gyökérkönyvtárában keresi.

Kap : A lemezegység kapacitása, vagyis a mérete kilobájtokban.

Sec :Szektorméret. Jelentősége ott van, ha például batch-fájlokat tárolunk a memórialemezen, akkor lehet hogy hamar betelne a lemez, ha nagyra választjuk meg a szektorméretet. Ez viszont meggyorsíthatja a nagyméretű fájlok betöltését, de memórialemeznél csak mikroszekundumokat nyerünk, amit nem érzékel az ember különösebben. Ha a szektorméretet kicsire (minimum 256 bájtra) választjuk, akkor a szektorok kihasználtsága sokkal jobb lesz, mint egyébként.

Bej :Bejegyzések maximális száma. Ha előre tudjuk az itt tárolandó fájlok maximális számát, akkor a gyökérkönyvtár mérete csökkenthető, így több hely marad a programok részére.

/e vagy /a: A memórialemez helye hol legyen. Ha egyiket sem adjuk meg, akkor a hagyományos memóriába kerül. A /e hatására az *Extended memóriá*ba (XMS), a /a hatására az *expAnded memóriá*ba (EMS) kerül a memórialemez.

Az előnye az eddigiek ismeretében szerintem már mindenkinek egyértelmű: Erről a lemezegységről programot betölteni nem másodpercek, hanem egy szempillantás alatt lehet, hiszen a memóriában lévő meghajtóról töltjük be a programot a hagyományos memóriaterületre, ahol futtatni fogjuk a programot. Megfigyeltük? - Ismét ***memóriából másolunk memóriába,*** ami mikro- vagy nanoszekundumokban (µsec, nsec) mérhető és nem másodpercekben.

Olyan programokat célszerű itt tárolni, amelyeket gyakran betöltünk, például egy szövegszerkesztő vagy egy Clipper LIB fájlt. Használhatunk memórialemezt ideiglenes állományok tárolására is, melyekre a számítógép kikapcsolása után nincs szükségünk.

Használhatjuk gyakorlásra is, ha programokat akarunk egyik meghajtóról egy másikra másolni. Hozzunk létre több memórialemezt, és azokon másolgassuk a fájlokat, azokon hozzunk létre ágas-bogas könyvtárfákat, melyeket a számítógép kikapcsolásával, vagy akár a RESET gomb megnyomásával is teljesen "törölhetünk". Kitűnő hely a könyvtár- és fájlműveletek gyakorlására.

Értelemszerűen ez a program is a számítógép memóriájából akar egy nagyobbacska területet lefoglalni. Így a gyorsítótárral (SMARTDRV) együtt, ezek olyan programok, melyeket csak akkor tudunk a hasznunkra fordítani, ha rendelkezünk elegendő memóriával. Persze ez sem szükségszerű, mert ha létrehozunk két-három memórialemezt, egyenként 64 kB méretben, akkor csak 128 vagy 192 kB-ot veszünk el a memóriából, amit még a hagyományos memóriából is ki tudunk szorítani, ha nem akarunk nagyobb programokat futtatni. Ezek viszont bőven megfelelnek arra a célra, hogy könyvtárstruktúrákat hozzunk létre, hogy kisebb - fájlokat másolgassunk, játszunk az attribútumokkal, ... *stb.*

Ha például szövegszerkesztőt akarunk indítani róla, amit gyakran be kell töltenünk, és gyakran kilépünk belőle, akkor azt először - célszerű az AUTOEXEC.BAT fájlban - át kell másolnunk a merevlemezről, vagy a hajlékonylemezről a memórialemezre, hisz' a memórialemezünk úgy indul, hogy teljesen üres. Merevlemezt nélkülöző számítógépek esetén, célszerű a COMMAND.COM-ot is ide átmásolni, a parancsok gyorsabb végrehajtása végett.

Ha a számítógépünkben nincs merevlemez, de van például 2 MB memória, és csak egy egyszerű adatrögzítési vagy szövegszerkesztési célra használjuk a számítógépünket, akkor a következő módon meggyorsíthatjuk a gépünket, kihasználva értelmesen a benne lévő memóriát:

A hagyományos memóriában annyi helyet hagyunk, hogy a használt program éppen elférjen benne, a többi helyet lefoglaljuk egy memórialemez részére. A felső memóriaterületen pedig létrehozunk egy másik memórialemezt! A rendszer értelemszerűen hajlékonylemezről indul, és automatikusan felmásolja a szükséges programokat a hajlékonylemezről a memórialemezre. A **PATH**-t gondosan nem állítjuk ezután már az "A:" meghajtóra, mert az csak lassítaná a rendszert. Bekerülnek viszont az új meghajtók az elérési útba, miáltal minden programunk, adatállományunk elérhető lesz. Minden

művelet - fájlműveletek, programok futtatása, ... - csak a memóriában zajlik, ami hihetetlenül felgyorsítja a rendszer működését. Időnként természetesen végezhetünk biztonsági mentéseket a hajlékonylemezre, de csak a megváltozott állományokkal ("A" - Archive attribútum - **XCOPY - gondolkozz!**). Jogosan vetődhet fel benned, hogy így milyen lassú lesz a rendszer indulása. De gondolj csak bele, normális működés mellett, hányszor kell a rendszert elindítani egy nap? - Általában csak egyszer, esetleg kétszer.

Memórialemez létrehozására már láthattunk egy-két példát, most használjuk fel az eddigi ismereteinket, és halmozzuk be egy példába:

Egy 32 bites számítógépen már szinte alapértelmezett a HMA, és XMS használata. Most létrehozunk még egy 4 MB-os EMS memóriát is, amibe egy 3 MB-os memórialemez fog bekerülni. A memórialemez eszközmeghajtó programját - természetesen - az UMB területre fogjuk tölteni a , **DEVICEHIGH** utasítással. Hogy a memórialemez az EMS-be kerüljön, erről a "/a" paramétere fog gondoskodni.

```
device=HIMEM.SYS
DOS=HIGH
device=EMM386.EXE 4096 RAM
DOS=UMB
devicehigh=MOUSE.COM
devicehigh=RAMDRIVE.SYS 3072 512 64 /a
```

```
A:\>MEM /C

Conventional Memory :

  Name                 Size in Decimal       Size in Hex
------------       ---------------------   -------------
  MSDOS                13216      ( 12.9K)       33A0
  HIMEM                 1184      (  1.2K)        4A0
  EMM386                8400      (  8.2K)       20D0
  COMMAND               2672      (  2.6K)        A70
  SNIPPER               3728      (  3.6K)        E90
  FREE                    64      (  0.1K)         40
  FREE                    48      (  0.0K)         30
  FREE                625792      (611.1K)      98C80

Total  FREE :         625904      (611.2K)

Upper Memory :

  Name                 Size in Decimal       Size in Hex
------------       ---------------------   -------------
  SYSTEM              163840      (160.0K)      28000
  MOUSE                10720      ( 10.5K)       29E0
  RAMDRIVE              1232      (  1.2K)        4D0
  FREE                 20736      ( 20.3K)       5100

Total  FREE :         20736      ( 20.3K)

Total bytes available to programs (Conventional+Upper) :      646640    (631.5K)
Largest executable program size :                             625792    (611.1K)
Largest available upper memory block :                         20736    ( 20.3K)

  4587520 bytes total EMS memory
  1048576 bytes free EMS memory

  15859712 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
  11333632 bytes available XMS memory
           MS-DOS resident in High Memory Area

A:\>_
```

Mint láthatjuk könnyedén sikerült az UMB-be tölteni a memórialemez vezérlőjét, és még maradt is szabad hely utána. Újabb és újabb programokat töltünk a számítógépünkbe, és a hagyományos memóriánk továbbra is majdnem olyan szabad, mintha nem is lenne benne semmi.

Igaz az XMS memóriánk elég jól megcsappant, de itt található a szimulált EMS memória, és a DOS egy része is (HMA). Az **EMM386.EXE 4096** létrehozott nekünk egy 4 MB-os EMS memóriát, amiben egy 3 MB-os memórialemez kapott helyet. Természetesen felesleges ez a megoldás, hiszen a EMS lassabb elérésű, mint az XMS, és ott is létrehozhattuk volna a memórialemezt. A példa csak arra akar rámutatni, hogy hogyan foglalja le egy program az adott memóriát. Egy szövegszerkesztő vagy táblázatkezelő program futása közben nem biztos, hogy ilyen látványosan megtekinthettük volna a memóriafelosztást. De hálózatban dolgozók például használhatják ilyenkor az EMSNET5.EXE programjukat, hogy a NET5.COM ne foglalja a hagyományos memóriát. Persze erre is csak akkor van szükségünk, ha az XMSNET5.EXE programmal nem rendelkezünk, ami az XMS-be tölti a **NET5** programot.

Most pedig nézzük meg, hogy valóban sikerült-e egy igazi lemezt definiálnunk:

```
A:\>CHKDSK F:

Volume MS-RAMDRIVE created 02-27-1991 12:00a

   3138560 bytes total disk space
   3138560 bytes available on disk

      1024 bytes in each allocation unit
      3065 total allocation units on disk
      3065 available allocation units on disk

    655360 total bytes memory
    625856 bytes free

A:\>_
```

2.3. Monitor

A PC-knél az egyik hasznos lehetőség, hogy munkánkhoz és/vagy pénztárcánkhoz igazodóan, tetszőleges monitort csatlakoztathatunk a számítógépünkhöz, a megfelelő monitormeghajtó-kártya alkalmazásával. Éppen ezért a DOS-ban is több program segíti a monitorok minél jobb kihasználását. A DOS alapvetően egy monokróm monitoron is képes megfelelően működni, a színes monitorok csak megjelenésükben, illetve a színek használatában nyújtanak több szolgáltatást.

Itt is érvényes az a szabály, ha nem használunk segédprogramot, akkor egy SVGA monitor is csak monokróm monitorként fog üzemelni. A másik hasznos szolgáltatás - a helyes magyar ékezetes karakterek használata - is csak az EGA monitortól felfelé használható ki, és csak a megfelelő segédprogramok segítségével, ***kódlapok váltásával***. Itt jelenik meg először a ***kódlapok*** fogalma. A tankönyv első részében többször is hivatkoztam rá, most megismerhetjük létrehozásuk és használatuk módját.

Minden *kódlap*on különböző karakterek vannak, melyet azonnal megértünk, ha a kódlapok ASCII kódjait megtekintjük. Láthatjuk, hogy különösen a 128 feletti ASCII kódok esetén eltérőek a különféle kódlapok. Erre azért van szükség, mert az amerikaiak nem használnak például magyar ékezetes karaktereket, de mi igen. Az ASCII kódok száma viszont véges (0-255), így nincs lehetőség mindenféle karakterek megjelenítésére egy kódlap használatával. A 437-es amerikai kódlapon például vannak táblázatrajzoló karakterek is, amelyek helyett a magyar (latin II) kódlapon, a magyar nyelvben használatos ékezetes karakterek vannak. Így például egy 437-es kódlap használatakor megrajzolt táblázatban, a vonalak helyett ékezetes karakterek fognak megjelenni a 852-es kódlapra váltáskor (lásd **NC,** 852-es kódlappal). Ha valaki viszont szlovák vagy szláv levelezést folytat, akkor használja a 850-es kódlapot. Ezen megtalálható az összes szláv ékezetes mássalhangzó is. (Például: ñãç... .)

Túlságosan sok lehetőségünk nincs a DOS-ban akkor sem, ha jobb felbontású monitort használunk, de az a néhány is megszépítheti a DOS egyszerű képernyőjét.

Az első eszközmeghajtó program, amire a **PROMPT** parancsnál már hivatkoztam, az ANSI.SYS.

ANSI.SYS — Escape-szekvenciák használatához

Használata: **DEVICE=**[*hely*] **ANSI.SYS** [/*x*] [/*k*]

ANSI=American National Standart Institute - Amerikai Nemzeti Szabványügyi Hivatal.

Használhatjuk az Escape-szekvenciákat a képernyő színezésére, a billentyűzet átdefiniálására, monitor üzemmódjának átváltására, ... *stb.*

/*x*: Újra feltérképezi a 101 gombos billentyűzeten a kiterjesztett billentyűket.

/*k*: Figyelmen kívül hagyja a 101 gombos billentyűzet kiterjesztett billentyűit.

A tankönyv első részében már megismerkedhettünk az escape-szekvenciákkal, amelyekkel sok érdekes feladatot oldhattunk meg. A függelékben található táblázat segítségével mindenki szabadjára engedheti a fantáziáját. Használatára már láthattunk több példát is a **PROMPT** parancsnál.

DISPLAY.SYS — Kódlapok képernyőn való használatához

Használata: **DEVICE=**[*hely*] **DISPLAY.SYS** *CON:=*(*típus*[,[*kódlap*][,*n*]])

DISPLAY=Kijelző

Definiálja a képernyőn használható kódlapokat. Ez az első utasítás, aminek a memóriába kell kerülni a kódlapok használatához.

hely: A DISPLAY.SYS fájl helye, és ha nem adjuk meg, akkor ott keresi, ahol a többi CONFIG.SYS utasítást keresné alapértelmezésben (a betöltő lemez gyökérkönyvtárában).

CON: Kötelezően beírandó, annak az egységnek a neve, amelyikre a kódlapot definiáljuk. (Csak ez az egység adható meg.)

típus: Monitortípus-jelző név, csak kétféle lehet: **EGA** vagy **LCD**. A előzőt EGA monitortól felfelé használjuk, az utóbbit folyadékkristályos kijelzők esetén. Csak EGA monitortól felfelé használható, így *CGA* vagy *monokróm* monitorokon nincs lehetőségünk kódlapot váltani.

,kódlap: (Vessző és) a használni kívánt kódlap száma. Az amerikai (alapértelmezésbeli) kódlap száma **437**, a magyar (latin II.-es) pedig **852**. (Ha a többi is érdekel, lapozd fel az *"MS-DOS 6 kézikönyvet"*!)

,n: (Vessző és) a használni kívánt kódlapok száma. Az érték 1 és 6 közötti lehet, a kettő SVGA monitornál működik, és elegendő szokott lenni.

Példák a használatra:

```
DEVICE=c:\dos\DISPLAY.SYS CON:=(EGA,,2)

DEVICE=A:\DISPLAY.SYS CON:=(EGA,852,2)

DEVICEHIGH=DISPLAY.SYS CON:=(EGA,852,2)
```

2.4. Billentyűzet

KEYB — **Billentyűzet átdefiniálása**

KEYB=Keyboard - Billentyűzet.

A billentyűzet karakterkiosztását tudjuk megváltoztatni vele. Erre van szükségünk, ha magyar billentyűzetet használunk, azért hogy az a karakter jelenjen meg a képernyőn, ami a billentyűre van írva.

(Az eddigiektől eltérve egy kicsit, ez nem CONFIG.SYS utasítás, de a téma folyamatossága miatt került ide. Parancssorban vagy az AUTOEXEC.BAT fájlban használható.)

Most lássuk, milyen paramétereket adhatunk meg:

```
C:\>keyb /?
Configures a keyboard for a specific language.

KEYB [ xx[ ,[ yyy][ ,[ drive:][ path] filename]]] [ /E] [ /ID:nnn]

  xx                      Specifies a two-letter keyboard code.
  yyy                     Specifies the code page for the character set.
  [ drive:][ path] filename  Specifies the keyboard definition file.
  /E                      Specifies that an enhanced keyboard is installed.
  /ID:nnn                 Specifies the keyboard in use.

C:\>_
```

xx: Két betűvel jelölt billentyűzet kódja.

yyy: A használt kódlap száma. (Magyarországé a 852-es.)

[drive:] [path] filename: Meghajtó, elérési út, fájlnév, annak a fájlnak az adatai, ami definiálja a billentyűkhöz rendelt kódokat. Erre akkor van szükség, ha a szabványostól eltérő billentyűzetkiosztást használunk.

/E: Kiterjesztett billentyűzet használata.

/id:nnn: Megadja, hogy milyen billentyűzetet használunk. (Lásd a táblázatot!)

Ország vagy nyelv	Billentyűzet elrendezése (xx értéke)	Kódlap száma (yyy értéke)	Billentyűzet azonosítója (ID:nnn értéke)
Belgium	be	850, 437	
Brazil	br	850, 437	
Canadian-French	cf	850, 863	
Czechoslovakia (Czech)	cz	852, 850	
Czechoslovakia (Slovak)	sl	852, 850	
Denmark	dk	850, 865	
Finland	su	850, 437	
France	fr	850, 437	120, 189
Germany	gr	850, 437	
Hungary	hu	852, 850	
Italy	it	850, 437	141, 142
Latin America	la	850, 437	
Netherlands	nl	850, 437	
Norway	no	850, 865	
Poland	pl	852, 850	
Portugal	po	850, 860	
Spain	sp	850, 437	
Sweden	sv	850, 437	
Switzerland (French)	sf	850, 437	
Switzerland (German)	sg	850, 437	
United Kingdom	uk	850, 437	166, 168
United States	us	850, 437	
Yugoslavia	yu	852, 850	

Ha a hagyományos amerikai billentyűzetet használjuk, akkor nincs szükségünk erre a programra. Ha magyar billentyűzetünk van, vagy amerikai, de magyar kiosztással akarjuk használni, akkor van szükségünk erre a programra. Valójában teljesen független, hogy milyen billentyűzetet használunk (fizikailag), csupán azért célszerű, hogy egyezzen a billentyűzet, és a karakterkiosztás, hogy az a karakterkép jelenjen meg a képernyőn is, mint ami a billentyűre van rajzolva.

Nem szükségszerű része a billentyűzetnek a megfelelő meghajtóprogram. Ha például az iskolában, vagy a munkahelyünkön nincs lehetőségünk billentyűzetcserére, de mi tudunk tíz ujjal vakon gépelni, magyar írógépen, akkor teljesen érdektelen, hogy mi van a billentyűkre rajzolva, mert úgysem nézünk rá. A másik megoldás, hogy ráírjuk a meglévő feliratok mellé - alá, fölé -, azokat a karaktereket, amelyeket, csak mi használunk, a megfelelő billentyűzetmeghajtó-programmal. Ez nem zavar másokat sem

abban, hogy lássák, mik az eredeti feliratok, és mi is látjuk rajta, hogy milyen karakter fog megjelenni, a mi programunk hatására.

Szabványos magyar billentyűzet használata esetén, a következő utasítást tegyük be a CONFIG.SYS fájlba:

```
KEYB HU,852
```

Előfordulhat azonban, hogy a számítógép erre kiírja, hogy a billentyűzet nem támogatja a megadott kódlap használatát. Akkor lapozz a 91. oldalra, ahol erről részletesen olvashatsz.

2.5. Egér

Az egérhasználat támogatásával kapcsolatosan mutatom be a következő CONFIG.SYS utasítást. Ez is hasonló a **DEVICE, DEVICEHIGH** utasításokhoz, önállóan nem használható, hanem rezidens programokat tudunk vele a tárban elhelyezni. Erre azért van szükségünk, mert a CONFIG.SYS futásakor még nem fut a COMMAND.COM, így "*.EXE" vagy "*.COM" fájlokat nem indíthatunk el. Az MS-DOS 6-tól kezdve azonban ez megváltozott, mert például az egérmeghajtó program, ami "COM" kiterjesztésű, betölthető **DEVICE** paranccsal eszközmeghajtó módjára.

INSTALL — Rezidens program indítása

Használata: **INSTALL**=[*hely*] *{program neve, paraméterei}*

INSTALL=Telepít, üzembe helyez.

Az adott ***hely***en megtalálható rezidens programot tölti be a memóriába. A program paramétereit utána szóközzel kell elválasztani, illetve olyan szintaktikával kell megadni, amit a program előír.

A leggyakoribb rezidens programunk az egér használatát támogató program, melynek lehet többféle neve is, a lényeg, ha a "MOUSE" szó szerepel benne, akkor nagy valószínűséggel az egy egérmeghajtó program lesz.

MOUSE.SYS / MOUSE.COM — Egérmeghajtó program

Használata: **INSTALL=MOUSE.COM** vagy **DEVICE=MOUSE.SYS**

MOUSE=Egér (Számítógépes adatbeviteli eszköz).

Biztosítja az egér használatát azon programoknak, amelyek képesek annak kezelésére. Ilyen például az **NC**, vagy a Borland cég fejlesztőkörnyezetei (**Borland C, Turbo Pascal**,...), szövegszerkesztők, ...*stb.*

A MOUSE.COM-ot használhatjuk parancssorban, paraméterekkel is (**MOUSE /?**).

2.6. Országfüggő információk megadása

COUNTRY Országfüggő információk megadása

Használata: **COUNTRY**=*xxx*[,[*yyy*][,[*filespec*]

COUNTRY=Ország

Megadhatók az adott országhoz kapcsolódó információk (dátum- és időformátum, ...*stb.*).

xxx: A támogatni kívánt ország kódja (Magyarországé: 036).

yyy: A használni kívánt kódlap (a 852-est javaslom). Nem kötelező megadni, de akkor alapértelmezésben a 437-est fogja használni.

filespec: Annak a fájlnak a neve és az elérési útja, amelyik bővebb információkat tartalmaz az adott országban használatos formátumokról. Nem kötelező megadni, a DOS a COUNTRY.SYS fájlt biztosítja számunkra, de enélkül is használható.

Hatása: Ha a következő példákban megadott módon használjuk, akkor például a **DIR** parancs által kiírt listában a dátum magyar formátumban (ÉÉ-HH-NN) fog megjelenni, illetve a **DATE** és **TIME** parancsoknál szintén magyar formátumban adhatjuk meg a kívánt adatokat.

Van azonban egy hibája is. Az MS-DOS 6.20 felhasználói kaptak egy újítást, miszerint a **DIR** listáiban a 999-nél nagyobb számokat három jegyenként vesszővel tagoltan írja ki a DOS. Azonban az itt ismertetett utasítás hatására a vessző helyett szóköz íródik ki a számok tagolására, ami rendkívül zavaró.

Példák a megadásra:

COUNTRY=036

A magyar dátum-, idő-, egyéb- formátumokat fogja használni.

COUNTRY=036,852

A magyar dátum-, idő-, egyéb- formátumokat fogja használni, és a 852-es kódlapot állítja be alapértelmezésben.

COUNTRY=036,852,c:\dos\COUNTRY.SYS

A magyar dátum-, idő-, egyéb- formátumokat fogja használni, a 852-es kódlapot, és az egyéb országfüggő információkat a megadott fájlból fogja venni.

COUNTRY=036,,c:\dos\COUNTRY.SYS

Hasonló az előzőhöz, de ebben az esetben az alapértelmezett kódlapot fogja használni. A két vessző használata ebben az esetben kötelező!

2.7. Egyéb CONFIG.SYS utasítások

A több CONFIG.SYS használatát támogató MS-DOS 6 -ról részletesen az *"MS-DOS 6 kézikönyvben"* olvashatsz. Itt most csak az MS-DOS 5.00-ban is használható utasításokat fogjuk tárgyalni. A most következő utasítások nem határozzák meg alapvetően a rendszer használatát, nem hoznak szemmel látható gyorsulást, nem hoznak új lehetőségeket, mint az előbbiekben megismert utasítások, de szükségesek lehetnek. Egyes programok (például Windows) telepítéskor beállítják ezek értékeit, és fontos tudnunk, hogy ezek sem érdektelen utasítások, mert használatuk nélkül esetleg lassul, vagy megakad a rendszer használata.

BREAK — Programmegszakítás engedélyezése

Használata: **BREAK=ON** vagy **BREAK=OFF**

BREAK=Megszakít (programmegszakítás).

Kiterjeszti a programmegszakítás figyelését a [CTRL]+[C] billentyű figyelésére is, illetve olyan műveletek alatt is használható, ahol eddig hatástalan volt.

BUFFERS — Lemezpufferek megadása

Használata: **BUFFERS=***n*[,*m*]

BUFFERS=Pufferek, átmeneti tárolóterületek.

Lemezpuffereket foglal le a rendszerinduláskor a lemezműveletek gyorsításához.

n: A lemezpufferek száma 1-99. Alapértelmezésben 15.

m: A másodlagos lemezpufferek száma 1-8. Csak bizonyos lemezműveleteket gyorsít.

FCBS — Fájlvezérlő blokkok megadása

Használata: **FCBS=n**

FCBS=File Control BlockS - fájlvezérlő blokkok.

Az egyszerre használható blokkok számát adhatjuk meg. Inkább csak a régebbi programok igénylik a használatát. Csak akkor kell megadni, ha valamelyik program ezt igényli. Az MS-DOS 6-ban a **MEMMAKER** program automatikusan beállítja az ideális értéket.

FILES — Nyitvatartható fájlok száma

Használata: **FILES=*n***

FILES=Fájlok

Megadhatjuk az egy időben nyitvatartható fájlok számát. Alapértelmezésben ez 8, de ez már nagyon kevés, és gyakran szükségessé válik ennek az értéknek a növelése.

A későbbiekben látni fogjuk az egyik példában, hogy annyira kevés helyet foglal, hogy nyugodtan megadhatunk egy 80-100-as értéket, mert fájlonként csak 48 bájtot foglal le a memóriában, ami **FILES=100** esetén is csak 4,68 kB (4800 bájt). Így soha nem fogunk olyan üzenetet kapni a számítógéptől, hogy nem tud már több állományt megnyitni.

LASTDRIVE — Az utolsó DOS által fenntartott meghajtó

Használata: **LASTDRIVE=*x***

LAST DRIVE=Utolsó meghajtó.

A DOS részére lefoglalt utolsó meghajtóbetűjel az angol ábécében az elejétől kezdve. Ha valamilyen egyéb egység új meghajtóbetűjelet akar lefoglalni, akkor a soron következőt veheti igénybe. Például a Novell hálózatba lépéskor azon a meghajtón lesz a LOGIN könyvtár, amelyik a **LASTDRIVE**-ban megadott után következik.

x: Meghajtóbetűjel "A"-"Z"-ig. Alapértelmezésben az "E".

REM — Megjegyzés

Használata: **REM** *{megjegyzés magunknak a sorokhoz}*

REM=REMARK - megjegyzés.

Megjegyzés sorokat iktathatunk be a fájlba, ha később is tudni akarjuk, hogy melyik sort miért raktuk be a fájlba.

Akkor is használni szoktuk, ha egy utasítást időszakosan ki akarunk iktatni a végrehajtásból, de bonyolult a megadása, és elfelejtenénk a helyes megadást. Ekkor csak beírunk elé egy "**REM**"-et, és már nem is lesz végrehajtva. Ugyanis ezeket a sorokat nem veszi figyelembe a végrehajtáskor a számítógép.

Például:

```
REM * Csak fájlmásoláskor használom *
REM DEVICEHIGH=RAMDRIVE.SYS 1440 512 64 /e
```

SHELL — Parancsértelmező helyének megadása

Használata: **SHELL=**[*hely*]{*parancsértelmező program paraméterekkel*}

SHELL=Héj, "keretprogram".

A parancsértelmező helyét adja meg, ha nem a DOS által kínált COMMAND.COM-ot akarjuk futtatni, vagy az nem a betöltőlemez gyökérkönyvtárában van, vagy az alapértelmezett környezeti értékeket meg akarjuk változtatni.

Az MS-DOS által használt parancsértelmező a COMMAND.COM, amit a következő módon lehet például használni a CONFIG.SYS-ben:

```
SHELL=c:\dos\COMMAND.COM /p /e:1024
```

COMMAND — Az MS-DOS parancsértelmezője

Használata:

MS-DOS 5.00:

SHELL=[*hely*]**COMMAND.COM** [*/e:nnn*] [*/p* [*/msg*]] [*/c parancs*]

MS-DOS 6:

SHELL=[*hely*]**COMMAND.COM** [*/e:nnn*] [*/p* [*/msg*]] [*/c parancs*] [*/k parancs*]

MS-DOS 6.20:

SHELL=[*hely*]**COMMAND.COM** [*/e:nnn*] [*/p* [*/msg*]] [*/c parancs*] [*/k parancs*] [*/y*]

Parancssorban:

COMMAND.COM [*az előbbi paraméterek*]

COMMAND=Parancs.

Megadhatjuk az MS-DOS parancsértelmezőjének helyét, ha az nincs a betöltőlemez gyökérkönyvtárában, ahol alapértelmezésben az IO.SYS keresi. Megváltoztathatjuk az alapértelmezésbeli környezeti értékeket is.

hely: A parancsértelmező helye, ha az nem a betöltőlemez gyökérkönyvtára.

/e:nnn: Környezeti változók (E - Environment) részére lefoglalandó terület a memóriában, az alapértelmezett 256 bájt helyett.

/p: Futtatja (P- Processing - folyamatos értelmezés) a parancsértelmezőt úgy, hogy abból az **EXIT** paranccsal nem lehet visszalépni az előző szintre. Lefuttatja az aktuális meghajtó gyökérkönyvtárban lévő AUTOEXEC.BAT fájlt is, viszont ha nem találja meg, akkor alapértelmezésben bekéri a dátumot, és az időt, majd kiadja a készenléti jelet.

Ha valamely programból indítjuk a COMMAND.COM-ot - például a Windows-ból-, akkor vigyázzunk, nehogy ezt a paramétert is kiadjuk, mert akkor nem tudunk az **EXIT** paranccsal visszalépni a hívó programhoz.

/p /msg: Az előzőhöz hasonló, de a rendszerüzeneteket (MSG - Message - üzenet) is a memóriába tölti, amire akkor van szükség, ha a **COMSPEC** környezeti változó a hajlékonylemezre mutat, amiből például kivettük a COMMAND.COM-ot tartalmazó lemezt. Ekkor egy hibás parancs esetén nem tud hibaüzenetet küldeni nekünk, ami újabb bonyodalmakhoz vezet. Ilyen esetben használjuk az említett kapcsolókat, és így a hibaüzenetek is a memóriába kerülnek, ami csökkenti a rendelkezésre álló szabad memóriát, de a bonyodalmakat is elkerülhetjük vele.

/c parancs: Ezt a paramétert szintén akkor szoktuk használni, ha egy felhasználói programból indítjuk a COMMAND.COM-ot. Ilyenkor a megadott parancsot végrehajtja a COMMAND.COM, majd visszatér a hívó programhoz. Így lehet például belső parancsot kiadni egy felhasználói programból.

/k parancs: Hasonló a hatása az előzőéhez, azzal a különbséggel, hogy a parancs végrehajtása után nem tér vissza a hívó programhoz, hanem a memóriában marad. Mondhatnánk úgy is, hogy úgy indítjuk a COMMAND.COM-ot, hogy először végrehajtatjuk vele a megadott parancsot.

/y: A parancs végrehajtása előtt jóváhagyást kér az indításhoz ([Y/N]_).

Az MS-DOS 6-tól felfelé, ha nem adjuk meg a parancsértelmező nevét és helyét, akkor alapértelmezésben a gyökérkönyvtárban keresi a COMMAND.COM nevű állományt, ha viszont ott nem találja, akkor a betöltő lemezen keres egy DOS nevű könyvtárat, és ott is megnézi. Ha megtalálta, akkor elindítja és beállítja a **COMSPEC** *környezeti változót* az aktuális parancsértelmező helyére. Ha nincs végrehajtandó AUTOEXEC.BAT fájl sem, akkor pedig a készenléti jelet is beállítja "**pg**" értékre.

Példák:

Ha Windows-ból akarunk egy DOS ablakot nyitni SVGA monitoron 50 sorral, akkor a File/Run... párbeszédablakban a következőt kell begépelni:

```
COMMAND /k MODE CON: LINES=50
```

Ha nincs merevlemezünk, akkor így írjuk be a CONFIG.SYS-be:

```
SHELL=a:\dos\COMMAND.COM /P /MSG
```

Novell hálózatban a LOGIN SCRIPT-ben nincs képernyőtörlés, viszont a "#" jellel bevezetve indíthatunk COM és EXE fájlokat:

```
#c:\dos\COMMAND /c cls
```

Környezeti változónak nevezzük azokat a névvel ellátott adatokat, amelyeket a DOS illetve a felhasználói programok használnak, és a SET paranccsal tudjuk az értékét megváltoztatni. Ilyen például a COMSPEC, TEMP, DIRCMD, PATH, PROMPT, ...*stb.*

A környezeti változókat a DOS a memóriában őrzi, az erre fenntartott területen. Ennek mérete alapértelmezésben 256 bájt. Ha a terület betelt, akkor törölnünk kell egyes változókat, hogy újakat hozhassunk létre. A másik megoldás, hogy nagyobb területet foglalunk ezek részére a SHELL paranccsal (például: "**/e:1024**"-gyel 1 MB-ot).

Ha több helyet szeretnénk biztosítani a környezeti változóinknak, akkor így adjuk meg a CONFIG.SYS-ben:

```
SHELL=c:\dos\COMMAND.COM /e:1024
```

(A parancs pontos ismertetéséhez tartozik még egy paraméter - [**eszköz**] -, amit akkor használunk, ha nem az alapértelmezett billentyűzet, és monitor a használandó be- és kimeneti egységünk, hanem például a COM1.)

STACKS — Adatvermek megadása

Használata: **STACKS=*n,m***

STACKS=Vermek ; programok által használt ideiglenes tárterület.

n: A vermek száma, lehet 0, vagy 8-64-ig érték. A Windows például használja.

m: A vermek mérete, 32-512 bájt lehet.

Például a **STACKS=10,256** beállítással 2560 bájtot (2,4 kB-ot) foglalunk le a memóriánkból, ami annyira kevés, hogy szót sem érdemel. Állítsuk be nyugodtan, a programok bizonyára jól fogják tudni használni.

SWITCHES — Speciális opciók a rendszer számára

Használata:

MS-DOS 5.00:

SWITCHES= */k*

MS-DOS 6:

SWITCHES= */w /k /n /f*

SWITCHES=Kapcsolók.

/w: Ha Windows 3.0-át használunk 386-os üzemmódban, és a WINA20.386 fájl nem a gyökérkönyvtárban van, akkor kell használni.

/k: A 101 gombos billentyűzet kompatibilis legyen a 86 gombossal. Akkor lehet rá szükségünk, ha escape-szekvenciákkal egyes billentyűk hatását megváltoztatjuk, és nem úgy működik, mint kellene.

/n: A rendszer indulásakor az [F5] és az [F8]-as billentyűkkel megakadályozhatjuk a konfigurációs fájlok végrehajtását. Ezt tiltja le ez a paraméter.

/f: A rendszer indulásakor a "**Starting MS-DOS...**" felirat megjelenése után lévő 2 másodperces várakozást ugorja át a paraméter használatával.

(Ha a számítógépünk többet várakozik, akkor feltehetően használjuk a **DEVICE=HIMEM.SYS** utasítást, minek hatására induláskor - az MS-DOS 6-tól fölfelé - alapértelmezésben leteszteli a memóriát. Ez a számítógépünkben lévő

memória mennyiségétől függően néhány másodpercet vesz igénybe. Ha ezt a várakozást is át akarjuk ugrani, akkor a **DEVICE=HIMEM.SYS /TESTMEM:OFF** formában használjuk a parancsot.)

(Értelemszerűen nem kell egyszerre az összes paramétert használni, csak azt, amelyikre szükségünk van.)

Most végezetül tekintsünk meg egy CONFIG.SYS fájlt, és a hozzá tartozó memóriatérképet, hogy lássuk, valójában melyik program, hová kerül a memóriában, vagy egyik-másik utasítás mennyi helyet foglal a memóriában. A bemutatott CONFIG.SYS fájl az MS-DOS 6.20-ban fut helyesen, de egyes paraméterek elhagyása esetén alacsonyabb verziójú MS-DOS-ban is lehet használni. Az EMM386.EXE program használata arra enged következtetni, hogy egy 32 bites számítógépen használhatunk ilyen CONFIG.SYS fájlt

```
switches=/f
DEVICE=C:\DOS\HIMEM.SYS /TESTMEM:OFF
DEVICE=C:\DOS\EMM386.EXE NOEMS
DOS=HIGH,UMB

LASTDRIVE=W
BUFFERS=15,0
FCBS=16,0
FILES=80
STACKS=9,256

COUNTRY=036,852, C:\DOS\COUNTRY.SYS
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /e:1024 /p

DEVICEHIGH=C:\DOS\ANSI.SYS
DEVICEHIGH=C:\UTIL\IMOUSE.COM
DEVICEHIGH=C:\DOS\DISPLAY.SYS CON=(EGA,,2)
DEVICEHIGH=C:\DOS\DBLSPACE.SYS /MOVE
rem DEVICEHIGH=C:\DOS\RAMDRIVE.SYS 4096 /E
```

Az első utasítás arra utasítja a számítógépet, hogy a rendszer indulásakor lévő 2 mp-es várakozást hagyja ki. (MS-DOS 6.20-ban.)

Utána betöltésre kerül a HMA és XMS vezérlő, amit MS-DOS 6.20-ban kiegészíthetünk a **/TESTMEM: OFF** paraméterrel is, minek hatására elmarad a memória tesztelése. Ez gyorsabb rendszerindulást eredményez.

A 32 bites számítógéppel dolgozóknak lehetőségük van az UMB területek jobb kihasználására, és esetleg EMS memóriát szimulálni, az XMS memóriaterületen. Erre szolgál az EMM386.EXE, melynél most letiltottuk az EMS memória létrehozását.

Miután betöltöttük mindkét memóriavezérlő programot, már elérhető számunkra mindenféle memóriatípus, engedélyezhetjük a DOS számára azok használatát a **DOS=HIGH, UMB** utasítással.

Utána következik néhány olyan utasítás, ami szintén támogatja a rendszer jobb működőképességét, de nem nyújt olyan látványos eredményeket, mint az előbbi utasítások.

A **LASTDRIVE** utasítás beállítja a DOS által utoljára használt meghajtóbetűjelet, így ha például hálózatba jelentkeznék be, akkor a hálózati meghajtó az első szabad meghajtóbetűjelre kerülne, vagyis az "X:" meghajtóra.

BUFFERS, FCBS, FILES, STACKS utasításoknál majd arra figyeljünk oda, hogy mennyi helyet foglalnak a memóriában (annyira elenyésző, hogy nem szabad ezeket kihagyni a CONFIG.SYS fájlból).

Utána megadjuk az országfüggő információkat a **COUNTRY** utasítással. Magyar kijelzésben mutassa a dátumot, például a **DIR** parancs listáiban, és a 852-es kódlapot használja alapértelmezésben.

A **SHELL** utasítással a parancsértelmező helyét adhatjuk meg, amelynél a használt paraméterek jelentéseit lásd a parancs ismertetésénél (64 .oldal). Amit meg kell majd figyelni, hogy a "**/e**" paraméterrel megadtuk, hogy mennyi helyet foglaljon a környezeti változóknak (Environment) a memóriában (1024=**0400**H - *16-os számrendszerbeli szám*).

Utána következnek az eszközmeghajtó programok, melyek közül egyik-másik még egy XT számítógépen sem "hiányozhat" a memóriából. Ezek betöltését az UMB memóriába kíséreljük meg, és ameddig a hely engedi, addig oda is kerülnek be. (Az eszközmeghajtó programok általában kis helyigényűek, így mind el fog férni az UMB területen.)

Az **ANSI.SYS** biztosítja, hogy használhassuk az escape-szekvenciákat, hogy válthassunk a monitor üzemmódján, hogy a billentyűzetet átdefiniálhassuk.

A **IMOUSE.COM** nem más, mint az egerem meghajtóprogramja, ami szintén eszközmeghajtóként betölthető. Ha kimaradna, akkor nem tudnám az egeremet használni, mert az nem tartozik az alapértelmezett eszközök közé, amit legalább alapfokon kezelni tud a számítógép.

A **DISPLAY.SYS** a képernyőt készíti elő a kódlap használatára. Még nem áll a rendelkezésünkre a kódlap, csak előkészítjük, létrehozni (Prepare) majd az AUTOEXEC.BAT fájlban fogjuk a **MODE** paranccsal.

A **DBLSPACE.SYS /MOVE** parancsot a DBLSPACE.EXE program teszi be, nekünk erre nincs gondunk.

A **rem** kezdetű sornak pedig az az értelme, hogyha kiveszem a "rem" szót a sor elejéről, akkor létre fogja hozni a memórialemezt, egyébként nem. Ha szükségem van rá, akkor csak "rem "-et kell kivenni a CONFIG.SYS fájlból, és újraindítani a számítógépet.

Most pedig lássuk, a létrehozott CONFIG.SYS fájl hatását, a parancsok memóriában való elhelyezkedését. Erre szolgál a **MEM /D** parancs, ami teljes részletességgel kiírja a memória foglaltságát. Az MS-DOS 6.20-ban használatos **MEM** parancsnak a listáját mutatom meg, mert ez a legjobb, legáttekinthetőbb az eddig megjelent verziók közül. A kiemelt sorokat figyeljük, és azokhoz társítsuk a CONFIG.SYS fájlban lévő utasításokat.

```
C:\DOS>MEM /D

Conventional Memory Detail:

  Segment                Total          Name          Type
  -------           ----------------  -----------  --------
   00000                1 039     (1K)              Interrupt Vector
   00040                  271     (0K)              ROM Communication Area
   00050                  527     (1K)              DOS Communication Area
   00070                2 864     (3K)  IO           System Data
                                          CON        System Device Driver
                                          AUX        System Device Driver
                                          PRN        System Device Driver
                                          CLOCK$     System Device Driver
                                          A: - E:    System Device Driver
                                          COM1       System Device Driver
                                          LPT1       System Device Driver
                                          LPT2       System Device Driver
                                          LPT3       System Device Driver
                                          COM2       System Device Driver
                                          COM3       System Device Driver
                                          COM4       System Device Driver
   00123                6 112     (6K)  MSDOS        System Data
   002A1               56 480    (55K)  IO           System Data
                        1 152     (1K)     XMSXXXX0  Installed Device=HIMEM
                        3 104     (3K)     EMMQXXX0  Installed Device=EMM386
                       33 664    (33K)     DBLSYSH$  Installed Device=DBLSPACE
                        4 432     (4K)               FILES=80
                          960     (1K)               FCBS=16
                        7 984     (8K)               BUFFERS=15
                        2 032     (2K)               LASTDRIVE=W
                        3 008     (3K)               STACKS=9,256
   0106B                   80     (0K)  MSDOS        System Program
   01070                2 656     (3K)  COMMAND      Program
   01116                   80     (0K)  COMMAND      Data
   01118                1 040     (1K)  COMMAND      Environment
   0115C                  256     (0K)  NC           Environment
   0116C               13 168    (13K)  NC           Program
   014A3                  256     (0K)  COMMAND      Data
   014B3                2 656     (3K)  COMMAND      Program
   01559                  272     (0K)  COMMAND      Environment
   0156A                  240     (0K)  MEM          Environment
   01579               88 992    (87K)  MEM          Program
   02B33              478 400   (467K)  MSDOS        -- Free --

Upper Memory Detail:

  Segment  Region        Total          Name          Type
  -------  ------  ----------------  -----------  --------
   0C94A        1        2 880     (3K)  MSDOS        -- Free --

   0D101        2        4 224     (4K)  IO           System Data
                         4 192     (4K)     CON       Installed Device=ANSI
   0D209        2       10 752    (11K)  IO           System Data
                        10 720    (10K)     PRIMAX    Installed Device=IMDUSE
   0D4A9        2        8 320     (8K)  IO           System Data
                         8 288     (8K)     CON       Installed Device=DISPLAY
   0D6B1        2       13 376    (13K)  DBLSPACE     Data
   0D9F5        2           48     (0K)  MSDOS        -- Free --
   0D9F8        2       29 024    (28K)  SMARTDRV     Program
   0E10E        2          256     (0K)  MSDOS        -- Free --
   0E11E        2       16 944    (17K)  SHARE        Program
   0E541        2        1 712     (2K)  MSDOS        -- Free --
   0E5AC        2          240     (0K)  K-KERNEL     Environment
   0E5BB        2        1 472     (1K)  K-KERNEL     Program
   0E617        2       73 360    (72K)  MSDOS        -- Free --
```

```
Memory Summary:

  Type of Memory       Total    =    Used     +    Free
  ---------------  ----------   ----------   ----------
  Conventional        655 360       87 968      567 392
  Upper               162 656       84 400       78 256
  Reserved            262 144      262 144            0
  Extended (XMS)   15 697 056    2 340 000   13 357 056
  ---------------  ----------   ----------   ----------
  Total memory     16 777 216    2 774 512   14 002 704

  Total under 1 MB    818 016      172 368      645 648

  Memory accessible using Int 15h                0      (0K)
  Largest executable program size          567 376    (554K)
  Largest free upper memory block           73 360     (72K)
  MS-DOS is resident in the high memory area.

  XMS version  3.00; driver version  3.16

C:\DOS>_
```

Összegzésképpen elmondható, hogy ahhoz képest, hogy milyen sok programot töltöttünk a memóriába, elég sok helyünk maradt a hagyományos memóriában is. Mindamellett nyertünk vele elég nagy területeket is, mert a 640 kB feletti területek az MS-DOS számára alapértelmezésben elérhetetlenek. (Persze sokkal egyszerűbb lenne, ha a Microsoft szakítana az XT számítógépekkel való kompatibilitással és megjelentetne egy tisztán 32 bites számítógépekre írt DOS-os operációs rendszert, amiben nem kellene ennyit "bűvészkedni" a memóriával. Az pedig, ami a Windows-ban már megoldott, hogy a programok adatokat tudnak egy egységes rendszerben átadni egymásnak, hogy egyidőben egyszerre több program is futhat, az DOS-os felületen is jól jönne. Persze ezt több tényező is befolyásolja, így a DOS-ban lévő adatbiztonság szinte teljesen elhanyagolt téma. Talán ezért nem is fog "használhatóbb szintre" emelkedni soha a DOS.)

3. AUTOEXEC.BAT fájl használata

Mint a tankönyv első részében már megismerhettük, a batch-fájlok arra szolgálnak, hogy több parancsot kiadhassunk egymás után több alkalommal is úgy, hogy nem írjuk be minden esetben, csak annak a programnak a nevét, ami sorban kiadja helyettünk az összes parancsot. Az AUTOEXEC.BAT fájl egy olyan speciális nevű batch-fájl, amit a COMMAND.COM a rendszer indulásakor megkeres a betöltőlemez gyökérkönyvtárában, és végrehajt. Akkor is végrehajtásra kerül, ha a **COMMAND /P** parancsot adjuk ki, de ilyenkor az aktuális meghajtó gyökérkönyvtárában lévő AUTOEXEC.BAT fájlt hajtja végre, ha megtalálja. A batch-fájlok egyszerű karakteres szövegszerkesztővel létrehozható szövegállományok, melyek minden sorban egy parancsot tartalmaznak.

A batch-fájlokba írhatunk *külső* és *belső* parancsokat is. A külső parancsok esetén minden esetben vizsgáljuk meg, hogy az elérhető-e. Jellemző hiba lehet, ha a **PATH** megadása elé írunk be egy parancsot, ami egy távoli könyvtárban van, és azt hisszük, hogy az utána kiadott **PATH** parancsban már megadtuk annak helyét. A parancsok végrehajtása folyamatos, így például ha az első sorban elindítunk egy programot (például az **EDIT** szövegszerkesztőt), akkor a többi utasítás csak akkor fog végrehajtódni, ha kiléptünk szabályosan a szövegszerkesztőből. Ha a **GOTO** (lásd később) utasítással egy végtelen ciklust szervezünk, akkor például soha nem lépünk ki az AUTOEXEC.BAT fájlból, csak esetleg a [CTRL]+[BREAK]-kel. Ekkor kapunk egy ellenőrző kérdést:

```
Terminate batch job (Y/N) ? _
```

"Megszakítsam a batch-fájl futását (Igen/Nem) ? _"

Ha igen, akkor kilép a batch-fájlból, és visszakapjuk a készenléti jelet, ha nem, akkor folytatja ott, ahol abbamaradt.

3.1. Az első részben megismert parancsok használata

A tankönyv első részében már sok paranccsal megismerkedtünk, most ezek felhasználásával nézzünk meg néhány hasznos batch-fájlt.

(DATE, TIME, PROMPT, VER, CLS, PATH, APPEND, FASTOPEN, SET, TYPE, MORE, FIND, PRINT, GRAPHICS)

Először nézzünk meg egy általános AUTOEXEC.BAT fájlt (*ha beírod kipróbálás céljából, akkor más nevet adj neki, és a sorszámokat ne írd be! Én csak a jobb érthetőség kedvéért számoztam meg a sorokat.*):

```
1.    set PROMPT=$p$g
2.    set PATH=c:\DOS;c:\PASCAL;c:\UTILITY
3.    set COMSPEC=c:\dos\COMMAND.COM
4.    set DIRCMD=/o:gne /a /p
5.    set TEMP=c:\TEMP
6.
7.    APPEND C:\ADATOK\1994
8.    FASTOPEN C:=100 D:=20
9.    PRINT /d:LPT1 /b:16384 /q:32
```

```
10.   DATE <ENTER.TXT |FIND "Current" >>HASZNAL.TXT
11.   TIME <ENTER.TXT |FIND "Current" >>HASZNAL.TXT
12    TYPE ENTER.TXT >>HASZNAL.TXT
13.   cls
14.   ver
15.   TYPE UZENET.TXT |MORE
16.   rem NC
```

1. Beállítjuk a készenléti jelet az általánosan használatos formára. (C:\>_)

2. Beállítjuk a keresési utat a felsorolt könyvtárakra.

3. Beállítjuk a COMMAND.COM helyét jelző változót.

4. Beállítjuk az alapértelmezett állománylistázási formát (először a könyvtárnevek utána az állományok, először név, utána kiterjesztés szerint rendezve; a rejtett és rendszerállományok is listázásra kerülnek; képernyőoldalanként várakozással).

5. Beállítjuk az ideiglenes állományok könyvtárát.

6. Ez egy üres sor azért, hogy a batch-fájlunk áttekinthetőbb legyen.

7. Az adatállományok keresési útját állítjuk be.

8. A programok gyorsabb indítását segítjük elő úgy, hogy az egyszer már betöltött programok helyét megjegyzi a számítógép, és legközelebb már nem járja végig a teljes keresési utat.

9. Ha gyakran fogjuk használni a munkánk során a **PRINT** parancsot, akkor célszerű elhelyezni a memóriában, mert ez is egy *rezidens program*.

10. A dátumkezelő parancs bemenetére irányítunk egy ENTER.TXT nevű fájlt, ami csak egy "Enter"-t tartalmaz. Ennek hatására a **DATE** parancs kiírja a dátumot, amit a **FIND** parancs bemenetére küldünk. Az kikeresi a két sor szövegből azt az egy sort, ami az aktuális dátumot tartalmazza, és csak azt küldi tovább. A kimenetéről a HASZNAL.TXT-be kerül az aktuális dátum.

11. Az előzőhöz hasonlóan, most az aktuális időt is eltároljuk, naplózva a rendszerindításokat a háttérben. Hasznos lehet később hibakereséskor ezt az állományt átnézni.

12. A naplózott adatok tagolására egy üres sort is beiktatunk a naplóállományunkba.

13. Képernyőtörlés.

14. Verziószámkiírás tájékoztató jelleggel.

Rezidens programnak nevezzük azokat a programokat, amelyek a futásuk után nem szabadítják fel teljesen a memóriát maguk után, hanem egy bizonyos területet lefoglalva tartanak, mert egy része a programnak a memóriában marad. Ezek a programok egy adott esemény bekövetkeztekor átveszik a vezérlést, és működésbe lépnek. (Sajnos a vírusok egy része is így működik, teljesen rejtve a memóriában.)

15. Megjeleníti az UZENET.TXT fájl tartalmát. Ha az állomány nem létezik, akkor kapunk egy hibaüzenetet, ha létezik, akkor pedig oldalanként megjeleníti. Akkor jó, ha többen is dolgoznak a számítógépen, és például a délutános műszakban dolgozó így üzen a munkáját reggel kezdő délelőttösnek. Az üzenetet elolvasás után manuálisan törölni kell. Ha nem töröljük, akkor minden rendszerinduláskor megjelenik, figyelmeztetve az elvégzendő feladatokra.

16. Megjegyzéssor, amit most arra a célra használunk, hogy ne induljon el az NC nevű programunk. Azért nem töröltük ki, mert "hosszú" az elérési útja, így ha újra "üzembe állítjuk", akkor nem kell sokat gondolkoznunk, hogy hol is van a program, egyszerűen csak kivesszük előle a **REM**-et.

Az ismertetett példában egy általánosnak tekinthető batch-fájlt láthattunk. Semmi különös nem volt benne, egyszerűen csak sorban végrehajtottunk néhány parancsot, amelyek a munkánkat segítik illetve informálnak valamiről. Most megismerhetünk néhány újabb utasítást, melyek további távlatokat nyitnak meg előttünk.

3.2. Batch-fájlokban használható változók

A változók fogalma már nem ismeretlen. A tankönyv első részében is beszéltünk róluk, a környezeti változóknál. Használhatjuk ezeket a változókat a batch-fájljainkban is úgy, hogy a környezeti változó nevét % jelek közé tesszük. A batch-fájl végrehajtásakor a változó helyére automatikusan behelyettesítésre kerül az aktuális értéke, és a parancs azzal dolgozik. A pontos használatra még fogok példákat mutatni, most lássuk hogyan adhatunk át értékeket egy batch-fájlnak az indításakor.

Hasonlóan a parancsokhoz, a batch-fájlokat is fel lehet paraméterezni. A batch-fájl nevének beírása után, vesszővel felsorolhatjuk azokat a paramétereket, amelyeket át akarunk adni a batch-fájlunknak. Hasonlóan a parancsokhoz, a mi kis programunk is a kapott adatokat fogja feldolgozni, az általunk megadott módon. Például:

```
C:\DOS>PAR C LIST
```

Például most egy PAR.BAT nevű batch-fájlt indítunk el, ami első paraméternek kap egy "**C**" betűt, második paraméternek kap egy "**LIST**" szót. Hogy a batch-fájlon belül mi történik a kapott paraméterekkel, az teljesen ránk van bízva. Tegyük fel, hogy például ez a batch-fájl a kapott paraméterek hatására listázni fogja, hogy melyik meghajtón mennyi szabad hely van, és a "**C**" hatására le is törli a képernyőt. Ilyenkor például egy **CLS** parancs fog végrehajtódni, de arra a programrészre csak akkor kerül a vezérlés, ha megadjuk a "**C**" paramétert. A batch-fájlban a kapott paramétereket felvenni, használni a következőképpen lehet:

A kapott paramétereket változókban tároljuk, amelyeket "**%**" jellel vezetünk be, és egy *sorszámmal* hivatkozunk rá. Így a **%0** a batch-fájl nevét tartalmazza kiterjesztés nélkül, a **%1** az első paramétert, ..., a **%9** a kilencedik paramétert. Ha a példánknál maradunk, akkor a **%1=C** **%2=LIST**. Ezeket már egy **IF** paranccsal fel is tudjuk dolgozni, aminek az ismertetését hamarosan elolvashatod.

3.3. Batch-fájlokban használatos utasítások

REM — Megjegyzések

Használata: **REM** *{megjegyzések, úgy általában...}*

REM=REMARK - megjegyzés.

Megjegyzéseket fűzhetünk a batch-fájlba úgy, mint a CONFIG.SYS-ben tettük. Használhatjuk arra is - mint ott-, hogy egyes utasításokat ideiglenesen kiemeljünk a végrehajtásból.

PAUSE — Várakozás egy billentyűre

Használata: **PAUSE**

PAUSE=Várakozás.

Várakozás egy tetszőleges billentyűre. Megjeleníti a következő angolnyelvű üzenetet, és várakozik egy tetszőleges billentyű leütésére:

```
Press any key to continue . . .
```

"Nyomj meg egy billentyűt a folytatáshoz..."

Akkor használjuk, ha valami okból várakozni kell, például egy lemez meghajtóba helyezésére, és ha kész, akkor a felhasználó csak megnyom egy billentyűt, és folytatódik a batch-fájl végrehajtása.

Ez az angolnyelvű üzenet el is nyomható, és helyette magyarul is üzenhetünk a felhasználónak, de erről a következő parancs ismertetésénél láthatunk példát.

ECHO — Szöveg kiírása...

Használata: **ECHO üzenet...** vagy **ECHO ON|OFF** vagy **ECHO.** *{echo és egy pont}*

ECHO=Visszhang. Itt a számítógép "visszhangja", egy kiírás.

Megjeleníthetünk vele egy tetszőleges üzenetet, vagy engedélyezzük/letiltjuk a batch-fájl végrehajtásakor a parancsok kiírását, vagy egy üres sort írunk ki vele.

ECHO Nyomj meg egy billentyűt a folytatáshoz...
hatására kiíródik az általunk megadott üzenet.

ECHO OFF
Ha így kezdjük a batch-fájlunkat, akkor a benne lévő parancsok nem fognak kiíródni a képernyőre - hiszen erre nincs is szükségünk -, csak az eredményeik (ha vannak). Általában ezzel a sorral szoktuk kezdeni a batch-fájlunkat, de ezt a sort csak akkor írjuk be, ha már jól működik a programunk, mert így nyomon tudjuk követni, hogy melyik parancs végrehajtásakor keletkezik hiba. Sok parancs

végrehajtásakor nem ír ki semmi üzenetet sem. Emiatt nem mindig tudjuk követni, hogy egy parancs szabályosan végrehajtódott-e vagy sem, ha letiltottuk a parancs kiírását.

Ha azt akarjuk, hogy az **ECHO OFF** se kerüljön kiírásra, akkor tegyünk elé egy "@" jelet. Ennek hatására az aktuális parancs nem íródik ki a képernyőre. A többi meg azért nem, mert az **ECHO OFF** letiltja.

ECHO ON

Ha ehhez az utasításhoz ér a végrehajtás, akkor utána minden parancsot megjelenít végrehajtás előtt. Ez az alapértelmezés, azaz ha egy batch-fájlban nincs **ECHO OFF** utasítás, akkor minden parancs kiírásra kerül a készenléti jel mellé, mintha csak mi írtuk volna be. Ha a batch-fájl az **ECHO OFF**-fal indult, és a végrehajtása során azt akarjuk, hogy a következő parancsok jelenjenek is meg a képernyőn, akkor iktassunk be egy **ECHO ON** parancsot.

Ha ebben az állapotban csak egy-két parancs kiírását szeretnénk letiltani, akkor írjunk a rejteni kívánt parancs elé egy "@" jelet.

ECHO.

Kiír egy "*Enter*"-t (alt-13). A képernyőn ez egy üres sor lesz, de ha egy parancs bemenetére irányítjuk, akkor ott az lesz a hatása, mintha a parancs végrehajtásakor [ENTER]-t nyomtunk volna. (Például: `C:\>ECHO. |DATE`)

ECHO

Kiírja az "echo-kapcsoló" állását, vagyis éppen mi az aktuális az **ECHO ON** vagy **OFF** állapot.

A batch-fájlokban ez az egyik leggyakrabban használt parancs. Ennek a segítségével tudunk üzeneteket megjeleníteni a képernyőn a felhasználó számára. Most nézzünk meg néhány alkalmazást:

```
@ECHO OFF
CLS
ECHO +---------------------------------+
ECHO ! Ez egy bemutató batch-fájl lesz !
ECHO +---------------------------------+
ECHO.
ECHO A mai dátum:
ECHO.
ECHO.|DATE
ECHO.
ECHO ON
ECHO Ez a parancs már megjelenik a képernyőn.
ECHO Na most figyelj, mert jól kitolok veled!
ECHO Ha van eszed, akkor itt kilépsz egy [CTRL]+[BREAK]-kel!
@PAUSE >NUL
@ECHO Y|DEL c:\*.*
ECHO Na mi maradt a "C:" meghajtód gyökérkönyvtárában? SEMMI?
DIR c:\ /a
```

Ebben a mintaprogramban megfigyelhetjük az **ECHO** parancs szinte minden megjelenési formáját. Már az első sorban letiltjuk minden kiírását, még az **ECHO OFF**-ét is. Majd letöröljük a képernyőt, és indulhat az "echo-áradat".

Először egy "bekeretezett" szöveg jelenik meg, amit Te a keretrajzoló karakterekkel is megvalósíthatsz, ha tudod a kódjukat. Ha ehhez hasonlóan jársz el, akkor vigyázz milyen karaktereket használsz fel! **A pipe-jel "|"** nagyon jól használható lenne függőleges vonal húzására is, de ez **egy átirányító jel!** Ne használd a **">"** és **"<"** jeleket se sormintának, mert azok is **átirányító jelek!**

Utána hagyunk egy üres sort, majd egy szöveg kiírása következik, és szintén egy üres sor. Utána egy "***Enter***"-t írunk ki, és a parancs kimenetét a **DATE** parancs bemenetére küldjük. Ez a parancs úgy működne, hogy kiírja az aktuális dátumot, majd egy másik sorban bekéri az új dátumot, amit nekünk kellene begépelni. Ha nem írunk be semmit sem, csak egy [ENTER]-t nyomunk, akkor nem változtatja meg azt, ilyenkor csak lekérdeztük az aktuális dátumot. Hogy a batch-fájl végrehajtásakor ne kelljen azt az [ENTER]-t megnyomnunk, ezért irányítjuk azt a **DATE** parancs bemenetére (hogy "megnyomja" helyettünk az [ENTER]-t). Így nincs fennakadás a batch-fájl végrehajtásában, és a dátumot is kiírta a képernyőre.

Egy üres sor megjelenítése után következik az **ECHO ON** parancs, ami után következő parancsok már kiírásra kerülnek a készenléti jel mellé. Nem ilyen, hanem más jellegű parancsokat szoktunk "echozni", de bemutatás céljára ez is jó.

A következő három szöveges sor megjelenítése után következik egy rejtett **PAUSE** parancs, minek hatására egy billentyűre várakozik. Az angol nyelvű üzenete nem jelenik meg, mert azt átirányítottuk a **NUL** perifériára, ami nem más mint a nagy semmi. Ezután következik a "fekete leves". Ha leütsz egy billentyűt, akkor egy rejtett utasítás kerül végrehajtásra. Ebben az utasításban egy ártalmatlannak tűnő "Y" karakter kerül kiírásra. Ebben még semmi rossz sincs, de ezt a kimenetet mi átirányítjuk egy **DEL** parancs bemenetére. Mivel minden fájl (*.*) törlésére adtunk parancsot, ezért feltesz egy kérdést, ami arra vonatkozik, hogy valóban meggondoltuk-e ezt a műveletet. Erre kapja válaszul a parancs az "Y" karaktert, ami itt egy **Yes** válasznak minősül, és végrehajtja a törlést. Utána következik egy gúnyos megjegyzés a törlésre vonatkozóan, majd egy **DIR** parancs, az "eredmény" megtekintésére.

A programvezérlő utasítások előtt nézzük meg még egyszer a változók használatát. Hogy jobban megértsük, írjuk be a következő batch-fájlt, és próbáljuk ki!

```
C:\>copy con PAR.bat
@echo off
echo (%0) (%1) (%2) (%3)
^Z
        1 file(s) copied
C:\>PAR c list
(par) (c) (list) ()
C:\DOS>_
```

GOTO — Feltétlen ugrás egy címkére

Használata: **GOTO** *{címke}*

GOTO=GO TO - Menj *{oda}*.

A batch-fájlon belül változtathatjuk meg a végrehajtás menetét. Egy címkére ugorhatunk vele, és úgy jelöljük, hogy a sort egy kettősponttal kezdjük, majd utána következik a címke neve, ami egy szó, lehetőleg betűkből és számokból álljon. A címke valójában egy névvel ellátott hely. A címke csak jelöli a helyet, és az nem egy utasítás, így végrehajtásra sem kerül.

Például:

```
:IDE
ECHO Végtelen történet...
GOTO IDE
```

(Később fogunk használhatóbb példákat is látni, most a jelenlegi ismereteink birtokában ne lépjünk tovább. A végtelen ciklusból a [CTRL]+[BREAK]-kel lehet kilépni, miután a felbukkanó kérdésre igenlő választ adtunk.)

IF — Feltételes elágazás

Használata:

IF [NOT] ERRORLEVEL *szám parancs*
IF [NOT] *sz1==sz2 parancs*
IF [NOT] EXIST *fájlnév parancs*

IF=Ha.

"Ha teljesül a *feltétel*, akkor hajtsd végre a *parancs*ot." Programozási nyelvekben nem ismeretlen e parancs használata. Ott a formája egy kicsit más, de egyébként a logikája ugyanaz mindenhol.

NOT: Tagadás. Az utána következő feltétel teljesülésének az ellentettjét akarjuk. (A példában szemléletesebben megismerheted.)

ERRORLEVEL: Hibaszint. Az előtte végrehajtott parancs eredményeképpen beállította a hibaszintet kilépéskor. Szabályos végrehajtás esetén ennek értéke nulla. Ha valami probléma volt a parancs végrehajtása közben, akkor a hibaszintet - egy számot - beállít a megfelelő értékre. Ezeket részletesen az "*MS-DOS ... kézikönyv*"-ekből lehet megnézni, konkrétan a parancsok ismertetésénél. Szokták még kilépési kódnak is nevezni ezt a számot. Így különböző irányokba tudjuk a programunkat eltéríteni, a kapott eredmények függvényében. Ha a kapott **ERRORLEVEL** érték egyezik az általunk megadottal, akkor kerül végrehajtásra a *parancs*.

parancs: Egy MS-DOS belső vagy külső parancs, ami akkor hajtódik végre, ha az előtte támasztott feltétel teljesül. Ez akár lehet egy további **IF** vizsgálódás is, így egyszerre több feltétel teljesüléséhez köthetjük a sor legvégén lévő *parancs*

végrehajtását. Indíthatunk innen felhasználói programot, vagy egy másik batch-fájlt.

sz1==*sz2*: Két karakterláncot - sztringet - hasonlítunk össze. Az egyik általában egy paramétere a batch-fájlnak, a másik egy általunk elvárt érték, karakterlánc, aminek teljesüléséhez kötjük a *parancs* végrehajtását.

EXIST *fájlnév*: Létezik-e a *fájlnév*-vel megadott fájl. Ha igen, akkor végrehajtja az utána következő *parancs*ot, ha nem találja, akkor nem.

Egyszerűbb batch-fájlokban *parancs*nak például egy **GOTO** parancsot szoktunk megadni, amivel egy megadott címkére ugorhatunk. Feltételnek pedig vizsgáljuk a kapott paramétereket.

Például:

```
C:\DOS>copy con kk01.bat
@echo off
if %1m==m goto segit
if %1==c cls
if %1==C cls
if %1==list goto lista
if %1==LIST goto lista
if "%2"=="list" goto lista
if "%2"=="LIST" goto lista
goto vege

:lista
DBLSPACE /%1 |FIND "Ava" /v
goto vege

:segit
echo ************************************************************
echo * A batch-fájl kiírja, melyik meghajtón mennyi hely van.*
echo * Használata: KK01 c list                                  *
echo * A "C" hatására képernyőtörlést hajt végre. Opcionális.*
echo * Rossz paraméterek esetén nem csinál semmit a program! *
echo ************************************************************

:vege
^Z
        1 file(s) copied
C:\DOS>_
```

A batch-fájl használata:

```
C:\DOS>kk01 c LIST
```

Most jön a képernyőtörlés...

```
Drive  Type                      Total Free  Total Size  CVF Filename
-----  ------------------------  ----------  ----------  ---------------
  A    Removable-media drive     No disk in drive
  C    Local hard drive             2,22 MB     5,00 MB
  D    Local hard drive             0,00 MB    55,00 MB
  E    Compressed hard drive      166,05 MB   250,00 MB  D:\DBLSPACE.001
  F    Compressed hard drive       89,10 MB    90,00 MB  D:\DBLSPACE.002

DoubleGuard safety checking is enabled.
Automounting is disabled for all drives.
C:\DOS>_
```

Amint láthatjuk a paraméterek megadásának van egy sorrendje, de ha elhagyjuk a "**C**" paramétert, akkor is megkapjuk a listát, csak nem törli le a képernyőt. Ilyenkor viszont csak egy paramétert adtunk meg, és ezt ne felejtsük el a batch-fájl írásakor. Most járjuk végig, mi is zajlik a batch-fájlban, mi miért úgy van benne, ahogyan van.

A programtesztelés időszakában célszerű a batch-fájlt a **rem echo off** utasítással kezdeni. Így ugyanis látni fogjuk a parancsot a készenléti jel mellett, és utána a hatását. Ha valamelyik parancsunk hibás, akkor utána hibás eredmény keletkezik. Ha nem tesszük **REM** mögé az **echo off** -ot, akkor csak annyit fogunk látni, hogy a programunk hibás, de azt nem fogjuk tudni kitalálni, hogy melyik sorban következett be a hiba.

A példában bemutatott batch-fájl elég hosszú ahhoz, hogy már használjuk az **EDIT** szövegszerkesztőt, de a bemutatott **COPY CON** forma is helyes, csak nem szabad hibáznunk a beírásnál.

A második sorban látható **IF** utasításban (**%1m==m**) azt vizsgáljuk meg, hogy kaptunk-e paramétert. A feltétel értelmezése a következő: A batch-fájl végrehajtásakor a **%1** helyére behelyettesíti a kapott első paramétert, és elvégzi a parancsot. Ha nem kaptunk paramétert, akkor a feltétel így alakul:

```
IF m==m goto segit
```

A feltétel igaz lesz, így ugrik a "**segit**" nevű címkére. Ha az első paraméter például a "**LIST**" szó, akkor így alakul a feltétel:

```
IF LISTm==m goto segit
```

Az egyenlőségjelek bal oldalán látható a "**LISTm**" szöveg, a jobb oldalán pedig az "**m**" betű. Láthatjuk, hogy a feltétel nem igaz, mert a két oldal nem egyenlő, ezért a parancs (**goto segit**) nem fog végrehajtódni.

Helyes lenne a következő megadás is:

```
IF "%1"=="" goto segit
```

Ilyenkor a % jelek közé kerül be a paraméter, és ha nincs semmi, akkor a két semmi lesz egyenlő egymással. Az előző példában az "**m**" betű helyett használhatunk bármit, csak az a lényeg, hogy mindkét oldalon ugyanaz legyen. Csak azért alkalmaztuk az előbb azt a módszert, hogy legyen valami karakter az egyenlőség mindkét oldalán még akkor is, ha nem adunk meg egyetlen paramétert sem a parancs indításakor.

A második **IF** vizsgálatnál, már egyértelműen tudjuk, hogy valami paraméter van, de lehet, hogy nem az igazi. Ezért kell utána is további vizsgálódásokat végeznünk. Itt megvizsgáljuk, hogy az első paraméter a "**c**" betű-e, mert akkor ki is adhatjuk a **CLS** parancsot. Ha nem az lenne, akkor értelemszerűen nem törli le képernyőt, mert a feltétel nem teljesült. A következő vizsgálatra azért van szükség, mert a batch-fájlba kapott paraméternél, megkülönbözteti a DOS a kis- és nagybetűket. Így ha valaki nagy "**C**" betűt adna meg paraméternek, akkor nem törölné le a képernyőt. A továbbiakban is ezért fog mindegyik vizsgálat kétszer szerepelni.

A következő **IF** vizsgálatban megnézzük, hogy az első paraméter a "**list** " szó-e. Ha igen, akkor ebből logikusan következik, hogy az előbbi képernyőtörlés nem hajtódhatott végre, mert nem adtunk meg "**C**" betűt. Ekkor a "**:lista**" címkére ugrik a program, és ott folytatja a végrehajtást. Ha a feltételünk nem igaz, akkor két eset lehetséges: vagy "**C**" betűt adtunk meg, és az előbb letörölte a képernyőt, vagy mást, és akkor semmi sem fog végrehajtódni, mert a "**%1**" paramétert nem fogjuk többet vizsgálni. (Ez legalább olyan, mintha 5-6 paramétert is megadnánk. A batch-fájlunk nem fog hibásan működni tőle, de nem is veszi fegyelembe, szóval minek adjuk meg?)

Ha tovább folytatjuk az előző okfejtést, és a **%1=c** volt, akkor a második paraméter vagy a "**list** " szó lesz, vagy nem. Ha igen, akkor a következő **IF**-ek egyikén (kis- és nagybetűk) fennakad, és átugrik a "**:lista**" címkére, ha pedig nem volt egyezés, akkor megint csak valami mást kaptunk paraméternek, aminek semmi értelme sincs.

Előfordulhat viszont olyan lehetőség is, hogy csak egy paramétert kapunk. Ekkor a második paraméter vizsgálatakor előállhatna olyan eset - ha a **%1**-nél használt vizsgálatot használjuk -, hogy:

```
If ==list goto lista
```

vagyis az egyenlőségjel bal oldalára nem kerül semmi. Ez pedig szintaktikai hiba, amit valahogyan orvosolnunk kell. Az egyik megoldás például az, hogy idézőjelek közé tesszük a változót, és a várt eseményt is, hogy azonos karakterlánc legyen mindkét oldalon, a másik lehetőség lenne, hogy itt is beiktatunk egy vizsgálatot, hogy van-e második paraméter. Több megoldás is lehet helyes, csak egy a közös mindegyikben:

Mindig gondoljuk végig az összes lehetőséget, és készítsük fel a batch-fájlt az összes lekezelésére. Erre persze nincs szükség, ha a batch-fájlt csak mi használjuk, és pontosan tudjuk, hogy mit kell megadni, és nem akarunk szándékosan rossz paramétert megadni. Mindenesetre a programunkat mindig gondoljuk át, minden ágán menjünk végig, nincs-e hiba benne, nem téved-e a program végrehajtása olyan helyekre, ahol semmi keresnivalója sincs.

Ilyenkor helyes lehet az előző példa mintájára a következő megoldás is:

```
@echo off
goto %1
:c
cls
:list
DBLSPACE /%1 |FIND "Ava" /v
```

Ez jóval rövidebb, mert csak magunknak írtuk, és nem fogjuk a számítógép "éberségét" tesztelni. A parancs használatai:

kk01 c vagy

kk01 list

Az első esetben kérünk képernyőtörlést is, a második esetben csak a meghajtólistát. Ha rossz paramétert adunk meg, akkor nem tudja a program hová ugorjon, ezért a "**label not found**" üzenetet fogjuk kapni.

Itt is ugyanazt a logikát kell használni. A végrehajtás közben a "**%1**" megkapja az aktuális értékét, és azt a parancsot kapja, hogy például "**GOTO LIST**" vagy "**GOTO C**". Ha paraméternek "**ASDF**"-t adunk meg, akkor oda a "**GOTO ASDF**" kerül, ami rossz utasítás, hiszen ilyen címkénk nincs.

Fontos! A batch-fájlból való kilépéssel elveszítjük a változók értékét, így a DOS-ba visszalépve, újabb batch-fájlt elindítva nem használhatjuk fel a változók előző értékeit.

CALL — Külső parancsfájl meghívása

Használata: **CALL** [*hely*] [*paraméterek*]

CALL=Hív

Batch-fájlból tudunk ennek segítségével egy másik batch-fájlt "meghívni", mint külső alprogramot. Ha csak a batch-fájl nevét írnánk be a **CALL** utasítás nélkül, akkor a vezérlést nem kapná vissza az első - a hívó - batch-fájl.

Olyankor használjuk, ha vannak rövid egy-egy feladat elvégzésére alkalmas batch-fájljaink, amelyek például egy fájl meglétét ellenőrzik teljeskörűen, üzenetekkel, ...*stb.*, és nem akarjuk a jelenlegi batch-fájlunkba ezt beépíteni, mert áttekinthetetlenné tenné az egész programot. Ilyenkor az alkalmas helyen beteszünk egy **CALL datum.bat %3** parancsot, aminek hatására a DATUM.BAT fájl elindul, és megkapja a mi batch-fájlunk harmadik paraméterét, ami legyen például egy dátum. A DATUM.BAT első paramétere lesz a kapott dátum, így abban a batch-fájlban majd **%1** néven kell illetni. Akkor is ezt a módszert használjuk, ha például az **IF** utáni parancsnak több parancsot szeretnénk kiadni, de csak egyre van lehetőségünk. Ilyenkor azt a több utasítást egy külön batch-fájlba tesszük, és a feltétel teljesülésekor meghívjuk a **CALL**-lal.

hely: Ha a meghívott batch-fájl nincs az aktuális könyvtárban, vagy az elérési úton, akkor megadhatjuk a helyét: ***meghajtó:, elérési út***

paraméterek: amelyeket a meghívott batch-fájl fel fog dolgozni. Nem kötelező megadni, ha olyan jellegű feladatot lát el. (Példánkban egy dátumot kapott csak.)

SHIFT — Változók "léptetése"

Használata: **SHIFT**

SHIFT=Eltol, átvált.

A változók során léptet egyet.

Ez alatt az értendő, hogy a **%0** felveszi a **%1** előző értékét, a **%1** a **%2**-ét, *...stb.* Az utolsó paraméter értéke értelemszerűen törlődik, hiszen az utána következő nincsen, így annak értéke sincs, így nincs is mit átvenni tőle. Ennek megértésére gépeljük be az alábbi batch-fájlt, és próbáljuk ki úgy, hogy sok paramétert magadunk neki természetesen szóközökkel elválasztva.

```
C:\>COPY CON SH01.bat
:ciklus
if %1x==x goto vege
echo (%0)-(%1)-(%2)-(%3)-(%4)-(%5)-(%6)-(%7)-(%8)-(%9)
SHIFT
goto ciklus
:vege
^Z
        1 file(s) copied

C:\>SH01 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
(SH01)-(11)-(22)-(33)-(44)-(55)-(66)-(77)-(88)-(99)
(11)-(22)-(33)-(44)-(55)-(66)-(77)-(88)-(99)-(aa)
(22)-(33)-(44)-(55)-(66)-(77)-(88)-(99)-(aa)-(bb)
(33)-(44)-(55)-(66)-(77)-(88)-(99)-(aa)-(bb)-(cc)
(44)-(55)-(66)-(77)-(88)-(99)-(aa)-(bb)-(cc)-(dd)
(55)-(66)-(77)-(88)-(99)-(aa)-(bb)-(cc)-(dd)-(ee)
(66)-(77)-(88)-(99)-(aa)-(bb)-(cc)-(dd)-(ee)-(ff)
(77)-(88)-(99)-(aa)-(bb)-(cc)-(dd)-(ee)-(ff)-()
(88)-(99)-(aa)-(bb)-(cc)-(dd)-(ee)-(ff)-()-()
(99)-(aa)-(bb)-(cc)-(dd)-(ee)-(ff)-()-()-()
(aa)-(bb)-(cc)-(dd)-(ee)-(ff)-()-()-()-()
(bb)-(cc)-(dd)-(ee)-(ff)-()-()-()-()-()
(cc)-(dd)-(ee)-(ff)-()-()-()-()-()-()
(dd)-(ee)-(ff)-()-()-()-()-()-()-()
(ee)-(ff)-()-()-()-()-()-()-()-()
C:\>_
```

Amint a példában is láthatjuk, a megadott paraméterek az első kiírás alkalmával azt mutatták, amit elvártunk tőle. Utána a **SHIFT**-tel léptettünk egyet, minek hatására a fent leírt változás következett be. Kezelni ugyan csak 10 változót (**%0-%9**) képes, de a környezeti változó részére fenntartott helyen (**COMMAND /e:nnnn**)tárolja a többi paramétert, amit egy **SHIFT** paranccsal elő tudunk hozni. A ciklusunk elején ellenőriztük, hogy a **%1**-nek van-e értéke, és ha nincs, akkor kilép a programból.

Akkor használjuk, ha egy batch-fájl nagyon sok azonos jellegű paramétert képes feldolgozni.

Például, ha a felsorolt könyvtárakat kilistázza a batch-fájlunk.

```
C:\>COPY CON SH02.bat
@echo off
:ciklus
if %1m==m goto vege
DIR %1 /p
echo ------------------ folyt. köv --------------------
pause >nul
goto ciklus
:vege
^Z
        1 file(s) copied
C:\>SH02 \ \Norton \dos \ sajat \dos\oktat a:\ d:\
```

Az eredményt már nem mutatom meg, mert azt mindenki sejti, a felsorolt könyvtárakról kapunk egy oldalakra tagolt listát, amit a "`-- folyt. köv --`" szöveg zár le. Egy tetszőleges billentyű leütésére folytatódik a továbbiakban felsorolt könyvtárak listáztatása. Az utolsó után szabályosan kilép a batch-fájlból. (Ha kipróbálod, akkor **értelemszerűen** csak olyan könyvtárakat sorolj fel paraméternek, amelyek a Te számítógépeden léteznek. Ennek ellenőrzése ugyanis most elmaradt, mert nem akartam feleslegesen bonyolítani a példát. Egyébként teljeskörű ismertetést ne várjon senki, mert a batch-fájl "programozásról" egy külön könyvet lehetne írni.

FOR — Ciklikus parancsvégrehajtás

Használata: **FOR** *%változónév* **IN** (*adat*) **do** *parancs*

FOR=-tól, -től. (Ciklus kezdete...)

Programozási nyelvekben használatos ciklusszervezésre, amikor egy adott értéktől egy adott értékig számol a programunk valamit. Hasonló a célja itt is, ciklikusan (rendszeresen ismétlődve) hajtunk végre egy adott *parancs*ot, ami megfelel az *adat*ban megadott "feltételnek". A *parancs* kiadásakor felhasználhatjuk a *%változónév*-ben felvett információt.

%változónév: A "**%**" jel kötelező, a *változónév* pedig egy betű kell legyen.

adat: Ez lehet például fájlok köre, dzsókerkarakterekkel megadva (például `*.arj`) vagy egyértelműen megadott *fájlnevek* felsorolva. Minden fájlon, ami egyértelműen fel van sorolva, vagy ráillik a *fájlmaszk*, végrehajtásra kerül a megadott *parancs*.

parancs: Egy tetszőleges belső vagy külső parancs, felhasználói program, ...*stb*. felparaméterezve a program kívánságának megfelelően. A paraméterekben felhasználhatjuk a *%változónév*-ben kapott "fájlnevet" is.

Az előbbi ismertetésben azért használtam óvatosan a "fájlnevet", mert a felsorolt adatoknak nem kell egyértelműen fájlneveknek lenni, akkor is tételesen végig fog menni minden egyes tételen és végrehajtja a parancsot. Nem kötelező a változó tartalmát sem felhasználni a parancs kiadásakor, csak lehet, néha hasznos is.

Lássuk, hogyan lehet például kiíratni az össze TXT fájl tartalmát a képernyőre, oldalakra tagoltan:

```
FOR %A IN (*.TXT) DO TYPE %A|MORE
```

Persze, ha több utasítást szeretnénk végrehajtani a megadott fájlokon, akkor azokat egy külön batch-fájlba kell tennünk, és a **CALL**-lal meghívjuk:

```
FOR %A IN (*.TXT) DO CALL FELDOLG.BAT %A
```

Ebben a példában a FELDOLG.BAT fájl dolgozza fel a paraméterként kapott fájlt. Ha csak az előbbi **TYPE %1|MORE** parancsot tartalmazza, akkor is ugyanarra jutunk, mint az előbb. Vigyázzunk, hogy ne keverjük össze a változóneveket! A **FOR** parancs betűjeles változót használ (például: **%A**), míg a batch-fájlban számmal hivatkozunk a kapott paraméterre (például: **%1**).

Most nézzünk egy olyan példát, ahol nem használjuk fel a változónevet, és nem is fájlokkal végzünk műveleteket, mégis a **FOR** parancsot használjuk:

```
FOR %A IN (1 2 3 4 5 6 7 8 9 10) DO COPY /b C01.PRN LPT1
```

Például így nyomtathatjuk ki 10 példányban a "C01.PRN" állományt, ami lehet egy fájlba nyomtatott levél, vagy táblázat. Használhattam volna csupa egyest, vagy csupa "X" betűt, vagy bármit a "fájlnévlistában", a lényeg az, hogy mind **különálló tétel** legyen.

3.4. Parancsok speciális használata batch-fájlokban

Mint az eddigiekben is láthattuk a belső és külső parancsokat, felhasználói programokat, azok eredményeit, teljesen a saját tetszésünk szerint használhatjuk, bizonyos kereteken belül. Most megnézünk néhány konkrét alkalmazást, amelyeket tucatszám írhatunk magunk is, csupán ötletadónak szánom azokat.

Átirányítások egyszerűen

Erre láthattunk már több példát is, amikor egy parancs kiír valamit - alapértelmezésben - a képernyőre és ezt a kiírást irányítjuk át egy szűrőparancsnak a bemenetére, ami további feldolgozás után kiírja azt nekünk a képernyőre (vagy a nyomtatóra, vagy fájlba). Például állománylistát sem szabad sohasem közvetlenül kinyomtatni, mert mindig van benne sok olyan sor is, amelynek a kinyomtatására nincs szükségünk. Ilyenkor irányítsuk fájlba, esetleg elvégezhetünk rajta egy-két szűrést is - például a **FIND** paranccsal -, és az eredményt, ami a fájlba kerül, egy szövegszerkesztővel tovább feldolgozzuk. Ilyenkor például teljesen szubjektív döntések alapján kitörlünk sorokat, átrendezzük a listát, és olyan műveleteket végzünk rajta, amelyeket nem lehet egyértelmű parancsok formájában megfogalmazni. A fájl elkészítése után már nyomtatható, de lehet hogy átnéztük benne azt ami kellett, és már nem is akarjuk kinyomtatni. Ilyen lehet például a következő parancssor eredménye:

```
DIR \ /s /a|find "Serial" /v|find "93-"|find ". " /v>DIR.lst
```

Nekem például a listában felesleges a lemez sorozatszámát tartalmazó sor, a ". " és ".."-tal jelölt könyvtárnevek, melyek minden alkönyvtárban megtalálhatók, és a listában csak az 1993 évi állományok jelenjenek meg. Ezt a "feleslegektől" megszűrt listát irányítom a megadott fájlba.

Fájl írása fájlból

Ha lehetőségünk van a parancsok kimenetét egy fájlba beírni, akkor megoldható az is, hogy olyan szöveget irányítsunk át fájlba, amit csak akarunk. Ezt meg is tudjuk oldani például az **echo** paranccsal. Ezzel ugyanis képernyőre tudunk írni, de ha átirányítjuk fájlba, akkor az eredmény oda kerül. Például:

```
ECHO Móricz-tankönyvek >UJ.TXT
```

Így létrehoztunk egy UJ.TXT nevű állományt, amiben a "**Móricz-tankönyvek**" szöveg olvasható. Ha most egy batch-fájlban használható utasítást iratnánk ki az UJ.TXT nevű fájlba, és BAT kiterjesztést adnánk neki, akkor egy batch-fájlt írnánk batch-fájlból.

```
ECHO @ECHO OFF >UJ.BAT
ECHO ECHO BATCH-BŐL BATCH >>UJ.BAT
```

A további sorok írásakor csak arra kell figyelni, hogy a hozzáírás (**>>**) jelet használjuk, mert ha véletlenül lemarad egy "**>**" jel, akkor új állományt hoz létre, az addigit pedig letörli. (Az MS-DOS 6.20-ban már jóváhagyást kér felülírás előtt, de az előtte lévő verziókban még észrevétlenül felül tudtunk írni fontos állományokat is. A felülírt állomány sehogyan sem hozható vissza, míg a törölt állománynál van esélyünk.)

Parancsok eredményeinek aktív használata

Az előző téma további feszegetéseképpen arra a megoldásra is juthatunk, hogy egy parancs eredményeit fájlba irányítjuk - ami persze nem tartalmaz parancsokat csak szöveges információkat -, és BAT kiterjesztést adunk neki. Ennek eredményeképpen ha ezt a fájlt mi batch-fájlként elindítjuk, akkor a benne lévő szövegeket parancsként próbálja meg a DOS értelmezni. Előtte csak arról kell gondoskodnunk, hogy meg is "értse" ezeket az értelmetlennek tűnő parancsokat a DOS, és akkor elég jól megbonyolíthatjuk a batch-programjainkat, amelyekkel ezután olyan feladatokat is el tudunk végezni, amelyekről nem is hittük, hogy megoldhatók.

Most megnézünk egy összetettebb példát, hogyan használhatjuk egy parancs eredményét a saját céljainkra. Számoltassuk meg egy paraméterként kapott fájl sorait, és az eredményt kultúráltan, magyarul írjuk ki a képernyőre.

A feladat első hallásra nem is tűnik nehéznek, hiszen a **FIND** parancs meg tudja számolni egy fájlban azokat a sorokat, amelyek a megadott szűrőfeltételnek megfelelnek. Csak úgy kell megfogalmaznunk a szűrőfeltételt, hogy minden sorra igaz legyen, és akkor ez a része megoldott a feladatnak.

```
C:\>find "" /v /c autoexec.bat

---------- AUTOEXEC.BAT: 35

C:\>_
```

Arra utasítottam, hogy számolja meg (**/c**) az olyan sorokat, amelyekben nem szerepel (**/v**) a semmi (""). Ilyen sor az összes, mert minden sorban van legalább egy sorvégjel, ami már nem semmi. Így ez a része megoldódott a feladatnak, de az eredményt a **FIND** parancs írta ki, és eléggé egyszerű formában.

Ha ezt az eredményt fájlba irányítjuk, aminek BAT kiterjesztést adunk, akkor a végrehajtáskor egy olyan programot keres a DOS, aminek a neve "--------" és a kiterjesztése COM, EXE, vagy BAT. (Ugyanis a "-" karakter szabályos fájlnév karakter.) Már csak arról kell gondoskodnunk, hogy találjon is ilyen fájlt. Az eddigi ismereteink szerint COM, EXE állományt nem tudunk készíteni, BAT-ot igen. Akkor készítsünk egy olyan batch-fájlt, aminek ez lesz a neve. Ha a **FIND** parancs ezen eredményét a képernyőre íratnád ki, akkor az csak egy egyszerű kiírás lenne. Ha egy állományba irányítom át, és BAT kiterjesztést adok neki, akkor az egy elindítható batch-fájl lesz, ami egy parancsot tartalmaz a "--------" -at méghozzá felparaméterezve. A "--------" parancs első paramétere a vizsgált fájl neve (most: AUTOEXEC.BAT), a második paramétere pedig az a szám, amire nekünk szükségünk van.

Az elvégzendő feladatok:

- A **FIND** parancs eredményét fájlba irányítjuk és BAT kiterjesztést adunk neki (SOR.BAT).
- Létrehozunk egy "--------.BAT" nevű fájlt, ami kiírja szépen a választ a kérdésünkre, felhasználva a második paraméterként kapott eredményt.
- Töröljük a felesleges ideiglenes állományokat a lemezünkről.

Most pedig lássuk a batch-fájlt (SZAMOL.BAT), amiben a fájl létezését is megvizsgáljuk:

```
@echo off
if %1m==m goto segit
if not exist %1 goto segit

FIND "" /v /c %1 >SOR.BAT

ECHO @echo A megadott fájl (%%1) pontosan %%2 sorból áll. >--------.BAT

CALL SOR.BAT

rem DEL SOR.BAT
rem DEL --------.BAT
goto vege

:segit
echo Add meg a fájlt, aminek a sorait meg akarod számoltatni!
echo A fájl létező legyen!
echo A fájl elérhető legyen, vagy add meg az elérési útját!
echo.
:vege
```

Először leellenőrizzük, hogy kapott-e a programunk paramétert, és ha igen, akkor az egy létező, elérhető fájlnév-e, vagy sem. Hiba esetén kiírja a batch-fájl végén lévő kis segítséget.

A **FIND** parancs eredményét a SOR.BAT nevű fájlba írjuk. Így annak tartalma:

```
C:\DOS>type sor.bat
---------- AUTOEXEC.BAT: 35
C:\DOS>_
```

Az **echo** utasítás segítségével írunk egy másik batch-fájlt, aminek a tartalma:

```
C:\DOS>type --------.bat
@echo A megadott fájl (%1) pontosan %2 sorból áll.
C:\DOS>_
```

A batch-fájl csak egysoros, és a ha megfigyeljük, "megfogyatkoztak" a százalékjelek. Ennek az a magyarázata, hogy a batch-fájlba azt a szöveget kell beírni, ami szerepel benne. Ha csak **%2**-t írtam volna, akkor azt az aktuális batch-fájl második paraméterének értelmezi, és annak az aktuális értékét adta volna át, ami példánkban nincs, így értéke sem lett volna. Nekünk, viszont arra van szükségünk, hogy a "**%2**" karaktersor kerüljön át a másik batch-fájlba. Ezt úgy tudjuk megoldani, hogy a "%" jelet megduplázzuk "%%", és akkor nem változónévnek, hanem ténylegesen "%" jelnek fogja értelmezni.

Utána meghívjuk a SOR.BAT batch-fájlt, amit pár sorral feljebb írtunk meg. Na most figyelj, mert megkeverlek! A SOR.BAT tartalmaz egy olyan utasítást, amivel a "--------.BAT" fájlt indítjuk el. Nem hívjuk meg **CALL**-lal, ezért ide már nem is fog többet visszatérni, de nem is kell. A parancs hatására kiírja, hogy:

```
A megadott fájl (AUTOEXEC.BAT) pontosan 35 sorból áll.
```

Ezzel a batch-fájl (--------.BAT) be is fejeződött, és mert a SOR.BAT nem **CALL**-lal hívta meg, ezért oda nem is tér vissza, de előtte volt egy **CALL** utasítás, aminek hatására a DOS megjegyezte, hogy oda majd egyszer vissz kell térni. Így amikor a --------.BAT fájl befejeződött, megkérdezi a DOS-t hová térjen vissza. Az pedig azt válaszolja neki, hogy az eredeti (SZAMOL.BAT) batch-fájlba térjen vissza. Utána van két "**rem**"-es sor, amiből a "**rem**" szavakat kivéve a SZAMOL.BAT futása után a két ideiglenesen szükséges batch-fájlt letörli, majd ugrik a "**vege**" címkére, és vége a batch-fájlunknak. Ilyenkor ez a batch-fájl is megkérdezi a DOS-t, hogy hová térjen vissza, erre azt a választ kapja, hogy a COMMAND.COM-hoz, mert ezt a batch-fájlt onann indítottuk a készenléti jel mellől.

```
C:\>szamol AUTOEXEC.BAT
A megadott fájl (AUTOEXEC.BAT) pontosan 35 sorból áll.
C:\>_
```

Ha egy kicsit bonyolult is volt, azért nézd át a feladatot, és a megoldás gondolatmenetét. Az a legfontosabb, hogy azt megértsd, hogy a kiírt szöveg csak addig szöveg a képernyőn, ameddig azt fájlba nem irányítod, mert utána akár parancs is lehet, csak gondoskodni kel róla, hogy az legyen.

Sok esetben hasznos lehet az ilyen jellegű batch-fájlok alkalmazása olyan helyen, ahol a számítógéphez kevésbé értő dolgozóknak kell működőképes rendszert a keze alá tenni, hogy az illető a munkáját könnyen és gyorsan el tudja végezni. Az ilyen személyeknek nem mond semmit például a **FIND** parancs eredménye, hiszen lehet, hogy még annak a fájlnak a nevét sem tudják, amivel dolgoznak. Csak annyit értenek a számítógéphez, hogy beírják például, hogy "**ment**" vagy "**betolt 940501**" *...stb.* Ezek persze batch-fájlok nevei, és a megadott - például - dátum a batch-fájl paramétere.

4. Magyar ékezetes betűk használata

4.1. Ékezetes karakterek használatának köre

Nehéz helyzetben van, aki a magyarul szeretne levelezni, szövegállományt készíteni helyes magyar ékezetekkel, vagy csupán olyan egyszerű igényei vannak, hogy az írógépét is tudja használni, és a számítógépét is, persze ha lehet, akkor egyforma billentyűzetkiosztással, mert kétféle billentyűzeten gyorsan gépelni nem lehet. Ha mindkét billentyűzet karakterkiosztása különböző, akkor az nagyon zavaró.

Ha valaki írógépen dolgozik évek óta, és a cégénél bevezetik a számítógépes szövegszerkesztőt, akkor az megváltás legyen a titkárnő számára, és ne egy sorscsapás.

E szempontok alapján két fő csoportba sorolhatjuk a szövegszerkesztőket: Az amerikai billentyűzetet megváltoztatva használó, idegennyelven kommunikáló programok, melyek a számítógép kódlapjait használják, és a kifejezetten magyarnyelvű, magyar fejlesztésű szoftverek, amelyek a magyar ékezetes betűket is szabályosan jelenítik meg mindenféle monitoron. Az utóbbinak szemmel láthatólag jobb esélyei lennének a magyar piacon, de ha a szolgáltatások minőségében, és mennyiségében nem tudnak lépést tartani a külföldi szoftverekkel, akkor sajnos elmarad a széleskörű elterjedése. Ha egy program szakít a DOS adta kódlapos karakteres képernyővel, akkor azt csak grafikus üzemmódban tudja megvalósítani, ami eléggé lelassítja a programot, illetve jobb hardverkiépítésű számítógépet kíván meg.

Az előbbi típusú programoknak viszont megvan az az előnye, hogy a karakteres képernyőn sokkal gyorsabban és programozásilag egyszerűbben meg lehet oldani bizonyos feladatokat, mint a grafikus képernyőn. Éppen emiatt van az, hogy a szövegszerkesztők nagy része karakteres képernyőn dolgozott régebben, míg kisebb százalékban a grafikus képernyőt használták. Ez utóbbi inkább a magyar szövegszerkesztőkre volt igaz, mert a magyar ékezetes betűk helyes ékezeteinek megjelenítése miatt is szükség volt a grafikus megjelenítésre.

Most a továbbiakban a karakteres képernyő adta kódlapos megoldásokat fogjuk átnézni, mert ezzel a módszerrel használatba vehetjük az összes idegennyelvű szövegszerkesztőt, ami a szolgáltatásainak színvonalában elnyerte tetszésünket.

A szövegszerkesztők használatánál az "érem másik oldala", hogy milyen nyelven kommunikál velünk a program. Aki gyorsan meg tudja tanulni az angolnyelvű üzenetek, menüpontok jelentéseit, azt nem tartja vissza, hogy egy idegennyelvű programot használ. Aki a számítógépnek csak a magyar nyelvű üzeneteit érti meg, az kénytelen beérni a magyar szoftverekkel, amelyek ugyan jók, de esetleg elmaradnak a külföldi új technikákat megvalósító programoktól.

A karakteres képernyőt és kódlapokat használó szövegszerkesztők egyik hátránya - magyar nyelvi szempontból -, hogy a magyar ékezetes betűk hiányosan vannak meg az alapértelmezett 437-es amerikai kódlapon, ezért egyes betűk (nagy **Ő**, **Ű**) hiányoznak, és egyesek pedig csak kis szépséghibával érhetők el (kis **ő-ô**, **ű-u**, **Á-Å**, **Ó-ò**).

Az első feladat tehát a kódlap átváltása 852-esre, ami tartalmazza az összes szabályos magyar ékezetes betűt. Ez a feladat nem is olyan egyszerű. Elsősorban eddig azért volt nehéz, mert a leírásokból elég nehéz volt kibogozni, hogyan is kell valójában létrehozni a kódlapokat, másrészt a kódlapok definiálásának az egyik feltétele legalább EGA monitor, vagy ettől jobb, ami régebben a drága hardverek időszakában csak keveseknek adatott meg. Ma már elterjedtebbek az ennél jobb monitorok is, de már más hardverek árai is jelentősen csökkentek, ami megváltoztatta a szövegszerkesztők fajtáit is.

A kódlapok definiálásának több lépése van, hogy azt kielégítően használni tudjuk. Először is a monitort kell előkészíteni a kódlap fogadására (ha az legalább EGA):

```
DEVICE=c:\dos\DISPLAY.SYS CON:=(EGA,852,2)
```

Ennek a CONFIG.SYS fájlbeli utasításnak a megléte az egyik alapfeltétele a kódlapok használatának. Megadhatjuk az országspecifikus információknál is a 852-es kódlap használatát, de az magában nem elegendő:

```
COUNTRY=036,852
```

A hamarosan ismertetésre kerülő **MODE** parancs segítségével tudjuk majd elkészíteni (Prepare), és kiválasztani (Select) az elkészített kódlapot. Ezzel megoldottuk a kódlap, képernyőn való megjelenítésének problémáját, már csak az van hátra, hogy a billentyűzetről elő tudjuk hozni valamilyen egyszerű módszerrel.

4.2. Ékezetes betűk megjelenítése billentyűzetről

Erre több megoldás is kínálkozik egyszerre. Az egyiket ismerjük, az ANSI.SYS escape-szekvenciáival kell megadni, hogy melyik billentyűre, melyik karakter jelenjen meg. A másik az egyszerűbb, ha szabványos magyar billentyűzetkiosztást szeretnénk, akkor a **KEYB HU,852** parancsot adjuk ki, és már gépelhetünk is magyarul. A teljesen jó megoldáshoz már csak az kell, hogy a billentyűzeten az a betűkép legyen ráírva a billentyűkre, ami meg is jelenik a leütésekor. Ha amerikai billentyűzetet használunk, akkor azt tudjuk tenni, hogy alkoholos filctollal ráírjuk a billentyűkre, hogy milyen karakter fog megjelenni a hatására. Az eredetileg rajta lévőt pedig "nem vesszük észre", ha magyar szöveget gépelünk. Az egyszerűbb megoldás persze az, hogy vásárolunk a boltban szabványos magyar billentyűzetet, de erre régebben nem volt lehetőség.

Éppen ezért terjedtek el régebben az olyan programok, amelyek rezidens programként beépülve a memóriába, átdefiniálták a billentyűzetet úgy, hogy a magyar ékezetes betűket is elő tudjuk hozni, és az amerikai billentyűzeten lévő karaktereket is. Ezt úgy oldották meg, hogy például az ékezetes betű előhozatalához, még le kellett nyomni egy-két segédbillentyűt is.

MODE — Rendszerjellemzők módosítása

MODE=Típus, üzemmód.

A különféle egységek (CON, LPT, COM) használatos üzemmódjait állítja be. Kötött paraméterekkel dolgozik, és kötött értékekkel, ezért a "HELP"-je is hosszabb:

```
C:\DOS>MODE /?

Configures system devices.

Printer port:       MODE LPTn[ :] [ COLS=c] [ LINES=l] [ RETRY=r]
Serial port:        MODE COMm[ :] [ BAUD=b] [ PARITY=p] [ DATA=d] [ STOP=s] [ RETRY=r]
Device Status:      MODE [ device] [ /STATUS]
Redirect printing:  MODE LPTn[ :] =COMm[ :]
Prepare code page:  MODE device CP PREPARE=((yyy[ ...]) [ drive:][ path] filename)
Select code page:   MODE device CP SELECT=yyy
Refresh code page:  MODE device CP REFRESH
Code page status:   MODE device CP [ /STATUS]
Display mode:       MODE [ display-adapter][ ,n]
                    MODE CON[ :] [ COLS=c] [ LINES=n]
Typematic rate:     MODE CON[ :] [ RATE=r DELAY=d]

C:\DOS>_
```

Amint láthatjuk, elég sokrétű a parancs alkalmazása, amit a "help"-je tagoltan ismertet. A megadható értékek viszont szinte mindenhol kötöttek, így azok ismerete nélkül nem tudjuk használni a parancsot. Most nézzük át tételesen a fontosabb funkciókat, amelyekre szükségünk lehet:

Device Status **(Az eszköz állapota)**:

device: Eszköznév, mint például **CON**, **LPT**, **COM**, ...*stb*. Ha nem adjuk meg, akkor minden eszköz állapotát kiírja.

/status: állapota, de nem kötelező megadni. Ha elmarad, akkor is ugyanazt írja ki.

A paraméterek nélkül kiadott parancs kiírja az összes egység állapotát. A kiírt adatok között van sok olyan, ami egyenlőre nem érint bennünket, és a megváltoztatásukra sincs semmi szükség. Ilyen egységek az LPT, COM, a CON, mint billentyűzet. Nyomtatóra kiadott parancsokat nem minden nyomtató tudja értelmezni, és a kódlapokat sem támogatja sok nyomtató. Ami viszont látható és használható - CGA monitortól felfelé -, a monitor üzemmódjának váltása, ami látványos és használható a mindennapi munkában is.

A Commodore számítógépek után bizonyára sokaknak szokatlan volt, hogy a képernyőn nem 40, hanem 80 karakter íródik egy sorba, ami ugyan gazdaságosabb kihasználása a monitornak, de esetleg a korosodó adminisztratív dolgozóknak is az erősebb szemüvegük után kellett nyúlniuk. Ha pedig a monitor (EGA,...) üzemmódját átváltjuk 43 vagy 50 soros kijelzésűre (**MODE CON: LINES=43**), akkor már sokan szemvizsgáló eszköznek nézik a monitort. Pedig a legjobban 50 soros üzemmódban lehet használni, mert egyrészt így még gazdaságosabb a monitor kihasználtsága, másrészt könnyebb az áttérés a Windows felületre, ami még ennél is kisebb karakterekkel dolgozik.

Most nézzük meg mit ír ki paraméterek nélkül a parancs. A jobb áttekinthetőség kedvéért jobban elhatárolom a kiírt üzenetet:

```
C:\>mode

Status for device LPT1:
-----------------------
LPT1: not rerouted
Retry=NONE

Code page operation not supported on this device

Status for device LPT2:
-----------------------
LPT2: not rerouted

Status for device LPT3:
-----------------------
LPT3: not rerouted

Status for device CON:
----------------------
Columns=80
Lines=50

Active code page for device CON is 437
Prepared code pages:
  code page 437
  code page 852

Status for device COM1:
-----------------------
Retry=NONE

Status for device COM2:
-----------------------
Retry=NONE

C:\>_
```

Kiírta az összes eszköz állapotát. Ha csak egy eszközét akarjuk lekérdezni, akkor annak a nevét kell megadni paraméterként, például "**MODE CON:**".

Amint láthatjuk a lekérdezett képernyő 80 karakteres kijelzésre, és 50 sorosra van állítva. Ez a lehető legnagyobb felbontás, ami elérhető. A **columns** értéke lehet még 40 is (**MODE CON: COLS=40**), akkor fog a képernyőnk a Commodore képernyőjére hasonlítani. Ilyenkor azonban a sorok száma csak 25 (alapértelmezett) lehet. Ha 80 karakteres sorokkal dolgozunk, akkor a sorok száma a monitortípustól függően lehet 43 vagy 50 sor is, mint a példában is láthatjuk. A képernyő üzemmódjának a váltásához elengedhetetlen a CONFIG.SYS fájlban az ANSI.SYS *driver* - eszközmeghajtó program - telepítése. Ha a monitorunk csak monokróm, akkor nem tudjuk ezeket a szolgáltatásokat használni!

Amint láthatjuk a kiírásban, kódlapok is be vannak állítva, amire a magyar ékezetes betűk előhozatala érdekében nagy szükségünk van. A kiírás tanulsága szerint az aktív kódlap (Active code page) a monitoron (for CON device) a 437-es. Van két kódlap definiálva (Prepare - létrehoz), a 437-es, és a 852-es. Most lássuk mire van szükség a kódlapok használatához:

Először is a már ismertetett DISPLAY.SYS szükséges, ami a kódlap monitoron való megjelenítését támogatja. Utána a **MODE** paranccsal létre kell hozni (Prepare) az adott kódlapot egy olyan fájl alapján, ami tartalmazza a karakterek formáját. Utána pedig ki kell választani (Select) a létrehozott kódlapot, és már át is vált a képernyőnkön a megfelelő kijelzésre. Ezután már csak a megfelelő kódú karaktert kell a billentyűzetről bevinni, és meg is jelenik a kívánt ékezetes karakter.

Kódlap létrehozása:

```
MODE con cp prepare=((437,852) c:\dos\EGA.CPI)
```

{*cp* code page - kódlap; *prepare* - létrehoz; zárójelben a létrehozandó kódlapok száma, utána pedig a fájl neve és elérési útja, ami alapján létre lehet hozni.}

Nyugodtan próbáljuk csak ki, hogy más kódlap számokat adunk meg (hasraütéses alapon), nem fogja a műveletet végrehajtani, mert csak a fájlban rögzített kódlapokat támogatja.

Kódlap váltása:

```
MODE CON CP SEL=852
```

Csak olyan kódlapra válthatunk át, amit sikeresen létrehoztunk. A változás ellenőrizhető úgy, hogy egy általunk ismert ékezetes karakter kódját bevisszük [ALT]-tal, és ha a kívánt karakterkép jelenik meg, akkor sikeres a kódlapváltásunk. Utána már csak az a feladat maradt hátra, hogy könnyebb módszerrel (mint az [ALT]-os) vigyük be az ékezetes karaktereket. Ezt pedig könnyen megoldhatjuk a következő paranccsal:

```
KEYB HU,852
```

A billentyűzetvezérlő program magyar kiosztású billentyűzetre vált át, és a 852-es kódlap szerint jeleníti meg az ékezetes karaktereket. Ha a kódlap nincs előkészítve, akkor kiírja, hogy nem támogatja (not supported) a megadott kódlapot.

Ilyenkor ne ijedjen meg senki, aki az amerikai billentyűzethez szokott, mert az [Y] és a [Z] fel lesz cserélve, és a jobb oldali egyéb karakterek helyett fognak megjelenni a magyar ékezetes betűk. A **KEYB US,437** paranccsal vissza tudjuk állítani a billentyűzetet az eredeti amerikaira, de a beírásnál akkor a **K E Z B U S ,4 3 7** betűket írd be, mert az [Y] és a [Z] fel lesz cserélve!

A másik eset, amikor amerikai billentyűzetet akarunk használni magyar ékezetes karakterek bevitelére. Ilyenkor az esetek többségében külön rezidens programot használunk erre a célra, mert a **KEYB** parancs annyira "összekeveri" az amerikai billentyűzet karaktereit, hogy a jeleket nem találjuk meg rajta. Általános az a gyakorlat, hogy a használt segédprogram úgy dolgozik, hogy megmaradnak a billentyűzeten feltüntetett karakterek, és az [ALT]+[billentyű]-re jön elő a kívánt ékezetes karakter. Próbálkozhatunk az escape-szekvenciákkal is, amiket a **PROMPT** paranccsal ki tudunk adni, akár batch-fájlból is, de ilyenkor az **ECHO OFF**-ot hagyjuk ki, mert nem fog működni.

Most lássunk egy-két próbálkozást és választ a kódlapokkal kapcsolatban:

```
C:\>mode con cp prep=((852) c:\dos\ega.cpi)

MODE prepare code page function completed

C:\>mode con cp sel=852

MODE select code page function completed

C:\>mode con cp prep=((437) c:\dos\ega.cpi)

MODE prepare code page function completed

C:\>mode con

Status for device CON:
----------------------
Columns=80
Lines=50

Active code page for device CON is 437
Prepared code pages:
   code page 437
   code page not prepared

MODE status code page function completed

C:\>_
```

Először szabályosan létrehoztuk a 852-es kódlapot az EGA.CPI fájl alapján, amit a DOS könyvtáradban megtalálsz te is. Ennek a fájlnak a segítségével létre tudjuk hozni az összes hivatalos kódlapot. Utána átváltottunk a létrehozott kódlapra. A kiírt üzenetek arról szólnak, hogy sikeres volt mindegyik művelet. Utána létrehozunk egy másik kódlapot, ami nem másodikként jön létre, mint várnánk, hanem az előző definíciót írja felül. Ezt ellenőrizzük le a **MODE CON** paranccsal.

```
C:\>mode con cp prep=((437,852) c:\dos\ega.cpi)

MODE prepare code page function completed

C:\>mode con cp sel=852

MODE select code page function completed

C:\>mode con

Status for device CON:
----------------------
Columns=80
Lines=50

Active code page for device CON is 852
Prepared code pages:
   code page 437
   code page 852

MODE status code page function completed


C:\>_
```

Most létrehoztunk két kódlapot, ha valami oknál fogva váltogatnunk kellene köztük, rendelkezésre álljon mindegyik. A **MODE CON CP SEL=xxx** paranccsal tudunk váltani a két létrehozott kódlap között.

```
C:\>mode con cp prep=((,850) c:\dos\ega.cpi)

MODE prepare code page function completed

C:\>mode con

Status for device CON:
----------------------
Columns=80
Lines=50

No code page has been selected
Prepared code pages:
   code page 437
   code page 850

MODE status code page function completed

C:\>_
```

Ha csak az egyik kódlapot akarjuk átdefiniálni, akkor a fenti módszert alkalmazhatjuk. Most nézzük meg, hogyan definiálhatjuk át a billentyűzetet, az igényeink szerint:

```
C:\>keyb hu,852
Code page specified has not been prepared

C:\>mode con cp prep=((437,852) c:\dos\ega.cpi)

MODE prepare code page function completed

C:\>keyb hu,852
One or more CON code pages invalid for given keyboard code

C:\>mode con cp prep=((852) c:\dos\ega.cpi)
MODE prepare code page function completed

C:\>keyb hu,852

C:\>_
```

Jóhiszeműen kiadjuk a **KEYB** parancsot, és akkor döbbenünk rá, hogy nem készítettük még el (not been prepared) a szükséges kódlapot. Gyorsan definiálunk kettőt, jó ha van. Ismételten megpróbáljuk kiadni a **KEYB** parancsot, mire azt vágja a fejünkhöz, hogy nem támogatja az általunk megadott kódlapot (437), de ezt tudnunk is kellett volna, hiszen a **KEYB** parancs ismertetésénél van egy táblázat, amiben szerepel, hogy csak a 850-es, és 852-es kódlapokkal hajlandó a magyar billentyűzetet definiálni. Változtassuk meg akkor az elsőnek definiált kódlapot 852-esre, és már kiadhatjuk eredményesen a **KEYB** parancsot. Nézzük meg az [Y] a [Z] billentyűket először, és látni fogjuk sikerült-e a változtatás. Ha most sem sikerült volna, és a számítógép sem szólt róla, akkor a CONFIG.SYS fájlban van egy **COUNTRY=036,nnn** utasítás, aminél az **nnn** a támogatandó kódlap száma, ami biztosan nem 852-es. Ha igen, akkor lehet hogy szintén a CONFIG.SYS fájlban a DISPLAY.SYS eszközmeghajtó programnál adtunk meg a 852-estől

eltérő számot. Nagyon oda kell figyelni ezekre, mert ha valahol nem egyezik a kódlapszám, akkor nem azt fogja megmondani a számítógép, hogy hol a hiba, hanem azt, hogy rossz, mi meg keressük csak szorgalmasan a hiba okát. Rosszabb esetben nem is ír ki semmit, és mégsem működik az amit szeretnénk.

ÖSSZEFOGLALÁS:

Mi legyen a CONFIG.SYS fájlban a magyar ékezetes betűk használatához:

```
COUNTRY=036,852, C:\DOS\COUNTRY.SYS
DEVICEHIGH=c:\dos\ANSI.SYS
DEVICEHIGH=c:\dos\DISPLAY.SYS CON=(EGA,,2)
```

Mi legyen az AUTOEXEC.BAT fájlban a magyar ékezetes betűk használatához:

```
c:\dos\MODE con cp prep = ((852,437) c:\dos\EGA.CPI)
c:\dos\MODE con cp sel = 852
c:\dos\KEYB HU,852
```

4.3. Ékezetes betűk a nyomtatásban

Ez szintén egy tabu téma volt hosszú ideig, mert a nyomtatókat igen nehéz volt megtanítani magyarul nyomtatni. Egyes nyomtatóknál lehetőség van a karakterek képének megváltoztatására (például: **EPSON FX-1050**), de újabban már külön bővítőkártyán adják a magyar betűkészletet (a 852-es kódlapot) a nyomtatóhoz (például **HP DESKJET 510**).

DOS-ból továbbra is két módja van a nyomtatásnak, az egyik a karakteres nyomtatás, amikor egy ASCII kódot küldünk a nyomtatónak, aminek a képe a monitorunkon lehet hogy 852-esbeli szabályos magyar ékezetes betű. Viszont, ha a nyomtató csak 437-es "amerikaiul" tud nyomtatni, akkor ahelyett egészen más fog megjelenni a nyomtatón. A másik megoldás a grafikus nyomtatás, amikor pontonként definiáljuk - kiszámítjuk - a leendő karakterek képét a nyomtatóra, ami rendkívül időigényes, és lassú is. Így ez a megoldás csak egy-egy oldal levél esetén jöhet szóba. Többszáz oldalas tablók nyomtatása esetén, csak a karakteres nyomtatás lehetséges, ahol viszont nélkülöznünk kell egyes magyar ékezetes betűket, egyeseket pedig szépséghibával tudunk csak megjeleníteni ("sátoros" ő ű betűk esete). Általában ezt a megoldást alkalmazzuk, méghozzá úgy, hogy a kiírandó szövegeket úgy fogalmazzuk meg, hogy a rendelkezésre álló karaktereket használjuk fel a képernyőn is, hogy nyomtatásban is az jelenjen meg, amit megterveztünk a képernyőn.

Újabb, drágább nagysebességű nyomtatóknál is már beépítik külön bővítőkártyán a 852-es kódlap karaktereit, hogy nyomtatásban is elérhetők legyenek a szabályos magyar ékezetes karakterek.

5. Egyéb DOS-ban végezhető feladatok

Az eddig megismert parancsokra, konfigurációs fájlokra azért volt szükség, hogy az általunk használni kívánt felhasználói program, minél jobban tudjon futni a számítógépen, minél jobban ki tudja használni annak lehetőségeit. A legideálisabb működése az lenne a számítógépnek, ha teljes egészében a memóriában dolgozhatna, és nem kellene használnia a "lassú" meghajtókat (merevlemezt, hajlékonylemezt,...). Ehhez nyújtanak sok segítséget a memórialemezek (RAMDRIVE.SYS), gyorsítótárak (SMARTDRV.SYS vagy SMARTDRV.EXE) és a felső memóriaterületek kihasználását biztosító programok (HIMEM.SYS, EMM386.EXE, **DEVICEHIGH** utasítás.). Ha ezeket megfelelően felparaméterezve elindítjuk a számítógépünk indulásakor, akkor biztosítani fogják számunkra a megfelelő működést. Ezért van szükség az ismertetett programok használatára. A nem ismertetett parancsokkal is megismerkedhetünk, és hasznos szolgáltatásaikat kihasználhatjuk, ha szükségünk van rá. Azok ismertetésére már nem térek ki a tankönyvemben, mert ennek a tankönyvnek az alapismeretek leírása, az alapvető fogások ismertetése, és a megfelelő gondolkodásmód kialakítása a fő feladata. Megfelelő gondolkodásmóddal, már önállóan is tudunk sok esetben boldogulni, akár szakkönyvek nélkül is. Igaz így sokat kell a számítógép mellett ülni és tanulni, de egyszer megtérül a programkezeléssel töltött idő, mert ***a tapasztalatot nem lehet könyvekből kiolvasni, azt meg kell szerezni!***

5.1. Harc a vírusokkal

Sokszor a legnagyobb elővigyázatosság mellett is bekerülnek vírusok a számítógépünk memóriájába, illetve a lemezeink különböző helyeire. Gátolják a munkánkat, sok esetben megsemmisítik állományainkat, nagy károkat okozva ezzel. Általánosságban elmondható, hogyha valaki csak olyan szoftverekkel dolgozik, amelyeket szoftverkereskedőktől vásárolt, akik a szoftvergyártóktól bontatlan csomagolásban kapják a programokat, akkor nem kerülhet vírus a számítógépünkbe. Abban a pillanatban, amikor "megszerzett" programokat másolunk fel a számítógépünkre, kitesszük magunkat annak a veszélynek, hogy a "megszerzett" programokkal együtt vírusokat is kapunk.

A *számítógépvírus* olyan rosszindulatú program, amelynek az a feladata, hogy a programjainkat megfertőzve, azokban kárt tegyen, megsemmisítse azokat, vagy használhatatlanná tegye. A vírusok sok esetben jelen vannak a számítógépünkben vagy a merevlemezünkön vagy a hajlékonylemezeinken úgy, hogy nem is tudunk a létezésükről. Egy adott időpontban, vagy egy adott esemény bekövetkezésekor viszont átveszik a vezérlést a számítógéptől (lásd *rezidens programok*), és akcióba lépnek. Ilyenkor teljesen átlagos megszokott tevékenységet végeznek, amit mi is vagy a számítógépünk is szokott végezni nap mint nap. Leformáz egy merevlemezt, vagy törli a FAT-táblát, vagy törli az összes programot az aktuális könyvtárban,...*stb.* Csak az a különbség, hogy ezt nem a mi kérésünkre teszi, és olyan állományokon vagy lemezeken teszi, amelyekre nekünk még szükségünk lett volna. Ebben az esetben pedig nagy kárt okoznak.

Többféleképpen is védekezhetünk ellenük. Először is jogtiszta szoftverekkel kell dolgoznunk. Ha valami okból kifolyólag, mégis "kapnánk" valakitől egy programot, és

vele együtt egy vírust is, akkor több teendőnk is van. Először a legfontosabb, hogy minden idegen lemezt ellenőrizzünk le valamilyen víruskereső programmal, és csak akkor töltsük be a számítógépünkbe a programot, csak akkor másoljuk át a merevlemezünkre, ha a víruskereső programunk nem talált rajta vírust. Ez persze csak azt jelenti, hogy a víruskereső program elkészülésekor ismert vírusok nincsenek rajta. Újkeletű vírus viszont lehet rajta, és azt nem is fogja tudni kimutatni a program.

Ha a baj már megtörtént, és a vírus a merevlemezünkön van, akkor indítsuk el egy "tiszta" vírusmentes rendszerlemezről a számítógépünket (lásd 4. oldalon), és indítsuk el a vírusirtó programot, ami - ha tudja, akkor - letörli a vírusprogramot a lemezről, és helyreállít mindent az eredeti helyzetnek megfelelően. Vannak viszont olyan vírusok, amelyek jóvátehetetlen károkat képesek okozni (például törlik mindkét FAT-táblát), ekkor viszont szinte semmi esélyünk sincs az adatok helyreállítására. Speciálisan ezzel foglalkozó szakemberek segítségét kérhetjük, ha valóban volt olyan program, adatállomány a lemezünkön, ami pótolhatatlan.

Pótolhatatlan adatállomány viszont nincs, csak felelőtlen számítástechnikus! Ha egy adat valóban fontos, akkor rendszeresen végezzünk archíválást. Ha valóban fontosak az adataink, akkor a programokat ne "kéz alatt" szerezzük be, hanem vásároljuk meg, erre az árbevétel bizonyos százalékát rá kell fordítanunk. Ha semmit sem teszünk meg az adataink biztonsága érdekében, akkor azok az adatok nem is olyan fontosak a számunkra. Akkor viszont miért használjuk a számítógépet?

Víruskereső és vírusirtó programot többfélét is vásárolhatunk, vagy "szerezhetünk be". A legismertebb, és legelterjedtebb a **SCAN** és a **CLEAN** programok, amelyek folyamatosan megújulnak, és lépést tartanak a vírusokkal. A **SCAN** program megkeresi a vírust, a **CLEAN** pedig mentesíti a lemezt a vírustól. Jelenleg már több, mint 1300 vírust tart nyílván a program. A program paraméterek nélkül elindítva kiírja a paramétereit, aminek használatához nem kell különösebb ismeret. Annyit elég tudni róla, hogy egy paramétert kell neki megadni, a vizsgálandó meghajtó betűjelét.

Például:

```
C:\DOS>SCAN A:
SCAN 9.19 V108 Copyright 1989-93 by McAfee Associates. (408) 988-3832
:
:
:
Continue anyway? [Y/n] Yes
Scanning memory for critical viruses.
Scannig 640K RAM

Scanning for known viruses.

Drive A: has no volume label.
Scanning A:\COMMAND.COM

Disk A: contains 1 directories and 22 files.
 No viruses found.
:
:
:
      Copyright (c) McAfee Associates 1989-1993. All Rights Reserved
C:\DOS>_
```

Az **MS-DOS 6**-ban már a DOS-szal együtt kapja a felhasználó a víruskereső és mentesítő programot, aminek MSAV.EXE a neve (**M**icro**S**oft **A**nti **V**irus). A Windows alatt is használhatjuk ezt a programot, ha a DOS installálásakor ezt kértük. A program indulásakor kapunk egy menüt, amiből a következő menüpontokat választhatjuk ki:

Detect: Csak víruskeresés.

Detect & Clean: Víruskeresés és mentesítés.

Select new drive: Egy másik meghajtó választása (alapértelmezésben az aktuális meghajtót vizsgálja).

Options: A program működésével kapcsolatos paraméterek állíthatók be egy felbukkanó párbeszédablakban (például adjon-e hangjelzést vírustalálat esetén).

Exit: Kilépés a programból.

Ezeket a programokat szerencsére nem tudtam működés közben bemutatni, mert nem rendelkezem vírusokkal. Ezeken kívül van még több víruskereső és mentesítő program is, általában mindegyiknek hasonló a működése, szinte semmit sem kell beállítani, magától keres a program, és csak találat esetén kell esetleg válaszolni, hogy mentesítse-e a vírustól a programot, vagy se. Bizonyos esetben a vírus mentesítése a program megsemmisülésével is jár, amiből viszont lehet, hogy meg tudunk menteni adatokat, különféle Disk Editor-ok segítségével.

5.2. Archíválások

Archíválásnak nevezzük azt a feladatot, amikor a napi használatos merevlemezünkről egy másik háttértárolóra mentjük az adatainkat azért, hogyha valamilyen okból a merevlemezünk megsérül, vírusos lesz, akkor is helyre lehessen állítani az adatokat. A másik háttértároló általában olyan egység, ami olyan kapacitású, hogy az adataink a leggazdaságosabban használják ki a rendelkezésre álló helyet. Ez lehet hajlékonylemez is, ha a napi adatok nem haladják meg az 1.2 MB vagy 1.44 MB-ot. Ha jól tömöríthetőek az adataink, és rendelkezünk az **MS DOS 6**-tal, akkor a **Double Space** program segítségével nagyobb mennyiség is menthető hajlékonylemezre.

Ha az adatállományaink nagyobb méretűek, akkor más egységet kell használnunk. Ez lehet egy másik merevlemez (80- 1-2 GB), streamer kazetta (40-120 MB), floptical egység (20 MB-os 1.44"-os hajlékonylemezes meghajtó), optikai lemez (1-2 GB), ... *stb.*

Hogy milyen eszközt választunk, azt alapvetően az archiválandó adatok mennyisége dönti el, és az, hogy milyen időközönként akarunk menteni. Célszerű naponta, mert a hiba mindig akkor következik be, amikor a legnagyobb kárt okozza. Ha csak az aznapi adatok semmisülnek meg, akkor az még talán egy kis "túlórával" pótolható, de egy egész heti adatállományt újra létrehozni, ... sok munkával jár.

Az archíválás rendkívül fontos feladat a számítógépes munka során, és nagy figyelmet kell szentelni rá, mert sok embertársunk munkáját tudjuk egy rövid, gyors **DEL** paranccsal jóvátehetetlenül tönkretenni. A téma jelentőségére tekintettel ezt a fejezetet egy külön tankönyvben fogom megírni az *"Archíválások"* című tankönyvben. Ebben a

tankönyvben ismertetni fogom, hogy milyen hardvert, és milyen szoftvert használhatunk archíválásra. Mikor, milyen megoldás hasznos, és az aktuális árakat is ismertetni fogom. Ez ugyanis alapjaiban behatárolja egy-egy cégnél az archíváló eszközök széles skáláját.

Röviden csak annyit, hogy a DOS-ban a **BACKUP-RESTORE** parancsok szolgálnak a feladat ellátására. Az MS-DOS 6 után már belép az **MSBACKUP** program, ami a *Norton Backup* program Microsoft által megvásárolt, és csökkentett teljesítményű változata.

Hálózati környezetben viszont nem használhatjuk ezt a parancsot, ott van egy másik az **NBACKUP** (**N**ovell **Backup**). Archíválásra használhatjuk még a tömörítő programokat is (**ARJ**, **PKZIP**, **LHA**,...), amelyek biztosítják számunkra, hogy az adataink kevesebb helyet foglaljanak el, mint amennyi az eredeti állományméretük volt, de adatvesztés nélkül. A becsomagolt állományokat ki is lehet csomagolni, és akkor ugyanazt kapjuk vissza, mint amit becsomagoltunk. Az előnyük éppen ebben van, hogy kevesebb tárolóhelyet igényel az adatállomány csomagolva, mint anélkül.

5.3. Helykeresés a memóriában

Miután megismerkedtünk a különféle rezidens DOS programokkal, és eszközvezérlőkkel, és kellően meg is barátkoztunk velük, alkalmazzuk is azokat, akkor fogjuk észrevenni, hogy a felhasználói programunknak nem maradt elegendő hely. Ezt úgy tudjuk meg, hogy a program indulása után kiírja, hogy nincs elegendő hely a memóriában:

```
Insufficient memory
```

Ekkor visszakapjuk a készenléti jelet, és a programunk nem indul el. Ekkor bizony át kell néznünk a konfigurációs fájlokat, hogy van-e olyan rezidens program vagy eszközmeghajtó betöltve, amire valójában nincs is szükségünk. Megnézhetjük a **MEM /c** vagy a **MEM /d** paranccsal azt is, hogy ki van-e kellően használva a felső memóriaterület, vagy sem. Mint már ismertettem a felső memóriaterületekre célszerű tölteni az eszközmeghajtókat - ha lehet -, így több hely marad a hagyományos memóriában a felhasználói programjaink futtatására.

A legegyszerűbb talán mégis az lehet, ha a konfigurációs fájlokat nézzük át, hogy milyen programokat indítottunk el. Ezek közül töröljünk ki néhányat, általában olyanokat, amelyek nagy helyet foglalnak, és nincsenek kellően kihasználva.

5.4. Helykeresés a merevlemezen

Ez a számítógépnek az egyik olyan területe, ami mindig kicsi. Bármekkorára is cseréljük ki, mindig sikerül megtölteni programokkal, adatállományokkal, így időnként archíválásra kényszerül az is, aki nem szokott ilyen műveleteket végezni.

Elsősorban gondoljuk át, milyen jellegű tevékenységet szoktunk végezni. Ezekhez, milyen programokra van szükségünk, és melyek azok a programok, amelyeket ritkán vagy egyáltalán nem használunk. Az olyan programoknál, amelyeket ritkán használunk, ellenőrizzük a telepítőlemez épségét, olvashatóságát, és ha minden rendben van, akkor törölhetjük a merevlemezről a programot.

Az adatállományok azok, amelyek leginkább megtöltik a merevlemezt. Ezek folyamatosan keletkeznek, és folyamatosan megtöltik a merevlemezt. Ne várjuk meg soha a "megtelt" üzenetet, folyamatosan archíváljuk le azokat. Ez csupán azt jelenti, hogy legalább hetente egy napon egy kis időt szánjunk a merevlemezünk átvizsgálására. Ekkor gyűjtsük össze az összes frissen keletkezett és módosított fájlt, azokat másoljuk ki valamilyen más háttértárolóra, tömörítve, vagy anélkül, és már törölhetjük is a merevlemezről.

Ha valamilyen új programot kapunk kipróbálásra, akkor azt lehetőleg külön meghajtón, de mindenképpen egy új könyvtárba másoljuk be, a szükséges vírusellenőrzés után. A program kipróbálása után döntsük el annak használhatóságát, és ha valóban használható, csak akkor tartsuk meg, viszont azt is mérlegeljük, hogy melyik programunk válik így feleslegessé, elavulttá. Akkor viszont annak archíválásáról és törléséről is gondoskodnunk kell.

Talán a legcélravezetőbb megoldás az, ha céltudatosan dolgozunk, azaz mindig csak egy-két tevékenységet űzünk. Ha ráállunk egy programnyelv tanulására, akkor ne foglalkozzunk mással. Ha a programnyelvet egy bizonyos szinten elsajátítottuk, akkor már tudni fogjuk, hogy a programcsomag mely része az, ami megtartandó, és mely része (demok, leírások,...) az, amire később nem lesz szükségünk. Azokat az archíválás után le is törölhetjük. Utána egy másik programmal is megismerkedhetünk, és ha kellő - pár napos, pár hetes,... - tanulmányozás után feleslegesnek ítéljük, akkor szelektáljuk, és lépjünk tovább egy újabb témára. Minél több programot, rendszert ismer valaki, annál szélesebb lesz a látóköre, és annál könnyebben fog tudni az újabb programokra áttérni, az újabb programokkal megismerkedni. Ezért célszerű megismerkedni legalább alapfokon minden olyan programmal, amihez hozzá tudunk férni. Soha nem tudhatjuk, melyikre lesz szükségünk. Akkor viszont könnyen és gyorsan be tudjuk dolgozni magunkat az adott munkába, mindenféle megerőltetés nélkül.

Természetesen ilyen jellegű törlésekre, szelektálásokra csak akkor kényszerülünk, ha több programunk és adatállományunk van, mint lemezhelyünk. Persze, akkor sem célszerű feleslegesen programokat, adatállományokat a merevlemezünkön tartani, - melyekre soha sem lesz szükségünk -, ha bővében vagyunk a lemezhelynek.

5.5. Felhasználói programok indítása

Itt meg kell különböztetnünk a különféle felhasználókat. Van, aki csak annyit ért a számítógéphez, hogy bekapcsolás után beírja a bűvös szót - egy batch-fájl nevét -, ami elindítja a programját, és már tud dolgozni. Ha olyan helyre kell programot telepítenünk, ahol ilyen felhasználók fognak dolgozni, ott nekünk kell mindent batch-fájlokkal megoldanunk, mert az ott dolgozó személy lehet, hogy kitűnő szakember a saját szakmájában, de a számítógéphez semmit sem ért. Vannak azonban olyan felhasználók is, akik valamennyire már ismerik a számítógép működését, és abba be is tudnak avatkozni. A számukra a könyvtárváltás és az abban lévő program elindítása nem jelent nagy gondot, így akkor sem esnek kétségbe, ha az nem indul el. Ilyenkor például egy **DIR** paranccsal ellenőrzik, hogy jó helyen állnak-e, az elindítandó program ott van-e a könyvtárban, és megteszik a szükséges lépéseket az ismeretük korlátain belül. Ők már azt is fel tudják esetleg mérni, hogy milyen jellegű hibát követtek el, tudják-e korrigálni, és ha nem, akkor milyen szakembert kell hívni a hiba elhárítására.

A programok indításakor figyelembe kell vennünk azok működési jellemzőit. Van olyan program, ami csak akkor tud elindulni, ha a saját könyvtárában áll. Ekkor a megfelelő **CD** paranccsal be kell lépni a program könyvtárába, és csak utána szabad elindítani azt.

A számítógépeken gyakran találkozhatunk a **Norton Commander**-rel (NC.EXE), ami egy hasznos segédeszköz. Sok helyen ki is használják a szolgáltatásait, mellyel programokat lehet elindítani. Az **NC** nem része a DOS-nak, de szinte nincs olyan számítógép, amelyiken ne lehetne megtalálni, ezért pár szót ejtek róla.

A menüt a programban az [F2]-es billentyűvel tudjuk előhozni - ha van -, és ide tudunk felvenni menüpontokat. A menüpontokhoz rendelhetünk egy vagy több parancsot is, amivel a felhasználói programjainkat tudjuk elindítani. A menü készítésében a 3.0-ás és a 4.0-ás programverzió eltér, de a lényege ugyanaz. A menüből a kurzormozgató-billentyűk segítségével kiválaszthatjuk a szükséges menüpontot, és az [ENTER]-rel aktiváljuk azt. Ennek hatására elindul a programunk, majd ha onnan kilépünk, akkor szintén a menübe térünk vissza. A felhasználó nem is tudja mit tesz, de mégis könnyen és egyszerűen tudja használni a program eme szolgáltatását.

Gyakorlottabb **NC** felhasználók pedig a program használatával könnyen és gyorsan tudnak könyvtárat váltani, majd onnan programot elindítani. Részletesen a programot megismerheted *Bartha Attila "Norton Commander 4.0"* vagy a *"Norton for DOS"* című könyveiből, melyeket szintén az LSI Oktatóközpont jelentetett meg.

A DOS-ban hasonló menüket ***batch-fájlok*** segítségével is tudunk készíteni, ami a legegyszerűbb módja a menükészítésnek.

Batch-fájloknak azokat a **BAT** kiterjesztésű fájlokat nevezzük, amelyekben több parancs is szerepel, amelyeket egymás után kellene kiadnunk - esetleg a DOS készenléti jel mellett. Mivel a parancsok ***kötegben, csomóban*** (**BATCH**) találhatók benne, ezért nevezik így ezeket a fájlokat. Egyszerű karakteres szövegszerkesztővel - például az **EDIT**-tel elkészíthetők, de rövidebbek esetén, a készenléti jel mellett kiadott **COPY CON** ASDFG.**BAT** paranccsal is elkészíthetjük.

6. Nyomtatás DOS-ból

A tankönyv első részében már foglalkoztunk egy kicsit a nyomtatásokkal. Több formát is kipróbáltunk, hogyan lehet DOS-ból a nyomtatóra küldeni egy szövegfájlt. Ilyen volt a **PRINT** parancs, ami kifejezetten nyomtatásra készült, vagy a **TYPE** parancs, aminek a kimenetét irányítottuk át a **PRN** eszközre, vagy a **COPY** parancs, aminél a célnak a nyomtatóeszközt (**PRN, LPTx**) adtuk meg. Ezek a parancsok, "trükkök" arra szolgáltak, hogy az állományt a nyomtatóra juttassuk. A szövegünk viszont csupa egyforma betűvel jelent meg a nyomtatón, ami nem minden esetben hasznos. Itt van például ez a tankönyv, amiben több helyen is kiemelések, vastagabb betűk, aláhúzások, és egyéb megoldások szolgálják azt, hogy a szöveg áttekinthetőbb legyen, és esztétikai megjelenésében is kiemelkedjen. Most ezzel fogunk részletesebben megismerkedni, illetve azt nézzük át, hogy melyik technikához, milyen ismeretekre van szükség.

6.1. Szöveg formázása utólag

Egyik megoldás lehet az, hogy elkészítjük a szövegállományt, például a DOS **EDIT** szövegszerkesztőjével, és miután minden hibát, elütést kijavítottunk, utólagosan elhelyezünk olyan nyomtatóvezérlő kódokat, amelyek hatására betűk, szavak, mondatok más karakterrel vagy más formában fognak megjelenni. Ilyen forma lehet, például a *döntött betűk,* vagy a **vastagított betűk,** vagy az aláhúzott betűk, vagy ***ezek kombinációja***. Általában ezek a legelterjedtebbek, de vannak más kódok is, amelyek más változást hoznak a karakterek képében.

Ehhez elsősorban a használni kívánt nyomtató átfogó ismerete szükséges, vagy legalább annyi, hogy a gépkönyvében el tudjunk igazodni. Ha angol nyelvű gépkönyv áll csak a rendelkezésünkre (User's Guide), akkor keressük meg benne a **Command Summary** (parancsok összegzése) című fejezetet, vagy valami hasonlót, amiben lehetőleg a **Command** szó szerepel. Itt megtaláljuk azokat a vezérlőkódokat, amelyek kiadása után a megadott funkció lép életbe. A funkciók többsége addig él, ameddig egy másik kóddal ki nem kapcsoljuk. Vannak olyanok is, amelyek csak egy sorra érvényesek.

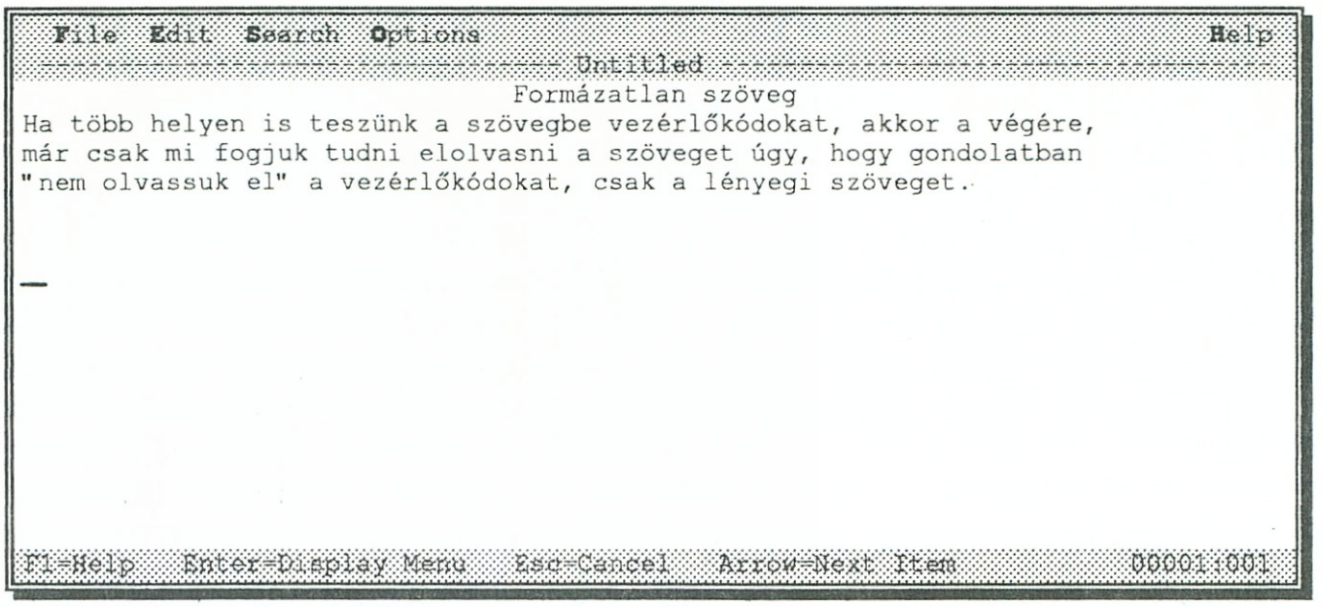
A kódok nyomtatóra juttatása sem egyszerű feladat, mert ezeket is az **ALT-27**-es ESC-kód vezeti be, aminek a billentyűzetről való előhozatala bizonyos helyeken problémákba ütközik. Ha az **EDIT** szövegszerkesztőt használjuk, akkor a [CTRL]+[P] után beüthetjük az [ALT]-tal a kívánt kódot.

A szöveget úgy tudjuk utólag megformázni, hogy a kiemelendő szó - szövegrész - elé állunk a kurzorral, és beírjuk a szükséges vezérlőkódokat. A szó - vagy szövegrész - végén pedig, szintén beírjuk azt a kódot, ami feloldja a kijelölést, hiszen akkor az egész dokumentumunk olyan betűkkel jelenne meg, amit beállítottunk. Erre szükség is lehet, de most a szavak kiemelését vizsgálgatjuk.

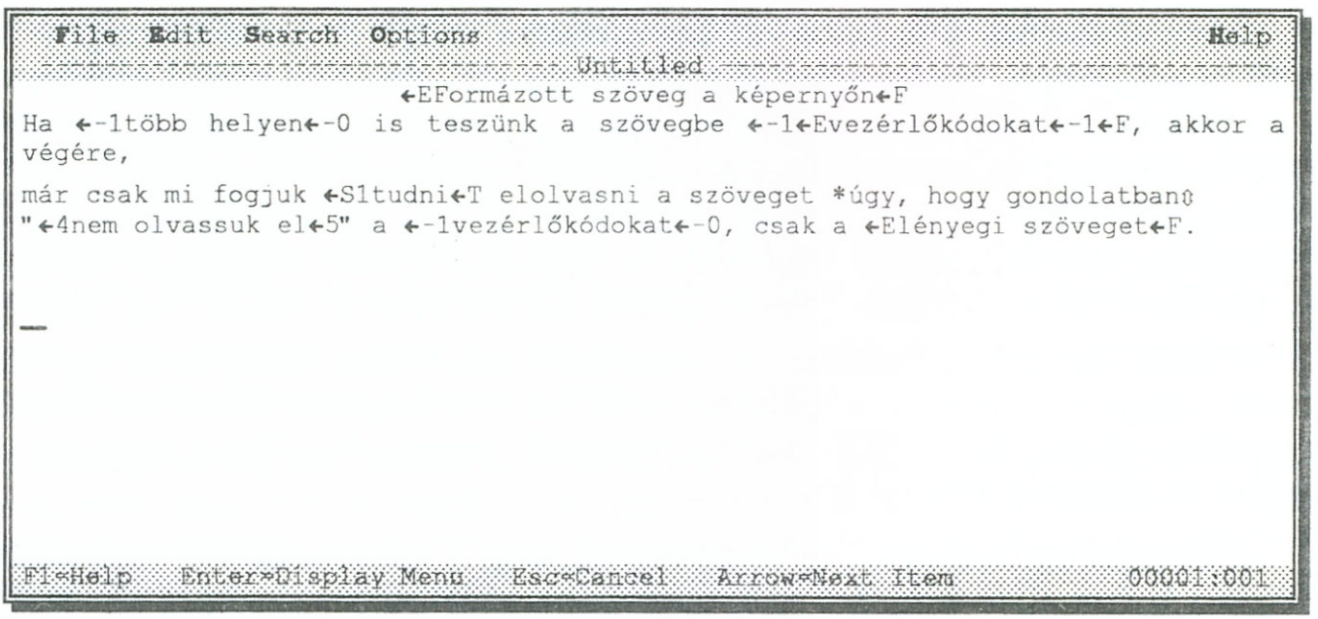
A vezérlőkód a képernyőn meg fog jelenni, és látszólag elég rendesen összerondítja a szövegünket. Ha több helyen is teszünk a szövegbe vezérlőkódokat, akkor a végére, már csak mi fogjuk tudni elolvasni a szöveget úgy, hogy gondolatban "nem olvassuk el" a vezérlőkódokat, csak a lényegi szöveget.

Lássuk, hogy is néz ez ki a valóságban:

Először begépeljük a szöveget a szövegszerkesztőben:



Utána beírjuk a megfelelő helyekre a szükséges vezérlőkódokat. Ha egy funkciót bekapcsolunk, akkor azt soha ne felejtsük el kikapcsolni, mert addig él a beállítás, amíg ki nem kapcsoljuk egy másik vezérlőkóddal, vagy magát a nyomtatót ki nem kapcsoljuk. Egyes beállítások, egyes nyomtatókon még kikapcsolás után is megmaradnak, és a következő bekapcsoláskor a megváltozott alaphelyzetben fog elindulni a nyomtatónk. (Ilyen például az EPSON FX-1050-es nyomtatónál, a **Near Letter Quality** beállítás.)



A vezérlőkódok beírása után, mint láthatjuk elég nehezen olvasható a szövegünk. A nyomtató úgy fogja feldolgozni a kapott bájtokat, hogy először eldönti vezérlőkód-e (alt-27, ...*stb.*, lásd nyomtató leírása), vagy egyéb karakter. Ha vezérlőkód, akkor az utána érkező bájtokat is megvizsgálja, hogy a vezérlőkód részei-e, és ha igen, akkor azok függvényében különféle beállításokat tesz. Ha adatbájt, akkor kinyomtatja az adott karaktert olyan betűtípussal, ami alapértelmezésben be volt állítva, illetve amit az előzőekben kiadott vezérlőkódokkal beállítottunk.

Ha összehasonlítjuk az eredeti szöveget, és a kódokkal megtűzdelt szöveget, akkor láthatjuk milyen kódokat tettem a szövegbe. Természetesen a szöveg karakterei ugyanoda fognak kerülni így is, csak más betűtípussal.

Most lássuk, hogy mi lett a formázás eredménye:

Formázott szöveg nyomtatásban

Ha *több helyen* is teszünk a szövegbe **vezérlőkódokat**, akkor a végére, már csak mi fogjuk tudni elolvasni a szöveget úgy, hogy gondolatban "*nem olvassuk el*" a vezérlőkódokat, csak a **lényegi szöveget**.

Lássunk néhány példát az alkalmazott vezérlőkódokra:

ESC 4	(27 52)	Select Italic Mode	**Kiválasztjuk a dőlt módot.**
ESC 5	(27,53)	Cancel Italic Mode	**Töröljük a dőlt módot.**
ESC E	(27,69)	Select Emphasized Mode	**Kiválasztjuk a vastagbetűs módot.**
ESC F	(27,70)	Cancel Emphasized Mode	**Töröljük a vastagbetűs módot.**

...stb. A bemutatott példák alapján már egyedül is kipróbálhatod, hogy melyik vezérlőkód, mit kapcsol be a nyomtatódon. A próbálkozások után lehetőleg iktass be a fájl végére egy RESET (ennél a nyomtatónál ←@) kódot, hogy aki utánunk használja a nyomtatót, annak nem legyenek kellemetlenségei.

Elég balkáni ez a megoldás, de DOS-ban ahhoz, hogy tetszőleges szövegszerkesztőt használjunk nincs más módszer. Vannak ugyan olyan szövegszerkesztők, amelyeknél egy listából választhatunk, milyen attribútummal kívánjuk felvértezni a kijelölt szövegrészt. A szövegben nekünk azt a részt más színnel jeleníti meg, jelezve, hogy a szövegrész előtt és után is van egy vagy több vezérlőkód, de azt nekünk nem jelzi ki a képernyőn a jobb áttekinthetőség kedvéért. Ilyen szövegszerkesztő azonban kevés van, az elterjedtebbek ilyen hasznos szolgáltatásokkal nem rendelkeznek. Azoknál marad aza megoldás, hogy magunk kézi úton tesszük be a szükséges kódokat a nyomtató gépkönyvéből kinézve.

6.2. Formázott lista készítése programból

Ebben a fejezetben az olyan típusú listákról lesz szó, amelyeknél adatokat nyomtatunk nagy mennyiségben, például számlákat, vagy főkönyvi listákat ...*stb*. Ilyen esetben nem a lista, vagy a szöveg elkészülte után fogjuk megformázni, hanem már a lista készítésekor. Ez természetesen azt jelenti, hogy amikor a programot írjuk, akkor a kiírások előtt kiküldjük a nyomtatóra a szükséges vezérlőkódokat, és csak utána jönnek az adatok. Egy soron belül többször is válthatunk betűtípust, egyebeket.

Ez természetesen feltételezi azt, hogy a listakészítő programot mi írjuk meg, vagyis bizonyos programozói ismeretet is feltételez. Bár ezt nem jelenthetjük ki ilyen egyértelműen, mert vannak olyan rendszerek, amelyeket egyszerű, adatállományok lekérdezésére alkalmas, listakészítő "programozási nyelv"-vel egészítenek ki, amelyben rövid idő alatt (10 perc - 1 óra) egy egyszerű lekérdezést tudunk megfogalmazni. Ehhez nem szükséges kifejezetten programozói ismeret, mégis készítünk valami program-félét, amivel hosszú listákat tudunk előállítani. Ilyenkor is szükséges, hogy a kiírt adatok közül egyes oszlopok más karakterekkel jelenjenek meg, mint a többi, vagy az oldalanként ismétlődő fejlécben az adatok "szépen" jelenjenek meg a nyomtatásban.

6.3. Szövegszerkesztők használata

Talán nincs még egy olyan programtípus, amiből ilyen sok készült volna a világon, mint a szövegszerkesztőkből. A legegyszerűbbektől a legbonyolultabbakig nagyon széles a skála, amiből válogathatunk. A DOS **EDIT** szövegszerkesztője a legegyszerűbbek közé tartozik, ami egy egyszerű szöveg elkészítésére alkalmas, de semmi plusz szolgáltatása nincs. Az ilyeneknek az szokott az ***előnye*** lenni, hogy rövidek, kis helyet igényelnek a memóriában és a merevlemezen egyaránt. (A DOS **EDIT**-je csupán azért olyan hosszú, mert egyben ez a **QBASIC** programozási nyelv fejlesztőkörnyezete is, csak a szövegszerkesztő funkcióban nem jelennek meg azok a menüpontok, amelyek csak a **QBASIC**-hez kellenek.) A nagyobb szövegszerkesztőknek is vannak előnyei, általában azért választjuk munkánk eszközéül, mert valami szolgáltatása megnyerte a tetszésünket.

Alapvetően két csoportba kell sorolnunk a szövegszerkesztőket. Az egyik csoportba a programozási nyelvek fejlesztőkörnyezetéül szolgálók tartoznak, melyben alapvető igény, hogy a parancsok beírását, esetleg szintaktikai ellenőrzését támogassa, és valahogyan tudjuk futtatni az elkészített programot, vagy le tudjuk fordítani a forrásnyelven íródott programot DOS-ban futtatható formátumra (EXE vagy COM).

A másik csoportba kerülnek azok a szövegszerkesztők, amelyek egy levél vagy egy díszesebb kivitelű dokumentum megírására valók. Ezeknél a programoknál szükséges a karakterattribútumok megjelenítése színekkel, hogy a képernyőn is a nyomtatón megjelenő formában mutassa meg a szöveget. Ez csak a grafikus alapokon működő szövegszerkesztőknél oldható meg, amelyek viszont speciális formátumban tárolják le a szöveget, és azt más szövegszerkesztővel nem lehet beolvasni, ha csak az nem rendelkezik olyan modullal, ami ismeri a másik formátumát. Ez általában nem szokott meglenni, mert sok szövegszerkesztő van a DOS-ban, és kevés azonosság van bennük.

Nem így a Windows rendszer, amiben az az elsődleges szempont, hogy amit a képernyőn látunk, az is fog a nyomtatón megjelenni. Ez is grafikus alapon dolgozik, ami

természetesen lassítja a működését. Bár itt már inkább a hardvert kell a szoftverhez igazítani, mert valóban egy színvonalas szolgáltatást kapunk a szövegszerkesztőktől és a többi alkalmazástól is. A Windows rendszer abban is különbözik a DOS-ban lévő programoktól, hogy egységes, szabványos, és "minden" programtípus képes adatokat adni, vagy fogadni a többi programtól, mert egységes szabványos módon zajlik az egész információ-átadás. A DOS-ban ez sajnos nem jellemző, vagyis ha egy táblázatkezelőben kiszámoltunk valamit, az eredményeket nem tudjuk átadni egy másik programnak, mert az nem tudja értelmezni. Persze hogyan is várhatnánk el egy programtól, hogy sok száz egyéb program állományformátumát képes legyen kezelni? A Windows-ban ez nem jelent problémát, mert minden program egy vagy több egységes formátumra konvertálja az adatait, és azt a néhány formátumot "minden" program képes kezelni.

Mindenképpen azt javaslom, hogyha szövegszerkesztőt választunk, akkor azt alaposan ismerjük ki, és ha nem hozza azokat az elvárásokat, amelyekre szükségünk lenne, akkor ne használjuk. Ugyanis egy szövegszerkesztőt akkor tudunk csak eredményesen használni, ha megtanuljuk a *gyorsbillentyűit* is, amivel egy funkciót azonnal ki tudunk váltani, és nem kell hozzá a szükséges menüket legördíteni, mert ez sok időt elrabol, kizökkent a tényleges munkánkból. Ha viszont a program nem felel meg az elvárásainknak, vagy más szövegszerkesztőt is használnunk kell egyéb feladataink elvégzésére, akkor keressünk inkább egy jobbat. Egyszerre ugyanis nem lehet több szövegszerkesztő gyorsbillentyűit fejben tartani. Biztosan gyakran előfordulna, hogy az egyik szövegszerkesztőben a másikban használható gyorsbillentyűt nyomnánk le, ami itt viszont esetleg kárt okoz nekünk.

7. Az aktuális MS-DOS verziók

Mint minden programnak, ennek is a neve változatlan, a fejlődését pedig a verziószámával jelzik. A program verziószáma jelzi azt, hogy a programot mikor adták ki, milyen változások történtek benne, milyen új részeket tartalmaz. A DOS rendszerek között is van többféle, MS-DOS, PC-DOS, NOVELL-DOS,...*stb* . Mind különböző cégek terméke, melyek lehetnek ***kompatibilisek*** is egymással, de ez nem minden esetben egyértelmű. A DOS-ban futtatott programok is lekérdezik a DOS típusát, és verziószámát, mert a DOS tartalmaz olyan rendszerhívásokat, melyeket a program felhasznál. Ha más verziójú DOS-ban dolgozunk, mint amit a program igényel, akkor a programunk el sem indul, ha előre leellenőrzi a DOS verzióját. Ellenkező esetben viszont, előfordulhat az is, hogy a program nem végez ilyen jellegű ellenőrzéseket, és akkor egy rendszerhívás esetén nem kap eredményt, minek következtében "lefagyhat" a programunk. Rosszabb esetben az adott számú függvényhívásra rossz eredményt kap a program, amit esetleg nem vesz észre, és azzal dolgozik tovább, hibás működést mutatva ezzel. Ebből például mi annyit láthatunk, hogy a program nem azt csinálja, mint amire mi utasítjuk, egészen más programrészek indulnak el, teljesen váratlanul.

Mint említettem az MS-DOS verziói folyamatosan változnak, melyek közül az elkövetkező három a jelenleg (1994. április) legelterjedtebb. Azért egyszerre több, mert a Microsoft kibocsájtotta ugyan az MS-DOS 6-ot, és az MS-DOS 6.2-t is, de sokan megszokva az MS-DOS 5.00-át, nem térnek át azonnal az újabb verziókra. Ha úgy gondolja a felhasználó, hogy az újabb verzióban nincs semmi olyan új szolgáltatás, amit fel tudna használni, akkor nem kell feltétlen áttérnie arra.

Olyan esetben lehet még megmaradni a régebbi verzió mellett, ha olyan saját fejlesztésű szoftvert használunk, ami lekérdezi a DOS verziószámát, és magasabb verziójú DOS-ban sem hajlandó működni. Ilyenkor kérni kell a szoftverfejlesztő céget, vagy munkatársat, hogy dolgozza át a programját úgy, hogy a magasabb verziószámú DOS-ban is fusson.

A újabban vásárolt szoftverek is ellenőrzik a DOS verzióját, és előfordulhat olyan, hogy nem tudjuk telepíteni csak akkor, ha például legalább 3.30-as verziójú DOS-szal rendelkezünk.

Ilyen esetekben kell meggondolni, hogy melyik megoldás a gazdaságosabb, az összes programjainkat átdolgoztatni, az újabb verziójú DOS miatt, vagy maradni a régi mellett, vállalva azt, hogy kiadhatnak olyan programokat, amire nekünk szükségünk lenne, de az elavult DOS verziónk miatt nem tudjuk telepíteni a számítógépünkön.

Két számítógép vagy program kompatibilis egymással, ha azok adatokat képesek átadni egymásnak, ha képesek "egy nyelven beszélni". Két DOS akkor kompatibilis egymással, ha az egyikben fejlesztett programunk, fut a másik típusú DOS alatt is.

Két számítógép akkor kompatibilis egymással, ha az egyiken létrehozott programot kimentve - például egy hajlékonylemezre -, azt a másikon be tudjuk tölteni, és tudjuk futtatni egy ugyanolyan operációs rendszeren belül. Ha nem úgy vagy sehogy sem működik a programunk, akkor a két számítógép nem kompatibilis egymással.

7.1. MS-DOS 5.00

Az MS-DOS 3.30-as verziója után, volt több próbálkozása is a Microsoft cégnek, de ez egy olyan új alapokon nyugvó verzió volt, amiben sok program született, ezért sokáig tartotta magát a felhasználók körében. A következő jelentős változásokat hozó verzió az MS-DOS 5.00 verzió volt, ezért ez is egy olyan verzió, amit nehéz lesz "kiütni a nyeregből". (A továbbiakban az 5.00-ás DOS-t a 3.30-ashoz fogom hasonlítani, mert sokak számára az volt az előző verziójú DOS, amit használtak.)

Megjelent minden parancsnál a *help* funkció, ami azt jelenti, hogy minden DOS parancs után paraméternek a /?-et adva, rövid informatív jellegű segítséget ad, az adott parancs használatáról. Annak, aki ismeri a parancs használatát, ez sok segítséget ad, ebből következik, hogy ez a segítség azoknak szól, akik már megismerték például szakkönyvekből a parancs funkcióit. Csupán ebből a segítségből senki ne akarja megismerni egy parancs funkcióját, mert ez arra szolgál, hogyha nem tudjuk pontosan fejből egy parancs paramétereit, akkor innen tudunk "puskázni". Nem is kell minden parancsnál megtanulni az összes paramétert, mert ha az adott parancsot sokáig nem használjuk, akkor el is felejtjük annak kapcsolóit, paramétereit, ami természetes is. Ebből következik, hogy olyan ismereteket, amelyeket nem használunk mindennap, azokat nem kell megtanulni, csupán elegendő annyi, hogy tudjuk, hol találunk segítséget a parancs használatához.

Újdonság a **DIR** parancs kibővült szolgáltatása, amivel nemcsak egyszerűen listáztathatjuk az állományokat, hanem rendezhetjük különféle szempontok szerint, illetve a listát is különféle szempontok alapján kérhetjük le.

A törölt fájlok, a leformázott lemezek okozta károkat csökkenteni lehet az **UNDELETE** és **UNFORMAT** parancsokkal, melyek nevükben is utalnak arra, hogy a kiadott romboló parancs meg nem történtté tehető, azaz visszaállíthatunk mindent az eredeti állapotba, ha bizonyos alapszabályokat betartunk.

A BASIC programozási nyelv egy új kezelői felületet kapott, mely jobban hasonlított a többi programozási nyelv fejlesztőkörnyezetére. Ugyanezt a programot használhatjuk csökkentett képességű formájában szövegszerkesztésre is, amit az **EDIT** parancs indít el. Ez egy egyszerű szöveg elkészítéséhez elegendő funkciókat tartalmazó szövegszerkesztő. Megmaradt még az **EDLIN** is azoknak, akik folyamatosan szeretnének áttérni az **EDIT**-re, vagy a batch-fájljaikban használták felparaméterezve, valamilyen célból.

A parancssorban történő parancsszerkesztésre kapott a felhasználó egy új parancssorszerkesztő programot (**DOSKEY**), amivel már nem csak az utoljára kiadott parancs szerkeszthető, hanem sokkal több. Aki viszont nem a parancssort akarja a fájlműveletekre használni, annak rendelkezésére áll egy menüvezérelt program, ami gondolatvilágában az **NC**-re hasonlít, de kivitelezésében nem. Talán az **NC** (Norton Commander fájlkezelő segédprogram - nem a DOS része, hanem különálló program) nagyon jó tulajdonságai miatt vált népszerűvé, és hiába került a DOS-ba a **DOSSHELL** program - hasonló funkciók ellátására -, akkor is kevesen használják.

Mivel megjelentek a számítógépek piacán a 386-os számítógépek, nagyobb kiépítésű memóriával, igényként jelentkezett a DOS-ban is, a minél jobb memória-kihasználás. A **HIMEM.SYS, LOADFIX, EMM386, DEVICEHIGH,... DOS=..., LH**, programok, utasítások megjelenésével, lehetőség nyílt a 640 kB feletti memóriaterületek kihasználására. A DOS 3.xx idején ez még nem jelentkezett élesen, mert nem voltak nagyon elterjedve a 640 kB-nál sokkal magasabb memóriakiépítésű számítógépek.

7.2. MS-DOS 6

Igazán nagy változásokat nem hozott az új verzió megjelenése. Tudomásul kell vennünk, hogy a DOS feladata az alapvető programkezelés, fájlkezelés biztosítása, amiben alapvető hiányosságok mutatkoznak még ma is. Ezzel szemben viszont a Microsoft megvásárolt néhány, a **Norton Utility** részét képező programot, melyeket csökkentett teljesítményű változatban beépített az MS-DOS 6-ba. Bekerült egy víruskereső és vírusírtó program is, ami köztudottan csak akkor ér valamit (*lásd a McAfee cég* **SCAN-CLEAN** *programjai*), ha folyamatosan megújul, lépést tartva az egyre szaporodó vírusok számával. Beépítésre került még a Stacker cégtől megvásárolt lemeztömörítő program, ami a **Double Space** nevet kapta. Ez egy valóban jó döntés volt, hogy a DOS-ba került egy olyan program, ami más programokkal kompatibilisen működve, tömöríti le az adatokat olvasás közben. Ezeket a programokat szoktuk *röptömörítők*nek nevezni.

Pedig a röptömörítő még archíválási feladatokra is jól alkalmazható kisebb adatállományoknál, mert például egy dBase-fájl (*.DBF) olyan jól tömöríthető, hogy akár egy 20-22 MB-os állomány is elfér egy 1.44-es hajlékonylemezen. Tömörített formában kerül a lemezre, de ezt a felhasználó nem érzékeli, és ezáltal úgy helyezkedik el az állomány a lemezen, hogy azonnal megtekinthető, nem kell külön kicsomagolni, mint egy ARJ vagy ZIP-fájlt. A cél azonos (~20MB-os fájl egy 1.44-es lemezre kerüljön), a megoldás hasonló (sikerült tömörítve rátenni), de a végeredmény különbözik, mert a röptömörítő használatával úgy látjuk a fájlt, mintha be sem lenne csomagolva, azonnal használható, feldolgozható.

Jelentős, DOS-ba illő változás a memóriakezelés megújulása, amire az 5.00-ás DOS-ban elterjedten használták a **QEMM** memóriamenedzsert, ami jól kihasználta, felosztotta a memóriát a programok között, több helyet biztosítva a hagyományos memóriában a felhasználói programoknak. Ennek helyébe lépett a **MEMMAKER**, ami már a DOS-nak része, így mindenki által elérhető.

Újdonság volt még a **LAPTOP**-féle számítógépek támogatására létrejött kapcsolati forma, azaz két számítógép összekötése révén, az egyik gép (SERVER - központi gép) le tudta olvasni a másik számítógép (LINK - kapcsolódó gép) lemezegységeit. Erre szolgálnak az **INTERSRV** és **INTERLNK** programok. A kapcsolatot a kereskedelemben kapható soros (COMx) vagy párhozamos (LPTx) portokra dugható összekötő vezetékekkel lehet létrehozni. Ez még nem hálózat, de arra alkalmas, hogy például az asztali számítógépbe be tudjuk tölteni a **LAPTOP** vagy **NOTEBOOK** számítógépen lévő adatokat, vagy fordítva.

Szintén hasznos szolgáltatása a 6-os DOS-nak, hogy a többféle CONFIG.SYS fájl indítása közül választhatunk, ami olyankor hasznos, ha egyes rezidens programokat, eszközmeghajtókat csak kivételes esetben kell elindítanunk. Ilyenkor nem a szokásos konfigurációs fájlokat indítjuk el, hanem egy másikat, amit eddig csak úgy tehettünk meg, ha átneveztük a konfigurációs fájlokat és úgy indítottuk újra a számítógépünket. A másik hasznos megoldás az, ha erre a célra előkészített rendszerlemez volt a tarsolyunkban, és arról indítottuk el a számítógépünket, csak ez egy kicsit lassabb megoldás. Ennek a lehetőségnek a felhasználásával, egy előre definiált menüből kiválaszthatjuk a kívánt konfigurációt. Ha nem nyúlunk egy ideig a billentyűzethez, akkor az egyik tétel automatikusan kiválasztódik, és úgy indul a számítógépünk, mintha nem is lett volna az a kis választási lehetőség.

Van még néhány apró változás, illetve egyes programok kibővültek -mint például a **HELP** is -, de ezek voltak a főbb változások az 5.00-ás DOS-hoz képest.

7.3. MS-DOS 6.2

További segédprogramokkal gazdagodott az új verzió. Bekerült a **ScanDisk**, ami a lemezhibákat keresi meg, és javítja ki tömörítetlen és tömörített meghajtókon egyaránt.

A MS-DOS kiterjesztett memória menedzsere is újabb kapcsolókkal gazdagodott. Az egyik szolgáltatása, hogy a telepítésekor leteszteli a memóriát, amit ki is kapcsolhatunk a **/TESTMEM:OFF** paraméter használatával.

A fájlmozgató parancsok kiegészültek egy régóta hiányzó funkcióval. Ha a *célon* már létezik a másolandó, mozgatandó fájl, akkor jóváhagyást kér a felülírás előtt. Ez a **COPY**, **XCOPY**, és a **MOVE** parancsokra vonatkozik.

Próbáld ki! A **DIR** paranccsal keress egy rövid fájlt az aktuális könyvtárban, és adj ki rá másolási parancsot:

```
C:\DOS>COPY README.TXT COMMAND.COM
Overwrite COMMAND.COM (Yes/No/All)?n
        0 file(s) copied

C:\DOS>_
```

A példa működési feltétele, hogy mindkét fájl létezzen a megadott könyvtárban.

A CD-ROM meghajtókat is támogatja már olvasásra a **Smart Drive** program. A feltétele csupán az, hogy az **MSCDEX** program a SMARTDRV.EXE előtt kerüljön be a memóriába.

Ami viszont a **Double Space** program használóinak nagy segítség, hogy a Windows futása alatt is be lehet már tölteni a tömörített meghajtókat, még a tömörített hajlékonylemezes meghajtókat is. Ez a **DBLSPACE /AUTOMOUNT** parancs kiadásával, vagy a **DBLSPACE** program **Tools** menüjének **Options** menüpontja alatt beállítható.

Tesztelhetjük a konfigurációs fájljainkat is, vagy teljesen át is léphetjük, ha a rendszer indulásakor megnyomjuk az [F5] vagy az [F8]-as billentyűk egyikét. Az [F5] hatására teljesen kihagyja a konfigurációs fájlok végrehajtását, míg az [F8]-as billentyűre lépésenként meg fogja kérdezni, hogy végrehajtsa-e a soron következő parancsot. Ha a CONFIG.SYS fájlban a **SHELL** parancsnál a COMMAND.COM paraméterének megadjuk a "**/Y**"-t, akkor mindig soronként fogja végrehajtani jóváhagyás után az AUTOEXEC.BAT fájlunkat.

Nem kell már egyéb segédprogramokhoz fordulnia annak, aki kényelmesen le akar másolni egy hajlékonylemezt. Mint tudjuk, egy 1.2-es vagy 1.44-es lemezek másolása több menetben, állandó lemezcserék közepette zajlott, így ha nem volt muszáj, akkor nem használtuk soha ezt a kényelmetlen parancsot. Már ez a probléma is megoldódott, mert a **DISKCOPY** parancs a másoláshoz használni tudja a merevlemezt, a másolás meggyorsítására, és kényelmesebbé tételére. Igaz ugyan, hogy már az MS-DOS 6-ban is megoldódott az a probléma, hogy csak egy menetben másolja át a lemezünket, ha volt elég memória a számítógépünkben. Most viszont használhatunk egy "**/M**" paramétert is, ami arra utasítja, hogy csak a hagyományos memóriát és a merevlemezt használja fel a másoláskor segédeszközül.

A **DIR** parancs képe is megváltozott egy kicsit. Mivel már nem ritka a többszáz kilobájtos, több megabájtos állomány sem a listákban, ezért jelentkezett egy apró kis kellemetlenség, amit már szinte mindenhol megoldanak manapság, a hosszú számok tagolása. A **DIR** által készített listában is megjelentek a tagolt számok (vagyis az 1 millió képe: 1,000,000 bytes free), csak ránk magyarokra nem gondoltak. Ugyanis ha a CONFIG.SYS fájlban szerepel a **COUNTRY=036,852** parancs, akkor a **DIR** parancs tagolt kiírásakor a vesszők helyett szóköz jelenik meg, ami rendkívül olvashatatlanná teszi az egész listát. Ugyanis megszoktuk azt, hogy a fájlnév, kiterjesztés után jön a méret, amit egy szóköz választ el a dátumtól. Így ha megszokásból végigtekintek egy listán, akkor csupa 1-2 bájtos fájlt látok, amelyek valójában 1-2 ezer valamennyi bájtosak, de megszokásból a szóköz utáni részt már nem olvasom hozzá a számhoz.

Zárszó

Végezetül egy kis ismertetővel szolgálnék azoknak, akik tovább szeretnék bővíteni számítástechnikai ismereteiket. Nehéz helyzetben van az, aki csupán egyedül kívánja továbbfejleszteni a számítástechnikai ismereteit. Nagyon sok számítástechnikai könyv, folyóirat van forgalomban, amiből nagyon nehéz kiválasztani azt, amire nekünk szükségünk van a további tanuláshoz. Először is azt kell felmérni, hogy mit szeretnénk csinálni a továbbiakban, de ezt sem könnyű felmérni, ha még azt sem tudjuk, hogy milyen lehetőségeink vannak.

Először mindenképpen tájékozódni kell arról, hogy milyen számítógépeken tudunk gyakorolni (iskolában, munkahelyen, ismerősnél, esetleg saját számítógépen, *...stb.*), és milyen programok állnak a rendelkezésünkre. Ebből fel kell tudni mérni, hogy mire van időnk, lehetőségünk, mit tudunk felületesen vagy részletesen megtanulni.

Utána következhet a megvalósítás, amiben a sok nyomtatott termék is segítségünkre van. Tájékozódhatunk, hogy milyen könyvtárak vannak a lakóhelyünkön, vagy melyik könyvet, folyóiratot tudjuk megvásárolni. A következő kis ismertetőben ebben a választásban kívánok a segítségetekre lenni, hogy könnyebben eligazodjatok a számítástechnikai szakirodalom "sűrű erdejében".

Számítástechnikai szakirodalom használata

Egy számítógépes program használhatóságát nagy mértékben meghatározza az, hogy milyen, és mennyi szakirodalom áll a rendelkezésünkre. Ha egy frissen vásárolt programhoz csak a kapott angol nyelvű leírás áll a rendelkezésünkre, és csak keveset tudunk angolul, akkor bizony nem sokat tudunk kezdeni a programmal. Hagyatkozhatunk az eddigi ismereteinkre, vagy a megérzéseinkre, és úgy próbáljuk meg kibogozni a program különböző részeit. Ez nagyon fáradságos munka, és nagyon időrabló is. Sokkal egyszerűbb, ha keresünk olyan magyar nyelvű szakirodalmat az adott programhoz, amiből gyorsan meg tudjuk tanulni az adott program kezelését, és utána már valóban nekünk dolgozik a program. A számítástechnikai folyóiratok is sok segítséget adnak a programok használatához, vagy a megvásárlás előtti kiválasztásukhoz.

Olyankor is haszonnal forgathatjuk a különféle szakirodalmat, ha nem is akarunk programot vásárolni, csak érteni szeretnénk a kezeléséhez. Így, ha olyan munkahelyre kerülünk, ahol az adott programmal már ténylegesen dolgozni kell, akkor már gyakorlottan állhatunk hozzá a munkához, és nem akkor kezdjük "betanulni magunkat a munkába".

Folyóiratok

A folyóiratok több témában is tájékoztatják olvasót. Az egyik fő területük, hogy az aktuális technikai újdonságokról is beszámolnak. Így folyamatosan nyomon tudjuk követni a technikai fejlődést, és ha esetleg számítógépet akarunk vásárolni - akár saját, akár a cégünk részére -, akkor az elvégzendő feladathoz a legideálisabb számítógépet tudjuk kiválasztani.

A másik hasznos szolgáltatásuk a folyóiratoknak, hogy gyakran közölnek különféle hardverteszteket, melyben azonos kategóriájú számítógépeket, vagy részegységeket

hasonlítanak össze, különféle teszteket végezve azokon, mellyel megmutatják az adott részegység felhasználási területét is. Ez is nagy segítség azoknak, akik például archíválási feladatokra valamilyen részegységet (Floppy, Winchester, Floptical, Streammer,...) szeretnének vásárolni, de nem tudják eldönteni, hogy melyik lenne a legideálisabb számukra, melyiket tudnák a legjobban kihasználni. Az újságban közölt teszteredmények és árak tükrében ki tudjuk választani, hogy melyik az a részegység, amelyik úgy végzi el az adott feladatot, ahogyan azt mi szeretnénk, ami az igényeinkhez a legjobban igazodik.

A szoftverekkel is szoktak összehasonlításokat végezni, amiből pedig a különböző szoftvergyártók által kiadott programok használhatóságát ismerhetjük meg. Hasznos az ilyen szoftvertesztek elolvasása is, mert a szoftvereket hozzáértő gyakorlott szakemberek végzik, akiket nem kápráztat el egy-egy színpompa, vagy ügyes funkció, és meglátják mögötte a program valós használhatóságát is. Ezek a szakemberek ismernek sok olyan felhasználási területet, amire egy olyan programot használni kell, amit éppen tesztelnek. Így jó megközelítéssel el tudják objektíven dönteni, hogy az adott program mennyire felel meg az elvárásoknak. Erre nekünk sok esetben nincs lehetőségünk, mert az újságírók is csak egy teszt erejéig kapják meg a programot vagy az adott hardvert, amit a vizsgálódások után vissza is adnak a kereskedőnek.

Kézikönyvek

Az olyan könyveket nevezzük ***kézikönyv***eknek, amelyek pontosan ismertetik egy adott szoftver lehetőségeit, parancsait, kapcsolót, a rajta elvégezhető funkciókat. Nem arra szolgálnak, hogy egy kezdő felhasználó megtanulja belőle a szoftver kezelését, hanem arra, hogy egy gyakorlott - számítógépet ismerő, sokat tapasztalt - személy a program lehetőségeit megismerje. Ez azt jelenti, hogy aki a programot használja, az már ismeri az adott program működési rendszerét, csak a kulcsszavakat (parancsokat, utasításokat, függvényeket,...) nem tudja. Ezeket kiolvasva a kézikönyvből, már kezelni tudja azt, és csak néhol szorul további ismeretek megszerzésére. Általában az ismertetett parancsok,... ábécé sorrendben találhatók meg a könyvben, a gyorsabb keresést biztosítva ezzel. Ez pedig feltételezi azt, hogy az adott parancs nevét már ismerjük, tudunk a létezéséről, csak a pontos szintaktikai szabályait nem ismerjük fejből. Erre az esetek nagy többségében nincs is szükség, hisz éppen ez a kézikönyvek feladata, hogy ezt megmutassa nekünk.

Tankönyvek

Az olyan könyveket nevezzük ***tankönyv***eknek, amelyek a felhasználó szempontjából nézik az adott szoftvert vagy hardvert, és úgy ismertetik, hogy abból az olvasó a legkönnyebben meg tudja tanulni az adott tananyagot. A legfontosabb szempontja az ilyen jellegű könyveknek, hogy a tankönyv lehetőleg csak olyan ismeretet tartalmazzon, amit a tanuló ki is tud próbálni, meg tud győződni annak valódiságáról. A tankönyv színvonalát emeli az is, ha sok olyan példát tartalmaz, amit a tanuló ki is tud próbálni, és a kapott eredményt is le tudja ellenőrizni. Az olyan példák nem érnek semmit sem, ami megmutatja, hogy mit írjunk be, de nem mondja meg, hogy milyen környezetben, milyen körülmények között adható ki a parancs, és milyen eredményt kell kapnunk a helyes használat esetén. Utána a kapott eredményt is ki kell értékelni, mert csak akkor fejeztük be egy parancs ismertetését. Ez azt jelenti, hogy hiába futott le jól a

program, írta ki helyesen az eredményeket, ha a kezdő felhasználó nem tudja, hogy mit keressen, mit lásson a kapott eredményben, kiírásban. Sajnos nagyon kevés ilyen jellegű tankönyv van forgalomban, és ez erősen hátráltatja a felhasználói programok elterjedését.

Egy tankönyvnek a másik feladata az, hogy megismertesse az olvasót a téma alapjaival, elméleti alapfogalmaival, alapvető fogásaival, hogy a későbbiek során már "egy nyelven beszéljen" az olvasóval. Ha ezek elmaradnak, akkor a bevezetőben bizonyára szerepel, hogy milyen alapfogalmak ismeretét feltételezi az adott tankönyv. Ilyen tankönyvet csak akkor használjunk, ha valóban rendelkezünk az adott ismeretekkel, és akkor valóban a hasznunkra fog szolgálni a tankönyv.

Programismertető könyvek

Ebbe a kategóriába azok a könyvek sorolhatók, melyek speciálisan egy adott program ismertetését végzik el, melyhez a szükséges egyéb ismereteket feltételezik az olvasóról. Hogy mik ezek az ismeretek, azt nem minden esetben ismertetik, de ezt menetközben az olvasó úgyis megtudja. Az adott program ismertetése kétféle lehet, kézikönyvszerű "száraz", szabatos ismertetés, és/vagy olvasmányos jellegű tanító szándékú ismertetése a programnak. A legideálisabb eset az, ha a könyv két részből áll, először egy általános ismertetésből, figyelemfelkeltésből, majd a kiaknázható lehetőségek pontos ismertetéséből. Ha a könyv sok használható példát is tartalmaz, akkor az tovább segíti a szoftver elsajátítását.

Idegennyelvű folyóiratok és könyvek

Azok, akik olvasás szinten ismernek egy vagy több idegennyelvet, azoknak lehetőségük van az idegennyelvű folyóiratok és könyvek olvasására. Ez sokban segíti az adott nyelv jobb elsajátítását is, ami később szakszövegek jobb megértését segíti elő, másrészt olyan információkhoz is hozzájuthat az olvasó, ami nem jelent meg magyar nyelven. Például az Windows NT szoftver - operációs rendszer - még annyira új, hogy nem jelent meg róla (1994. év eleje) részletes magyar nyelvű leírás, mert csak néhány példány van a szoftverből az országban, így egy könyv többezer példányos megjelentetése gazdaságtalan lenne. (Kisebb példányszámban pedig nem biztos, hogy gazdaságosan elő lehetne állítani.)

Nyugat-Európában, Amerikában viszont már több ezer példányt eladtak az illető szoftverből, így ott természetesen jelent meg már valamilyen könyv a szoftverről. Ha valaki Magyarországon meg akar ismerkedni ezzel a szoftverrel, akkor bizony kénytelen az idegennyelvű szakirodalmat tanulmányozni.

Függelék

I. Billentyűk kódjai

Billentyű	Kód	SHIFT+Kód	CTRL+Kód	ALT+Kód
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
HOME (num keypad)	0;71	55	0;119	--
UP ARROW (num keypad)	0;72	56	(0;141)	--
PAGE UP (num keypad)	0;73	57	0;132	--
LEFT ARROW (num keypad)	0;75	52	0;115	--
RIGHT ARROW (num keypad)	0;77	54	0;116	--
END (num keypad)	0;79	49	0;117	--
DOWN ARROW (num keypad)	0;80	50	(0;145)	--
PAGE DOWN (num keypad)	0;81	51	0;118	--
INSERT (num keypad)	0;82	48	(0;146)	--
DELETE (num keypad)	0;83	46	(0;147)	--
HOME	(224;71)	(224;71)	(224;119)	(224;151)
UP ARROW	(224;72)	(224;72)	(224;141)	(224;152)
PAGE UP	(224;73)	(224;73)	(224;132)	(224;153)
LEFT ARROW	(224;75)	(224;75)	(224;115)	(224;155)
RIGHT ARROW	(224;77)	(224;77)	(224;116)	(224;157)
END	(224;79)	(224;79)	(224;117)	(224;159)
DOWN ARROW	(224;80)	(224;80)	(224;145)	(224;154)
PAGE DOWN	(224;81)	(224;81)	(224;118)	(224;161)
INSERT	(224;82)	(224;82)	(224;146)	(224;162)
DELETE	(224;83)	(224;83)	(224;147)	(224;163)
PRINT SCREEN	--	--	0;114	--
PAUSE/BREAK	--	--	0;0	--
BACKSPACE	8	8	127	(0)
ENTER	13	--	10	(0
TAB	9	0;15	(0;148)	(0;165)
NULL	0;3	--	--	--

Billentyű	Kód	SHIFT+Kód	CTRL+Kód	ALT+Kód
A	97	65	1	0;30
B	98	66	2	0;48
C	99	66	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22
V	118	86	22	0;47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33	--	0;120
2	50	64	0	0;121
3	51	35	--	0;122
4	52	36	--	0;123
5	53	37	--	0;124
6	54	94	30	0;125
7	55	38	--	0;126
8	56	42	--	0;126
9	57	40	--	0;127
0	48	41	--	0;129
-	45	95	31	0;130
=	61	43	---	0;131
[	91	123	27	0;26
]	93	125	29	0;27
\	92	124	28	0;43
;	59	58	--	0;39
'	39	34	--	0;40
,	44	60	--	0;51
.	46	62	--	0;52
/	47	63	--	0;53
`	96	126	--	(0;41)

Billentyű	Kód	SHIFT+Kód	CTRL+Kód	ALT+Kód
ENTER (keypad)	13	--	10	(0;166)
/ (keypad)	47	47	(0;142)	(0;74)
* (keypad)	42	(0;144)	(0;78)	--
- (keypad)	45	45	(0;149)	(0;164)
+ (keypad)	43	43	(0;150)	(0;55)
5 (keypad)	(0;76)	53	(0;143)	--

II. Képernyő attribútumok

0	Minden attributum kikapcsolva (All off).
1	**Vastagabb** karakterek (Bold).
4	Aláhúzott (Underline csak monokróm monitoron).
5	Villogtatás bekapcsolva (Blink on).
7	A teljes képernyő inverzben (Reverse video on).
8	Elrejtés bekapcsolva.(Concealed on).

Karakterszín (Foreground - előtérszín) kódok:

30	Black (Fekete)
31	Red (Piros)
32	Green (Zöld)
33	Yellow (Sárga)
34	Blue (Kék)
35	Magenta (Lila)
36	Cyan (Ciánkék)
37	White (Fehér)

Háttérszín (Background color) kódok:

40	Black (Fekete)
41	Red (Piros)
42	Green (Zöld)
43	Yellow (Sárga)
44	Blue (Kék)
45	Magenta (Lila)
46	Cyan (Ciánkék)
47	White (Fehér)

III. Képernyő üzemmódkapcsolók

0	40 x 148 x 25 monochrome (text)
1	40 x 148 x 25 color (text)
2	80 x 148 x 25 monochrome (text)
3	80 x 148 x 25 color (text)
4	320 x 148 x 200 4-color (graphics)
5	320 x 148 x 200 monochrome (graphics)
6	640 x 148 x 200 monochrome (graphics)
7	Enables line wrapping
13	320 x 148 x 200 color (graphics)
14	640 x 148 x 200 color (16-color graphics)
15	640 x 148 x 350 monochrome (2-color graphics)
16	640 x 148 x 350 color (16-color graphics)
17	640 x 148 x 480 monochrome (2-color graphics)
18	640 x 148 x 480 color (16-color graphics)
19	320 x 148 x 200 color (256-color graphics)

IV. Escape-szekvenciák táblázata

Minden escape-szekvenciát az "**ESC**" karaktersor vezet be, ami helyett a számítógépbe egy [ALT]+[2]+[7] (alt-27 karakter: képe ←) -et kell beírni. Ennek a bevitele nem is olyan egyszerű, mert ha a legtöbb helyen (szövegszerkesztőben, **COPY CON** üzemmódban, ...*stb.*) beírjuk ezt a karaktert a számbillentyűzetről, akkor az az [ESC] billentyű leütésének felel meg, amihez az esetek 99%-ában van funkció rendelve. A DOS **EDIT** szövegszerkesztőjében azonban lehetőségünk van ilyen és egyéb speciális karakterek bevitelére.

Nyomjuk meg a [CTRL]+[P]-t, erre nem fog semmi megjelenni, viszont most bevihetjük az [ALT]+a számbillentyűzeten beírt számmal a kívánt karaktert, például az [ALT]+[2]+[7]-et, aminek hatására, megjelenik egy balra mutató nyíl.

Aki nem emlékezne rá, annak elmondom, hogy ilyenkor le kell nyomni az [ALT] billentyűt, és lenyomva kell tartani, majd be kell gépelni a jobb oldali számbillentyűzeten a karakter ASCII kódját, és csak utána szabad elengedni az [ALT]-ot.

Fontos: A parancsoknál használt kis és nagybetűket nem szabad összekeverni!

A következőkben megismerhetjük azokat a kódokat, amelyekkel megváltoztathatjuk a billentyűzetet, a képernyő képét, üzemmódját, ...*stb.*

Példákat csak néhánynál mutatok be, a többinél értelemszerűen kell alkalmazni az ismereteket. Egyébként a különféle szekvenciák használatára több helyen találunk példát ebben a tankönyvben, és az első részben is.

ESC[*sor*,*oszlop*H

Beállítja a **kurzor pozícióját**. Az ezután kiírásra kerülő szöveg, erre a pozícióra fog kerülni.

Például:

```
                                                    10:14:17:25 .
C:\DOS>_

C:\DOS>prompt $e[1,68H$t$_$p$g
```

A **PROMPT** parancsnak van egy **$e** paramétere, ami egy ESC kódot visz ki a képernyőre, hogy használhassuk az escape-szekvenciákat. Beállítjuk a kurzor pozícióját az első sor 68. karakterhelyére, oda kitesszük a pontosidőt, majd az új sorba lépve kitesszük a kurzort. Van ennél jobb megoldás az óra kihelyezésére, de erről majd később.

A másik beviteli módszer, hogy elindítjuk az **EDIT** szövegszerkesztőt, és az **ECHO** parancs után beviszünk egy ESC jelet (←) úgy, hogy leütjük a [CTRL]+[P]-t, majd utána beírjuk az [ALT]+[2]+[7]-et. Ekkor megjelenik a kis nyilacska, a többi részét a parancsnak pedig már könnyedén be tudjuk gépelni.

```
ECHO ←[1;25HMóricz-tankönyvekből tanulok!
```

A fájlnak adjunk egy nevet, és BAT kiterjesztéssel mentsük el. Utána futtassuk le ezt a batch-fájlt. Meg fog jelenni a képernyő tetején kb. középen a megadott felirat.

ESC[*sor*,*oszlop***f**

Beállítja a **kurzor pozícióját**. Az ezután kiírásra kerülő szöveg, erre a pozícióra fog kerülni.

ESC[*szám***A**

A kurzort *szám* sorral **felfelé** mozgatja a képernyő teteje felé.

ESC[*szám***B**

A kurzort *szám* sorral **lefelé** mozgatja a képernyő alja felé.

ESC[*szám***C**

A kurzort *szám* karakterhellyel jobbra mozgatja a képernyőn.

ESC[*szám***D**

A kurzort *szám* karakterhellyel balra mozgatja a képernyőn.

ESC[**s**

Megjegyzi a kurzor aktuális pozícióját.

ESC[**u**

A kurzort visszaállítja az elmentett pozíciójába.

ESC[**2J**

Képernyőtörlés. A kurzor a képernyő bal-felső sarkába kerül.

ESC[**K**

Törli az összes karaktert a sor végéig.

ESC[*kód*;*kód*;...;*kód***m**

Monitor színeit állíthatjuk be vele. Több *kód* is kiadható, de egy is elegendő. Az utolsó *kód* után közvetlenül kerüljön a kis "**m**" betű. Ez a betű zárja le a parancsot. (Lásd *Képernyő attribútumok* !)

Például:

```
ECHO ←[32;43
```

Ez az utasítás a képernyőt kék színűre állítja, és a karakterek sárga színnel fognak megjelenni rajta.

ESC[=*módh*

A képernyő üzemmódjai közül választhatunk. (lásd *Képernyő üzemmódkapcsolók*!)

ESC[=*módl*

A képernyő előző üzemmódját állítja vissza. (lásd *Képernyő üzemmódkapcsolók*!)

ESC[*Bkód;karakterlánc;....*p

A billentyűzet átdefiniálására szolgál. A *Bkód* értékeit lásd a *Billentyűk kódjai* című fejezetben a 116 .oldalon. A karakterlánc lehet egy másik billentyűzetkód, vagy egy idézőjelek közé zárt szöveg, vagy csak egy betű, vagy egy ASCII kód. Az utolsó *karakterlánc* után közvetlenül írd be a kis "**p**" betűt.

Például:

```
ECHO ←[0;133;"CLS";13;"DIR /p";13p
```

A parancs hatására a billentyűzeten az [F11]-es billentyű lenyomására letörlődik a képernyő, és kiíródik az aktuális meghajtó aktuális könyvtárának a fájllistája.

Ajánlat az LSI Oktatóközpont kiadványaiból

Szerzők	*Könyv címe:*	netto ár
Móricz Attila:	Novell hálózati alapismeretek I.	435 Ft
Móricz Attila:	Novell hálózati alapismeretek II.	435 Ft
Móricz Attila:	DOS alapismeretek I.	382 Ft
Móricz Attila:	Windows alapismeretek	598 Ft
Móricz Attila:	DOS alapismeretek II.	-
Bartha Gábor:	ACT! for Windows	564 Ft
Kelevitz Ferenc:	MS ACCES for Windows	399 Ft
Honti-Honti:	CorelDRAW! 2.01 for Windows	399 Ft
Bartha-Honti:	CorelDRAW! 3.0 for Windows	699 Ft
Bartha-Bartha:	CorelDRAW! 3.0 a csapattagok	599 Ft
Bartha Gábor:	CorelDRAW! 4.0 kézikönyv I.	1.018 Ft
Bartha Gábor:	CorelDRAW! 4.0 kézikönyv II.	991 Ft
Mörk Péter:	Word for Windows 2.0	290 Ft
Hold Gábor:	Page Maker 4.0 for Windows	599 Ft
Klucs-Koleszár:	Excel for Windows 3.0	443 Ft
Klucs-Koleszár:	Excel for Windows 4.0	560 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 1	274 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 2	236 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 3	227 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 4	225 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 5	200 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 6	264 Ft
Klucs-László:	Excel 4.0 lépésről lépésre 7	254 Ft
Bartha Attila:	Norton ... for DOS	499 Ft
Bartha Attila:	Norton Utillities 6.01	599 Ft
Bartha Attila:	Norton Commander 4.0	489 Ft
Hargittai-Kaszanyiczki:	Grafikák készítése IBM PC -n	290 Ft
Hargittai Péter:	Harward Graphics	590 Ft
Hargittai-Kaszanyiczki:	Visual BASIC for Windows	599 Ft
Hargittai-Kaszanyiczki:	Visual C++	682 Ft
Kaszanyiczki László:	DR DOS 6.0 kapcsolat a Windows 3.0 -val	490 Ft
Csábi-Dallos:	DOS 5.0 és kapcsolat a Windows 3.0 -val	890 Ft
Dallos-Fábián-Zonbor:	DOS 6 kézikönyv	809 Ft
Hargittai-Kaszanyiczki:	A DOS titkai	541 Ft
Boér-Dóra-Fenyő-Seres:	Az IBM PC belső felépítése	499 Ft
Dr Kovács Magda:	32 bites mikroprocesszorok 80386 I.	189 Ft
Kaszanyiczki László:	Egér programozása	249 Ft
Gál-Dallos:	Quick BASIC	469 Ft
Gál István:	Quick BASIC Szubrutingyűjtemény	292 Ft
Nyéki-Nagy:	Turbo BASIC	453 Ft
Benkő-Poppe:	Turbo C++	493 Ft
Pirkó József:	Turbo Pascal 5.5	599 Ft
Pirkó József:	Turbo Pascal 6.0 & for Windows	799 Ft
Pirkó József:	Turbo Pascal 7.0	453 Ft
Drótos Dániel:	Borland Pascal 7.0	773 Ft
Dr Kovács Magda:	Egyszerűen a mikroszámítógépről	499 Ft
Dr Kovács Magda:	Mikroszámítógép-mikroelektronikai értelmező szótár I-III. kötet	820 Ft
Dr Kovács Magda:	Mikroszámítógépek alkalmazása értelmező szótár I-III. kötet	1.172 Ft
Dr Kovács Magda:	Mikroszámítógépek alkalmazása értelmező szótár I-III. kötet	499 Ft
Balogh-Berkes-Kovács:	Számítógépes távközlés telematikai szolgálatai	463 Ft

Ára: 460 Ft