Master DOS

pro SAM Coupé

Napsal Andrew Wright s pomocí některých materiálů, kterými přispěl Alan Miles.

Přeložil a napsal Ing.Vladislav Viták, Smetanovo nám.1857 H.Brod
Ing.Petr Havlišta

S případnými podněty nebo problémy s tímto programem kontaktujte fy SAMCO nebo pište na adresu:

Dr.Andrew Wright, 24 Wyche Ave, Kings Heat, BIRMINGHAM B14 6LQ

- VITASOFT -

1991

Úvod k užívání Master DOSu

Pro Váš klid je disketa s Master DOSem zabezpečena ochranou proti přepsání, aby nedošlo k jejímu náhodnému smazání. V jednom rohu můžete vidět malou dírku. Jestli-že ne posuňte malé plastikové šoupátko tak, aby vidět byla.

Abyste si mohli nahrát Master DOS do Vašeho počítače, zapněte jej a vložte disketu s Master DOSem do drajvu č.1, který je po levé ruce. Disketu vkládejte do mechaniky štítkem nahoru a kovovou částí od těla. Potom stiskněte klávesu F9 - led dioda na disketové mechanice se rozsvítí a "Diskový operační systém" se začne natahovat do počítače.Počítač teď bude rozumět řadě nových příkazů, které mu umožňují ovládat operace s diskem.

Měli by jste vědět, že pokud není do počítače natažen DOS, SAM směruje příkazy LOAD a SAVE na magnetofon. Stisknutím klávesy F9 je natažen DOS použitím klíčového slova BOOT - je připojena disketová mechanika d1, SAVE, LOAD, MERGE, a VERIFY teď automaticky užívají disketovou mechaniku častěji než magnetofon.

To znamená, že programy v BASICu, které pracují s magnetofonem, obvykle nemusí být kompletně převedeny na spolupráci s diskem. Pro opětovné použití magnetofonu stačí vložit do názvu souboru "t:". Například SAVE"t:test", nebo SAVE "t 45:faster" nebo LOAD "t:test". Existuje jedna vyjímka, kdy "t:" nemusíte vkládat LOAD " " vždy použije magnetofon, protože natahování souboru bez vyznačení jeho jména v DOSu je nesmysl. Pod klávesami F7 a F8 jsou uložena klíčová slova LOAD " " a LOAD " " CODE; a ty budou vždy pracovat s magnetofonem. Můžete také napsat: DEVICE t, abyjste se vrátili zpět do pracovního režimu s magnetofonem.

Kopírování Vašeho disku s Master DOSem

První věcí kterou by jste měli udělat, je zkopírování Vašeho disku s Master DOSem. Aby jste to mohli udělat,stiskněte tlačítko na drajvu, vytáhněte disketu s Master DOSem a místo ní vložte čistou disketu. Potom napište: FORMAT " D1 " a stiskněte klávesu RETURN (od této chvíle budu předpokládat, že víte, že RETURN je potřeba stisknout po každém napsání příkazu). Počítač bude formátovat všechny stopy, a potom zkontroluje, zda jsou všechny O.K Nestane-li se tak, zkuste to znovu raději s jinou disketou. Když je formátování ukončeno, vyjměte disketu a vlože zpět disketu s Master DOSem. Napište: BACKUP "d1" TO "d1". Všechny soubory bu-

dou čteny z diskety. Když se objeví sdělení "....."(vložte cílový disk a stiskněte klávesu), vytáhněte disketu s Master DOSem, vložte disketu, kterou jste naformátovali dříve a stiskněte klávesu (jakoukoliv). Když se objeví zpráva "O.K.,0:1", kopírování je ukončeno.

(To se stane v případě, že na disketě není umístěno mnoho rozsáhlých souborů. Jinak budete vyzváni "Insert source disk"(vložte zdrojový disk). Následují vyzvání dokud není kopírování ukončeno).

Nyní uložte originální disketu s Master DOSem na bezpečné místo a dále používejte jen kopii.

Každou disketu před jejím použitím musíte zformátovat. Není to nezbytné pro disketu,která obsahuje kopii Master DOSu, ale je to vhodné. DOS musí být první soubor na disketě, jinak nebude nalezen, proto je vhodné kopírovat jej až po zformátování. Můžete také použít příkaz COPY "mdos1" TO "*" ke zkopírování DOSu z diskety na další disketu - následují vyzvání.

RAM disky - proč máte alespoň šest diskových jednotek?

RAM Disk je sekce paměti počítače, která se chová jako disketová jednotka. RAM Disky jsou detailně popsány dále, a tak se zde o nich jen zmíním,aby jste věděli, že máte ve svém Coupé minimálně šest "drajvů", ačkoliv si myslíte, že máte jen jeden.Můžete obvykle použít dvou drajvové podoby příkazů vysvětlených v tomto manuálu později jen nezapomeňte, že RAM disk ztrácí svůj obsah, když počítač vypnete.

Vysvětlení adresáře:

Vložte Váš nový disk s MASTER DOSem do mechaniky č.1 a napište DIR. DIR zobrazí na obrazovce adresář aktuálního disku (t.j toho, který je určen příkazem DEVICE)
-tedy v mechanice č.1 na obrazovce uvidíte sdělení podobné tomuto:

```
Master DOS1:

MDOS1          memuse         prog1
prog2          prog3          rafprog

Number of Free K-Bytes = 758   (Počet volných K-bajtů = 758)
6 Files, 74 Free Slots         (6 souborů, 74 volných stop)
```

Toto je jednoduchý adresář, který nám dává pouze jména souborů na disketě a některé informace o disku samotném. Jména souborů jsou setříděna podle abecedy (tedy přesněji jsou setříděny podle ASCII kódů znaků. "1"je před "A", které je před "a"). Pro získání detailnějších infomací napište: DIR1(nebo DIR2 pro drajv č.2). Uvidíte něco podobného, jako je toto:

```
Master DOS1:

1 MDOS          31 C 65536,15700
2 prog1          2 BASIC
3 prog2          3 BASIC
4 prog3          3 BASIC
5 rafprog        2 BASIC
6 memuse         3 BASIC    1

Number of Free K-bytes = 758
6 Files, 74 Free Slots
```

Nelekejte se, pokud obrazovka nevypadá přesně jako toto. Jméno disku je napsáno nad adresářem vlevo.Zde je to "Master DOS". Napravo od jména disku je "1:", která nám říká, že adresář náleží drajvu č.1. Potom následuje dlouhý seznam jmen. Pod každým jménem je soubor.

Soubory mohou být: programy v BASICu, programy ve strojovém kódu, obrazovky nebo soubory dat. Na konci seznamu jsou některé informace o disku: ještě volné místo v kilobajtech, počet ukázaných souborů a počet "volných stop" pro vedlejší jména souborů v adresáři. Disk v příkladu může obsahovat více než 80 souborů, a ještě je na disku místo. Později objevíte,jak připravit disk aby byl schopen absorbovat stovky souborů (viz.FORMAT).

Každý záznam v seznamu souborů začíná číslem nahrávky. Kdykoliv ukládáte soubor,Coupé mu přiřadí první dostupné číslo souboru. Takže, máme-li adresář ukázaný v příkladu, další soubor, který uložíte bude automaticky začínat programovým číslem 7.Toto programové číslo zůstává stejné, dokud není soubor vymazán. Ale jestli-že vymažete (ERASE), řekněme číslo 9, potom další soubor, který uložíte na disk bude začínat novým číslem souboru 9.

Druhý sloupec nám ukazuje jména souborů,která mohou mít délku 10 znaků. Jména mohou obsahovat téměř všechny znaky; i když pravděpodobně je nejlepší začínat jméno písmenem a pokračovat písmeny, čísly a třeba dělat mezery. Vyvarujte se jménům začínajícím "d1", "d2", "d3" atd., a jménům začínajícím čísly - ty mohou být dvojznačné (označují např. čísla drajvů....)

Tečky mají speciální roli v oddělování hlavní části jména od rozšíření (extense) a znaky "?", "*", "/" a "\" mají také speciální účel. Vyhněte se užití těchto znaků ve jménech souborů, jejich užití bude podrobněji popsáno později.

Třetí sloupec v adresářovém souboru zobrazuje počet využitých sektorů disku. Každý sektor obsahuje 512 byte (=0,5 kilobyte), takže chcete-li zjistit, kolik kilobyte zabírá ten který soubór, vydělte číslo ve třetím sloupci 2.

Čtvrtý sloupec je určen pro typ souboru. Zde jsou různé typy souborů, které budete nejčastěji používat:

BASIC	= program v BASICu	
C	= Code file	- soubor ve strojovém kódu
SCREEN$	= obrazový soubor	
D.ARRAY	= pole číselných dat	
$.ARRAY	= znakové pole	
SNP 48k	= soubor fotogr. snímku (kopie paměti Spectra)	
DIR	= podadresář	

Pravděpodobně nebudete ještě rozumět významu všech těchto typů souborů, ale nemějte obavy, budou popsány později.

Soubory, které jsou typu CODE (strojový kód) mají za označením C startovací adresu a délku programu,to bude vhodné pro pokročilejší programátory.

Když číslo následuje za slovem BASIC, pak je to číslo řádku odkud bude program po natažení do počítače automaticky spuštěn.

Je-li Váš adresář delší než obrazovka, budete dotázáni na "SCROLL?" (scrolovat?). Pokud jste přečetl adresář, stiskněte < RETURN >, abyste si mohl prohlédnout další obrazovku adresářových záznamů. Prozatím to bude o adresářích stačit. Podrobnosti dalších forem budou popsány později.

Jednoduché ukládání a natahování (SAVING) (LOADING)

Vložte naformátovaný disk do disketové mechaniky, potom zapište do počítače tento krátký program, který použijeme pro ilustraci různých operací s diskem.

```
10 REM circles
20 FOR r=1 TO 255 STEP 2
30 CIRCLE 120,77,r
40 NEXT r
```

K uložení (SAVE) programu na drajv č.1 pod jménem "circles", napište: SAVE "circles". Maximální počet znaků ve jménu souboru je 10. Velké písmena jsou rovnocenná malým, takže " circles " je to samé jako " CIRcles ". Mezery uvnitř jména jsou rozlišovány, takže "file 1" je odlišný název od "file1".

Nyní si ověříme, zda byl soubor správně uložen - napíšeme: VERIFY "circles" a měli byste obdržet zprávu O.K. (VERIFY porovnává program v počítači s programem na disku - takže jestliže program zeditujete a pak opět provedete VERIFY obdržíte zprávu "VERIFY failed" (neúspěšná kontrola)).

Teď vymažte program v počítači tím, že napíšete NEW, poté natáhněte program zpět z disku příkazem LOAD "circles".Objeví-li se zpráva O.K., program byl natažen. Stiskněte klávesu RETURN, tím program vylistujete a ověřte si ho.

Je také možné udělat program, který se automaticky spustí poté co je natažen do počítače. Udělejte to tedy příkazem: SAVE "circles" LINE 10.

Ale my už na disku jeden program se jménem " circles " máme. Coupé Vám to oznámí a zeptá se Vás, zda-li si přejete přepsat existující soubor. Stiskněte Y (Yes-Ano) při souhlasu nebo N (No -Ne) při nesouhlasu. V tomto případě není třeba potvrzovat volbu stlačením klávesy RETURN.Pro minimalizaci špatné volby,kterákoli klávesa mimo Y je počítačem akceptována jako Ne.

Napište DIR1, abyste viděli "circle" v adresáři. Všimněte si čísla řádku automatického spuštění za nápisem "BASIC"(Vlastně je to číslo řádku automatického příkazu GO TO, ale každý tomu říká "auto-run" (samospoušť)).

Pro ukládání a natahování z drajvu č.2 stačí napsat: DEVICE d2 a potom psát přesně to co pro drajv č.1, nebo můžete použít jméno souboru, před který je číslo drajvu; např.: "d1:circles". "d2:" je odděleno od jména souboru před použitím SAVE nebo LOAD. Proto to není počítáno jako součást 10-ti znaků, které máte přiděleny pro vlastní jméno souboru.

Zkrácené natahování souboru

Po zobrazení podrobného adresáře můžete natáhnout program jednoduše užitím jeho čísla z levého sloupce - např. LOAD 3. Nemusíte dělat ani mezeru za LOAD. Tento způsob pracuje u všech typů souborů s vyjímkou souborů polí. Jsou natahovány soubory z aktuálních disků, takže příkaz DIR 2:LOAD 12 natáhne dvanáctý soubor z adresáře drajvu č.2

Tento způsob natahování je zvláště vhodný, chcete-li natáhnout soubory SCREEN$ nebo CODE - klidně použijte správné číslo.

Soubory AUTO a BOOT

Stisknete-li klávesu F9, je vloženo speciální klíčové slovo BOOT a to způsobí natažení DOSu.BOOT také vyhledává všechny soubory startující pomocí "aut" a natahuje je, když je najde. Pokud uložíte program v BASICu za použití SAVE " auto "LINE 1, program bude automaticky natažen a spuštěn bezprostředně po natažení DOS

Stisknete-li znovu F9, nebo použijete-li klíčové slovo BOOT, poté co byl natažen DOS, soubor "auto" bude opět natažen, ale natažení DOSu bude vynecháno.

Jestliže napíšete BOOT1, bude natažen DOS bez toho, aby byl natažen soubor "auto".

Další formy SAVE a LOAD

Několik informací o dalších formách SAVE a LOAD můžete najít v příručce pro Coupé, ale já Vám zde dám jejich krátký soupis.

SAVE "name" LINE n - uloží soubor v BASICu "jméno" a GO TO řádek
SAVE "name" SCREEN$- uloží běžný (aktuální) obrazovkový obraz včetně jejich barev PALETTE
SAVE "name"CODE a,b- uloží soubor"name" ve strojovém kódu od adresy A v délce B
SAVE"name"CODE a,b,c jako předcházející,ale spustí strojový program z adresy C po jeho natažení
SAVE"name"DATA xyz$- uloží string, nebo pole stringů jako soubor jména "name"
SAVE"name"DATA xyz() uloží číselné pole xyz() jako soubor jména "name"

Všechny formy SAVE mohou být zapsány i formou SAVE OVER"name" V tomto případě bude jakýkoliv existující soubor se stejným jménem přepsán bez dotazování.
Toto se týká i souborů chráněných proti přepisu.

LOAD"name" SCREEN$ - natáhne soubor obraz obrazovky, nebo natáhne soubor strojového kódu na obrazovku.Soubor SCREEN$ automaticky nastaví správný MODE (mód) obrazovky.

LOAD"name" LINE n - natáhne soubor " jméno " v BASICu a provede GO TO na řádek n. Ignoruje jakýkoliv samospouštěcí řádek.

LOAD"name"CODE - natáhne soubor"name" strojového kódu na adresu uvedenou příkazem SAVE (viz adresář). Také pracuje se soubory typu SCREEN$

LOAD"name"CODE A - jako předešlý příkaz, ale natahuje od adresy A.

LOAD"name"DATA xyz$- natáhne soubor "jméno" string, nebo stringové pole a nazve jej xyz$

LOAD"name"DATA xyz() natáhne soubor číselného pole"jméno" a nazve jej xyz()

VERIFY a MERGE

Disky jsou docela kvalitní a VERIFY bude skoro vždy pracovat.

VERIFY "name" - Porovná soubor v BASICu s programem v počítači.

VERIFY"name"CODE - Porovná soubor ve strojovém kódu s odpovídající oblastí paměti počítače

VERIFY"name"CODE A - Porovná soubor ve strojovém kódu s paměťovou oblastí od adresy A

VERIFY také pracuje se soubory polí a soubory SCREEN$, ačkoliv později budete muset příkaz používat během programu (programování), protože Vaše psaní bude měnit obazovku.

MERGE"name" - spojí soubor "name" v BASICu a jeho proměnné s běžným (aktuálním) programem. Pomáhá šetřit čas, když je nějaký program příliš rozsáhlý.

MERGE"name"CODE - jako LOAD"name"CODE, ale zastaví jakýkoliv auto-start.

MERGE nemůžete použít s ostatními typy souborů.

Použití sítě

Příkazy SAVE a LOAD můžete použít i při práci v síti a to tak, že před jméno souboru vložíte "n:", například: LOAD"n:" na přijímacím počítači a potom napište SAVE"n:"testing" na odesílajícím počítači. Soubor se jménem "testing" se objeví na obrazovce jako po natažení z magnetofonu, ale přenos je mnohem rychlej-

ší. Můžete jednodušeji podávat sdělení dalšímu počítači, prostřednictvím ukládání a natahování souborů stringových polí.

BACKUP

Příkaz BACKUP Vám dovoluje zkopírovat celý disk.Disk na který chcete kopírovat (tzv. cílový disk) musí být vždy naformátován.V opačném případě (není-li naformátována)jsou všechny soubory na takovém disku již ztraceny. Můžete kopírovat z drajvu č.1 na drajv č.2, z drajvu č.2 na drajv č.1, z drajvu č.1 na drajv č.3 atd.., nebo můžete použít pouze jednu mechaniku. Například:

BACKUP"d1"TO"d2" kopíruje z drajvu č.1 na č.2
BACKUP"d1"TO"d1" kopíruje z drajvu č.1 na č.1 -jednou mechanikou

Čtení ze zdrojového disku začíná ihned. Použité oblasti disku jsou čteny do volné počítačové paměti dokud se nezaplní, nebo dokud je z disku co číst. Poté jste vyzváni ke vložení cílového (target) disku, pokud používáte jednu mechaniku a informace jsou zapsány na cílový disk. BACKUP často končí v tomto bodě,obzvláště tehdy, máte-li počítač s pamětí 512k, ale je-li to nezbytné, budete vyzváni k opětovnému vložení zdrojového disku a čtení a zapisování bude pokračovat tak dlouho, dokud všechny informace nebudou zkopírovány. Jestliže se přihodí před ukončením BACKUP jakákoliv chyba, nepoužívejte kopii disku i když Vám adresář ukáže, že všechny soubory jsou na disku.

Můžete snížit počet výměn disket potřebných pro BACKUP s jednou mechanikou tím, že uvolníte tolik paměti počítači, kolik je to jen možné. Můžete zkusit:

CLEAR 32767:OPEN TO 1 (těchto volných 48k normálně využívá BASIC nebo pokud jste použili více jak jednu obrazovku:
SCREEN 1:FOR s=2 TO 10:CLOSE SCREEN s: NEXT s
To bude volných 32k za každou obrazovku, která byla otevřena.RAM disk může být smazán použitím příkazu FORMAT - např:FORMAT"d3";0

Pojmenování cílového disku

Jména použitá v přík. BACKUP mohou být delší než "d1.Například:

BACKUP "d1:ignored"TO"d1:FILES 5"

Nehledě na určení zdrojového disku je první jméno ignorováno ale druhé jméno je použito k pojmenování cílového disku. V tomto příkladu bude disk (cílový) nazván "FILES 5". Jméno se objeví na

vrcholu seznamu adresáře (vlevo).Použijete-li jen "d1" nebo "d2" je použito jméno MDOSu. Pro pojmenování disku můžete také použít příkaz RENAME.

BACKUP označuje každý cílový disk rozdílným náhodným číslem, takže MASTE DOS může sdělit kdy měníte disk - i když vyměníte používaný disk při BACKUP.

COPY

Příkaz COPY můžete použít ke kopírování jednoho, nebo několika souborů.
Např: Proto, abyste udělali kopii souboru jména "Test",které se bude jmenovat "Testcopy" musíe napsat:

COPY "Test" TO "Testcopy"

Soubor "Test" bude natažen do dočasné (přechodné) oblasti paměti počítače a vy krátce uvidíte zprávu "LOADING Test". Potom se objeví zpráva "INSERT TARGET DISK ..." (vlož cílový disk a stiskni klávesu). Ke zhotovení nové kopie na stejný disk jednoduše stiskněte klávesu. Nebo, chcete-li mít kopii na jiném disku, vložte tento do mechaniky a stiskněte klávesu. (Nezapomeňte, že nový disk musí být naformátován).

Pokud se na cílovém disku již vyskytuje soubor nazvaný "Test copy", budete dotázáni zdali si přejete jej přepsat. Stiskněte Y pro potvrzení, nebo N pro anulování kopírování.

Jakmile je kopie ukládána (na disk), uvidíte "...." (Ukládání Testcopy) a poté uvidíte "...." (Vložte zdrojový disk a stiskněte klávesu). Coupé kontroluje, zda-li na zdrojový disk není více souborů, které mají být zkopírovány. (uvidíte proč, jakmile za chvíli dojdete ke kopii hvězdičkové konvence). Takže znovu vložte originální disk a stiskněte klávesu. Jestliže na disku není žádný další soubor ke zkopírování, uvidíte zprávu O.K.

Zneužití počítače je rostoucí sociální problém.Společnou příčinou těchto zneužívání je záměrné zničení, nebo ztráty "živých" souborů. Nevěřte, že Vám se to nemůže přihodit. Může! SAM Coupé se může stát stejnou obětí. Proto si zvykněte pořizovat kopie klíčových (nejdůležitějších) souborů. Používejte různé diskety ale stejná jména stejných souborů, abyste si zjednodušili život.

Rozsáhlé soubory je třeba kopírovat v několika sekcích, protože mohou být použity jen volné stránky paměti. Následují nápovědy, jsou-li nezbytné.

Používáte-li dvě diskové mechaniky, můžete také kopírovat soubor z jednoho drajvu na druhý. Tady je několik příkladů:

```
COPY "Test" TO "d2:Test"
COPY "d2:name" TO "d1:name"
COPY "d1:xyz" TO "d3:arghhhh"
COPY "name" TO "d2"
```

Poslední příklad používí "d2", což znamená "kopíruj na drajv č.2 a použij originální jméno", což je jistě vhodné a vysvětluje to proč byste neměli zkoušet pojmenovávat soubory "d1" nebo "d2"

Obecně lze říci o příkazech disku,použijete-li "d1"nebo"d1:" nebo "d1:*" po klíčovém slově TO, znamená to " drajv č.1, použij originální jméno" a obdobně pro ostatní diskové jednotky.

Má-li Váš počítač jen jednu reálnou diskovou jednotku můžete být roztrpčeni počtem výměn disket při příkazu COPY "d1" TO "d1" a příkaz BACKUP bude možná jednodušší k užití. (Můžete vždy výběrově mazat soubory pro provedení BACKUP). Samozřejmě můžete si také nastavit (připojit) RAMdisk jako přechodnou paměť.

```
10 FORMAT "d3",1,24:REM
20 COPY "d1" TO "d3"
30 PRINT "vložte cílový disk a stiskněte jakoukoliv klávesu"
40 PAUSE
50 COPY "d3" TO "d1"
60 FORMAT "d3",0
```

Máte-li počítač s pamětí 512k nebo rozšířenou RAM, budete schopni nastavit rozsáhlejší RAM disk. Také pokud nemáte na mysli malé programování,výše uvedené rutiny mohou být rozšířeny tak aby zvládly větší počty souborů v několika výměnách (disků). Užitečné přitom mohou být nové funkce DIR$, FSTAT nebo DSTAT.

V tomto bodě musíme vysvětlit několik zvláštností, které využijete v mnoha povelech pro práci s diskem - jsou to tzv. hvězdičkové konvence, rozšíření jména souboru a volba "ASK ME" (Zeptej se mne).

Hvězdičková konvence

Doslovně měnící se karty = žolík. Pod tímto pojmem rozumíme speciální symboly ve jméně souboru, které jsou ekvivalentní různým znakům nebo řadě znaků.Doslovný název je odvozen od speciálních karet v některých hrách, kterými můžete nahradit ostatní karty = žolík. Mohou být použity nejen v příkazu COY, ale také v příkazech DIR, ERASE, RENAME, HIDE atd. Hvězdičkou (*) nahradí řadu znaků,takže COPY"*" TO "d2" bude přijato jako příkaz k překopírování všech souborů z běžného (aktuálního) disku na disk 2 pod stejnými jmény. "d1:*" nebo "d4:*" znamená všechno na tomto

disku. Pro zpohodlnění práce může být místo toho použito "d1"nebo "d4", ale příkaz s hvězdičkou je mnohem pružnější. Chcete-li zkopírovat všechny soubory začinající písmenem "n", napište COPY "n*" TO "d2". Protože "*" nahrazuje kterýkoliv znak, je zde ještě symbol "?", který nahrazuje jednotlivé znaky, takže příkaz COPY "a??cs"TO"d3" bude kopírovat všechny soubory nazvané a..cs. Druhý a třetí znak může být jakýkoliv.Hvězdičkovou konvenci můžete také použít ve jméně po TO. Hvězdička znamená " nahraď " všechny znaky ze zdrojového jména".

Otazník znamená:"nahraď tento znak znakem ze zdrojového jména" a cokoliv jiného znamená " použij mne! ignoruj zdrojové jméno"

Například: COPY "M*" TO "X???TWO" bude kopírovat soubor "mrt" do souboru "Xrt TWO" a soubor "mrt2" do souboru "Xrt2TWO".

Toto je jen malá část toho, jak hvězdičkové konvence využít - ostatní později. Ale i toto Vám může být v některých případech užitečné.

Rozšíření jména souboru.

Už víte, že jména souborů mohou být maximálně 10 znaků dlouhá - včetně mezer, ale bez označení záznamového zařízení jako je "d2:" nebo "t50". Ale na pomoc organizování Vašich souborů Vám Coupé povoluje použít rozšíření jména souboru, stejným způsobem jako u obchodního počítače.

Můžete například ke všem Vašim souborům, kde máte dopisy, připojit rozšíření ".let" (zkratka letter = dopis), nebo k databázovým souborům připojit ".dat" a všechny Vaše grafické soubory označit rozšířením ".gra". Váš adresář by pak mohl vypadat takto

```
bank  .dat bank  .gra bank  .let
bank2 .dat invite.let lettr1.let
lettr2.let stamps.dat stamps.gra
```

Jména souborů však musí mít stále max. 10 znaků, včetně desetinné tečky. Povšiměte si způsobu,jak úhledně počítač odřádkuje všechna rozšíření. Vypadá to hezky, ale na druhou stranu to znamená, že soubor nazvaný " x.12345678 " bude zobrazen jako "x .123", ačkoliv jeho jméno je "x.12345678" a vy jej právě tak potřebujete natáhnout příazem LOAD. Nezapomeňte, že jméno je to co jste vložili, nikoliv to, co je zobrazeno. Pokud chcete,je možné úhledné seřazení rozšíření jmen souborů zrušit-viz proměnné DOSu.

Symbol "*" bude fungovat odděleně pro každou část rozšířeného jména, takže použijete-li např: COPY "d1:*.dat" TO "d2", budou zkopírovány všechny datové soubory, nebo příkaz ERASE "bank.*" smaže soubory "bank.dat", "bank.let" a "bank.gra". DIR ".let" Vám ukáže soubory s rozšířením ".let".

Volba ASK ME

Dalším způsobem výběru některých souborů je ten, že se Váš počítač před každou operací zeptá. Tato vložená volba pracuje s příkazy ERASE, RENAME, PROTECT a HIDE a také s COPY. Doplníte-li příkaz na konci otazníkem, sdělujete tím DOSu, že chcete, aby se Vás dotázal co zamýšlíte s každým souborem zvlášť. To je rozdíl od použití otazníku uvnitř jména souboru - viz hvězdičková konvence. Například:

COPY "d1" TO "d2"?

se Vás bude ptát

COPY "jméno souboru" (y/n/a/e)

pro každý soubor zvlášť. Můžete odpovědět stisknutím kláves "Y", přejete-li si soubor zkopírovat nebo klávesou "N" chcete-li jej přeskočit a pokračovat u dalšího souboru. Stisknutím klávesy "a" bude kopírován daný soubor a celý zbytek bez dalšího dotazování. Stisknete-li klávesu "e" vystoupíte z příkazu COPY. Stlačení jakékoliv jiné klávesy má za následek stejnou odezvu jako stlačení písmene "M".

Něco více o DIR

Příkaz DIR má mnoho odlišných forem. Zatím jste se setkali s příkazem DIR, který nám dodává jednoduchý, tříděný adresář aktuálního disku a s příkazy DIR 1, DIR 2 atd..., které nám dávají podrobné adresáře specifických disků.Pouze následně zmíněná forma dělá CLS první.
Zde jsou příklady některých dalších forem:

DIR* - podrobný adresář aktuálního disku
DIR 1! - jednoduchý adresář specifického disku
DIR "*.let" - podrobný adresář využívající hvězdičkovou konvenci
DIR "asd??"!- jednoduchý adresář využívající hvězdičkovou konvenci

DIR "D2:*.DAT"-podrobný adresář specifického disku využívající hvězdičovou konvenci.
DIR2;"*.dat" -jiná forma výše uvedeného příkazu
DIR DATE -podrobn adresář s datem uložení, byl-li užit.Snad nejlépe zobrazený v MODE 3. Viz příkaz CLOK
DIR DATE"k*" -jako výše uvedené, ale vybírající soubory
DIR "sub1/*" -adresář jmenovaného podadresáře. Viz příkazy SUB-DIRECTORIES

Můžete ovládat aby jednoduchý adresář byl abecedně setříděn či nikoliv a kolik sloupců bude použito na šířku obrazovky.(Běžně je počet sloupců maximální v běžném MODE a WINDOW). Viz DOS VARIABLES.

ERASE

K vymazání souboru napište: ERASE "jméno". Jako obvykle, i zde můžte vložit číslo drajvu, chcete-li jej specifikovat - např ERASE "d2:fred". Také zde znamená v příkazu "d1" všechno na disku 1 a "*" znamená "všechno na aktuálním disku". Takže buďte velice opatrní a uvědomte si co znamená ERASE "d1" když tento příkaz napíšete.

ERASE s volbou "ASK ME"

Předpokládejme, že máte disk se 30 soubory a z nich chcete 25 smazat. Je to nepříjemná práce napsat 25 jmen, ale ERASE "*" nebo ERASE "d1" smaže soubory všechny,takže co jiného Vám zbývá? Odpověď je nasnadě, použijte:

ERASE "*"?

Otazník sdělí DOSu, aby se Vás před každým smazáním souboru dotázal,zda jej smazat či ne.Je-li první soubor nazván "letter2" uvidíte: ERASE "letter2" (y/n/a/e) na spodní části obrazovky.

Můžete pokračovat stisknutím klávesy "y",chcete-li daný soubor smazat nebo stisknutím klávesy "n", chcete-li tento přeskočit a pokračovat na dalším souboru. Stisknutím klávesy "a" bude tento soubor a soubor za ním následující smazán již bez dalšího dotazování. Stisknete-li klávesu "e" vystoupíte z příkazu ERASE. Stisknutí kterékoliv jiné klávesy má stejný důsledek jako stisknutí klávesy "n".

Význam kláves Y = yes - ano
N = no - ne
A = all - všechno (smazat)
E = EXIT-východ (ven z příkazu)

Můžete také využít hvězdičkové konvence jako v příkazu ERASE "*.bas"?

Speciální formy příkazu ERASE pracují jen na podadresářích - - viz SUBDIRECTORIES

PROTECT (OCHRANA)

Jednotlivé soubory můžete před smazáním příkazem ERASE ochránit napsáním PROTECT "jméno"
Takto chráněný soubor bude v podrobném adresáři zobrazen s hvězdičkou v levém sloupci místo čísla souboru.
Ochrana může být odstraněna příkazem:

PROTECT OFF =jméno"

Příkazy PROTECT a PROTECT OFF můžete také doplnit hvězdičkovou konvencí a volbou "ASK ME" např:

PROTECT "d2:gr*"?

Chcete-li použít normální příkaz ERASE na chráněný soubor, nebo odpovíte-li "y" na sdělení " OVERWRITE? " (přepsat) během provádění příkazu SAVE,počítač pípne a objeví se zpráva "PROTECT ed FILE" (Chráněný soubor). Mažete-li více souborů za použití hvězdičové konvence, počítač pípne u každého chráněného souboru a příkaz ERASE bude pokračovat u dalšího souboru.
Pokud bude nalezen a smazán alespoň jeden nechráněný soubor, závěrečna zpráva bude znít "OK". V opačném případě bude zobrazeno opět sdělení "PROTECTed FILE"
Je-li soubor chráněn, přeci jen existuje jak jej přepsat,nebo smazat a to použitím příkazů:

SAVE OVER " jméno "
ERASE OVER " jméno "

Rovněž chcete-li překopírovat obsah jednoho souboru (řekněme souboru "nový") příkazem COPY pěs obsah jiného souboru (řekněme souboru "starý") a tento soubor "starý" je chráněn příkazem PROTECT, napište:

COPY OVER "nový" TO "starý"

Tento příkaz překopíruje obsah souboru " nový " do souboru " starý ", přičemž soubor "starý" si zachová své původní jméno.

HIDE

Soubory můžete také utajit. Jednoduše napište:

HIDE "jméno"

a soubor "jméno" se už v adresáři neobjeví. Příkaz HIDE OFF"jméno" udělá přesně to,co očekáváte (zruší utajení). Jako obvykle můžete využít hvězdičovou konvenci a volbu "ASK ME". Utajíte-li soubor příkazem HIDE, soubor je zároveň utajen i chráněn proti přepisu. Nezapomeňte na to! Soubor může být přepsán, nebo přemazán příkazy SAVE OVER, ERASE OVER nebo COPY OVER.Příkaz HIDE OFF způsobí, že soubor se objeví v adresáři, ale stále ještě bude chráněn proti přemazání jako po použití příkazu PROTECT.

RENAME (Přejmenování)

Příkaz RENAME může být využit ke změně jména souboru nebo jména disku.Soubory mohou být také přemístěny z jednoho podadresáře do druhého (viz SUBDIRECTORIES).

K přejmenování souboru příkazem RENAME používáme následující formu:

RENAME "starý" TO "nový"

Použijme RENAME k přejmenování souboru " Circles " na soubor " Example 1 ":

RENAME " CIRCLES " TO " EXAMPLE 1 "

Podívejte se zpět do adresáře,abyste si ověřili,že tato změna byla skutečně provedena. Jestli že byste měli na disku soubor "EXAMPLE 1", uviděli by jste zprávu " File name used " (Použité jméno souboru) a příkaz RENAME by nebyl akceptován.

K přejmenování disku v aktuálním drajvu příkazem RENAME použijeme následující formuli:

RENAME TO "jméno"

Vyzkoušejte si to a potom proveďte příkaz DIR. Uvidíte, že "jméno" se zobrazí na vrcholu adresáře, v místě rezervovaném pro jméno disku. Pojmenování disku je užitečná věc k tomu, abyste si připoměli, jak máte setříděné soubory, které jste uložili na disk, a pro pojmenování výpisů adresářů, abyste věděli, ze kterých disků pochází. Také je dobré si jméno disku napsat na štítek na disketě.

RENAME s mnohonásobnými soubory

K přejmenování několika souborů najednou můžete využít hvězdičkovou konvenci. Můžete použít, řekněme " * " jako první jméno v příkazu RENAME k přejmenování všech souborů, nebo " SNAP " k přesnému přejmenování snímkových souborů. Druhé jméno vyžaduje trochu přemýšlení, protože samozřejmě nemůžeme použít např."testing", neboť DOS by zkoušel pojmenovat všechny soubory stejným jménem! (Nebojte se -bude Vám sděleno "File name used" - použité jméno souboru - pokud to přeci jen náhodou uděláte.) Mnohem rozumnější příklad by byl:

RENAME "SNAP*" TO "GAME???"

Také by byly přjmenovány všechny snímkové soubory, ale pátý šestý a sedmý znak (které jsou reprezentovány ve druhém jménu otazníky) budou stejné jako byly v originálních jménech souborů

Můžete také použít volbu "ASK ME" takže můžete rozhodnout, které soubory přejmenovat - v tomto případě dopňte za druhé jméno otazník. Příkaz RENAME může být velice užitečný v podadresářích - viz SUBDIRECTORIES.

FORMAT (Formátuj)

Příkaz FORMAT připravuje disk zápisem volných sektorů a kontroluje jej aby z něj bylo možno zpětně čist. (Je-li příkaz opakovaně neúspěšný, je povrch disku pravděpodobně poškozen). Nejjednodušší formou příkazu FORMAT je FORMAT"d". Tento příkaz formátuje disketu v aktuálním disku. FORMAT "d1" nebo "d2" formátuje disketu v drajvu číslo 1 nebo 2. Můžete specifikovat, kolik stop má být využito k uložení diskového adresáře a tímto určit, kolik souborů bude možno na disk uložit. Nebudete-li tuto hodnotu specifikovat, bude předpokládáno uložení maximálně 80 souborů a pro adresář budou rezervovány 4 stopy. Zde jsou některé příklady:

FORMAT "d1"
FORMAT "d1",4 - oba příkazy umožňují uložit max. 80 souborů a na disku je z 800k k dispozici pro soubory 780k
FORMAT "d1",5 - dovoluje až 98 souborů, k dispozici je 775k
FORMAT "d1",10 - dovoluje až 198 souborů, k dispozici je 750k
FORMAT "d1",39 - dovoluje až 778 souborů, k dispozici je 605

Každá stopa , na které je uložen adresář, obsahuje 20 záznamů o souborech, vyjímku tvoří pátá stopa, která jich obsahuje pouze 18, protože část této stopy je rezervována pro DOS. Maximální počet stop, které můžete přidělit adresáři je 39 a minimální počet jsu 4, vyjímku tvoří RAM disky, u kterých je povolena minimálně jedna stopa. Formát disku užívaný Master DOSem je nepatrně odlišný od formátu SAM DOSu; tento umožňuje přenos dat na disk a z disku o 10% rychleji. (Diskety formátované Master DOSem mohou být použity SAM DOSem, pokud adresář nebude delší než čtyři stopy.)

I při zapisování prázdných sektorů na disk zapisuje Master DOS některé informace o disku do prvního sektoru. Tento sektor obsahuje ve dvou bajtech náhodné číslo, které umožňuje aby byly disky odlišeny,a jméno disku bude zobrazeno nad adresářem. Jestliže nespecifikujete jméno disku,bude použito jméno "Master DOS" Chcete-li použít jiné jméno, udělejte například toto:

FORMAT "NUMBER 6" nebo FORMAT "d2:ALAN'S"

RAM DISKY

RAM disk je sekce počítačové paměti (RAM),která pracuje jako disková jednotka (Bylo-li uvedeno dříve, nazýval jsem je RAM drajvy, ale nemějte obavy, je to to samé). Jejich hlavní výhodou je, že jsou mnohem rychlejší než reálné diskové jedotky. To kompenzuje tu nevýhodu, že okamžikem, kdy vypnete počítač, ztratíte z nich všechny soubory. Obvyklá metoda práce s nimi je, že se do nich z reálných disků překopírují ty soubory, které používáte nej častěji. Potom mohou být používány jako velice rychlé reálné disky. Na konci programování, dříve než vypnete počítač, nezapomeňte uložit zpět na reálný disk ty soubory, které byly změněny.

Další výhodou RAM disku je, že máte dodatečnou diskovou jednotku, pokud ji potřebujete. Má-li Váš počítač pouze jednu disketovou jednotku, pak se mnoho operací kopírování souborů stává nudnou záležitostí z důvodů neustálých výměn disket. Tyto operace mohou být do značné míry redukovány vytvořením RAM disku pro dočasné uložení těchto souborů. Ve skutečnosti můžete mít těchto

RAM disků až 5, každý z nich může mít rozsah od 5 do 780K. Jedinou omezující podmínkou je dostupná paměť, máte-li přídavný paměťový modul, bude automaticky využíván pro RAM disky předtím, než pro ně bude používána hlavní paměť (instalovaná v počítači).

RAM disky jsou číslovány od 3 do 7 a většinou jsou využívány právě jako reálné drajvy, např: DIR 3 nebo SAVE "d3:test" nebo DEVICE d5:LOAD "xyz" nebo OPEN#4;"d3:seriál". Samozřejmě,že před použitím těchto RAM disků, je musíte naformátovat - a tento příkaz (FORMAT) má pro RAM disky drobné odlišnosti.

Nastavení RAM disků

K naformátování RAM disku použijte následující formuli:

FORMAT "d3",D,T

D - je počet stop, které chcete přidělit pro adresář
T - je celkový počet stop, které chcete mít na disketě.

Stejně jako reálný disk i zde lze uložit na každou stopu adresáře 20 záznamů o souborech. Na ostatní stopy může být uloženo 5K dat na stopu. Musíte určit nejméně 1 stopu pro adresář a jednu přídavnou stopu (pro data), takže nejmenší RAM disk je dán příkazem FORMAT "d3",1,2

Ve skutečnosti požaduje DOS pro RAM disk přidělení celé 16K stránky, takže byste měli raději použít FORMAT "d3",1,3 protože tento příkaz požaduje právě jednu stránku. Každá stránka může zabrat 3,1 stopy.

Funkce FPAGES

Často budete chtít, aby byl RAM disk tak velký,jak jen to je možné.Jak velký může být? Běžně má 256k Coupé k dispozici 9 volných 16k stránek, zatím co 512k počítač jich má k dispozici 25. Každý vnější paměťový modul přidává dalších 64k stránek po 16k. (Z rezervovaných 64 stránek jsou obyčejně 4 přiděleny pro BASIC, 2 pro obrazovku a 1 pro DOS).Samozřejmě počet volných stránek se bude měnit podle toho, kolik je nastaveno obrazovek a RAM disků, nebo podle množství programů natažených do počítače. Přístupné stránky budou narůstat, pokud bude omezováno přidělení místa pro BASIC např: CLEAR 32767: OPEN TO 1. Nejjednodušší způsob jak to zvládnout je použít funkci FPAGES.Zkuste teď napsat PRINT FPAGES Není to zrovna dobrý nápad použít každou stránku pro RAM disky, protože COPY a BACKUP potřebují při své práci nejméně 1 stránku pro přechodné uchování dat. Pro využití všech stránek s vyjímkou jedné by jste mohli použít:

```
10 LET tks = INT ( ( FPAGES - 1 ) * 3.1 )
20 FORMAT "d3",1,tks
```

Toto nám dává obvykle 24 stop pro RAM disk na 256K počítač nebo 74 stop na 512K počítač. To nám umožní uložit 115K respektive 365K dat. Po využití paměti určené pro BASIC můžete tento počet rozšířit na 34 stop (165K) nebo 83 stop (410K). S vnější paměťovou jednotkou budete muset mít na paměti, že celkový počet stop je 160 nebo méně, protože kapacita každého RAM disku nemůže být větší než kapacita reálného 80-ti stopého oboustranného disku. Rovněž tak počet stop pro data musí být 156 nebo méně. Vhodné je použít FORMAT "d3",4,160 protože to je obvyklý formát reálného disku.

Můžete změnit velikost RAM disku opětovným použitím FORMATu. Soubory na RAM disku budou zničeny. K uvolnění celé paměti použijte FORMAT "d3"0.

Soubory mohou být přeneseny do RAM disku jednoduše jejich natažením příkazem LOAD z reálného disku a potom užitím SAVE"d3: name", ale použití příkazu COPY je pravděpodobně jednodušší: COPY "d1" TO "d3" zkopíruje všechny soubory do RAM disku č.3,pokud je v něm místo. Hvězdičkovou konvenci můžete použít, ale pouze k výběru přenášených dat. Další způsob je použití příkazu BACKUP "d1" TO "d3", který bude pracovat v případě, že všechny sektory použité ve zdrojovém disku, jsou přístupné i na RAM disku, jinak obdržíte zprávu o chybě.

Pokud máte sadu souborů na Vašem RAM disku, můžete zkusit použít tyto soubory, jakoby byly na drajvu č.1. Mohli byste udělat DIR 3 a potom natáhnout soubor podle čísla příkazem LOAD, nebo použít příkaz LOAD "d3:jméno", nebo příkazy DEVICE d3: LOAD "jméno". Celkem snadno může vyvstat problém, že program, který natáhnete může obsahovat řádek jako DIR1 nebo LOAD "d1;udgs"CODE který vede zpět k užívání pomalého starého disku č.1. To, co bychom obyčejně chtěli sdělit DOSu je,aby použil disk 3 kdykoli je zmínka o disku č.1 - a to můžeme udělat následujícím způsobem.

Master DOS si vede tabulku o tom,který disk je právě aktuální, když je volán drajv. Tato tabulka obyčejně obsahuje čísla od 1 do 7, takže např: DIR 1 ukazuje v tabulce na posici 1, čte 1 a používá drajv 1. Ale pokud je tabulka změněna tak, aby na této posici byla 3, bude samozějmě použit drajv č.3. (Případně můžete "přehodit" drajvy č.1 a 2 tak, že tabulka bude obsahovat 2,1,3,4 5,6,7!). Tabulka je uložena v DVAR 111 až 117, takže přeneste Vaše soubory na disk č.3, potom proveďte POKE DVAR 111,3. Nyní pracuje drajvč.3 jako kdyby měl č.1. Rovněž tak klávesa F9 natahuje soubory AUTO z drajvu č.3. K návratu do normálního stavu napište POKE DVAR 111,1.

SUBDIRECTORIES (podadresář)

Pokud budete používat počítač s diskem jen chvíli, budete pravděpodobně shledávat, že máte sklon k akumulování disket s velkým množstvím krátkých a jednoduchých programů. Dokonce i prohlédnutí adresáře obsahujícího 80 souborů je problém, natož pak jsou-li těchto souborů stovky. Samozřejmě můžete použít více disket, ale to je plýtvání a pro Vás to znamená hledat správný disk. Vysvobozením je rozdělení souborů do podadresářů. Vytvoření podadresáře je skoro jako tvorba nového disku bez disku. Abyjste viděli co s tím myslím, vezměte nový disk a nahrajte na něj nějaké soubory. Potom vytvořte podadresář tímto způsobem:

OPEN DIR "sub 1"

Samozřejmě, jméno může být takové, jaké se Vám bude líbit, jako je to u jména souboru. Uděláte-li nyní DIR 1, uvidíte v obsahu souborů "sub 1", napsáním DIR si ukážeme, že je to jméno podadresáře.

Ze "sub 1" můžeme udělat běžný (aktuální) adresář napsáním: DIR "sub 1"

Teď udělejte DIR a na vrchu obrazovky uvidíe něco podobného tomuto: Master DOS:\sub 1. "sub 1" prozrazuje jméno podadresáře. Budete vidět, že v adresáři zatím nejsou soubory,takže jich běžným způsobem několik nahrajte. Volné stopy a místo na disku se zmenší,ale povšiměte si, že tento podadresář může zaplnit všechny volné stopy a místo na disku, pokud to bude třeba. Nyní vytvořte pod-podadresář řekněme OPEN DIR "games". Potom použijte DIR="games" k aktualizaci tohoto podadresáře. DIR nám nyní ukáže jméno podadresáře takto: 1:\sub 1\games, což znamená, že "games" je podadresář "sub 1",který je zase podadresářem hlavního adresáře. Můžeme pokračovat v tvorbě adresářů uvnitř adresářů dokud nebudou využity všechny volné stopy nebo dokud nebude vytvořeno 254 podadresářů. Každý adresář může obsahovat několika násobné podadresáře.

Hlavní adresář je obvykle nazýván "kmen" (nebo také "kořen"), protože někdo může vidět v uspořádání podadresářů strom, kde hlavní adresář tvoří kmen větvící se do podadresářů, které se dále větví tak dlouho, dokud neskončí u větve, která už nemá další výhonek podadresáře nebo souboru. (Jediný problém, který v této analogii vyvstává je, že říkáme-li "jdeme hlouběji" myslíme tím, že jdeme do podadresáře dále od kmene-takže strom by musel růst obráceně.

Ale nyní se vraťte zpět ke kmeni. Existuje několik způsobů, jak to udělat. Můžeme se vrátit zpět cestou,kterou jsme k podadresáři došli použitím příkazu DIR=" "(Symbol SHIFT - H), který nás posune o jednu úroveň výše do podadresáře "sub 1" a potom se stejnou cestou vrátit ke kmeni. Nebo to můžeme provést najednou

použitím DIR="\", což znamená "udělej kmen aktuálním adresářem" a který nás vrátí ke kmeni z kteréhokoliv místa ve stromu.

Tyto dva způsoby jsou užívány docela často. Abyste si zjednodušili psaní můžete vynechat úvozovky,příkaz DIR=/ bude pracovat i bez nich. Měli by jste si všimnout, že v případě bez úvozovek je lomítko obrácené. Master DOS zdařile akceptuje jiný způsob.

Jsme-li zpět u kmene, můžeme experimentovat s některými odlišnými způsoby příkazů SAVE a LOAD. Vyzkoušejte: SAVE"sub1/test file1". Půjdete-li zpět do adresáře "sub1" (DIR="sub1") a provedete DIR, zjistíte, že soubor byl nahrán sem. Rovněž tak můžeme soubor nahrát do pod-podadresáře použitím např: SAVE"sub1/games/ /file-name". String říká příkazu SAVE kam soubor uložit a jak ho nazvat. Má specielní jméno-nazývá se "path" (cesta). Pouze poslední část tohoto stringu je aktuální jméno souboru, zbytek určuje cestu adresářem "stromem" a nepočítá se do desetiznakového omezení jména souboru.

Normálně začíná cesta od aktuálního adresáře, ale můžete začít od hlavnío adresáře, pokud bude prvním znakem v popisu cesty lomítko \.Např.:SAVE"\subdr2\fred" nahraje soubor nazvaný "fred" do podadresáře "subdr2", který je podadresářem kmenového adresáře, ať je aktuální adresář kterýkoliv.Bez prvního lomítka by byl "subdr2"podadresářem právě aktuáního adresáře. Můžete také použít " " na místě prvního znaku, to Vám umožní vrátit se do předešlého podadresáře před tím, než se použije zbytek cesty.To Vám dovolí přístp k podadresáři, který je na stejné větvi jako podadresář aktuální-pokud to budete chtít (obvykle myslím, že ne).

DIR a podadresář

Použitím např. DIR "sub1/*" můžete zobrazit všechny soubory v podadresáři DIR? bude zobrazovat seznam všech souborů v každém podadresáři tak jako DIR1?. Čísla souborů v levém sloupci mohou být vždy použity k natažení souborů příkazem LOAD ať jste v jakémkoliv adresáři.

ERASE a podadresáře

Příkaz ERASE můžete použít tak, jak jste mohli předpokládat. Například: ERASE "sub1/*" vymaže všechny soubory v podadresáři "sub1" a příkaz ERASE "games\bad game" smaže soubor "bad game" z podadresáře "games". Běžná forma ERASE ignoruje DIR-typ souborů, ale takový typ souborů můžete smazat použitím např:

ERASE DIR "sub1"

Tento příkaz ale bude pracovat jen tehdy, je-li adresář zcela prázdný; jestli že není, obdržíte zprávu " Directory not empty " (Adresář není prázdný)

COPY, PROTECT, HIDE a podadresář

Jejich použití je všeobecně stejné, ale pro ilustraci několik příkladů:

```
COPY "games/lotto" TO "d2:games/*"
PROTECT "/sub1/imp.let"
HIDE "sub1/st*"
```

RENAME a podadresáře

RENAME je rychlý způsob přenosu souborů z jednoho podadresáře do jiného. Máte-li disk s několika soubory a chcete-li umístit některé z nich do podadresáře, vyzkoušejte

```
OPEN DIR "some name"
RENAME "*" TO "some name\*"?
```

Stisknete-li "Y", když jste otázáni,soubor bude pojmenován stejným jménem, ale bude v podadresáři "some name" a zmizí z hlavního adresáře. Vyzkoušejte DIR "some name*". Můžete použít i mnohem komplikovanější cesty jako např:

```
RENAME "\sub2\games\frogger" TO "\junk\hop"
```

Běžná forma příkazu RENAME se automaticky vyvaruje přejmenování kteréhokoliv nalezeného jména podadresáře. Pokud byste chtěli přejmenovat podadresář, musíte použít speciální tvar příkazu RENAME, který komunikuje pouze s podadresáři. Tento tvar je následující:

```
RENAME DIR "old name" TO "new name"
```

RENAME DIR můžete také použít s určením cesty (path).
RENAME DIR "old name" TO "subd1/*" odpovídá v naší stromové analogii adresáře odříznutí větve se všemi jejími výhonky a jejímu naroubování na jiné místo stromu. Adresář "old name" a všechny jeho soubory budou nyní přístupné pouze přes podadresář "subd1". Vyvarujte se použití takových příkazů jako je např:RENAME "fred" TO "fred/xyz"-to způsobí že podadresář "fred" bude dostupný pou-

ze přes podadresář "fred"... To je jako naroubování větve do nekonečné smyčky, nepřipojení ke stromu.

Funkce PATH$

Tato funkce dává zprávu o jménu aktuálního podadresáře a kde je umístěn za jménem disku v adresáři. Např: jste-li v hlavním adesáři PRINT PATH$ zobrazí "1:" nebo "2:" nebo "3:".Napíšete-li DIR="sub1", funkce PATH$ zobrazí "1:\sub1" a je možný i sožitější příklad jako např: "2:\sub1\games\chess\chdata". Maximální dosažitelná délka je 38 znaků; a všechny další znaky jsou odříznuty (a nejsou vytištěny v adresáři), ačkoliv se můžete nacházet v hlubší úrovni adresáře než může být zobrazeno.

Použití hodin a kalendáře

Hodiny a kalendář s bateriovým zálohováním jsou umístěny na základní desce od SAMCO. Příkazy DATE a TIME a funkce DATE$ a TIME$ Vám umožňují nastavit a číst datum a čas. Soubory jsou označeny datem a časem v okamžiku uložení na disk.

DATE a DATE$ (příkaz DATE a funkce DATE$)

Příkaz DATE má dva významy: dovoluje Vám nastavit datum na hodinách vestavěných do Coupé a pro Vaši informaci jej tiskne. I když nemáte vestavěné hodiny může být tento příkaz užitečný. K nastavení data doplňte příkaz DATE stringem obsahujícím 6 čísel např: DATE"010391" nastaví datum na 1.března 1991. Do tohoto stringu, pokud chcete zlepšit čitelnost, můžete vložit i nečíselné znaky - takže je povolen i tvar "01/03/91". Den v měsíci musí být mezi 1 - 31, měsíc musí být mezi 1 - 12 a rok 0 - 99, jinak obdržíte chybovou zprávu "Integer out of range"(číslo mimo rozsah).

Napsáním DATE bude vytisknuto aktuální datum ve tvaru dd/mm/yy - den,měsíc,rok. Pro mnohem pružnější metodu užívání aktuálního data v programu můžete použít funkci DATE$, která vrací 8mi znakový string obsahující datum. Např:

```
PRINT DATE$
PRINT "Month=";DATE$(4 TO 5)
```

Možná by Vás potěšilo napsání programu,který by např. z 03/07/91 funkcí DATE$ vygeneroal něco jako 3.července 1991

Máte-li vestavěné hodiny, potom budou všechny soubory, které vytvoříte označeny aktuálním datem a hodinou.Tato informace může být zobrazena příkazem DIR DATE (pro podrobnosti viz DIR). Nemáte-li vestavěné hodiny, datum bude začínat v 00/00/00, pokud je

nevestavíte. Takto neobsazená hodnota sděluje DOSu, aby se netrápil označováním vytvořených souborů datem a hodinou.Samozřejmě pokud chcete, můžete po každém zapnutí počítače nastavit datum pro řádné označení souboru datem.Nejprve proveďte POKE DVAR 150, 0 abyste sdělili DOSu, aby nezkoušel používat neexistujících hardware (normální hodnota této proměnné je 239, což je adresa portu hodin)

TIME a TIME$ (Příkaz TIME a proměná TIME$)

Příkaz TIME má dva významy: Dovoluje Vám nastavit čas na vestavěných hodinách a jeho tisk pro Vaši informaci. K nastavení času doplňte příkaz TIME stringem obsahujícím max.6 čísel, např: TIME "11:30". Doplníte-li méně než 6 čísel, je za poslední dosazena 0. Stejně jako v DATE i zde můžete použít oddělovací znaky, ale nejsou potřebné. Hodiny musí být v rozmezí 0 - 23 a minuty a sekundy 0-59, jinak obdržíte chybnou zprávu "Integer out of range" (číslo mimo rozsah). Je použito 24 hodinové čítání, takže např.: 2:00 odpoledne je vloženo jako "14:00"

Napsáním TIME bude zobrazen aktuální čas ve tvaru hh:mm:ss Pro pružnější metodu užití aktuálního času v programu můžete použít funkci TIME$, která zobrazuje 8-mi znakový string obsahující čas. Např.:

```
DO:PRINT AT 10,10;TIME$:LOOP
```

(čas se nebude měnit, pokud nemáte desku počítače s hodinami)

Adresář a funkce souboru

Funkce DIR$

Tato funkce sděluje všechna jména souborů v aktuálním adresáři jako string. Každé jméno zabírá ve stringu 10 znaků - kratší jména jsou doplněna mezerami. Jména za sebou následují v pořadí, v jakém je můžete vidět v podrobném adresáři. PRINT DIR$ vám dá adresář, ale tento není tak úhledně srovnaný jako při napsání DIR, takže k čemu to vlastně je? Tato funkce je vytvořena proto aby byla použita během programu na pomoc při manipulaci se soubory. Jméno každého souboru může být získáno programem podobným tomuto:

```
10 LET list$ = DIR$
20 FOR p = 1 TO LEN list$ STEP 10
30 LET nm$ = list$ (p TO p+9)
40 REM
50 NEXT p
```

Vhodné může bť vytvoření desetiznakového okénka v obrazovce

```
10 WINDOW 0,9,0,18:CLS 1
20 PRINT DIR$:WINDOW
```

Můžete také společně s proměnnou DIR$ použít hvězdičkovou konvenci. Například:

```
PRINT DIR$ ("let*")
```

Funkce DSTAT (drajv, N)

DSTAT je zkratka pro Disk STATus. Tato funkce může být využita k tomu, abyste se dozvěděli množství užitečných věcí o disku ve kterémkoliv drajvu. Např.:

```
PRINT DSTAT (1,1) Vám sdělí, kolik volného místa je na disku
                  v drajvu číslo 1
```

Můžete specifikovat číslo drajvu od d1 do d7, protože funkce pracuje stejně tak s RAM disky jako s reálnými diskovými jednotkami. Můžete také použít hvězdičku k označení aktuálního disku, takže:

```
PRINT DSTAT (*,1) Sdělí kolik volného místa je na  aktuálním
                  disku
```

Zde je kompletní seznam informací poskytovaných pro různé hodnoty druhého čísla (parametru N):

1. Sděluje množství použitelného místa na disku v byte. (doslovně 510 krát počet volných sektorů mínus 9 byte, které by mohly reservovány pro hlavičku při ukládání souboru). Není-li v adresáři místo (t.j. počet volných stop = 0), nebo je-li disk chráněn proti zápisu obdržíte hodnotu 0, když na disku volné místo je. Hodnota -1 sděluje, že v jednotce není disketa, nebo že RAM disk nebyl naformátován nebo není připojena disketová jednotka. To platí pro všechny hodnoty N.

2. Sdělí 1 pokud je disk chráněn proti přepisu a 0 je-li disk v jednotce a není-li chráněn proti přepisu. Pozn.:RAM disky nejsou nikdy chráněny proti přepisu.

3. Sdělí množství volného místa na disku v byte, stejně jako v bodu 1,ale bez toho aby rozlišil je-li disk chráněn proti přepisu, nebo je-li místo v adresáři.

4. Sdělí počet volných stop v adresáři pro jména souborů.

5. Sdělí celkový počet souborů na disku

6. Sdělí počet souborů v aktuálním adresáři. Bude stejný jako v bodě 5 jestliže nebude vytvořen podadresář. (viz podadresáře)

7. Sdělí počet stop použitých pro adresář.

8. Sdělí číslo aktuálního disku

Funkce FSTAT ("file name,N)

FSTAT je zkratka pro File STATus. Tato funkce poskytuje užitečné informace o specifikovaných souborech. Například:

PRINT FSTAT ("test",2) sdělí délku souboru "test".

Zde je kompletní přehled informací, poskytovaný při různých hodnotách čísla N:

1. Sděluje číslo specifikovaného souboru v adresáři. Je to číslo které můžete vidět těsně vedle souboru v podrobném adresáři. Pokud soubor neexistuje, je funkcí sdělena 0. Je to užitečné pro kontrolu, zda soubor existuje, před operacemi SAVE nebo LOAD. Pokud jakákoliv hodnota N sdělí -1, není disk v jednotce

2. Sděluje délku souboru v byte. Některé soubory mají "hlavičku" o délce 9 byte,která není zahrnuta do této délky.

3. Sděluje typ souboru podle následujících kódů:

1 = ZX BASIC
2 = ZX číselné pole
3 = ZX stringové pole
4 = ZX CODE (strojový kód)
5 = ZX SNP 128K (snímek)
6 = ZX Microdrive File (soubor mikrodrajvu)
7 = ZX SCREEN$
8 = Speciál
9 = ZX SNP 128K
10 = OPENTPE (otevřený typ)
11 = ZX EXECUTE (výkonný)
16 = SAM BASIC
17 = SAM číselné pole
18 = SAM stringové pole
19 = SAM CODE
20 = SAM SCREEN$
21 = SAM podadresář

4. Sděluje typ souboru jako 3, ale navíc přičítá 64, je-li soubor chráněn proti přepisu a 128, je-li soubor utajen

Tvorba sériových souborů

K tvorbě sériového souboru musíte nejdříve ze všeho otevřít soubor pro výstup příkazem OPEN, specifikovat číslo kanálu a jméno souboru, například

```
10 OPEN#4;"testfile" OUT
```

Diskový operační systém bude kontrolovat, zda soubor "testfile" již existuje na aktuálním disku a zeptá se Vás, zda-li budete chtít přepsat starou kopii bude-li nalezena. Můžete specifikovat jiný disk jako obvykle užitím jména a začínajícím "d1:", "d2:", "d3:" atd. Kanál 4 bude přidělen souboru, pokud již není použit. (Můžete použít jakékoliv číslo kanálu od 4 do 15).

Od této chvíle bude příkaz PRINT doplněný "#4" tisknout do diskového souboru, častěji než na obrazovku. Například.:

```
20 FOR n = 1 TO 50
30 PRINT n; "abcdefghi"
40 PRINT #4;n;"abcdefghi"
50 NEXT n
```

Můžete vidět co je posíláno na diskový soubor, protože řádek 30 tiskne kopii na obrazovku. Soubor bude obsahovat 50 stringů, od " 1abcdefghi " do =50abcdefghi". V mnoha aplikacích se mohou tyto stringy vztahovat k záznamům.

Kdyby měl být každý jednotlivý znak umísťován na disk zvlášť psaní souboru by trvalo velmi dlouho. Proto jsou znaky akumulovány ve zvláštní vyrovnávací paměti, dokud jich není tolik, aby zaplnily celý sektor na disku. Potom je aktivována disková jednotka (pokud je to třeba) a celý sektor je zapsán najednou. Podobná vyrovnávací paměť je použita když je soubor z disku čten. Coupé ukládá do každého sektoru 510 znaků dat. Dvakrát tolik byte je použito pro DOS, což je celkem 512 byte na sektor. Použití vyrovnávací paměti znamená,že program výše uvedený je nekompletní. Když je ukončena smyčka FOR - NEXT, vyrovnávací paměť bude jen částečně naplněna a když nic dalšího neuděláte, informace v ní budou navždy ztraceny. Proto je třeba soubor uzavřít.

```
60 CLOSE #4
70 STOP
```

Tím bude obsah vyrovnávací paměti zapsán na disk a také bude vytvořen záznam do diskového katalogu.

Čtení sériového souboru

Máte-li vytvořen sériový soubor tak, jak je popsáno výše,budeme jej nyní číst. Soubor musí být opět otevřen příkazem OPEN, ale nyní s klíčovým slovem IN. Číslo kanálu, které jste specifikovali v příkazu OPEN může být později použito ke čtení dat ze souboru použitím buď INPUT # nebo INKEY$ # nebo INP$.

```
100 OPEN #4;"testfile" IN
110 INPUT #4; a$
120 PRINT a$
130 GOTO 110
```

Příklad zobrazí obsah souboru, zastaví se a vydá zprávu "END of file" (konec souboru). K uzavření souboru je třeba napsat:

```
CLOSE #4
```

Ačkoliv uzavření vstupního souboru není tak podstatné jako uzavření výstupního souboru,výstupní soubor využívá vyrovnávací paměť stejně jako soubor vstupní a proto nemůže bý znovu použita bez uzavření příkazem CLOSE. Kromě toho je třeba kanál uzavřít pokud bude opět použit.

Máte-li otevřen soubor na disku, není dobré v jednotce měnit tento disk za jiný. Vstupy ze souboru jiného disku budou dávat nesmysly a ukládání na jiný disk, než na kterém byl otevřen daný soubor, je ten nejlepší způsob vedoucí k porušení souborů na tomto disku.

Je možné vložit do jednotky jiný disk dočasně a použít DIR, nebo LOAD. Pokud používáte diskety formátované Master DOSem, Coupé bude vědět, že disketa byla vyměněna a upozorní Vás na to píp-nutím a zprávou "OPEN file" (otevřený soubor) ve spodní části obrazovky, kdykoliv použijete DIR, LOAD nebo SAVE s jiným diskem než původním (na kterém byl otevřen soubor).

Jestliže ukládáte na jiný disk příkazem SAVE, využití dostupného místa nebude příliš účinné buď pro ukládání souboru příkazem SAVE nebo pozdější zapisování do otevřeného souboru příkazem PRINT (pokud jste do jednotky opět vložili původní disketu).

Kterýkoliv otevřený soubor můžete opustit (s možnou ztrátou dat, pokud budete psát do souboru) použitím CLEAR # MOVE (viz později) používá dočasné OPENTYPE soubory a tak CLEAR # může potřebovat použít, pokud je vykonávání příkazu MOVE přerušeno. Pokud by Vás to zajímalo, tak příkaz PRINT PEEK DVAR 20 Vám sdělí počet otevřených souborů.

Každý příkaz INPUT čte jeden ze stringů původně uložených do souboru příkazem PRINT a možná jste zvědavi, jak je to uděláno - - jinými slovy - jak DOS ví, kdy string končí? Odpověď zní. Každý ze stringů je "ukončen" speciálním znakem CHR$ 13, který je nazýván " carriage return" (návrat vozíku) - jméno pochází z dob dálnopisů. Když vložite příkaz podobný tomuto;

```
PRNT "one": PRINT "two"
```

Coupé ve skutečnosti odešle "o","n","e",CHR$ 13,"t","w","o",CHR$ 13 do podprogramu ROM, který normální písmena vytiskne na obrazovku, ale na CHR$ 13 odpoví přesunutím tisku na další řádek. PRINT do diskového souboru je podobný, ale znaky CHR$ 13 jsou na disk běžně ukládány, aniž by na ně bylo nějak reagováno. Když pak přijde na řadu INPUT ze souboru, DOS čte znaky ze kteréhokoliv místa souboru dokud nenajde CHR$ 13. Ten potom přidělí tyto znaky proměnné specifikované v příkazu INPUT (CHR$ 13 sám je oddělen).

Funkce INKEY$

INKEY$ může být použita ke čtení jednotlivých znaků diskového souboru. Na rozdíl od INKEY$ z klávesnice, dává INKEY$ z disku znak vždy. Zkuste výše uvedený řádek č.110 vyměnit za

```
110 LET a$ = INKEY$ #4
```

Podotýkám, že také musíte přidat nový řádek:

```
90 CLOSE #4
```

Tím předejdeme chybě " Stream used " (použitý kanál) a je vhodné na to nezapomínat. Nyní vyzkoušejte program příkazem RUN 90 nejprve s existujícím řádkem 120 a potom s následujícími variantami

```
120 PRINT a$;
```

nebo

```
120 PRINT a$; " ";CODE a$
```

Poslední verze nám jasně ukáže všechny znaky CHR$ 13. Můžete vytvářet další soubory, které budou obsahovat " control codes " (ovládací - kontrolní kódy)(které působí změny) jiné než CHR$ 13 Například k tabelaci obrazového výstupu (tiskovou čárkou) odešlete do diskového souboru znaky CHR$ 6. Také PEN, PAPER, TAB atd představují speciální znakovou řadu.

INPUT LINE

Jestliže některé stringy,které mají být z disku načteny příkazem INPUT, obsahují úvozovky - např: "Bide-a-wee"-název domu v adrese, budete potřebovat použít INPUT LINE, stejně jako byste jej použili při ukládání z klávesnice. Např.:

```
INPUT #4; LINE a$
```

Něco více o otevírání a zavírání souborů

Je možné otevřít soubor příkazem OPEN bez specifikování IN, nebo OUT,v tomto případě DOS předpokládá, že tím míníte OUT, pokud soubor ještě není v adresáři a předpokládá, že tím míníte IN pokud již v adresáři je. Je-li soubor chráněn proti přepisu příkazem PROTECT, DOS zabrání tomu, aby jste psali do souboru užitím IN, ať už specifikujete cokoliv.

Ačkoliv je obvyklé používat OPEN s IN nebo RND v souborech typu OPENTYPE,skutečně můžete každý soubor otevřít příkazem OPEN tímto způsobem, ačkoliv případy, kdy tento způsob můžete uplatnit, jsou dost exotické.(Změna snímku bez jeho natažení?) Některé typy souborů (BASIC, CODE, pole a SCREEN$) mají hlavičku o délce 9 byte před hlavní částí souboru.

Totéž platí pro uzavírání jednotlivých kanálů. Příkaz CLOSE Vám dovoluje uzavřít všechny otevřené soubory tohoto tvaru:

```
CLOSE *
```

CLEAR # vyprázdní všechny otevřené soubory, ale bez jejich uzavření (CLOSE). Není to důležité u souborů IN, ale soubory OUT budou ztraceny. Soubory RND budou ztraceny, pokud jsou nové,nebo mohou být částečně přepsány.

MOVE

Příkaz MOVE čte soubor, znak najednou a píše jej do dalšího souboru nebo kanálu. Ve skutečnosti k tomu využívá podobného postupu jako je střídání INKEY$ # a PRINT #, ale tento proces je pro uživatele neviditelný, a kanály a vyrovnávací paměť propojení s diskovými soubory jsou otevírány a zavírány DOSem automaticky. Například:

```
MOVE "testfile" TO "copyfile"
```

"testfile" musí existovat a "copyfile ne, pokud jej nechcete přepsat. Je to jednoduše pomalá metoda kopírování souboru a COPY "file TO "file" je pro dlouhé soubory mnohem rychlejší.Samozřejmě MOVE je mnohem pružnější. Například, máte-li dva soubory nazvané "první" a "druhý", můžete vytvořit jediný delší soubor,který bude obsahovat oba dva jako např. následující způsob:

```
10 OPEN #5;"combined" OUT
20 MOVE "first" TO #5
30 MOVE "second" TO #5
40 CLOSE #5
```

Můžete také přemístit příkazem MOVE na kanál 2, což je hlavní část obrazovky. Je to jednoduchý způsob jak prohlédnout soubor typu OPENTYPE a stejně tak Vám dovoluje prohlédnout programy v BASICu bez toho, aby jste je natáhli,i když nebudou tak úhledně vylistovány jako obvykle.Je to také dobrý způsob jak prohlédnout text, nebo data v programu ve strojovém kódu. Viz DOSVAR 24 pro podrobnosti. Příkazem MOVE můžete také přemístit soubory na kanál 3, což je tiskárna a náhodně přístupné soubory.(viz později).

RANDOM ACCESS FILES (Náhodně přístupné soubory)

Ačkoliv sériové soubory jsou velmi užitečné, mají velkou nevýhodu v tom, že k tomu,aby jste viděli část souboru,musíte přečíst všechna předcházející data.K tomu, aby jste změnili tu kterou část souboru, musíte přečíst a přepsat celý soubor. Pro některé aplikace to může být velice nevhodné, a tento problém je tím horší, čím je delší soubor.Master DOS umožňuje tvorbu náhodně přístupných souborů (RANDOM ACCESS). To znamená, že můžete

kontrolovat, nebo měnit jakoukoliv část souboru bez nutnosti celý soubor natáhnout do počítače. Jsou použity běžné soubory OPEN-TYPE ale jsou otevřeny příkazem OPEN a rozšířením RND; např

```
OPEN #5; d1"testx" RND
OPEN # strm; d2 "file name" RND
```

To Vám umožní zápis do souboru použitím PRINT # jako kdyby jste použili OUT a čtení použitím INPUT # nebo INKEY$ # nebo INP #, jako by byl použit IN. Skutečně můžete všechny IN a OUT v příkladech pro sériové soubory nahradit RND a programy budou pracovat jako předtím.

Tvorba jednoduchého souboru s náhodným přístupem

Ačkoliv příkazem OPEN můžete otevřít jakýkoliv existující soubor typu OPENTYPE použitím rozšíření RND, vysvětlení bude mnohem srozumitelnější, když si vytvoříme jednoduchý zkušební soubor, jako je uvedeno níže. Pro ty, kteří nenávidí psaní, bude jistě příjemné zjištění, že tento příklad je obsažen na jejich disku s Master DOSem. Natáhněte jej příkazem LOAD "prog1".

```
 10 CLOSE #4
 20 OPEN #4; "test" RND
 30 LET a$="abcdefghi"
 40 FOR n=1 TO 200
 50 LET a$( TO 3)=STR$ n
 60 PRINT AT 10,10;a$
 70 PRINT #4;a$
 80 NEXT n
 90 CLOSE #4
100 STOP
```

Nyní vytvořte soubor spuštěním programu příkazem RUN. První tři znaky každého stringu jsem použil k uložení čísla záznamu, abychom mohli snadněji kontrolovat, na který záznam se v našem pokusu právě díváme. Soubor bude obsahovat 200 stringů neboli záznamů, od "1defghi" do "200 defghi". Ačkoliv text každého ze stringů je 9 znaků dlouhý, každý zabere na disku 10 znaků, protože každý string je ukončen naším známým znakem CHR$ 13. Tyto stringy jsou příkladem záznamů s pevnou délkou (FIXED-LENGHT) a výhoda použití tohoto systému vyplyne později.

V našem příkladu je jednoduché spočítat znaky v a$ a zjistit jak dlouhý záznam vznikne, ale používáte-li delší stringy, je lepší použít DIM. Např:

```
25 DIM a$(99)
```

pak by reservoval pro každý string 99 znaků délky a 100 byte místa na disku. Tohle vše Vám bude jasné, pokud jste pochopili povídání o sériových souborech. Bude-li řádka 20 ukončena OUT, program bude pracovat úplně stejně. Diskový soubor, který jste vytvořili je běžný soubor typu OPENTYPE.

Čtení jednoduchého souboru s náhodným přístupem

Předpokládejme, že máte existující soubor typu OPENTYPE, můžete jej otevřít příkazem OPEN.... RND a manipulovat s ním, jako by bylo užito IN, číst jeden záznam za druhým. Když to děláte, pohybuje se vnitřní souborový ukazatel (FILE POINTER) po souboru tak jak je použit příkaz INPUT nebo INKEY$. Ten ukazuje na další znak, který má být čten. Například načítáte-li příkazem INPUT deseti znakový string, ukazatel se posune o 11 protože záznam končí znakem CHR$ 13 (soubory OUT mají podobný ukazatel, který ukazuje na poslední pozici zápisu, která je vždy na konci souboru)

Souborový ukazatel použitý Master DOSem začíná před čtením prvního znaku hodnotou 0, takže můžeme říci, že první znak je na pozici souboru 0. Před tím, než přečteme poslední znak souboru, bude mít ukazatel hodnotu o 1 menší než je délka souboru. Soubor si můžete představit jako řadu znaků číslovaných od 0 do (délka souboru -1). Jakmile je přečten poslední znak, hodnota ukazatele bude rovna délce souboru. Jakékoliv další čtení způsobí chybu "end-of-file" (konec souboru)

POINT (bod,pozice)

Použijte formát příkazu: POINT # kanál,pozice

Souborový ukazatel souboru připojeného k určenému kanálu je bezprostředně přesunut na specifikovanou pozici. (To je častý způsob jak natáhnout nový sektor z disku). Nyní vyzkoušejte následující program jeho spuštěním příkazem RUN 200.

```
200 CLOSE #4
210 OPEN #4; "test" RND
220 DO
230 INPUT "Record? " ;rec
240 EXIT IF rec=0
250 POINT #4, (rec-1) * 10
260 INPUT #4; a$
270 PRINT a$
280 LOOP
290 CLOSE #4
```

Tento program Vám umožní poznat výhody při použití záznamů (stringů) pevné délky - hodnota ukazatele pro kterýkoliv záznam může být jednoduše získána z čísla záznamu (stringu), což Vám umožňuje velice rychlý INPUT z kterehokoliv záznamu souboru.

Pokud je záznam, který požadujete, v aktuálním sektoru, bude získán velice rychle, ale i když je záznam na druhém konci 700 K souboru a disk stojí, může být záznam získán během 1 - 2 sekund. RAM disky jej dodají téměř okamžitě.

Poznámka: POINT s hodnotou menší než 1 způsobí chybu " integer out of range " (číslo mimo rozsah) a POINT s hodnotou větší, než délka souboru způsobí chybu "End of file" (konec souboru)

Změna souboru s náhodným přístupem.

Soubor typu OPENTYPE otevřený v módu RND může být zapisován stejně pružně jako čten. Ukazatel souboru indikuje pozici kam budou data zapisována i odkud budou čtena, takže POINT dovoluje změnit jakýkoliv záznam v souboru. Níže uvedený program nám ukazuje náhodné čtení i zapisování našeho spolehlivého souboru "test". Umožňuje poskytnout číslo souboru, takže vždy víme, se kterým záznamem jsme ve styku, jednoduchým způsobem z hodnoty souborového ukazatele, kterou jsme nastavili do použitého příkazu POINT: číslo záznamu = souborový ukazatel/délka jednoho záznamu (stringu) +1. Později uvidíte, jak číst hodnotu souborového ukazatele přímo. Teď natáhněte následující program příkazem LOAD "prog2".

```
 10 CLOSE #4
 20 OPEN #4; "test" RND
 30 DIM a$(9)
 40 PRINT "Read, Write or Exit? ( R/W/E)"
 50 LET c$=INKEY$
 60 IF c$= "W" OR c$= "w" THEN GOSUB 100
 70 IF c$= "R" OR c$= "r" THEN GOSUB 300
 80 IF c$<>"E" and c$<>"e" THEN GOTO 50
 90 CLOSE #4: STOP
100 INPUT "Record to write? "; r
110 IF r=0 THEN RETURN
120 POINT #4, (r-1)*10
130 INPUT #4; a$
140 PRINT "Old text:";a$
150 PRINT "New text"
160 INPUT n$
170 IF n$="" THEN GOTO 100
180 LET a$ ( 4 TO )=n$
190 POINT #4, (r-1)*10
```

```
200 PRINT #4;a$
210 GOTO 100
300 INPUT "Record to read? ";r
310 IF r=0 THEN RETURN
320 POINT #4, (r-1)*10
330 INPUT #4; a$
340 PRINT a$
350 GOTO 300
```

Doporučuji Vám, aby jste si s tímto programem trochu pohráli, četli si a zapisovali záznamy (stringy) celého soubory. Možná budete chtít přidat kontrolu proti použití čísla záznamu většího než je počet záznamů v souboru a možná, že si program rádi přepíšete celý, protože jsem jej napsal v ošklivém Spectrum BASICu. Vložte "record number" (číslo záznamu), nebo 0 k ukončení čtení nebo zapisování. (Ve skutečnosti můžete podprogram k zapisování na řádku číslo 100 poslouží stejně dobře i ke čtení. Pokud stisknete po vydání zprávy "New text?" (Nový text?) klávesu RETURN, záznam nebude změněn). Všimněte si, že příkaz POINT je použit na řádku číslo 120 k nastavení souborového ukazatele pro příkaz INPUT a potom opět na řádku 190 pro příkaz PRINT stejného záznamu, protože INPUT přesune ukazatel. Pokud jste ukončili práci, stiskněte klávesu "E" k opuštění a uzavřete soubor příkazem CLOSE. Disk může, nebo nemusí běžet, závisí to na tom, zda-li jste měnili aktuální sektor nebo ne.

Funkce souboru typu OPENTYPE

PTR - čtení souborového ukazatele

Stejně tak jako je možné přemístit souborový ukazatel použitím příkazu POINT, umožňuje Master DOS i čtení aktuální pozice ukazatele použitím funkce PTR (nahrajte níže uvedený program příkazem LOAD "rafprog". Funkce PTR (kanál) Vám sdělí pozici ukazatele pro ten soubor typu OPENTYPE, který je spojen se specifikovaným kanálem. Například:

```
10 OPEN #4; "test" RND
20 PRINT PTR 4;" ";
30 INPUT #4;a$: PRINT a$
40 GOTO 20
```

Tato funkce je nejužitečnější, jestliže soubor obsahuje stringy proměnné délky a souborový ukazatel je po každém příkazu INPUT posunut o jiný počet znaků. Můžete najít posici každého stringu a tento pak později získat zpět použitím příkazu POINT.

Funkce LENGTH - získání délky souboru

Zvětšení souboru

Často je užitečné znát přesnou délku souboru. Například: můžete snadněji rozšířit soubor typu OPENTYPE, pokud nastavíte souborový ukazatel na konec souboru použitím příkazu POINT předtím než použijete příkaz PRINT # pro přidání nových dat. K tomu aby jste to mohli udělat, potřebujete znát délku suboru a Master DOS Vám to umožňuje použitím funkce LENGTH. Například:

```
100 CLOSE #4
110 OPEN #4; "test" RND
120 POINT #4; LENGTH #4
130 FOR n=1 TO 10
140 PRINT #4; "extended"
150 PRINT LENGTH #4
160 NEXT n
170 CLOSE #4: STOP
```

Tento program připojí nějaká data na konec našeho už tolikrát použitého souboru test" a vytisknete aktuální délku souboru, jakmile je přidán každý string. Bude-li řádek č.120 vynechán,pak by byla první část souboru přepsána deseti stringy " extended ", protože souborový ukazatel by začínal na nule. Délka souboru by zůstala stejná.

Délka souboru je maximální hodnota, kterou můžete použít pro příkaz POINT, takže chcete-li rozšířit soubor, tak aby jste nevynechali žádný sektor, musíte příkazem POINT přesunout ukazatel na konec souboru a použít příkaz PRINT #, aby se zvětšila velikost souboru.

A nyní si přečtěme celý soubor, abychom si ověřili,že všechno je skutečně tak, jak jsme předpokládali:

```
200 CLOSE #4
210 OPEN #4; "test"
220 DO
230 INPUT #4; a$
240 PRINT a$
250 LOOP
```

Program skončí zprávou "End of file" (konec souboru), což může být problém, když píšete skutečný program. Je možné ukončit soubor "legráckou " jako třeba " zzz " a použít řádek jako tento 250 LOOP UNTIL a$="zzz", ale to není příliš vhodné. Master DOS Vám pomocí funkce EOF řekne kdy byl dosažen konec souboru, takže se můžete vyhnout dalším INPUTům - viz níže.

Funkce EOF - detekce konce souboru

Funkce dává 0, pokud ještě nebyl přečten poslední znak ve specifikovaném kanálu, nebo 1, pokud již přečten byl. Níže uvedený příklad nám ukazuje její použití v změněném řádku číslo 250

```
250 LOOP UNTIL EOF 4
```

Funkce INP$ - čtení pevného počtu znaků

Funkce INP$ čte specifikovaný počet znaků z kanálu.Například LET a$ = INP$ (#4,10)

Stejně jako INKEY$, ani INP$ se nestará o to, jaké znaky čte a bude zahrnovat i znaky CHR$ 13, které najde. Počet znaků může být až 16384, takže můžete často přečíst celý soubor najednou například použitím LET a$ = INP$ (#4, LENGTH #4): PRINT a$ Je to mnohem rychlejší než opakované použití INPUT nebo INKEY$.

Ukládání dat do souboru s náhodným přístupem

Náš soubor "test" je příliš malý a jednoduchý.Skutečný přehled by nejspíše využíval delší záznamy s rozdílnými částmi věnovanými jednotlivým účelům. (Tyto oblasti jsou nazývány "pole"). Například, je-li záznam vytvářen ukládáním 100 znakového stringu do souboru příkazem PRINT, mohli byste mít umístěny data v něm tak, jak je ukázáno níže. (Tento částečný program není na Vaší disketě).

```
100 DIM d$ (100)
110 INPUT "Author?"; a$
120 LET a$ ( TO 40 ) = a$
130 INPUT "Title?"; t$
140 LET d$ (41 to 96) = t$
150 INPUT "year of publication?"; y$
160 LET d$ (97 TO 100) = y$
170 PRINT # something; d$
```

Máte-li číst takovýto záznam ze souboru, mohli by jste si informace zobrazit podobným způsobem, jako je tento:

```
500 PRINT "Author:";d$( TO 40)
510 PRINT "Title:";d$(41 TO 96)
520 PRINT "Year:";d$(97 TO 100)
```

Některá data mohou mít nepříjemný počet polí - například adresa může obsahovat ulici, okres, město, kraj a směrovací číslo. Jestliže rezervujete dost místa pro maximální délku pole, tolik kolik předpokládáte, vyplýtváte mnoho místa na disku. Jednou z možností jak místo ušetřit a přitom se přiblížit výše uvedeným požadavkům, je zachovat pevnou délku pro každý kompletní záznam, ale obsah uspořádat trochu pružněji. Například bychom mohli uložit počet podsouborů proměnné délky v kompletním záznamu tak,jak je ukázáno v části programu uvedené níže:

```
600 DIM d$ ( 100 )
610 LET t$ = ""
615 DO
620 INPUT a$
630 EXIT IF a$ = ""
640 LET t$ = t$ + a$ + CHR$ 13
650 LOOP
660 LET t$ = d$
670 PRINT #4;d$
680 REM rest of program
```

Jednotlivé části záznamu jsou od sebe odděleny znakem CHR$ 13,což znamená, že k přečtení každého kompletního záznamu musíte několikrát použít INPUT; tak jak je ukázáno v níže uvedené časti podprogrmu pro čtení ze souboru.

```
900 LET poi = PTR 4
910 DO
920 INPUT #4; a$
930 PRINT a$
940 LOOP WHILE PTR 4 < ptr + 100
```

Použití funkce PTR v uvedeném programu dovoluje uložit do každého záznamu různý počet substringů (dříve jsem užil podsoubor). Alternativou INPUT je funkce INP$, která bude číst stále stejný počet znaků z kanálu.Všechny řádky výše uvedené (900-940) mohou být nahrazeny následujícími:

```
900 LET a$ = INP$ (#; 100)
910 PRINT a$
```

To od Vas vyžaduje pátrání po CHR$ 13 v a$, pokud by jste stringy chtěli rozdělit do jednotlivých substringů-zde může být užitečná INSTR (viz manuál Coupé)

Ukládání čísel

Předpokládejme,že do souboru chcete uložit číselná data.Kdykoliv tak můžete jednoduše učinit příkazem PRINT, například takto:

```
PRINT #5 ; x
```

Nicméně větší kontrolu nad tím, která část souboru je využita (pro uložení čísel) budete mít, použijete-li příkaz podobný tomuto:

```
LET d$ (10 TO 12) = STR$ x
```

Přesněji řečeno,nejlepší použitá metoda bude záviset na velikosti a přesnosti numerické hodnoty, kterou chcete uložit, a na tom kolik chcete ušetřit místa. Například: je-li číslo od 0 do 255 můžete např. použít LET d$(65)=CHR$ x (opak je LET x=CODE d$(65) Můžete omezit hodnotu na pevný počet desetinných míst použitím nap. LET x=INT (x*100)/100 před jejím uložením (tento příklad omezuje počet desetinných míst na 2)(Obecně LET x=INT(x*10 na n tou)/10 na n-tou. Kde n je počet desetinných míst - bez zaokrouhlení.

Záznamy (stringy) proměnné délky

Někdy jsou záznamy o pevné délce nevhodné, protože data,která společně ukládáte mají tak rozdílnou velikost, že by vyplýtvaly příliš mnoho místa na disku. V některých případech může být jednodušší pro program použít záznamy o proměnné délce, zvláště když jen zřídka nebo dokonce vůbec nechcete aktualizovat záznam v souboru (což je choulostivé, zvláště, jsou-li nová data delší než stará). Rovněž tak můžete chtít číst data z existujících sériových souborů tvořených záznamy o proměnné délce.Pokud byste použili náhodný přístup (použití několikanásobného INPUT), mohli byste v souvislosti se čtením některých záznamů souboru narazit na poněkud dráždivé problémy. Bohužel, proměnná délka záznamů je důvod, že není možné použít běžný tvar příkazu POINT k dosažení záznamu, například:

```
POINT # kanál, (číslo záznamu - 1)* délka záznamu
```

Existují ale způsoby, jak to obejít, třeba mít soubor dat pevné délky, který poskytuje hodnoty ukazatele pro každý záznam proměnné délky v jiném souboru nebo v další části téhož souboru. Ale je i jednoduché řešení; použít další tvar příkazu POINT. Jako třeba takto:

POINT #5, OVER 10

Tento příkaz začne tam,kde se právě nachází ukazatel aktuálního souboru a přeskočí 10 oddělovačů CHR$ 13 před nastavením nové pozice ukazatele. Takže,chcete-li ukázat na dvoutisící záznam souboru, musíte použít:

POINT #5,0: POINT #5, OVER 1999

Tento příkaz nejdříve ukáže na začátek souboru a potom přeskočí přes následujících 1999 oddělovačů CHR$ 13. Ačkoliv POINT bude muset číst prvních 1999 záznamů aby toto provedl, čte rychlostí 22,5 K/sekundu (nebo přes 50 K/sekundu z RAM disku), takže pro většinu souborů spotřeba času je bezvýznamná. Kromě toho, víte-li který záznam máte právě vložit příkazem INPUT, potom nemusíte znovu začínat na začátku souboru. Například jestliže jste vložili záznam (string) číslo 2000 a jako další chcete vložit záznam (string) číslo 2100, bude pracovat i příkaz

POINT #5, OVER 99

Je dokonce možné předefinovat znak, který má být přeskočen příkazem POINT OVER "přeskočit přes" při běhu programu, použitím funkce POKE DVAR 10 (kód znaku), takže záznamy můžete oddělit jedním znakem (řekněme CHR$ 128) a poté znakem jiným, takže potom můžete použitím POINT OVER nalést jak záznam, tak i pole, které požadujete, velice rychle. Abychom si to předvedli, níže uvedený program vytváří soubor 1000 záznamů (stringů) a každý záznam je ukončen CHR$ 128; obsahuje 6 polí různě dlouhých a ukončených CHR$ 13. Středníky na koncích řádků 60 a 80 zabraňují vložení CHR$ 13 do souboru. Tento program můžete natáhnout z diskety Master DOS příkazem LOAD "prog3". Příkazem RUN spusťte vytváření souboru. Zabere to trochu času,protože soubor bude asi 150 K dlouhý. Dejte si zatím čaj nebo kávu!

```
10 CLOSE #4
20 OPEN #4; "varifile" RND
30 FOR r= 1 TO 1000
40 PRINT AT 10,10;r
50 FOR f= 1 TO 6
60 PRINT #4; "Record:";r; "Field:";f;
70 PRINT #4; "ABCDEFGHIJKLMN" ( TO RND*14)
80 NEXT f: PRINT #4; CHR$ 128;
```

```
 90 NEXT r
100 CLOSE #4
110 STOP
```

Když se vrátíte zpět k počítači umožní Vám následující program dle Vašeho přání číst pole a záznam (string) ze souboru.

```
200 CLOSE #4
210 OPEN #4; "VARIFILE" RND
220 POINT #4,0
230 INPUT "RECORD?";r
240 IF r=0 THEN CLOSE #4: STOP
250 INPUT "Field?";f
260 POKE DVAR 10,128
270 IF r<>1 THEN POINT #4; OVER r-1
280 POKE DVAR 10,13
290 IF f<>1 THEN POINT #4; OVER f-1
300 INPUT #4; r$: PRINT r$
310 GOTO 220
```

Otevírání mnohonásobných souborů

Naše příklady pracovali pouze s jedním souborem otevřeným v daném časovém okamžiku, ale v módu náhodného přístupu je klidně možné mít otevřeno najednou několik souborů. Ale pokud soubory, které používáte, zahrnují zapisování do nového souboru nebo rozšiřování staršího, musí být tyto soubory na stejném disku a ve stejné jednotce, neboť místo na disku by bylo využito velice nepůsobivě. (Ostatní soubory mohou být na disku v druhém drajvu)

Přemístění souborů s náhodným přístupem

Je možné přemístit soubor do kanálu otevřeného pro soubor s náhodným přístupem. Protože MOVE využívá k přenosu dat obodby PRINT #, je snadné předvídat výsledek. Máte-li právě otevřen soubor s náhodným přístupem, souborový ukazatel bude ukazovat na nulu a přemísťovaná data přepíší existující dokud je všechny nepřepíší a soubor začne být rozšiřován. Použijete-li POINT a funkci délky souboru k nastavení souborového ukazatele na konec souboru, data rozšíří soubor bez přepsání jakýchkoliv existujících dat.

Čtení a zapisování souborů

Uživatelé s technickými sklony mohou použít příkazy READ AT a WRITE AT k přímému přístupu do sektorů disku. Oba příkazy mají podobnou syntaxi:
READ nebo WRITE AT drajv (jedotka), stopa, sektor, adresa, počet (použitých sektorů)

Příklad: WRITE AT 1, 10, 1, 100000, 20

Tento příkaz uloží obsah paměti do disku v drajvu č.1, začátek na stopě 10, sektor 1, nahraje data od adresy 100000 dále.Je ukládána na 20 sektorů, takže budou využity celé stopy 10 a 11. Je-li poslední číslo vynecháno, počítač to akceptuje jako 1 sektor. Přenosová rychlost pro velké počty souborů je kolem 22 K za sekundu s reálnými disky.

Upozornění!

WRITE AT zásadně nepodáva zprávu o tom, že již něco v sektorech je a že by to mohl přepsat!

READ AT 3, 0, 1, 65536, 42

Tento příklad čte z disku č.3 (RAM disk), od stopy 0, sektor 1 a natahuje do paměti 42 sektorů do adresy 65536 a dále.

Číslo drajvu musí být 1-7, nebo můžete použít hvězdičku, což znamená aktuání disk. Číslo stopy 0-79, nebo 128 - 207 (1. a 2. strana diskety). Číslo sektoru musí být 1 - 10. Adresa se musí pohybovat v rozmezí 16384-540671. Počet čtených sektorů musí být 1-1024. Teoreticky je tak umožněno ovládat až 512K pamět.

RAM disky budou mít často různý počet stop, ale vždy začínají od 0 a mohou mít čísla až do 79 a jako reálné disky od 128 výše.

Pokud některý příkaz dosáhne nejvyššího čísla stopy na jedné straně disku a má pokročit na další stopu, budou data směrována na první stopu druhé strany disku.

Proměnné DOSu a DVAR

Proměnné DOSu můžete změnit napsáním POKE DVAR n,x - kde n je číslo proměnné DOSu a x je její nová hodnota. Můžete také použít POKE se stringem, nebo zobrazit hodnotu DVAR použitím např. PRINT PEEK DVAR 2.

Pokud chcete změnit proměnnou DOSu, tak abyste ji nemuseli měnit po každém zasednutí k počítači (vzpomeňte si, že tak můžete učinit bez velkého úsilí pomocí AUTO souboru), postup je trochu komplikovanější. Nemůžete uložit příkazu SAVE proměnnou DOSu přímo z paměti, takže natáhněte DOS z disku na nějakou adresu, řekněme 65535. Potom tam, kde byste měli použít POKE DVAR n,d použijte POKE 65535+535,d.(+535 je kompenzace protože proměnné DVAR jsou uloženy mimo soubor DOSu). Potom uložte modifikovaný DOS použitím: SAVE "jméno" CODE 65535,15700. Napište CALL 0 abyste zresetovali počítač a když stisknete klávesu F9 k novému nahrání, zjistíte že v proměnné DOSu je nová hodnota.

Poznámka: Některé proměnné jsou vhodné jen pro využití značně technicky nadanými uživateli. Proměnné DOSu jsou:

0 řídí blikání podkladu (BORDER) když je soubor čten z disku.
POKE DVAR 0,0 - podklad obrazovky nebliká
POKE DVAR 0,7 - normální nastavení

1 Data stop a stavu drajvu č.1. Obsazuje počet stop na jedné straně disku (obvykle 80) plus 128,pokud je disk dvoustranný Obvyklá hodnota je 208, ale může být pomocí POKE změněna,máte-li jinou diskovou jednotku, nebo chcete-li - řekněme částečně formátovat disk.

2 Data stop a drajvu č.2 - jako výše. DOS automaticky nastavuje tuto proměnnou na 208, máte-li druhý drajv, pokud příkazem POKE neuložíte do proměnné jinou hodnotu. Máte-li jen jeden drajv, normální hodnota je 0.

3 Kroková rychlost drajvu č.1.Uložená hodnota 0 umožňuje drajvu krokovat tak rychle, jak jen to je možné. Jiná hodnota způsobí čekání (v milisekundách) po každém příkazu ke kroku. Např: POKE DVAR 3,3 způsobí čekání kolem 3 ms. Čekání může být nutné pro spolehlivou činnost diskových jednotek starší konstrukce.

4 Kroková rychlost drajvu č.2 - jako výše.

5 Znak použitý v nevyplněném místě v adresářovém seznamu. Obyčejně je to mezera, ale např.POKE DVAR 5,"-" vyplní prázdné místo v adresáři pomlčkami. Vyzkoušejte si to!

6 Rezervováno

7 Číslo verze DOSu krát 10 plus 20! (Krát 10, takže verze 1.0 dá 10 a plus 20 k odlišení Master DOSu od SAM DOSu, který využívá čísla do 20). PRINT DVAR 7 bude dávat 30 nebo více.

8 Nastavení počtu sloupců v jednoduchém adresáři. Obyčejně je to nula, což znamená "Použij tolik sloupců, kolik bude vhodné". Bude se měnit v závislosti na aktuálním módu obrazovky a nastavení okna. Pomocí POKE můžete dát jinou hodnotu např. 1 pro jednosloupcový seznam, nebo 7, chcete-li široký seznam na 80-ti znakové tiskárně.

9 Řízení abecedního třídění pro jednoduché adresáře. Normálně je její hodnota 1, což znamená "třídění zapnuto". POKE DVAR 9,0 vypíná třídění.

10 Znak použitý POINT # (kanál), OVER x jako oddělovač záznamů (stringů). Obyčejně to je CHR$ 13. Viz část o zacházení se záznamy proměnné délky, dozvíte se podrobnosti k použití.

11 Znak použitý k oddělení rozšíření jména souboru při tisku do adresáře. Normálně je to ".". Příkazem POKE může být změněn na jiný znak. POKE DVAR 11,0 vyřadí rozšíření jména souboru.

12 Hlavní symbol uznaný jako oddělovač mezi jmény v cestě. Používá se v PATH$ a adresářích. Normálně "\" (lomítko je obráceně)

13 Alternativní symbol uznaný jako oddělovač mezi jmény. Jako v bodě 12 ale lomítko se používá "/".(normálně)

14 Pootočení mezi stopami v diskovém FORMATu. Normálně je 255 (což zde působí jako mínus 1); znamená to, že pro každou stopu je poslední sektor na stopě (10) protilehlý sektoru č. 9 na následující stopě. To znamená, že čtecí hlavička má kolem 20 ms krok na novou stopu a usazení předtím než dojede sektor č.1, protože první sektor bude naproti sektoru č.10 na nové stopě, bude ignorován a jeho přejetí zabere 20 ms. SAM DOS používá pootočení o 255 což umožňuje 40 ms a to může být vhodnější pro starší typy diskových jednotek, ačkoliv LOAD a SAVE budou asi o 10% pomalejší. Mohli by jste vyzkoušet i posunutí o 0, poskytující pro krokování pouze několik málo ms. Je to na hranici zpracovatelnosti - ale jestliže Vám to zničí soubory, nenadávejte mi.

15 Opomenutí čísla drajvu používaného vnitřně

16 Počet řídících stop na posledně použitém disku

17 Hexa číslo pro aktuální podadresář

18 Hexa číso pro střídající podadresář v operacích typu soubor TO soubor. (ze souboru do souboru)

19 Ukaž data

20 Počet souborů otevřených příkazem OPEN

21 Dosud nejvyšší číslo podadresáře (v hexa)

22 (2 byte) Adersa adresové tabulky HOOK kódu, pokud je DOS umístěn v oblasti od 16384

24 Návěští MOVE TO #2. Určuje co se stane se znaky s kódy nad 127, když jsou příkazem MOVE přemístěny ze souboru na obrazovku. Normální hodnota 0 znamená, že takovéto znaky jsou zmenšeny o 128 a pak vytisknuty v módu INVERSE 1. Změnou pomocí POKE na hodnotu 1 je tiskne jako UDG nebo grafický blok

25 Kontrolní kód MOVE TO #2 náhradního znaku. Používá se místo CHR$ 0-31 a CHR$ 22, které mohou způsobit chyby. Normálně je to ".".

26 Stránka přepnuta na 32768, když je stisknuto tlačítko BREAK při natahování SAMova emulátoru Spectra a poté je stisknuta klávesa 1 nebo 5. Normálně je hodnota 4
27 (2 byte) Adresa volaná za výše uvedených okolností. Normálně 4, což nemá žádný efekt, ale umožní např. v budoucnosti uložení obrazovky.
29 Počet 0.25 sec. čekání před SAVE, mínus 1
30 (3 byte) Rezervováno
33 (2 byte) Adresa ON ERROR, pokud má příkaz chybnou syntaxi.
35 Rezervováno
36 použitá stránka ON ERROR, pokud je adresa ON ERROR > 32767
37 (2 byte) Přírůstek použití mezi sektory během několika sektorového READ AT a WRITE AT. Normálně 512. Jiné hodnoty mohou být použity, pokud pracujete s disky ne pro Coupé a 510 může být užitečné pro nahlédnutí do souborů SREEN$.
39 (5 byte) Stopy na drajv pro každý RAM disk, nebo 0
44 (5 byte) První 16k stránka pro každý RAM disk
49 (7 byte) Číslo aktuálního podadresáře pro každý drajv
56 (7 byte) Délka cesty pro každý drajv
63 (14 byte) Náhodný dvojitý bajt pro každý drajv
77 (2 byte) Náhodný dvojitý bajt když je otevřen první soubor
79 (2 byte) Dočasná proměnná hodin
81 (9 byte) datum ve formě dd/mm/rr po přečtení hodin
90 (8 byte) max/min hodnoty DATE pro zadávání dd/mm/rr
98 (9 byte) čas ve formě hh:mm:ss po přečtení hodin
105 (8 byte) max/min hodnoty TIME pro zadávání h:m:s
111 (7 byte) ukazuje počet existujících disků. Normálně 1-7
118 (32 byte) vnější tabulka RAM. Jeden bit představuje každou možnou 16k vnější paměťovou stránku. Bit je 0, je-li stránka přístupná, nebo je 1, pokud stránka neexistuje nebo je použitá. Bit č.7 prvního bajtu představuje první 16k stránku
150 hodnota portu použitá vestavěnými hodinami

Formát řídícího záznamu

Tyto informace jsou určeny pro ty, kteří se zajímají o použití příkazů READ AT a WRITE AT k přímému řízení disku při psaní užitkových programů.

První stopy na první straně disku jsou využity pro adresář, začínající na stopě 0, sektoru 1. Každý sektor obsahuje dva 256-bajtové řídící záznamy.Formát každého záznamu je následující:

BYTE

0 STATUS/typ souboru, jak je sdělen funkcí FSTAT s parametrem 4. Obsahuje 0, pokud není záznam využit nebo je smazán.
1 (10 bajtů) Jméno souboru. Je-li první bajt roven 0, záznam nebude použit a čtení řídícího záznamu nebude pokračovat.
11 MSB počtu sektorů použitých v souboru
12 LSB počtu sektorů použitých v souboru
13 číslo stopy, kde začíná soubor
14 číslo sektoru kde začíná soubor
15 (195 bajtů) Mapa užitých sektorů pro soubor. Bit č.0 prvního sektoru představuje 1.sektor 4.stopy, bit č.1 představuje 2. sektor atd. Je-li bit zapnut znamená to že sektor je využit souborem.
210 (10 bajtů) Ve všech záznamech na disku, kromě prvního záznamu, tyto bity obsahují informace PLUS D/zastoupení.
V prvním záznamu je uloženo jméno disku. Bit č.7 bajtu č.210 může být zapnut, nebo vypnut bez změny jména disku.
220 Návěští
221 (10 bajtů) Specifické informace o typu souboru. Je-li typ souboru typu SCREEN$ potom bajt 221 je mód obrazovky - 1.
232 (4 bajty) Reservováno.
236 V bitech 4-0 číslo počáteční stránky,bity č.7-5 nejsou definovány.
237 (2 bajty) kompenzace počátku v rozmezí 32768 - 49151
239 počet stránek v délce
240 (2 bajty) délka MOD 16384. Bity 15 a 14 nejsou definovány
242 Prováděcí stránka pro soubory ve strojovém kódu, nebo 255.
243 (2 bajty) Provedení vyrovnání (32768-49151) pro soubory ve strojovém kódu, nebo samospouštěcí řádek pro program v BASIC
245 Den uložení příkazem SAVE, nebo 255
246 Měsíc uložení
247 Rok uložení
248 Hodina uložení
249 Minuta uložení
250 Pro soubory DIR je zde uloženo číslo v HEXA které označuje soubory v tomto podadresáři
251 reservováno
252 (2 bajty) Pouze v prvním řídícím záznamu,náhodné slovo identifikující disk
254 HEXA číslo podadresáře ve kterém se nachází tento soubor
255 Pouze v prvním záznamu, počet stop adresáře minus 4

Chybové hlášení

Kromě běžných chybových hlášení, které podává BASIC SAM Coupé používá Master DOS následující chybová hlášení:

84 Vyžádané opuštění
Stisknuta klávesa ESC

85 Chyba TRK - ,SCT -
Sektor specifikovaný číslem stopy a sektoru nemohl být nataženo.

86 Ztracený formát TRK -
Specifikovaná stopa byla porušena

87 Zastavený disk v drajvu, nebo není disk v jednotce

91 Chybné zařízení

93 Chybná kontrola
Data na disku nesouhlasí s daty v počítači

94 Chybný typ souboru
Např. LOAD"jméno" když jméno je soubor ve strojovém kódu

99 Čtení psaného souboru
Čtení souboru otevřeného příkazem OPEN s OUT, nebo chybný OUT (nový soubor)

100 Zapisování čteného souboru
Zapisování souboru otevřeného příkazem OPEN s IN, nebo soubor chráněný proti zápisu příazem PROTECT.

101 Není soubor "AUTO*"
Druhé použití BOOT hledalo soubor "auto*" a nenašlo jej

103 Neexistující disk
Disk není vhodný, nebo RAM disk není naformátovaný

104 Disk je chráněn proti zápisu
Přesuňte plastikový čtvereček v rohu diskety

105 Disk je plný
Plocha disku pro data je zaplněna

106 Plný adresář
Volné stopy (slots) jsou vyčerpány

107 Soubor není nalezen

109 Použité jméno souboru
V aktuálním adresáři je již soubor stejného jména a nebo je již otevřen příkazem OPEN

111 Použitý tok
Můžete uzavřít kanál příkazem CLOSE a nebo použitím příkazu CLEAR #

112 Použitý kanál

113 Adresář nebyl nalezen

114 Adresář není prázdný
Vymažte adresář pokud obsahuje soubory

115 Nejsou volné stránky
COPY a BACKUP potřebují alespoň jednu volnou stránu

116 Soubor je chráněný příkazem PROTECT
Pokud je soubor chráněn lze použít pouze příkazy SAVE, COPY