# KAPITOLA 7

# Vytváření souborů CMD a BAT

**V této kapitole:**

- Spuštění příkazového interpreteru
- Práce v příkazovém řádku
- Vytváření dávkových souborů

V předchozích kapitolách jsme si představili příkazy a utility. Ty můžete použít v příkazovém řádku v interaktivním módu tak, že je napíšete společně s přepínači příkazových řádků a s argumenty. Ale pokud příkazovou řádku využíváte častěji, ruční zadávání všech těchto údajů vás rychle unaví. Je mnohem lepší nejpoužívanější příkazy nějakým způsobem zautomatizovat, aby po zadání jediného příkazu bylo provedeno hodně práce. Soubory CMD (příkazové) a BAT (dávkové), o kterých pojednává tato kapitola, vám právě toto umožní, a to jen za cenu nepatrného úsilí.

Uživatelé mnoho let spoléhali na dávkové soubory různých typů. Dají se rychle vytvořit, jsou snadno pochopitelné a lze je jednoduše upravit. Testování dávkového souboru je snadné – nepotřebujete žádné propracované nástroje nebo ladicí program (debugger). Práce s dávkovými soubory vlastně představuje ten nejjednodušší způsob, jak začít s programováním. Zadáváte příkazy doslova ve stejném pořadí, jako to děláte v příkazovém řádku. Ve Windows se s pomocí kopírování a vkládání snadno propracujete ke kompletnímu dávkovému souboru.

Jedním z nejzajímavějších příkladů využití dávkových souborů jsou soubory `AutoExec.NT` a `Config.NT`, které slouží ke konfiguraci příkazového řádku. Tyto dva soubory představují významnou možnost k nastavení příkazového prostředí podle konkrétních potřeb, aniž by to obnášelo o mnoho více práce než provedení volby v nabídce. Ale většina lidí tuto možnost opomíjí. Tato kapitola ukazuje, že krátký kód s několika základními instrukcemi může skutečně výrazně přispět k tomu, aby práce v prostředí příkazového řádku byla příjemná.

# Spuštění příkazového interpreteru

Příkazový interpreter je zvláštním druhem aplikace. Když otevřete příkazové okno, ve skutečnosti tak spouštíte příkazový interpreter. Ten přijímá vaše příkazy a něco s nimi provádí. Příkazový interpreter pro Windows se jmenuje `CMD.EXE`. Tato aplikace je zodpovědná za vytvoření příkazového okna a přijímání konfiguračních příkazů. Také poskytuje přístup k vestavěným příkazům, jako je příkaz `Dir`.

Příkazový interpreter můžete ve Windows nakonfigurovat pěti způsoby. První možnost spočívá v přidání přepínačů příkazových řádků do souboru CMD.EXE. Tímto postupem je příkazový interpreter nastaven jako celek. Jakmile je příkazový interpreter spuštěn, nemáte příliš velkou kontrolu nad touto konkrétní změnou. Samozřejmě si můžete vytvořit zástupce pro jednotlivé příležitosti, při kterých je pro každý úkol zapotřebí jiné nastavení přepínače příkazových řádků.

Druhou možností je změna obsahu v souboru `Config.NT`. Tento soubor se nachází v adresáři `\Windows\System32` a příkazový interpreter jej volá, aby nastavil prostředí příkazového okna. Soubor `Config.NT` mění ovladače zařízení, počet souborů, nahrávání Virtual DOS Machine (VDM) do vyšší paměti a další nastavení.

Třetím konfiguračním postupem je změna obsahu souboru `AutoExec.NT`. Tento soubor je vlastně dávkovým souborem, který můžete měnit stejně jako kterýkoli jiný dávkový soubor. Všechny postupy popsané v oddílu „Vytváření dávkových souborů“ v této kapitole se týkají tohoto souboru. Použitím odpovídajících programovacích postupů si můžete vytvořit jakékoli prostředí. Ve skutečnosti byste mohli uživateli ukázat možnosti a pak podle jeho výběru konfigurovat prostředí.

Čtvrtou možností konfigurace je soubor Program Information File (PIF). Informace o nastavení, které jste poskytli dosovým aplikacím prostřednictvím tohoto souboru, přímo ovlivňují běh těchto aplikací. Soubor PIF je vlastně prostředkem ke specifikaci alternativních variant souborů `Config.`

NT a AutoExec.NT. Když nějaká aplikace určená pro příkazový řádek vyžaduje pro spuštění zvláštní prostředí, můžete jej následně vytvořit.

Pátým postupem jsou ruční změny příkazového řádku. Některé možné změny jsou popsány v pasáži „Správa proměnných v prostředí pomocí příkazu Set“ ve 3. kapitole. Ovšem jiné změny lze provést prostřednictvím běžných utilit pro příkazový řádek. O těch pojednávají následující oddíly.

## Používání přepínačů CMD

Příkazový interpreter CMD.EXE je nejdůležitější částí příkazového řádku, protože má vliv na vše, co v tomto řádku děláte. Malá změna v příkazovém interpreteru může mít zásadní vliv na to, jakým způsobem se chovají vaše aplikace. Soudě podle výchozího nastavení příkazového řádku se předpokládá, že nechcete používat žádné přepínače příkazového řádku a že chcete začít v domovském adresáři.

Pokud jste někdy používali systém DOS (Disk Operating System), je třeba mít na paměti, že přepínače příkazového řádku, které nabízí systém Windows pro příkazový interpreter CMD.EXE, nemají v žádném případě nic společného s podobným stylem práce v Dosu. Společnost Microsoft z důvodu kompatibility zpřístupnila některé přepínače příkazového řádku. Například příkaz /X je stejný jako /E:ON, /Y je stejný jako /E:OFF a /R je stejný jako /C. Příkazový interpreter ignoruje všechny starší přepínače; namísto toho musíte použít přepínače příkazového řádku popsané v této části.

Možná si také pamatujete několik příjemných vlastností z dob Dosu, které už nejsou součástí Windows. Například jednu dobu bylo možné vytvořit spouštěcí nabídku prostřednictvím záznamu [MENU] v souboru Config.SYS. Soubor Config.NT toto nastavení nepodporuje. Jedinou alternativou je vytvořit několik souborů Config.NT a přiřadit je podle potřeby k aplikacím. Stručně řečeno, i když příkazový interpreter dělá mnoho stejných věcí jako v dosové verzi, je odlišný. Proto musíte být ve svých hypotézách obezřetní. Aplikace CMD používá následující syntaxi:

```
CMD [{/A | /U}] [/Q] [/D] [/E:{ON | OFF}] [/F:{ON | OFF}] [/V:{ON | OFF}] [[/S]
   [{/C | /K}] řetězec] [/T:FG]
```

V následujícím seznamu jsou popsány všechny argumenty příkazového řádku.

**/C *řetězec xxx*** Vykoná příkaz dle specifikace *řetězec* a poté ukončí relaci příkazového interpreteru. Při použití tohoto postupu uvidíte výstup z aplikace jen tehdy, pokud poskytuje grafický výstup nebo pokud použijete přesměrování, abyste uložili výsledky do souboru.

**Poznámka:** Při práci s přepínači příkazového řádku /C nebo /K můžete zadat více příkazů vytvořením jednoho řetězce, který obsahuje všechny příkazy. Oddělte každý příkaz dvěma znaky „&“ (&&). Celý řetězec musíte uzavřít do dvojitých uvozovek. Například řetězec „Dir *.DOC&&Dir *.TXT“ by vykonal dva příkazy Dir. První by hledal všechny soubory s příponou DOC, zatímco druhý by vyhledal všechny soubory s příponou TXT.

**/K *řetězec*** Vykoná příkaz dle specifikace *řetězec*. Příkazové okno zůstane po provedení příkazu otevřené, a tak můžete vidět výsledky aplikace.

**/S** Upraví postup, jakým je zpracován příkazový řetězec použitý s přepínači příkazového řádku /C a /K. Příkazový interpreter nabízí pro zpracování příkazového řetězce dvě možnosti. Pokud použijete přepínač příkazového řádku /S, příkazový procesor prohlíží řetězec spojený s přepínači příkazového řádku /C a /K. Ověřuje, zda jsou prvním znakem uvozovky, a odstraňuje je z řetězce. Potom příkazový procesor hledá uzavírací uvozovky a rovněž je odstraní. Tuto možnost můžete

využít, pokud uvozovky způsobují potíže při spouštění příkazu. Příkazový interpreter odstraní uvozovky také v případech, kdy:

- použijete v řetězci některý z následujících zvláštních znaků: &<>()@^|
- vložíte jednu nebo více mezer
- vložíte do řetězce název spustitelného souboru
- použijete v řetězci více než jeden pár uvozovek

**/Q** Vypíná výpis. Výpis je výstupem příkazového interpreteru, který zobrazuje, jaký příkaz běží.

**/D** Vyřazuje spouštění příkazů AutoRun z registru. O tomto záznamu v registru je pojednáno dále v tomto oddíle.

**/A** Stanoví, že výstupem interních příkazů bude roura nebo soubor s kódováním ANSI.

**/U** Stanoví, že výstupem interních příkazů bude roura nebo soubor s kódováním Unicode.

**/E:ON** Povoluje rozšíření příkazu. Tato rozšíření nabízejí další funkčnosti pro tyto příkazy: `Assoc`, `Call`, `ChDir (CD)`, `Color`, `Del (Erase)`, `EndLocal`, `For`, `Ftype`, `GoTo`, `If`, `MkDir (MD)`, `PopD`, `Prompt`, `PushD`, `Set`, `SetLocal`, `Shift` a `Start` (zahrnuje rovněž změny externích příkazových procesů). Oddíl „Porozumění rozšířením příkazů“ v této kapitole vám objasní, jaký má rozšíření na příkazy vliv.

**Poznámka:** Možná jste si všimli, že některé příkazy jsou uvedeny v kulatých závorkách. Například příkaz `ChDir` je uveden před příkazem `(CD)`. Tyto dva příkazy jsou totožné. Můžete použít kterýkoli z nich. Kulaté závorky neznamenají upřednostnění, ale jen alternativu.

**/E:OFF** Vypíná rozšíření příkazu.

**/F:ON** Povoluje dokončování názvu souboru nebo adresáře. To umožňuje rychlé psaní v příkazovém řádku. Například pokud chcete napsat `Dir Temp` a použít funkci dokončování, můžete napsat `Dir T` a pak stisknout Ctrl+D (adresář) nebo Ctrl+F (soubor). Příkazový interpreter za vás automaticky doplní adresář nebo název souboru. Pokud zadáte část řetězce, která neodpovídá žádným záznamům, příkazový interpreter pípne, aby upozornil, že zadání je nesprávné. Jestliže příkazový interpreter najde více záznamů, které by mohly odpovídat vašemu zadání, zobrazí první záznam ze seznamu. Ten můžete procházet opakovaným tisknutím Ctrl+D nebo Ctrl+F. Klávesové zkratky Shift+Ctrl+D a Shift+Ctrl+D slouží k posouvání směrem zpět seznamem možností. Používané ovládací klávesy můžete změnit úpravou příslušného záznamu v registru. Do uvozovek musíte vypsat názvy souborů a adresářů, které začínají zvláštními znaky. Mezi tyto znaky patří <space> &()[]{}^=;!'+,`~.

**/F:OFF** Vypíná dokončování názvu souboru nebo adresáře (detaily viz přepínač příkazového řádku `/F:ON`).

**/V:ON** Povoluje zpožděné rozšíření proměnné prostředí. Toto rozšíření považuje vykřičník (!) za oddělovač. Zadání `!MyVar!` do příkazového řádku by rozšířilo (zobrazení hodnoty) `MyVar` v době spuštění.

**/V:OFF** Vypíná zpožděné rozšíření proměnné prostředí.

**Poznámka:** Systém Windows Vista podporuje přepínač příkazového řádku `/T`, i když není uveden v nápovědě k aplikaci CMD. Je důležité si uvědomit, že Microsoft nedokumentované přepínače příkazových řádků často ruší, takže se může stát, že k následujícímu prvku nebudete mít v budoucích verzích přístup.

**/T:FG** Nastavuje barvu textu (`F`) a pozadí (`G`). Hodnoty musejí být zadány dohromady, bez mezer. V následujícím seznamu jsou uvedeny barvy, které můžete v příkazovém řádku použít, společně s číslem příslušné barvy.

| | |
|---|---|
| 0 – černá | 8 – šedá |
| 1 – modrá | 9 – světle modrá |
| 2 – zelená | A – světle zelená |
| 3 – modrá | B – světle modrá |
| 4 – červená | C – světle červená |
| 5 – fialová | D – světle fialová |
| 6 – žlutá | E – světle žlutá |
| 7 – bílá | F – zářivě bílá |

## Práce s příkazovým interpreterem v registru

Chování mnoha příkazových řádek závisí na nastaveních registru. Tato nastavení naleznete v klíči `HKEY_CURRENT_USER\Software\Microsoft\Command Processor` (pro lokálního uživatele) nebo v klíči `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Command Processor` (pro všechny uživatele téhož počítače). Příkazový interpreter vyhledává tato nastavení v případě, že neposkytnete odpovídající přepínač příkazového řádku. Pokud interpreter nevidí záznam v registru, pak příkazový interpreter použije výchozí nastavení. Nastavení lokálního uživatele má vždy přednost před nastavením počítače; nastavení přepínačů příkazového řádku má vždy přednost před nastaveními v registru. Níže jsou uvedena nastavení registru a jejich význam.

**AutoRun** Určuje příkaz, který má příkazový interpreter spustit při každém otevření příkazového řádku. Tato hodnota je typu `REG_SZ` nebo `REG_EXPAND_SZ`. Jednoduše zadejte název spustitelné aplikace společně s potřebnými přepínači příkazového řádku. Stejně jako u řetězce pro přepínače příkazových řádku `/C` a `/K` můžete více příkazů oddělit pomocí dvou znaků „&" (&&).

**EnableExtensions** Určuje, zda má příkazový interpreter povolena rozšíření. Seznam aplikací, které tento údaj ovlivňuje, najdete v popisu přepínače příkazového řádku `/E:ON`. Tato hodnota je typu `REG_DWORD`. Nastavte ji na „povoleno" zadáním hodnoty 0x1, nebo na „zakázáno" zadáním hodnoty 0x0.

**CompletionChar** Určuje znak pro dokončování názvů souborů (detaily viz přepínač příkazového řádku `/F:ON`). Výchozím znakem je Ctrl+F (0x06). Pokud chcete tento prvek vyřadit, použijte hodnotu pro mezeru (0x20), neboť mezera není platným řídicím znakem. Tato hodnota je typu `REG_DWORD`.

**PathCompletionChar** Určuje znak pro dokončování názvů adresářů (detaily viz přepínač příkazového řádku `/F:ON`. Výchozím znakem je Ctrl+D (0x04). Pokud chcete tento prvek vyřadit, použijte hodnotu pro mezeru (0x20), neboť mezera není platným řídicím znakem. Tato hodnota je typu `REG_DWORD`.

**DelayedExpansion** Určuje, zda příkazový interpreter používá zpožděné rozšíření proměnné. Další informace najdete v popisu přepínače příkazového řádku `/V:ON`. Tato hodnota je typu `REG_DWORD`. Nastavte ji na „povoleno" zadáním hodnoty 0x1 nebo na „zakázáno" zadáním hodnoty 0x0.

### Porozumění rozšířením příkazu

Rozšíření příkazu je další možností, kterou příkazový interpreter poskytuje pro určité příkazy. Výsledek se liší podle příkazu, ale obecně lze říci, že příkazy získávají dodatečnou funkčnost. V některých případech, jako u příkazu `del (erase)`, rozšíření jednoduše mění způsob, jakým příkaz funguje. V následujícím seznamu jsou popsány změny rozšíření příkazu pro jednotlivé ovlivněné příkazy.

**Assoc** Microsoft nezdokumentoval, jak rozšíření příkazu mění příkaz `Assoc`. Ačkoliv jej Microsoft uvádí jako jeden z příkazů, který se prostřednictvím rozšíření příkazu mění, v příkazovém řádku není žádný zjevný rozdíl.

**Call** Přijímá jako cíl volání návěští (místo obvykle používaného názvu souboru). Díky této vlastnosti můžete převádět ovládání z jedné části dávkového souboru do jiné. V důsledku použití rozšíření můžete volat návěští prostřednictvím `call :Label Arguments`. Všimněte si, že před návěští musíte napsat mezeru.

**Chdir (CD)** Zobrazí názvy adresářů přesně tak, jak jsou uvedeny na vašem pevném disku. Například pokud je v názvu adresáře mezera, uvidíte při změně adresářů mezeru. Použití velkých písmen je rovněž shodné. Název adresáře zobrazený s prvním velkým písmenem v Průzkumníkovi Windows se zobrazí právě tak v příkazovém řádku. Kromě toho, pokud vypnete rozšíření příkazu, příkaz nepovažuje mezery za oddělovače. To znamená, že nemusíte v příkazu před a za názvy adresářů doplňovat mezery nebo uvozovky, a přesto dosáhnete použitím tohoto příkazu správných výsledků.

**Color** Microsoft nezdokumentoval, jak rozšíření příkazu mění příkaz `Color`. Ačkoliv jej Microsoft uvádí jako jeden z příkazů, který se prostřednictvím rozšíření příkazu mění, v příkazovém řádku není žádný zjevný rozdíl.

**Del (Erase)** Mění způsob, jakým funguje přepínač příkazového řádku `/S`. Příkaz zobrazí odstraňované soubory, namísto aby zobrazil všechny soubory včetně těch, které nemohl nalézt.

**EndLocal** Obnovuje nastavení rozšíření příkazu do stavu před zadáním příkazu `SetLocal`. Příkaz `SetLocal` zpravidla neukládá stav rozšíření příkazu.

**For** Zavádí řadu možností pro příkaz `For`. Při práci s adresáři zpracováváte adresáře (místo seznamu souborů v adresáři) prostřednictvím tohoto volání: `for /D {%%| %}proměnná in (sada) do příkaz [argumenty_příkazového_řádku]`. Můžete také provést rekurzivní zpracování adresářové struktury. Při použití této vlastnosti může jediný příkaz zpracovat celou strukturu (na rozdíl od použití jednotlivých příkazů pro zpracování samostatných částí. Použijte příkaz `for /R [[jednotka:]cesta] {%% | %}proměnná in (sada) do příkaz [argumenty_příkazového_řádku]`, pokud chcete provést rekurzi. Je také možné opakovat různé hodnoty (podobně jako u funkčnosti smyčky For používané v pokročilých jazycích), a to pomocí tohoto příkazu: `for /L {%% | %}proměnná in (Start#,Step#,End#) do příkaz [argumenty_příkazového_řádku]`. Náhrada proměnné je další užitečnou vlastností při používání rozšíření příkazu (detaily naleznete v oddílu „Používání nahrazení proměnných“ v této kapitole). Pomocí rozšíření příkazu můžete rovněž na místě provést komplexní analýzu a iteraci souborů (detaily naleznete v oddílu „Provádění komplexní iterace souboru“ v této kapitole).

**Poznámka:** Vista poskytuje podporu pro utilitu ForFiles. Tato utilita vás zbavuje nutnosti používat některá rozšíření. Více o této utilitě se dozvíte v oddílu „Používání utility ForFiles“ v této kapitole.

**Ftype** Společnost Microsoft nezdokumentovala, jak rozšíření příkazu mění příkaz `FType`. Ačkoliv jej Microsoft uvádí jako jeden z příkazů, který se prostřednictvím rozšíření příkazu mění, v příkazovém řádku není žádný zjevný rozdíl.

**Goto** Definuje zvláštní návěští nazvané `:EOF`. Pokud v dávkovém souboru vytvoříte příkaz `Goto` s návěštím `:EOF`, systém přenese kontrolu na konec aktuálního dávkového souboru a ukončí činnost. Tato vlastnost bude fungovat i v případě, že v dávkovém souboru návěští nenadefinujete.

**If** Stanoví dodatečnou porovnávací syntaxi, díky čemuž je příkaz `If` mnohem flexibilnější. Podrobnosti se dozvíte v oddílu „Používání příkazu *If*" v této kapitole.

**MkDir (MD)** Umožňuje vytvořit jediným příkazem adresáře s pokročilou strukturou. Například strukturu celého podadresáře můžete nadefinovat pomocí příkazu `MD MůjAdr/Složka1/Složka2`. Pokud adresář `MůjAdr` neexistuje, systém jej vytvoří jako první. Následuje adresář `Složka1` a nakonec `Složka2`. Bez použití tohoto příkazu byste museli každý adresář vytvořit zvlášť a přesunout se na nižší úroveň (abyste mohli vytvořit další podadresář) pomocí příkazu `CD`.

**PopD** Odstraňuje písmeno jednotky přiřazené příkazem `PushD`.

**Prompt** Podporuje další příkazové znaky. Znak `$+` přidává do příkazového řádku pro každou úroveň příkazu `PushD` jedno nebo více znamének plus (+). Pomocí této vlastnosti můžete zjistit, kolik úrovní přesměrování příkaz `PushD` uložil do zásobníku a kolikrát musíte příkaz `PushD` použít, abyste je extrahovali. Znak `$m` přidává do příkazového řádku vzdálený název spjatý s jednotkou. Příkazový řádek nezobrazuje žádné doplňující informace o lokálních jednotkách.

**PushD** Umožňuje uložit cesty v síti do zásobníku stejně tak jako lokální jednotky a informace o cestě.

**Set** Při použití samotného příkazu `Set` se zobrazí všechny aktuálně definované proměnné v prostředí. Pokud doplníte název proměnné v prostředí, zobrazí zadanou proměnnou, ale ne příslušnou hodnotu. Pokud zadáte jen část názvu proměnné, příkaz `Set` zobrazí všechny proměnné, které by mohly tomuto názvu odpovídat.

**SetLocal** Umožňuje příkazu `SetLocal` podle potřeby a konkrétních jazykových nastavení povolit nebo zakázat rozšíření příkazu.

**Shift** Podporuje přepínač příkazového řádku `/N`, který umožňuje příkazu `Shift` posouvat proměnné, a to od *n*-té proměnné. Například pokud do příkazového řádku zadáte `Shift /2`, proměnné `%0` a `%1` nejsou ovlivněny, ale proměnné od `%3` do `%9` získají tímto posunem novou hodnotu.

**Start** Microsoft nezdokumentoval, jak rozšíření příkazu mění příkaz `Start`. Ačkoliv jej Microsoft uvádí jako jeden z příkazů, který se prostřednictvím rozšíření příkazu mění, v příkazovém řádku není žádný zjevný rozdíl.

## Úpravy souboru Config.NT

Soubor `Config.NT` obsahuje údaje, které mají vliv na práci systému s příkazovým řádkem. Kdysi konfigurační soubor obsahoval řadu ovladačů jednotek a povelů, které určovaly, jak příkazový řádek používá soubory a vyrovnávací paměť. Ovšem soubor `Config.NT` zřídka obsahuje ovladače jednotek; údaje o těchto ovladačích jsou obvykle součástí softwaru třetích stran. Jeden ze záznamů o ovladači zajišťuje přístup do systému NetWare z příkazového řádku (ale v mnoha případech postačuje k podpoře systému NetWare několik záznamů v souboru `AutoExec.NT`). Následující pasáže popisují běžná doplnění použitelná v souboru `Config.NT`.

**Poznámka:** Někteří z vás si možná pamatují na soubor `Config.SYS`, který systém DOS používá k načtení stejné konfigurace, jako to dělá systém Windows se souborem `Config.NT`. Někteří lidé se dokonce pokoušejí přesunout soubor `Config.SYS` do prostředí Windows. 32bitová verze Windows naštěstí toleruje mnoho starších dosových příkazů, i když je sama nepoužívá. Například utilita FastOpen zajišťuje v Dosu ukládání do mezipaměti cache, které zrychluje prohledávání adresářů. I když systém Windows tento soubor také nabízí, ve skutečnosti funkčnost nevyužívá a FastOpen neprovádí žádné operace. Při přechodu na 64bitovou verzi Windows je většina funkčnosti úplně odstraněna. Například není možné vytvořit soubor `Config.NT` s odkazem na utilitu FastOpen.

## Používání ovladače ANSI.SYS pro kontrolu prostředí

Ovladač zařízení `ANSI.SYS` poskytuje dodatečnou funkčnost pro aplikace v příkazovém řádku. Prostřednictvím speciálních kódů escape můžete pro vaše dávkové soubory vytvořit uživatelské rozhraní založené na znacích. Užitečný přehled ANSI escape kódů naleznete na adrese *http://www.evergreen.edu/biophysics/technotes/program/ansi_esc.htm* a na stránkách společnosti Microsoft na adrese *http://www.microsoft.com/technet/archive/msdos/comm1.mspx?mfr=true*. Tato utilita používá následující syntaxi:

```
device=[jednotka:][cesta]ANSI.SYS [/X] [/K] [/R]
```

Stejně jako u všech ostatních ovladačů jednotek, které lze přidat do souboru `Config.NT`, zahájíte záznam v souboru `ANSI.SYS` pomocí `device= položka`, načež následuje jednotka a cesta k souboru `ANSI.SYS`. V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/X** Nezávisle přemapuje doplňkové klávesy na klávesnici se 101 klávesami. Tento prvek ve skutečnosti funguje nezávisle na tom, kolika doplňkovými klávesami vaše klávesnice disponuje.

**/K** Přinutí soubor `ANSI.SYS` zacházet s klávesnicí, která má 101 nebo více kláves, jako s klávesnicí s 84 klávesami (jsou ignorovány doplňkové klávesy).

**/R** Mění vlastnosti posuvu řádku. Tím se zlepšuje čitelnost při práci s programy pro čtení obrazovek. Tyto programy interpretují obsah obrazovky a předkládají jej pomocí jiné formy výstupu. Programy pro čtení obrazovek pomáhají zrakově postiženým porozumět obsahu obrazovky.

## Nastavení umístění systému DOS v paměti

Tato utilita používá následující syntaxi:

```
DOS=[{HIGH | LOW}] [{,UMB | ,NOUMB}]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**HIGH | LOW** Určuje, zda se prostředí příkazu pokusí nahrát část sebe sama do oblasti vysoké paměti (HMA) – argument HIGH – nebo uchová celý kód v konvenční paměti – argument LOW. Výchozí nastavení je `LOW`. Nahrání příkazového prostředí do vysoké paměti je žádoucí pro zajištění dostatečné konvenční paměti pro aplikace.

**UMB | NOUMB** Určuje, zda má příkazové prostředí spravovat blok vyšší paměti (UMB – Upper Memory Block) vytvořený poskytovatelem této paměti. Systém Windows při výchozím nastavení využívá UMB. Uživatelé systému DOS byli zvyklí tento úkol provádět pomocí zvláštní aplikace nazvané `EMM386.EXE`. Argument UMB příkazovému prostředí říká, aby spravovalo blok UMB, což uvolňuje dodatečnou paměť pro nahrávání aplikací do oblastí mimo konvenční paměť. Výchozí nastavení je `NOUMB`.

### Běh pouze dosových aplikací

Z příkazového řádku můžete spustit jakýkoli program. Pokud chcete otevřít Poznámkový blok (Notepad), jednoduše napište Notepad a stiskněte klávesu Enter. Ale starší dosové aplikace mohou někdy v systému Windows způsobit problémy. Vývojáři psali dosové aplikace s tím, že tyto programy mají pod kontrolou celý počítač; to může při běhu aplikací ve Windows způsobit řadu problémů. Pokud máte některý z těchto starších programů (nejsou příliš často k vidění), můžete napomoci správnému spuštění takového programu tím, že do souboru `Config.NT` přidáte údaj `NTCMDPROMPT`. Tento údaj sděluje operačnímu systému, aby zakázal spouštění aplikací pro Windows z příkazového řádku. Následkem toho bude mít dosový program za to, že kontroluje celý počítač. Programy pro systém Windows samozřejmě můžete nadále otevřít z dalšího příkazového řádku nebo pomocí jiného obvyklého postupu, například z nabídky Start.

**Poznámka:** Někteří lidé se zajímali o to, zda údaj `NTCMDPROMPT` opravdu zabrání všem aplikacím v tom, aby přerušovaly dosový program. Přestane například díky tomuto údaji aplikace pro službu Instant Messaging (IM) kolidovat se hrou Tank Wars? Všechny aplikace, které jsem testoval, se podle všeho chovaly podle pravidel. Ale software od společnosti Microsoft podle pravidel někdy nehraje. Když se ostatní distributoři naučí triky Microsoftu, použijí je, aby se ani oni nemuseli řídit pravidly. Tudíž není žádná záruka, že vaše aplikace pro službu IM nebude kolidovat se hrou Tank Wars; je ale dostatečný důvod věřit, že údaj `NTCMDPROMPT` bude za normálních okolností fungovat.

### Zobrazování příkazů spouštěných souborem CONFIG.NT

Standardně neuvidíte žádné informace o příkazech, které jsou provedeny před otevřením příkazového okna; vidíte jen příkazový řádek. Doplnění údaje `ECHOCONFIG` do souboru `Config.NT` zobrazí jednotlivé příkazy tak, jak jsou spouštěny. Tato vlastnost vám pomůže diagnostikovat problémy s obsahem souboru `Config.NT`.

### Kontrola záznamu o rozšířené paměti EMM

Starší aplikace, zejména hry ve znakovém módu (DOS), spoléhají na specifikaci EMS, aby obešly omezení paměti příkazového řádku. Je důležité si uvědomit, že příkazový řádek účinně omezuje velikost paměti, dostupné pro dosové aplikace, na 640 kB (dále se odečte paměť využívaná operačním systémem). Obvykle je velikost této paměti nastavena jako součást souboru PIF dané aplikace. Ale soubor PIF vám neumožňuje kontrolovat správce rozšířené paměti (EMM), což je aplikace, která tuto paměť zpřístupňuje. Údaj `EMM` vám umožňuje změnit činnost správce EMM. Ten používá následující syntaxi:

```
EMM = [A=AltRegSets] [B=BaseSegment] [RAM]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**A=AltRegSets** Definuje, kolik alternativních sad pro mapování registru má správce EMM k dispozici pro mapování paměti mezi rozšířenou pamětí a konvenční pamětí. Můžete zadat jakoukoli hodnotu od 1 do 255. S výchozí hodnotou 8 je fungování ve většině případů bezproblémové. Ohledně dalších požadavků se podívejte do dokumentace k danému programu.

**B=BaseSegment** Definuje bázovou adresu segmentu, prostor, kam správce EMM podle potřeby umísťuje kód v oblasti dosové konvenční paměti z rozšířené paměti. Obecně platí, že jakékoli zvolené nastavení funguje dobře. Ale některé aplikace pro svůj běh využívají specifickou segmentaci paměti. Použití stejného paměťového segmentu pro dva účely způsobí poškození paměti a může

zapříčinit selhání aplikace. Jakékoli požadavky v této oblasti byste měli nalézt v dokumentaci příslušné aplikace. Segment můžete nastavit na jakoukoli šestnáctkovou hodnotu od 0x1 000 do 0x4 000. Výchozí nastavení je 0x4 000.

**RAM** Stanoví, že správce EMM má využívat jen 64 kB adresového prostoru z oblasti bloku UMB pro stránku EMM. EMM standardně používá pro stránku EMM celý blok UMB, aby zvýšil výkon. Ale některé programy mohou vyžadovat více konvenční paměti, než kolik zpřístupňuje tento postup. Použití volby RAM snižuje velikost stránky EMM. V důsledku toho je pro příkazové prostředí jednodušší nahrát více aplikací do vyšší paměti, čímž se uvolní konvenční paměť pro daný program.

### Nastavení počtu souborů, ke kterým lze přistupovat

Nastavení souborů se může jevit jako nedůležité, ale každý popisovač souboru, který poskytnete příkazovému prostředí, používá konvenční paměť. Mějte na paměti, že konveční paměť je i tak docela malá a mnohé starší programy se zřídka nahrají do poskytnutého prostoru. Výchozí nastavení Files=40 je zpravidla odpovídajícím kompromisem. Znamená to, že příkazové prostředí může otevřít 40 souborů, což je pro většinu starších aplikací víc než dostatečný počet. Toto číslo můžete zvýšit až na 255, pokud aplikace upozorňuje na nedostatek popisovačů souboru, nebo snížit až na 8, pokud má aplikace nedostatek paměti.

### Ovládání rozšířené paměti pomocí ovladače HIMEM.SYS

Ovladač `HIMEM.SYS` zajišťuje v příkazovém řádku podporu rozšířené paměti. Specifikace rozšířené paměti (XMS – eXtended Memory Specification) je metoda, kterou využívají aplikace, aby se vyhnuly omezení paměti v systému DOS. Velikost dostupné rozšířené paměti XMS nastavíte pomocí souboru PIF pro danou aplikaci. Funkčnost paměti XMS můžete dále upřesnit prostřednictvím přepínačů příkazového řádku, jak je to popsáno v tomto oddílu. Stejně jako u všech ostatních ovladačů jednotek, které lze přidat do souboru `Config.NT`, zahájíte záznam v ovladači `HIMEM.SYS` pomocí `device= položka`, načež následuje jednotka a cesta k ovladači `HIMEM.SYS`. Tento ovladač používá následující syntaxi:

```
DEVICE=[jednotka:][cesta]HIMEM.SYS [/HMAMIN=m] [/INT15=xxxx] [/NUMHANDLES=n]
   [/TESTMEM:{ON|OFF}] [/VERBOSE]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**Poznámka:** Ovladač `HIMEM.SYS` obsahuje řadu přepínačů příkazového řádku, z nichž mnohé jsou již zastaralé. Například ačkoliv `HIMEM.SYS` stále podporuje přepínač příkazového řádku `/A20CONTROL`, museli byste mít velmi starý počítač (starší deseti let), abyste tento přepínač potřebovali. Stručně řečeno, pokud nevlastníte velmi starý systém, pro tyto starší přepínače příkazového řádku nebudete mít nikdy využití. Kromě přepínače příkazového řádku `/A20CONTROL` se nezabývám rovněž přepínači příkazového řádku `/CPUCLOCK`, `/EISA`, `/MACHINE` a `/SHADOWRAM`. Tato pasáž popisuje ty přepínače příkazového řádku, které jsou stále užitečné.

**/HMAMIN=*m*** Určuje, kolik kilobajtů paměti HMA musí aplikace požadovat, aby ovladač `HIMEM.SYS` tento požadavek splnil. Některé aplikace vyžadují malé části z oblasti paměti HMA, což kouskuje již tak malou oblast paměti; ta pak kvůli tomu není dostupná pro jiné aplikace. Vyvstává otázka efektivního využití paměti. Aplikace, které mohou využít větší část paměti HMA, pravděpodobně uvolní více konvenční paměti pro ostatní aplikace. Můžete zadat jakoukoli hodnotu od 0 do 63. Výchozí hodnota je 0. Pokud je tento přepínač příkazového řádku nastaven na 0 nebo pokud je

v příkazovém řádku zcela vynechán, HIMEM.SYS může přiřadit paměť HMA první aplikaci, která o ni požádá, bez ohledu na to, kolik paměti HMA tato aplikace použije.

**/INT15=*xxxx*** Upřesňuje, kolik kilobajtů rozšířené paměti má ovladač HIMEM.SYS rezervovat pro rozhraní Interrupt 15h. Možná si říkáte, co vůbec rozhraní Interrupt 15h znamená – je to metoda, pomocí které aplikace komunikují s rozšířenou pamětí XMS. Tento přepínač příkazového řádku budete potřebovat jen tehdy, pokud máte starší dosovou aplikaci (velmi pravděpodobně nějakou hru nebo grafický program), který vyžaduje paměť XMS. Tato aplikace velmi pravděpodobně zobrazí neurčitou chybovou hlášku, ve které bude výslovně zmíněno rozhraní Interrupt 15h. Ujistěte se, že jste velikost paměti XMS nastavili na hodnotu o 64 kB větší, než je hodnota požadovaná touto aplikací. Můžete zadat jakoukoli hodnotu od 64 kB to 65,563 kB (ovšem samozřejmě nemůžete zadat vyšší hodnotu, než jakou disponuje váš systém). Pokud zadáte hodnotu nižší než 64, ovladač HIMEM.SYS nastaví hodnotu na 0. Výchozí hodnota je 0.

**/NUMHANDLES=*n*** Určuje maximální počet popisovačů pro bloky rozšířené paměti (EMB – Extended Memory Block), které může systém současně použít. Pokaždé, když aplikace požádá o více paměti, potřebuje popisovač ke zpřístupnění této paměti. Tento přepínač příkazového řádku nemusíte zadávat, pokud nepoužíváte starší graficky náročnou aplikaci. Můžete zadat hodnotu od 1 do 128. Výchozí hodnota je 32, což je pro většinu aplikací více než dostatečné. Změna počtu popisovačů spotřebovává více paměti pro procesy údržby, tento přepínač příkazového řádku tedy používejte obezřetně.

**/TESTMEM:{ON|OFF}** Stanoví, zda ovladač HIMEM.SYS při spuštění příkazového řádku provede kontrolu paměti. Většina lidi neví, zda je využívaná paměť v pořádku, takže občasnou kontrolou můžete předejít nežádoucím překvapením. Ovšem běh tohoto testu nějakou dobu trvá. Při použití tohoto přepínače příkazového řádku uvidíte nepřehlédnutelné zpoždění při vykreslování příkazového řádku. Ve většině případů je mnohem lepší testovat paměť za použití programu třetí strany, který není ovlivňován systémem Windows. V opačném případě si nemůžete být jisti, zda testujete veškerou paměť, a nezjistíte, jaká překvapení před vámi systém Windows tají. Test pomocí ovladače HIMEM.SYS je zevrubnější než test, který proběhne při spuštění počítače; můžete jej použít, když nemáte k dispozici jiný způsob, jak paměť otestovat.

**/VERBOSE** Zobrazuje doplňkové stavové a chybové hlášky při načítání ovladače HIMEM.SYS. Systém standardně žádné zprávy nezobrazuje, vyjma případů, kdy narazí na problém při načítání nebo spouštění HIMEM.SYS. Přidání tohoto přepínače příkazového řádku může odhalit případné problémy v nastavení vašeho systému a pomoci s diagnostikou těch problémů aplikací, které byste jinak neodhalili. Pro tento přepínač příkazového řádku můžete použít zkratku /V. Bohužel na rozdíl od dokumentace dostupné pro ovladač HIMEM.SYS na Internetu nemůžete v době, kdy systém načítá ovladač HIMEM.SYS do paměti, zobrazit tyto podrobné informace stisknutím klávesy Alt; abyste tyto údaje viděli, musíte použít přepínač příkazového řádku /VERBOSE.

## Úpravy souboru AutoExec.NT

I když soubor Config.NT nabízí některé zajímavé základní metody pro změnu prostředí příkazového řádku, soubor AutoExec.NT poskytuje mnohem více možností. Jakoukoli aplikaci, se kterou můžete pracovat pomocí příkazového řádku, je možné přidat také do souboru AutoExec.NT. Přidáním aplikací, které často používáte, můžete nastavit příkazový řádek od počátku tak, že uvidíte, co potřebujete, aniž byste museli zadávat jakékoli příkazy. Soubor AutoExec.NT můžete rovněž naprogramovat stejně jako kterýkoli jiný dávkový soubor. To znamená, že do nastavení můžete přidat různé nabídky, ve kterých pak vybíráte požadované volby. Detaily o vytváření naprogramovaného rozhraní pro

soubor `AutoExec.NT` naleznete v oddílu „Vytváření dávkových souborů“ v této kapitole. Následující oddíly popisují některé utility, které jsou v souboru `AutoExec.NT` nejčastěji používány. Tyto utility slouží k nastavení příkazového prostředí tak, aby byla vaše zkušenost s ním co nejlepší. Ale neomezujte se jen na tento výběr – do souboru `AutoExec.NT` můžete přidat libovolné příkazy nebo utility popisované v této knize.

**Poznámka:** Možná najdete některé starší utility, které systém Windows nainstaluje kvůli kompatibilitě a pak je nepodporuje. Utilita KB16 by měla poskytovat podporu klávesnice, ale zjistíte, že příkazový řádek tuto podporu zajišťuje automaticky, takže utilitu KB16 vlastně nepotřebujete. I když se tato utilita načte do paměti a tváří se, že pracuje, ve skutečnosti nic nedělá. Zjistíte, že systém Windows kromě utility KB16 nepodporuje ani utilitu MSCDex. Tato kapitola o těchto příkazech a utilitách, používaných kvůli kompatibilitě, nepojednává.

## Nastavení čísla znakové stránky pomocí utility CHCP

Znaková stránka definuje jazykovou podporu v příkazovém řádku. Za časů Dosu bylo zapotřebí dodat znakovou stránku, aby byl daný jazyk v příkazovém řádku řádně podporován, ale systém Windows obvykle nastavení znakové stránky nevyžaduje. Toto nastavení můžete potřebovat pro starší aplikace pracující ve znakovém módu. Pokud v příkazovém řádku spuštěném v okně použijete rastrový font, zobrazí se správně pouze font OEM, který jste nainstalovali jako součást Windows. Ale při běhu přes celou obrazovku nebo při použití fontu TrueType můžete použít jakoukoli podporovanou znakovou stránku. Tato utilita používá následující syntaxi:

```
CHCP [nnn]
```

Níže je popsán argument příkazového řádku.

***nnn*** Definuje, jaká znaková stránka bude použita. Čísla standardních znakových stránek jsou uvedena v tabulce 7.1. Znakové stránky 847 – 1258 jsou implementacemi fontů OEM a ANSI. Tyto implementace jsou dostupné pouze v systému Windows. Podle potřeby můžete nainstalovat další znakové stránky. Na internetové adrese *http://www.i18nguy.com/unicode/codepages.html#msftdos* můžete vidět, jak tyto znakové stránky vypadají.

**Tabulka 7.1:** Standardní znakové stránky OEM a OEM / ANSI

| Znaková stránka | Země nebo jazyk |
|---|---|
| 437 | Spojené státy |
| 850 | vícejazyčná (Latin I) |
| 852 | slovanské jazyky (Latin II) |
| 855 | cyrilice (ruština) |
| 857 | turečtina |
| 860 | portugalština |
| 861 | islandština |
| 863 | kanadská francouzština |
| 865 | skandinávské jazyky |
| 866 | ruština |
| 869 | moderní řečtina |

| Znaková stránka | Země nebo jazyk |
|---|---|
| 874 | thajština |
| 932 | japonština Shift-JIS |
| 936 | zjednodušená čínština GBK |
| 949 | korejština |
| 950 | tradiční čínština Big5 |
| 1258 | Vietnam |

## Přidání podpory pro rozhraní DPMI pomocí utility DosX

Rozhraní chráněného módu systému DOS (DPMI – DOS Protected Mode Interface) je jednou z metod, prostřednictvím které může dosová aplikace získat přístup k paměti větší než 640 kB (jak to obvykle umožňuje systém DOS, resp. příkazový řádek). Toto rozhraní navíc poskytuje chráněný přístup do paměti, takže daná dosová aplikace nekoliduje s během systému Windows. O rozhraní DPMI se můžete dočíst na stránce *http://whatis.techtarget.com/definition/0,,sid9_gci213913,00.html*. Aby bylo možné toto rozhraní používat, vývojář aplikace potřebuje zvláštní podporu v aplikaci – obvykle jako součást doplňkové knihovny, poskytované třetí stranou. Vše, co potřebujete vědět, abyste mohli tento prvek použít, je zda aplikace (obvykle hra) podporuje využití rozhraní DPMI. Tato utilita používá následující syntaxi:

```
DosX
```

Jak vidíte, tato utilita nepotřebuje žádné přepínače příkazového řádku a po nainstalování nezobrazuje žádná hlášení. Můžete použít utilitu MEM popsanou ve 3. kapitole v oddílu „Zjišťování stavu paměti pomocí utility Mem“, abyste zjistili, zda se utilita nahrála do paměti v souladu s očekáváním.

## Povolení podpory grafických znaků pomocí utility GrafTabl

Systém standardně zobrazuje všechny znaky s diakritikou, které vaše aplikace musí zobrazit, jako čistý text. V některých případech to znamená, že znaky s diakritikou se nezobrazí správně, protože váš systém nemusí být schopen tyto znaky řádně zobrazit. Utilita GrafTabl pomáhá systému Windows zobrazit znaky s diakritikou v grafické podobě, což znamená, že se vždy zobrazí správně – za předpokladu, že máte načtenu podporu správné znakové stránky. Utilita GrafTabl má vliv jen na zobrazení znaků s diakritikou; pokud chcete změnit výstup z konzoly, musíte použít utilitu Mode nebo CHCP. Tato utilita používá následující syntaxi:

```
GRAFTABL [xxx]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

***xxx*** Určuje číslo znakové stránky, která bude použita při zobrazení. Běžné znakové stránky platné v systému Windows jsou uvedeny v tabulce 7.1.

**/STATUS** Zobrazí znakovou stránku, kterou utilita GrafTabl nahrála pro účely zobrazování. Tento přepínač příkazového řádku nebere v potaz nastavení utilit Mode a CHCP.

**Poznámka:** Utilita GrafTabl nebude fungovat v systému Windows Itanium a v 64bitových verzích systému Windows.

## Tisk grafiky příkazového řádku pomocí utility Graphics

Podle nápovědy systému Windows by tento systém neměl v případě, že zkusíte použít utilitu Graphics, nic udělat. Ovšem ve skutečnosti se tato utilita načte a zjevně má nějakou funkčnost. Jestliže si toto uvědomíme, pak použití utility Graphics je „na vlastní nebezpečí". V případě, že potřebujete dosáhnout požadované funkčnosti aplikace, měli byste tuto aplikaci použít až jako poslední možnost. Adresář `\WINDOWS\system32` obsahuje soubory `GRAPHICS.COM` a `GRAPHICS.PRO`, zmíněné ve znalostní bázi na stránce *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q78123*.

**Důležité:** Použití utility Graphics může při práci v příkazovém řádku vyvolat neočekávané vedlejší účinky. Například se může stát, že přestane pracovat vyrovnávací paměť příkazové historie. Kromě toho možná nebudete schopni vyrovnávací paměť procházet, abyste mohli vidět starší informace. Utilita Graphics se snaží omezit vás tak, abyste současně pracovali jen s jedním příkazem a jen s jednou obrazovkou.

Utilitu Graphics použijte, pokud chcete nahrát do příkazového řádku podporu pro tisk grafiky. Některé starší aplikace mohou tuto podporu vyžadovat, ale standardně není nahrání této utility zapotřebí. Grafickou utilitu můžete potřebovat například tehdy, když chcete vytisknout snímek obrazovky se starší hrou. Pokud je tato utilita nahrána, obrázek se vytiskne po stisknutí kláves Shift+PrintScreen. Tato utilita používá následující syntaxi:

```
GRAPHICS [typ] [[jednotka:][cesta]název_souboru] [/R] [/B] [/LCD]
   [/PRINTBOX:STD | /PRINTBOX:LCD]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

***typ*** Určuje typ tiskárny. Ve většině případů – až na výjimky, kdy narazíte při tomto postupu na problémy – použijete výchozí typ. Mezi typy tiskáren patří: COLOR1, COLOR4, COLOR8, HPDEFAULT, DESKJET, GRAPHICS, GRAPHICSWIDE, LASERJET, LASERJETII, PAINTJET, QUIETJET, QUIETJETPLUS, RUGGEDWRITER, RUGGEDWRITERWIDE, THERMAL a THINKJET.

***[jednotka:][cesta]název_souboru*** Určuje soubor, který obsahuje informace o podpoře dané tiskárny. Ve většině případů tento soubor získáte od dodavatele tiskárny.

**/R** Tiskne výstup formou bílých písmen na černém pozadí, tedy shodně se standardním zobrazením na monitoru. Utilita za normálních okolností barvy přehodí, aby se šetřila náplň.

**/B** Vytiskne barevné pozadí, pro tiskárny typu COLOR4 a COLOR8.

**/LCD** Vytiskne obrazovku za použití poměru stran LCD monitorů, takže výstup vypadá stejně jako obrazovka.

**/PRINTBOX:STD | /PRINTBOX:LCD** Zobrazí rámeček kolem výstupu. Volby upřesňují velikost rámečku. Můžete si zvolit mezi standardním (STD) a LCD poměrem stran.

## Úspora paměti pomocí příkazu LH

Příkaz `Load High` (`LH`) se pokouší nahrát utilitu do vysoké paměti, místo aby použil paměť aplikace. Nahrání utility do vysoké paměti šetří paměť, do které se mohou načíst paměťově náročné aplikace. Za běžných podmínek byste měli nahrát do vysoké paměti všechny utility, které můžete, včetně DosX a ReDir. 64bitové verze systému Windows tento příkaz nepodporují. Tento příkaz používá následující syntaxi:

```
LH
```

Příkaz nevyžaduje žádné přepínače příkazového řádku. V příkazovém řádku za `LH` jednoduše doplňte utilitu nebo příkaz, který chcete nahrát do vysoké paměti. Pokud příkaz selže, systém nezobrazí žádnou chybovou hlášku. Umístění utility v paměti si ovšem můžete ověřit pomocí utility Mem.

### Přidání podpory CD pomocí utility MSCDexNT

Systém Windows zajišťuje veškerou potřebnou podporu pro CD a DVD. Název této podpory v systému DOS je Microsoft Compact Disk (nebo CD-ROM) Extensions (MSCDEX). Ovšem k této podpoře i tak potřebujete mít z příkazového řádku přístup. Při nahrání utility MSCDexNT se nahrají různé další aplikace včetně ovladače `VCDEX.DLL`. To je 32bitový ovladač virtuálního zařízení pro MSCDEX. Tato utilita používá následující syntaxi:

```
MSCDexNT
```

Tato utilita nevyžaduje žádné přepínače příkazového řádku.

### Instalace síťového směrovače pomocí utility ReDir

Tuto utilitu použijte, pokud chcete nahrát směrovač VDM ovladače virtuálního zařízení (VDD – Virtual Device Driver). Tento směrovač zajišťuje přístup k virtuálnímu zařízení z příkazového řádku a především umožňuje přístup do sítě. Tato utilita používá následující syntaxi:

```
ReDir
```

Tato utilita nevyžaduje žádné přepínače příkazového řádku.

### Přidání podpory pro systém NetWare pomocí utilit NW16 a VWIPXSPX

Pokud používáte server se systémem NetWare a chcete mít k aplikacím na tomto serveru přístup z příkazového řádku, musíte pro NetWare nainstalovat zvláštní utility. O tyto požadavky se obvykle postará při instalaci podpory pro klienta NetWare přímo společnost Microsoft. NetWare od společnosti Novell tento úkol také splní, ale za použití vlastního klientského softwaru namísto nástrojů od Microsoftu, které jsou v tomto oddíle probírány. Možná se stane, že se budete muset obrátit na podporu systému NetWare. Postup, jakým diagnostikovat problémy v případech, kdy aplikace NetWare nefungují, je popsán například ve znalostní bázi na stránce *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q136199*. Tyto utility používají následující syntaxi:

```
NW16
VWIPXSPX
```


Ujistěte se, že tyto dvě utility nahráváte v uvedeném pořadí a až poté, co jste nahráli utility ReDir a DosX. Tyto utility nevyžadují žádné přepínače příkazového řádku.

Systém Windows Vista neposkytuje starým protokolům IPX/SPX systému NetWare plnou podporu. Tato možnost není při instalaci ani nabídnuta, ale Vista zjevně potřebnou podporu nainstaluje, pokud detekuje server se systémem NetWare. Obecně řečeno: abyste předešli řadě problémů s konektivitou mezi systémem NetWare a systémem Vista, budete muset nastavit systém NetWare tak, aby používal protokoly TCP/IP. V případě nainstalování protokolů TCP/IP jsou uvedené utility zbytečné.

## Úprava chování aplikace pomocí souboru PIF

Soubor PIF je doplňkem pro dosové aplikace, který kontroluje, jakým způsobem systém Windows s aplikací pracuje. PIF neslouží ke spouštění aplikace (i když poklepání na ikonu PIF aplikaci spustí), ale k ovlivnění způsobu, jakým systém Windows s danou aplikací zachází. Z pohledu příkazového řádku můžete

použít soubor PIF ke dvěma zvláštním úkolům. Zaprvé můžete vytvořit pro aplikaci vlastní soubory `AutoExec.NT` a `Config.NT`, abyste tak mohli kontrolovat prostředí, ve kterém aplikace běží. A zadruhé můžete do příkazového řádku aplikace přidat přepínače příkazového řádku, takže aplikace bude spuštěna s vlastnostmi, které potřebujete. Následující oddíl popisuje tyto dva konfigurační postupy.

**Z praxe**

### Jak přimět aplikace, aby v příkazovém řádku fungovaly

Lidé se často zajímají, jak dokážu v příkazovém řádku zprovoznit tolik aplikací. Musíte si osvojit základní zásadu, která říká, že systém Windows se rozhodně nevzdá kontroly nad příkazovým řádkem, nad jeho zdroji (např. pamětí) ani nad jeho prostředím. Ovšem mnohé starší aplikace předpokládají, že mají systém zcela ve své moci. Výsledkem je zhroucení, při kterém starší aplikace selžou, protože Windows má vše pevně pod kontrolou. To znamená, že je třeba přimět starší aplikace, aby si myslely, že systém ovládají, i když tomu tak ve skutečnosti není.

Soubor PIF je jedním z nejdůležitějších nástrojů, který se používá ke zprovoznění starších aplikací v příkazovém řádku. Vytvořením souborů `AutoExec.NT` a `Config.NT`, které jsou upraveny podle potřeb aplikace, můžete kontrolovat prostředí a zlepšit jeho přístupnost pro danou aplikaci. Dokonce můžete vytvořit vlastní příkazový řádek zavoláním dávkového souboru a použitím přepínačů příkazového řádku pro aplikaci `CMD.EXE` (tyto přepínače jsou popsány v oddílu „Používání přepínačů CMD“ v této kapitole). Takováto změna prostředí může nefunkční aplikaci přeměnit ve funkční.

Systém Windows naštěstí nabízí určitou pomoc. V dialogu Vlastnosti příslušného souboru PIF prozkoumejte možnosti na záložce Kompatibilita. Pomocí uvedených možností můžete přinutit systém Windows k tomu, aby použil odlišné nastavení barev nebo aby spustil aplikaci v režimu kompatibility. Záložka Různé obsahuje volby, které kontrolují způsob komunikace Windows s danou aplikací; aplikaci můžete dokonce umožnit výlučnou kontrolu nad myší. Záložka Obrazovka nabízí užitečná nastavení způsobu, jakým systém Windows zachází s oknem aplikace. Můžete například nastavit, aby aplikace běžela přes celou obrazovku, pokud selže spuštění v okně. A nakonec, pokud jste si ještě nepovšimli, starší aplikace často vyžadují zvláštní nastavení paměti. Záložka Paměť obsahuje nastavení, která můžete použít, abyste změnili paměť podle potřeb aplikace.

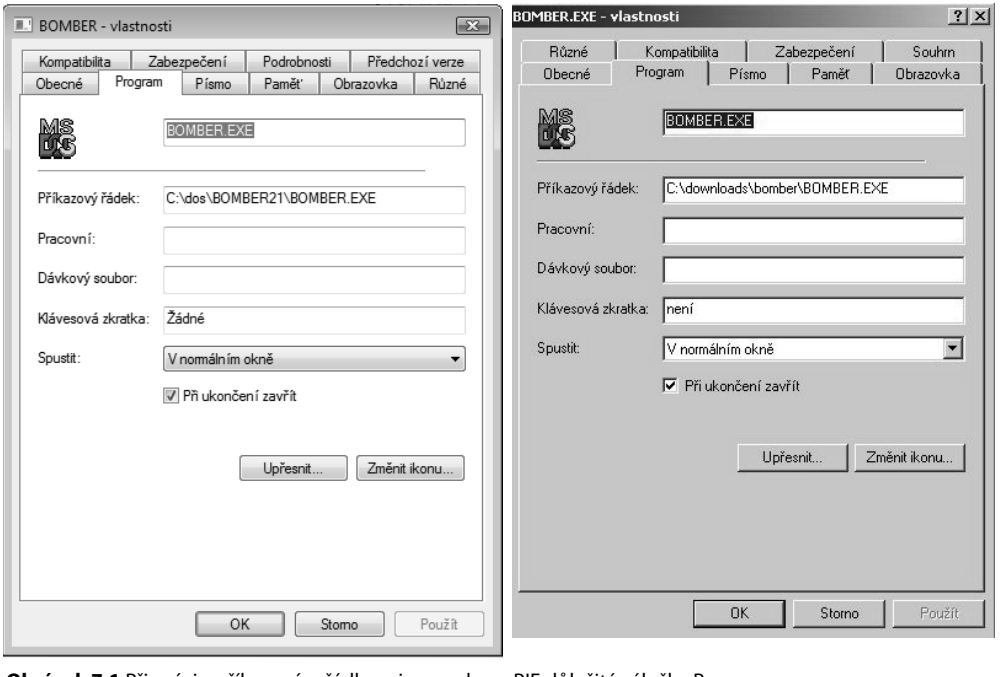
## Používání vlastních souborů AutoExec.NT a Config.NT

Pokaždé, když ve Windows otevřete příkazový řádek (i jako součást spouštění nějaké aplikace), systém Windows použije soubory `AutoExec.NT` a `Config.NT` v adresáři `\Windows\System32`, aby vytvořil prostředí pro aplikaci. Jediným problémem při tomto postupu je, že konfigurace, která funguje pro většinu aplikací, pravděpodobně nebude fungovat pro všechny aplikace. Následkem toho se můžete dostat do situace, kdy si budete hrát se soubory `AutoExec.NT` a `Config.NT`, abyste umožnili běh určité konkrétní aplikace. Samozřejmě asi nechcete měnit přímo výchozí soubory, které jsou pro jiné aplikace ve vašem systému funkční. Namísto toho vytvořte v adresáři aplikace nové verze těchto dvou souborů. Zkopírování existujících souborů je dobrým začátkem, protože tak budete mít povinné údaje již nastaveny.

VISTA

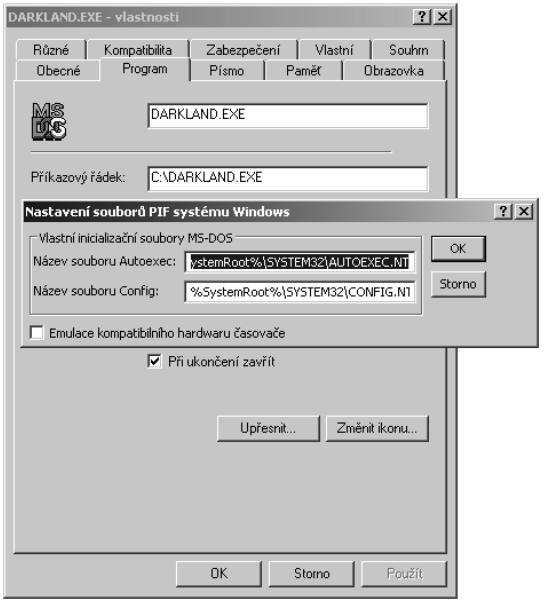
Uživatelem definovaný soubor `AutoExec.NT` nebo `Config.NT` může obsahovat cokoli, co obsahují běžné soubory. V souboru `AutoExec.NT` může být jakýkoli příkaz nebo utilita, pro kterou potřebujete nakonfigurovat zvláštní prostředí. Samozřejmě stěží vložíte do souboru `Config.NT` archaické dosové ovladače, protože systém Windows je nepoužije; nadto tento konkrétní případ může způsobit potíže při oživování některých starších aplikací. Pokud se toho vaše aplikace přesto dožaduje, možná je načase ji nahradit.

Jakmile máte pro svou aplikaci vytvořen nový soubor `AutoExec.NT` nebo `Config.NT` (nebo oba), je nutné nastavit soubor PIF tak, aby používal tyto nové soubory místo oněch univerzálních, které běžně aplikace používají. Pokud chcete přiřadit vlastní nastavení, začněte tak, že ve Windows klepnete pravým tlačítkem na dosovou aplikaci a v místní nabídce zvolíte Vlastnosti. Následky tohoto kroku jsou dva. Jednak se zobrazí vlastnosti pro danou aplikaci, jednak je vytvořen soubor PIF. Ten má stejný název jako soubor aplikace ve spustitelném formátu, jen má místo přípony EXE nebo COM příponu PIF. Zvolte záložku Program a uvidíte rozhraní příkazového řádku, jak je to znázorněno na obrázku 7.1. Na tomto obrázku je dialog ve Windows XP vlevo a ve Windows Vista vpravo. Zabezpečení systému Vista otázku kompatibility komplikuje, takže zprovoznění dané aplikace bude podstatně složitější. Ujistěte se, že jste při konfigurování své aplikace změnili v záložce Zabezpečení všechna potřebná nastavení.




**Obrázek 7.1** Při práci s příkazovým řádkem je v souboru PIF důležitá záložka Program

V oddílu „Přizpůsobení Průzkumníka Windows pomocí přepínačů příkazového řádku“ v této kapitole je popsáno, jak pracovat s polem Cmd Line. Klepněte na tlačítko Upřesnit a uvidíte dialog Nastavení – viz obrázek 7.2. Protože v tomto konkrétním případě se jedná o hru, která používá stejná nastavení jako všechny hry v mém systému, umístil jsem alternativní soubor `AutoExec.NT` do adresáře `\Windows\System32`. Pokud chcete aplikaci nabídnout vlastní soubor `AutoExec.NT` nebo `Config.NT`, stačí zadat jeho umístění do příslušného políčka (viz obrázek níže).



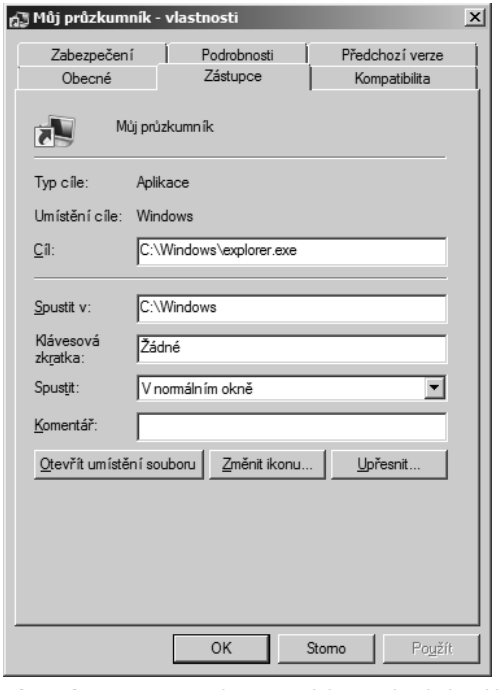

**Obrázek 7.2** Změňte obsah polí Název souboru Autoexec a Název souboru Config tak, aby odpovídal vašim vlastním souborům

**Poznámka:** Ani provedení změn v souborech `AutoExec.NT` a `Config.NT` nemusí stačit k tomu, aby vaše aplikace v systému Vista fungovaly. Ve většině případů je navíc zapotřebí v dialog Vlastnosti zaškrtnout na záložce Kompatibilita také volbu Spustit tento program jako správce. V minulosti všechny aplikace předpokládaly, že jste výhradním vlastníkem systému, a tudíž jeho správcem. Vista toto vnímání mění – všichni uživatelé jsou standardními uživateli s omezenými právy. Zaškrtnutí volby Spustit tento program jako správce snižuje selhání aplikace díky nízkým právům. (Přizpůsobení průzkumníka Windows pomocí přepínačů příkazového řádku ).

Pro jakoukoli aplikaci ve vašem systému můžete vytvořit zástupce nebo soubor PIF, který obsahuje uživatelem definovanou konfiguraci přepínače příkazového řádku. Ve skutečnosti dokonce můžete – za předpokladu, že se přepínače příkazového řádku nebudou měnit – vytvořit vlastní konfiguraci pro každou individuální situaci. (Pokud máte argumenty, které se mění, měli byste namísto souboru PIF nebo zástupce použít dávkový soubor nebo skript.) Jednou z aplikací, které bude asi každý chtít upravit, je Průzkumník Windows. Tento oddíl se tedy zabývá možnostmi přizpůsobení Průzkumníka pomocí příkazového řádku.

Průzkumník Windows není aplikací Dosu ani příkazové řádky, takže pro něj můžete nastavení příkazové řádky změnit pomocí postupu, který je popsán v oddílu „Používání vlastních souborů AutoExec.NT a Config.NT“ v této kapitole. V tomto případě začnete s úpravami tím, že vyhledáte soubor `Explorer.EXE` v adresáři `\Windows`. Klepněte pravým tlačítkem na soubor, který vidíte v Průzkumníkovi, a v místní nabídce zvolte Vytvořit zástupce. V příkladu níže je použit Můj Průzkumník. V tomto okamžiku můžete poklepat pravým tlačítkem na soubor a v místní nabídce zvolit Vlastnosti. Uvidíte dialog Vlastnosti, podobný tomu na obrázku 7.3. Pole Cíl obsahuje informace, které chcete

pro tuto aplikaci změnit. V závislosti na tom, jaké přepínače příkazového řádku aplikace podporuje, můžete upravit její chování. V následujícím seznamu jsou popsány přepínače příkazového řádku, které Průzkumník Windows používá.




**Obrázek 7.3** Upravte příkazový řádek pro jakoukoli aplikaci, aby nastavení vyhovovalo specifickým požadavkům

**/n** Spustí Průzkumníka Windows v režimu s jedním polem. Toto pole zobrazuje výchozí adresář, což je obvykle kořenová jednotka, ve které je nainstalován systém Windows.

**/e** Spustí Průzkumníka Windows s výchozím nastavením.

**/e, *Adresář*** Spustí Průzkumníka Windows s výchozím nastavením v zadané složce. Při použití této volby můžete nadále vstupovat do kořenového adresáře, stejně jako do všech ostatních adresářů v systému.

**/root, *Adresář*** Spustí Průzkumníka Windows s použitím zadané složky jako kořenového adresáře. Nemůžete procházet nadřazené adresáře, ale máte přístup do všech podadresářů.

**/select, *Objekt*** Otevře okno se zadanou složkou, souborem nebo aplikací.

Průzkumník Windows po otevření standardně zobrazí Tento počítač, což znamená, že pro většinu lidí není takto použitelný. Toto chování můžete změnit použitím jiného příkazového řádku. Například změna příkazového řádku na `%SystemRoot%\explorer.exe /e, C:\` přinutí Průzkumníka, aby po otevření zobrazil obsah jednotky C, takže můžete začít rovnou pracovat.

Až skončíte s úpravami nastavení zástupce, zavřete dialog klepnutím na tlačítko OK. Přetáhněte nového zástupce tam, kam potřebujete, abyste ho měli rychle k dispozici. Pokud například umístíte zástupce do složky Start → Programy → Po spuštění, systém při startu spustí Průzkumníka, ve kterém pokaždé bude otevřen správný adresář.

## Definování kompatibility aplikace pomocí utility SetVer

Některé starší aplikace očekávají při spuštění specifickou verzi Dosu (příkazového interpreteru) a v žádné jiné verzi se nespustí. Obvykle tyto aplikace ani nenastartují; jednoduše jen zobrazí chybovou hlášku, která vás nabádá k zajištění správné verze Dosu. Tento konkrétní problém můžete vyřešit tak, že přidáte záznam pro aplikaci do tabulky utility Set Version (SetVer). Pokud zkusíte aplikaci spustit, příkazový interpreter aplikaci oznámí, že se spouští s konkrétní verzí Dosu (kterou aplikace vyžaduje). Tato utilita používá následující syntaxi:

```
Zobrazit informace o SetVer
   SETVER [jednotka:cesta]
Přidat novou aplikaci
   SETVER [jednotka:cesta] název_souboru n.nn
Odstranit aplikaci
   SETVER [jednotka:cesta] název_souboru /DELETE [/QUIET]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

***[jednotka:cesta]*** Určuje umístění souboru `SetVer.EXE`.

***název_souboru*** Určuje název programu, který má být přidán do tabulky SetVer nebo z ní vymazán.

***n.nn*** Určuje, jakou verzi Dosu má aplikaci sdělit.

**/DELETE nebo /D** Vymaže aplikaci z tabulky SetVer.

**/QUIET** Provede zadaný úkol, aniž by zobrazil jakékoli výzvy. Tuto volbu použijte, pokud pracujete s dávkovými soubory; běh utility tak nebude uživatele rušit.

**Poznámka:** Soubor `SetVer.EXE` musíte nahrát pomocí `Config.NT` jako ovladače zařízení, abyste měli k dispozici uvedenou možnost nastavovat verzi. Stav utility SetVer můžete ověřit v příkazovém řádku, pokud napíšete `SetVer` a stisknete klávesu Enter. Příkaz vypíše seznam aplikací, které jsou zahrnuty v tabulce utility SetVer, a pak zobrazí stav této utility. V případě, že zařízení není v paměti načteno, se objeví chybová hláška.

## Používání běžných utilit systému DOS

Na pevném disku naleznete řadu běžných dosových utilit. Většina těchto utilit je k dispozici v každé verzi Dosu, dokonce i v těch, které nevyrobila společnost Microsoft (například verze od společnosti IBM). Největší podíl zaujímají utility, které mají na starosti úkoly, jež souvisejí s údržbou a jež můžete provést pomocí dávkového souboru. Mnoho lidí například mělo v dobách, kdy byly rozšířeny diskety, dávkové soubory pro práci s utilitou DiskCopy. Následující oddíl tyto utility podrobně popisuje.

### Porovnání obsahu disků pomocí utility DiskComp

Tuto utilitu můžete použít k porovnání obsahu dvou disket. S ohledem na to, jakým způsobem utilita pracuje, musí být obě diskety ve stejném formátu a není možné porovnat novější média, například kompaktní disky. Lepší volbou pro moderní média je utilita FC, popsaná v oddíle „Pokročilé porovnávání souborů pomocí utility FC“ ve 2. kapitole. Utilita DiskComp používá následující syntaxi:

```
DISKCOMP [jednotka1: [jednotka2:]]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

***jednotka1:*** Určuje zdrojovou jednotku.

***jednotka2:*** Určuje cílovou jednotku. Pokud cílovou jednotku nezadáte, utilita předpokládá, že budete porovnávat jednu jednotku, a vyzve vás podle potřeby k záměně zdrojové diskety za cílovou.

## Kopírování obsahu diskety na jinou disketu pomocí utility DiskCopy

Tuto utilitu můžete použít ke zkopírování obsahu jedné diskety na jinou. S ohledem na to, jakým způsobem utilita pracuje, musí být obě diskety ve stejném formátu. Lepší volbou pro moderní média je utilita XCopy, popsaná v oddíle „Hromadné přenosy souborů pomocí utility XCopy“ ve 2. kapitole. Utilita DiskCopy používá následující syntaxi:

```
DISKCOPY [jednotka1: [jednotka2:]] [/V]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

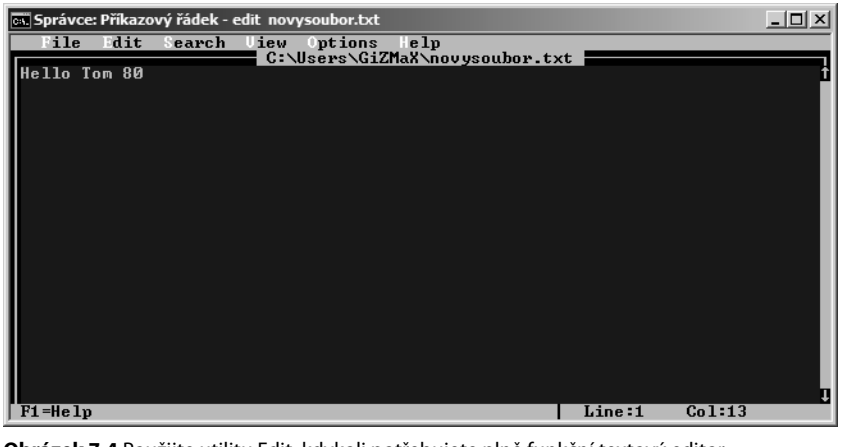
***jednotka1:*** Určuje zdrojovou jednotku.

***jednotka2:*** Určuje cílovou jednotku. Pokud cílovou jednotku nezadáte, utilita předpokládá, že pro porovnání použijete jednu a tutéž jednotku, a vyzve vás podle potřeby k záměně zdrojové diskety za cílovou.

**/V** Potvrzuje, že se soubory správně zkopírovaly. Používejte tuto volbu vždy, když chcete zajistit co nejvyšší spolehlivost kopírování.

## Úprava datových souborů pomocí utility Edit

Utilita Edit je aplikací, kterou je velmi užitečné znát, protože funguje tehdy, kdy většina ostatních editorů nikoli. Tato utilita je vcelku malá, takže ji můžete nahrát na disketu nebo na CD, na kterém je váš diagnostický software. Pro používání utility Edit ani nepotřebujete grafické rozhraní, funguje v příkazovém řádku Dosu, v konzole pro zotavení systému Windows a takřka ve všech jiných situacích, kdy se vám může editor hodit. Ovšem i s těmito omezeními tato utilita nabízí propracované rozhraní pro textový editor, jak můžete vidět na obrázku 7.4.




**Obrázek 7.4** Použijte utilitu Edit, kdykoli potřebujete plně funkční textový editor

Jak vidíte, tato utilita nabízí mnoho funkcí stejných jako Poznámkový blok, ale bez toho, aby měla takové nároky. Utilita Edit používá následující syntaxi:

```
EDIT [/B] [/H] [/R] [/S] [/<nnn>] [/?] [soubor(y)]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/B** Nastavuje utilitu Edit tak, aby se spustila v černobílém režimu.

**/H** Nastavuje utilitu Edit tak, aby se spustila s co nejvyšším počtem řádek textu pro váš hardware. Při výchozím nastavení se zobrazuje 25 řádek textu.

**/R** Nahrává soubory pouze ke čtení. Tuto volbu použijte, pokud chcete prohlížet obsah souboru; nestane se, že byste jej nedopatřením změnili.

**/S** Přinutí utilitu Edit, aby používala krátké názvy souborů.

***/nnn*** Nahraje binární soubor a zobrazí jeho obsah tak, že zalomí dlouhé řádky podle zadaného počtu znaků.

***soubor*** Určuje název souboru, který se má nahrát. Můžete použít zástupné znaky a zadat více názvů, abyste nahráli více souborů. Tento argument musí být v příkazovém řádku zadán až jako poslední.

## Formátování disku pomocí utility Format

Tato utilita naformátuje jednotku a připraví ji k použití. Tato utilita odstraní z vámi vybraného svazku všechna existující data. V tomto oddílu je popsána utilita Format ve verzi pro příkazový řádek. Konzola pro zotavení nabízí mírně odlišnou verzi této utility. Abyste tuto utilitu mohli používat, musíte být členem správcovské skupiny. Utilita Format před formátováním pevného disku vždy zobrazí varování, které musíte potvrdit, než dojde k naformátování. Tato utilita používá následující syntaxi:

```
FORMAT svazek [/FS:souborový_systém] [/V:název] [/Q] [/A:velikost] [/C] [/X]
   [/P:passes]
FORMAT svazek [/V:název] [/Q] [/F:velikost] [/P:passes]
FORMAT svazek [/V:název] [/Q] [/T:stopy /N:sektory] [/P:passes]
FORMAT svazek [/V:název] [/Q] [/P:passes] FORMAT obsah [/Q]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**Důležité:** Utilita Format přivedla k zoufalství více lidí než kterákoli jiná utilita. Jednoduše řečeno, tato utilita naformátuje váš pevný disk. V mnoha případech můžete soubory, které utilita Format odstranila z pevného disku, obnovit pomocí speciální utility, ale tento proces je časově náročný a mohou se během něj vyskytnout chyby. Utilitu Format používejte s mimořádnou obezřetností.

***svazek*** Určuje písmeno jednotky, přípojný bod nebo název svazku, který má být naformátován. Tento argument vždy zadejte, neboť výchozí hodnotou je aktuální jednotka.

**/FS:*souborový_systém*** Určuje souborový systém, který bude při formátování svazku použit. Při práci s pevným diskem máte na výběr ze souborových systémů typu FAT (starší 16bitová verze), FAT32 nebo NTFS. Pro diskety lze použít pouze systém FAT.

**/V:[*název*]** Určuje název svazku. Jestliže tento přepínač příkazového řádku použijete bez zadání názvu svazku, utilita Format vás během procesu formátování vyzve, abyste název zadali.

**/Q** Bude provedeno rychlé naformátování média. Rychlé formátování vymaže z média tabulku souborů a kořenový adresář. Neproběhne skenování sektorů média, takže nebudou detekovány špatné sektory. Tuto volbu byste měli používat jen u osvědčených médií.

**/C** Vytvoří svazek v systému NTFS, který bude systémem zkomprimován – nebudete tedy muset kompresi provádět později.

**/X** Pokud je to třeba, přinutí systém před zahájením formátování k odpojení jednotky. Odpojením jednotky se ukončí všechny aktivní popisovače.

**/A:*velikost*** Změní výchozí velikost alokační jednotky pro pevný disk. Toto nastavení můžete použít, abyste optimalizovali vlastnosti úložiště pro specifické úkoly. Pokud například zamýšlíte ukládat mnoho malých souborů, budete možná chtít použít malé alokační jednotky. Pomocí argumentu velikost můžete pro jednotky v systému NTFS stanovit velikost clusterů na 512 bytů, 1 024 bytů, 2 048 bytů, 4 096 bytů, 8 192 bytů, 16 kB, 32 kB a 64 kB. U jednotek s NTFS nelze použít clustery větší než 4096 bytů. Jednotka naformátovaná v systému FAT nebo FAT32 může použít clustery o velikosti 512 bytů, 1 024 bytů, 2 048 bytů, 4 096 bytů, 8 192 bytů, 16 kB, 32 kB, 64 kB, 128 kB nebo 256 kB. Pro sektory o velikosti větší než 512 bytů můžete použít pouze volbu 128 kB nebo 256 kB.

**Poznámka:** Souborové systémy FAT a FAT32 stanovují limit pro počet clusterů v jednom svazku. Jednotka naformátovaná systémem FAT32 může mít od 65 526 do 4 177 918 clusterů. Pokud utilita Format zjistí, že počet clusterů na formátované jednotce při použití zadané velikosti clusterů nebude v uvedeném rozmezí, okamžitě se zastaví.

**/F:*velikost*** Určuje velikost (kapacitu) formátované diskety. Výchozím nastavením je hodnota 1,44 MB. Tuto hodnotu můžete zadat ve formátu 1 440, 1 440k, 1 440 kb, 1,44, 1,44 m nebo 1,44 mb. Utilita Format umí naformátovat také diskety o kapacitě 720 kB. Žádná dokumentace společnosti Microsoft neuvádí, zda tato utilita formátuje také diskety o kapacitě 2,88 MB. Teoreticky můžete naformátovat také velmi staré diskety o velikosti 5 a půl palce a o kapacitě 640 kB nebo 1,2 MB. Kdykoli je to možné, používejte přepínač `/F` místo `/T` nebo `/N`.

**/T:*stopy*** Určuje počet stop připadajících na jednu stranu disku.

**/N:*sektory*** Určuje počet sektorů připadajících na jednu stopu.

Příkaz Format nabízí několik ukončovacích kódů, které můžete použít při práci s dávkovými soubory. V následujícím seznamu jsou popsány jednotlivé ukončovací kódy.

**0** Formátování bylo úspěšně dokončeno.

**1** Formátování selhalo, protože jste zadali nesprávné argumenty.

**4** Došlo ke kritické chybě. Utilita například nemohla naformátovat jednotku, protože byla kvůli používání zamknuta systémem. Utilita Format oznámí tuto chybu, pokud nelze použít kód 0, 1 nebo 5.

**5** Když se utilita dotázala, zda má naformátovat disk, uživatel stiskl klávesu N. Stisknutí klávesy `N` proces formátování vždy ukončí.

**/P:*passes*** Hodnota „passes“ určuje, kolikrát vynuluje každý sektor na pevném disku. Toto je nový bezpečnostní prvek v systému Vista. Přepínač příkazového řádku /P umožňuje zevrubnější smazání dat na pevném disku. Ovšem jediným způsobem, jak zajistit, že data jsou nenávratně pryč, je pevný disk úplně zničit a médium magneticky vymazat.

## Nahrávání starších dosových aplikací pomocí utility LoadFix

Některé starší aplikace se nenačítají správně. Při pokusu o nahrání uvidíte chybovou hlášku „Packed file corrupt“. Utilita LoadFix upravuje nahrávací proces těchto aplikací, takže se nahrají mimo oblast paměti o velikosti 64 kB, která je využita pro potřeby operačního systému. Tato utilita používá následující syntaxi:

```
LOADFIX [jednotka:][cesta]název_souboru
```

Dále je popsán argument příkazového řádku.

**[jednotka:][cesta]název_souboru** Určuje jméno a umístění souboru, který má být nahrán.

## Nastavení systémových zařízení pomocí utility Mode

Utilita Mode připravuje systémové zařízení k použití. Nadto ji můžete použít ke zobrazení stavu jednoho nebo všech zařízení v systému. Pokud chcete zobrazit stav všech dostupných zařízení, napište příkaz `Mode` a stiskněte klávesu Enter. Pokud vás zajímá stav jen jednoho zařízení, napište příkaz `Mode`, za něj uveďte název zařízení, např. LPT1, a pak stiskněte Enter. Tato utilita používá následující syntaxi:

```
Sériový port
        MODE COMm[:] [BAUD=b] [PARITY=p] [DATA=d] [STOP=s] [to={on | off}]
           [xon={on | off}] [odsr={on | off}] [octs={on | off}] [dtr={on | off
           | hs}] [rts={on | off | hs | tg}] [idsr={on | off}]
Stav zařízení
        MODE [zařízení] [/STATUS]
Přesměrování tisku
        MODE LPTn[:]=COMm[:]
Volba znakové stránky
        MODE CON[:] CP SELECT=yyy
Stav znakové stránky
        MODE CON[:] CP [/STATUS]
Režim zobrazení
        MODE CON[:] [COLS=c] [LINES=n]
Rychlost opakování znaků
        MODE CON[:] [RATE=r DELAY=d]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**COM*m*[:]** Určuje port COM, který bude nakonfigurován. Utilita Mode rozpozná všechny porty COM, pro které nalezne název zařízení. Obecně řečeno to znamená, že podporuje porty COM1: až COM9:, pokud nemáte speciální hardware, který by zpřístupnil další porty COM.

**Poznámka:** Při práci s utilitami v příkazovém řádku zjistíte, že různé utility podporují rozdílné počty portů. Většina utilit podporuje přinejmenším (paralelní) porty LPT1: až LPT3:, ale narazíte i na několik utilit, které podporují porty až po LTP9:. Podpora pro porty COM (sériové) je ještě rozmanitější. Některé utility podporují jen COM1: až COM3:. Běžnější u utilit je podpora od COM1: po COM3:. Ovšem setkáte se i s nemnoha utilitami, které podporují porty až po COM9:, nebo dokonce tolik portů, kolik jich je ve vašem počítači nainstalováno. Jedno zařízení je neměnné: každá pracovní stanice podporuje zařízení typu CON: (konzola).

**BAUD=*b*** Určuje přenosovou rychlost portu COM v bitech za sekundu (bps). Povšimněte si, že hodnota bps vždy neodpovídá hodnotě v baudech; podrobnosti naleznete v definici na internetové stránce *http://webopedia.internet.com/TERM/B/baud.html*. Tabulka 7.2 zobrazuje hodnoty, které musíte použít pro určení požadované rychlosti.

**Tabulka 7.2** Rychlosti (v baudech) podporované utilitou Mode

| Číselná hodnota | Odpovídající rychlost v baudech |
|---|---|
| 11 | 110 baudů |
| 15 | 150 baudů |
| 30 | 300 baudů |
| 60 | 600 baudů |
| 12 | 1 200 baudů |
| 24 | 2 400 baudů |
| 48 | 4 800 baudů |
| 96 | 9 600 baudů |
| 19 | 19 200 baudů |

**PARITY=*p*** Určuje, jakým způsobem systém kontroluje chyby přenosu. Příkaz mode podporuje hodnoty uvedené v tabulce 7.3.

**Tabulka 7.3** Typy parity podporované utilitou Mode

| Hodnota parity | Typ kontroly parity |
|---|---|
| n | žádná |
| e | sudá |
| o | lichá |
| m | značka |
| s | mezera |

**DATA=*d*** Určuje počet datových bitů na jeden znak. Můžete použít jakoukoli hodnotu od 5 do 8. Výchozí nastavení s hodnotou 7 poskytuje podporu pro standardní znaky ASCII. Hodnotu 8 použijte pro znaky ASCII s diakritikou (sada 256 znaků). Některé počítače 5 nebo 6 bitů na jeden znak nepodporují.

**STOP=*s*** Určuje počet stop-bitů po každém znaku. Jako hodnoty stop-bitu můžete zadat 1, 1,5 nebo 2. Hodnota výchozího nastavení pro 110 baudů je 2. Všechny ostatní přenosové rychlosti používají výchozí hodnotu 1. Některé počítače stop-bity s hodnotou 1,5 nepodporují.

**to={on | off}** Určuje, zda počítač používá časově neomezené zpracování. Výchozí hodnota je „off". Nastavení této hodnoty na „on" bude mít za následek, že počítač bude čekat doslova navždy na odpověď od počítače nebo hosta.

**xon={on | off}** Určuje, zda systém povolí protokol XON/XOFF. Tento protokol poskytuje řízení toku pro sériové komunikace, čímž získávají na spolehlivosti, ale také jsou náročné na výkon. Více o protokolu XON/XOFF se můžete dozvědět na internetové stránce *http://docs.hp.com/en/32022-90051/ch09s08.html*.

**odsr={on | off}** Určuje, zda systém povolí výstupní metodu handshaking používající signál Datová sada připravena (DSR – Data Set Ready).

**octs={on | off}** Určuje, zda systém povolí výstupní metodu handshaking používající signál Volno k odeslání (CTS – Clear to Send).

**dtr={on | off | hs}** Určuje, zda systém povolí výstupní metodu handshaking používající signál Datový terminál připraven (DTR – Data Terminal Ready). Režim „on" zajišťuje konstantní signál, který ukazuje, že terminál je připraven přijímat data. Režim „hs" zajišťuje signál metody handshaking mezi dvěma terminály.

**rts={on | off | hs | tg}** Určuje, zda systém povolí výstupní metodu handshaking používající signál Požadavek na odeslání (RTS – Request to Send). Režim „on" zajišťuje konstantní signál, který ukazuje, že terminál je připraven posílat data. Režim „hs" zajišťuje signál metody handshaking mezi dvěma terminály. Režim „tg" zajišťuje přepínání mezi stavy „připraven" a „nepřipraven".

**idsr={on | off}** Určuje, zda systém povolí citlivost signálu DSR. Tuto vlastnost musíte povolit, abyste mohli používat metodu handshaking pro DSR.

***zařízení*** Určuje název zařízení, se kterým chcete pracovat. Standardní názvy zahrnují LPT1: až LPT3:, COM1: až COM9: a CON:.

**/STATUS** Určuje, že chcete zjistit stav určitého zařízení nebo všech zařízení dostupných v systému.

**LPT*n*[:]** Určuje číslo LPT portu, který bude nakonfigurován. Obecně řečeno to znamená přiřazení názvů LPT1: až LTP3:, pokud vás systém nenabízí pro paralelní porty speciální podporu.

**CON[:]** Určuje konzolu jako cíl konfigurace nebo kontroly stavu. Určuje, že se změna týká okna příkazového řádku.

**Poznámka:** Utilita Mode podporuje jen fyzické jednotky. Nelze ji použít k nastavení virtuálního zařízení. Pokud například přesměrujete síťovou tiskárnu na port LPT1:, utilita Mode o tomto nastavení nezobrazí žádnou stavovou informaci a ani nebude schopna toto zařízení nakonfigurovat. Ale pokud máte ve vašem počítači fyzický port LPT1, můžete jej prostřednictvím utility Mode nastavit. Protože každá pracovní stanice disponuje fyzickou konzolou, můžete vždy použít utilitu Mode k nastavení zařízení CON a k získání stavových informací o tomto zařízení.

**CP** Nastavuje nebo zjišťuje informace o znakové stránce pro zvolené zařízení. Tabulka 7.1 obsahuje seznam znakových stránek, které systém Windows podporuje automaticky. Podporu pro další znakové stránky můžete přidat sami.

**SELECT=*yyy*** Určuje číslo znakové stránky, která má být pro zvolené zařízení používána.

**COLS=*c*** Určuje počet sloupců zobrazených na obrazovce. Výchozí nastavení je 80 sloupců. Ačkoliv můžete nastavit počet sloupců na jakoukoli hodnotu, další standardní hodnoty jsou 40 a 135. V případě použití nestandardní hodnoty může dojít k problému v aplikaci běžící prostřednictvím příkazového řádku.

**LINES=*n*** Určuje počet řádků ve vyrovnávací paměti. Výchozí hodnota je 25 (další standardní hodnota je 50). Pro stanovení počtu řádků můžete použít jakoukoli hodnotu.

**RATE=*r*** Určuje, jakou rychlostí systém klávesnice opakuje znak. Tato rychlost určuje, jak rychle systém Windows opakuje znak, když držíte stisknutou příslušnou klávesu na klávesnici. Můžete použít jakoukoli hodnotu od 1 do 32. Výchozí nastavení je 20 znaků za vteřinu. Pokud nastavíte hodnotu Rate, musíte nastavit také hodnotu Delay.

**DELAY=*d*** Určuje prodlevu, která uběhne od stisknutí klávesy do chvíle, kdy systém Windows začne příslušný znak opakovat (pokud klávesu stále držíte stisknutou). Platné hodnoty pro d jsou 1 (0,25 vteřiny), 2 (0,50 vteřiny), 3 (0,75 vteřiny) a 4 (1 vteřina). Výchozí hodnota je 2.

# Práce v příkazovém řádku

Některé příkazy a utility vám pomáhají vytvářet pro pracovní řádek lepší pracovní prostředí. V mnoha případech mají tyto příkazy jen estetickou úlohu, nejsou užitečné pro úpravu souboru nebo změnu stavu systému. Následující oddíly popisují tyto účinné příkazy a utility.

**Z praxe**

### Vytváření příjemného pracovního prostředí

Možná se ptáte, proč byste měli ztrácet čas s příkazy a utilitami, které nedělají žádnou užitečnou práci. Pro mě je tím motivem stejný důvod, proč se na podlahu dává koberec nebo proč existuje dálkové ovládání k televizi – pohodlí. Například zjistíte, že vyrovnávací paměť obrazovky se posléze zaplní nepotřebnými údaji, které už opravdu nepotřebujete sledovat. Jistě, můžete takový nepořádek na obrazovce ignorovat, ale je mnohem lepší se ho zbavit, abyste mohli pracovat s větším pohodlím. Příkaz CLS vymaže vyrovnávací paměť obrazovky a vy tak můžete dále efektivně pracovat.

Uživatelé vašich dávkových souborů a skriptových aplikací také chtějí hezké pracovní prostředí. Následující příkazy budete ve svých dávkových souborech často používat proto, abyste nabídli zvláštní efekty. Příkaz CLS vymaže obrazovku, takže uživatel sledující výstup z dávkového souboru nebude zmaten množstvím informací.

## Přesměrování výstupu z příkazového řádku do schránky Windows pomocí utility Clip

VISTA

Utilita Clip je zajímavou novinkou v systému Vista. Každý, kdo používá přesměrování, ví, jak výhodné je posílání výstupních dat na jiné místo (například do souboru) nebo naopak získávání vstupních dat odjinud (například z portu COM). Utilita Clip vám umožní provést přesměrování pomocí schránky systému Windows. Použijte přesměrování nebo rourový příkaz jako obvykle. Například příkaz `Dir | Clip` odešle výstup pro přikaz Directory do schránky systému Windows. Tento příkaz používá následující syntaxi:

```
Clip
```

Tento příkaz nevyžaduje žádné přepínače příkazového řádku.

## Vymazání obsahu obrazovky pomocí příkazu CLS

Příkaz `Clear Screen` (`CLS`) vymaže vyrovnávací paměť obrazovky a zobrazí prázdnou obrazovku. Vidíte pouze příkazový řádek. Tento příkaz používá následující syntaxi:

```
CLS
```

Tento příkaz nevyžaduje žádné přepínače příkazového řádku.

## Správa uživatelských jmen a hesel pomocí utility CmdKey

VISTA

Utilita CmdKey je novým doplňkem v systému Vista, který umožňuje správu uživatelských jmen a hesel. Prostřednictvím této utility můžete podle potřeby zobrazovat, vytvářet a mazat pověření. Ovšem tato utilita funguje jen pro aktuálního uživatele. Jinak řečeno: pověření, které spravujete, se

týká aktuálního uživatele, ne jiného uživatele v témže systému. Pokud chcete utilitu použít pro jiného uživatele, musíte se nejdříve jako tento uživatel přihlásit. To znamená, že tento příkaz dobře funguje v případě využití v přihlašovacích skriptech, které provádějí úkoly za zády daného uživatele, ale není již tak dokonalým nástrojem pro správu uživatelů prostřednictvím příkazového řádku.

Systém může mít dva typy hesel, které lze kontrolovat pomocí utility CmdKey. Prvním jsou obecná hesla, která můžete použít kdekoli. Například můžete vytvořit jméno a heslo pro přístup ke vzdálenému systému pomocí Virtuální privátní sítě (VPN – Virtual Private Network). Druhým typem jsou doménová hesla, která použijete pro přístup na doménový server. Tyto typy hesel se objeví ve výstupu pole Type, když vypíšete pověření pro váš systém.

### Zobrazování

V VISTA

Přepínač příkazového řádku /list vypíše všechna pověření přiřazená k aktuálnímu účtu. Tento režim používá následující syntaxi:

```
cmdkey /list
cmdkey /list:cílový_název
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/list** Zobrazuje všechny údaje, nebo jen ty, které upřesníte pomocí cílového názvu.

***cílový_název*** Určuje cílové pověření. Jako cíl použijte název pověření. Tento argument příkazového řádku neumožňuje použít zástupné znaky, takže jej můžete v jeden okamžik použít k vypsání pouze jednoho pověření.

### Vytváření

V VISTA

Vytvoří na základě uživatelského jména a hesla nový doménový nebo obecný údaj. Použijte přepínač příkazového řádku /add, abyste vytvořili doménové údaje, a přepínač příkazového řádku /generic, abyste vytvořili údaje obecné. Tento režim používá následující syntaxi:

```
cmdkey /add:cílový_název /user:uživatelské_jméno /pass:heslo
cmdkey /add:cílový_název /user:uživatelské_jméno /pass
cmdkey /add:cílový_název /user:uživatelské_jméno
cmdkey /add:cílový_název /smartcard
cmdkey /generic:cílový_název /user:uživatelské_jméno /pass:heslo
cmdkey /generic:cílový_název /user:uživatelské_jméno /pass
cmdkey /generic:cílový_název /user:uživatelské_jméno
cmdkey /generic:cílový_název /smartcard
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/add** Vytvoří doménový údaj.

**/generic** Vytvoří obecný údaj.

***cílový_název*** Určuje cílové pověření. Jako cíl použijte název osobních údajů. Tento argument příkazového řádku neumožňuje použít zástupné znaky, takže jej můžete v jeden okamžik použít k vypsání pouze jednoho pověření.

**/user:*uživatelské_jméno*** Určuje uživatelské jméno, které bude používáno pro přihlašování. Podoba uživatelského jména může být určena podmínkami, bez jejichž splnění není přihlášení možné.

**/pass nebo /pass:*heslo*** Určuje heslo, které bude používáno pro přihlašování. Pokud zadáte /pass bez hesla, systém vás vyzve k zadání hesla, které má být používáno ve vzdáleném systému. Na

rozdíl od mnoha jiných výzev souvisejících s heslem tato výzva nezobrazuje při zadávání hesla počet znaků formou hvězdiček (*), takže může snadno dojít k chybě. Při zadávání hesla proto buďte opatrní.

**/smartcard** Vytvoří heslo na základě obsahu čipové karty smart card. Systém vás vyzve, abyste jako součást vstupních údajů poskytli tuto kartu.

### Mazání

Pomocí utility CmdKey nelze pověření měnit. Pokud chcete provést změnu, musíte nejprve smazat starý záznam a pak vytvořit nový. Možnost mazat údaje budete také potřebovat pro odstranění starých pověření, která již nepotřebujete. Tento režim používá následující syntaxi:

```
cmdkey /delete:cílový_název
cmdkey /delete /ras
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/delete:*cílový_název*** Vymaže zadané pověření. Pokud nemažete pověření pro přístup na vzdálený server (RAS), musíte zadat cílový název.

**/ras** Určuje, že chcete odstranit pověření pro přístup na vzdálený server (RAS).

## Změna barvy obrazovky pomocí příkazu Color

Příkaz Color mění barvu textu a pozadí v příkazovém okně. Tento příkaz používá následující syntaxi:

```
COLOR [FG]
```

Níže je popsán argument příkazového řádku.

**FG** Nastavuje barvu textu (F) a pozadí (G). Hodnoty musejí být zadány dohromady, bez mezer. Pokud použijete příkaz Color bez zadání hodnot pro barvy, příkaz změní barvy na výchozí hodnoty, které byly platné při otevření příkazového okna. V následujícím seznamu jsou uvedeny barvy, které můžete v příkazovém řádku použít, společně s číslem příslušné barvy.

| | |
|---|---|
| 0 – černá | 8 – šedá |
| 1 – modrá | 9 – světle modrá |
| 2 – zelená | A – světle zelená |
| 3 – modrá | B – světle modrá |
| 4 – červená | C – světle červená |
| 5 – fialová | D – světle fialová |
| 6 – žlutá | E – světle žlutá |
| 7 – bílá | F – zářivě bílá |

## Práce se systémovým datem pomocí příkazu Command

Pomocí příkazu `Date` lze zobrazovat a nastavovat systémové datum. Tento příkaz používá následující syntaxi:

```
DATE [{/T | datum}]
```

Níže jsou popsány jednotlivé argumenty příkazového řádku.

**/T** Zobrazí datum bez výzvy k zadání nového data. Tento přepínač příkazového řádku funguje jen tehdy, pokud povolíte rozšíření příkazu.

***datum*** Určuje nové systémové datum.

## Sledování aktivity v příkazovém řádku pomocí utility DosKey

Utilita DosKey provádí tři činnosti. Zajišťuje výpis historie, který většina lidí používá, aby měli přehled o existujících údajích zadaných do příkazového řádku. Stisknutím klávesy se šipkou dolů se přesunete na další příkaz, stisknutím klávesy se šipkou nahoru se přesunete na předchozí příkaz, stisknutím klávesy se šipkou dolů na nejnovější příkaz a stisknutím klávesy se šipkou nahoru na nejstarší příkaz v historii.

Dále můžete tento příkaz použit k editaci předchozích příkazů. V následující seznamu jsou popsány editační vlastnosti.

**šipka vlevo** Přejde na předchozí znak v příkazu.

**šipka vpravo** Přejde na další znak v příkazu.

**Ctrl+šipka vlevo** Přejde na předchozí slovo v příkazu.

**Ctrl+šipka vpravo** Přejde na další slovo v příkazu.

**Home** Přejde na začátek řádku.

**End** Přejde na konec řádku.

**Esc** Vymaže příkaz z obrazovky.

**F1** Zkopíruje další znak ze stejného sloupce v příkazu, který jste předtím zadali.

**Poznámka:** Systém umístí předchozí příkaz do zvláštní oblasti paměti nazvané šablona a umožní vám pracovat s tímto příkazem (založeným na pozici aktuálního sloupce). Pokud jste například zadali jako předchozí příkaz `Dir*.BAK`, spustili jej a pak zadali do příkazového řádku `Dir`, po stisknutí klávesy F1 by byla přidána mezera, po dalším stisknutí pak hvězdička (*) a tak dále. Pokud zůstaneme u tohoto příkladu, pak po stisknutí klávesy F2 a pak klávesy `A` by se v příkazovém řádku zobrazilo `Dir *.BA`. Používání kombinace šablony a funkčních kláves vám pomůže snížit počet úderů potřebných pro zadání příkazu.

**F2** Vyhledává v předchozím příkazu další znak, který jste zadali po stisknutí F2.

**F3** Zkopíruje zbylou část předchozího příkazu do příkazového řádku.

**F4** Vymaže znaky, které se nacházejí mezi aktuální pozicí kurzoru a vámi zadaným znakem. Pokud je například v příkazovém řádku právě napsáno `Dir .BAK` a kurzor bliká na pozici hvězdičky (*), po stisknutí klávesy F4 a klávesy B by se příkaz změnil na `Dir BAK`.

**F5** Zkopíruje předchozí příkaz do aktivního příkazového řádku.

**F6** Vloží na aktuální pozici kurzoru znak označující konec souboru (Ctrl+Z). Tato možnost se obvykle využívá při vytváření souboru prostřednictvím konzoly.

**F7** Zobrazí v dialogu všechny příkazy uložené v příkazové historii. Příkazy vybíráte pomocí klávesy se šipkou nahoru a se šipkou dolů, po zvolení příkazu klávesou Enter jej utilita DosKey přepíše do příkazového řádku. Po dalším stisknutí klávesy Enter bude příkaz proveden. Před příkaz můžete také zapsat sekvenční číslo, které můžete použít pomocí klávesy F9.

**Alt+F7** Vymaže všechny příkazy uložené v aktuální paměti příkazové historie.

**F8** Zobrazí jeden příkaz z příkazové historie, který začíná stejnými znaky jako aktuální příkaz. Opakovaným tisknutím klávesy F8 můžete procházet seznam všech odpovídajících příkazů.

**F9** Vyzve vás k zadání čísla příkazu v historii vyrovnávací paměti a pak zobrazí příkaz spojený se zadaným číslem. Stisknutím klávesy Enter příkaz provedete. Stiskněte klávesu F7, abyste viděli seznam příkazů uložených v příkazové historii a příslušná příkazová čísla.

**Alt+F10** Vymaže definice všech maker.

A nakonec můžete tento příkaz použít také k vytváření maker. Makra automaticky provádějí některé úkoly v příkazovém řádku, podobně jako dávkové soubory, ale mnohem méně pohodlně. Tato makra můžete také použít ke komunikaci s aplikacemi. Ovšem počet aplikací, které mohou makra utility DosKey používat, je velmi omezený. Makro DosKey můžete například použít s utilitou FTP, která je popsaná v oddílu „Správa serverů FTP pomocí utility FTP" v 5. kapitole. Aby mohla nějaká aplikace utilitu DosKey využít, musí běžet v příkazovém řádku a poskytovat výstup z vyrovnávací paměti. S ohledem na omezení maker ze strany utility DosKey se jimi tato kniha nezabývá. Mnoho příkladů maker DosKey ale můžete nalézt na Internetu. Například na stránce *http://www.palmtoppaper.com/ptphtml/17/pt170037.htm* je uvedena řada užitečných maker, která lze vytvořit. Dobrý popis jejich tvorby najdete na adrese *http://www.atarimagazines.com/compute/issue137/S18_How_to_create_keyboa.php*. Článek na stránce *http://www.melbpc.org.au/pcupdate/9405/9405article7.htm* popisuje tvorbu maker DosKey krok za krokem. Utilita DosKey používá následující syntaxi:

```
DOSKEY [/REINSTALL] [/LISTSIZE=velikost] [/MACROS[:ALL | :aplikace]] [/HISTORY]
   [/INSERT | /OVERSTRIKE] [/EXENAME=aplikace] [/MACROFILE=název_souboru]
   [název_makra=[text]]
```

Níže jsou popsány jednotlivé argumenty příkazového řádku.

**/REINSTALL** Nainstaluje novou kopii utility DosKey. Tuto možnost použijte, pokud byla vaše aktuální kopie poškozena nebo pokud je zaplněna nepotřebnými daty.

**/LISTSIZE=velikost** Určuje počet příkazů, které budou uloženy v paměti příkazové historie. Výchozí nastavení je 10.

**/MACROS** Zobrazí seznam všech maker DosKey. Výstup zahrnuje jak makra pro příkazový řádek, tak makra spojená s určitou aplikací.

**/MACROS:ALL** Zobrazí seznam všech maker DosKey použitelných pro spustitelné aplikace kompatibilní s utilitou DosKey. Pokud například vytvoříte makro DosKey pro utilitu FTP, tento přepínač příkazového řádku jej zobrazí.

**/MACROS:*aplikace*** Zobrazí seznam všech maker DosKey spojených s určenou aplikací.

**/HISTORY** Zobrazí seznam všech příkazů v paměti příkazové historie.

**/INSERT** Přepne utilitu DosKey do vkládacího režimu. Jakýkoli text nově zadaný do příkazového řádku se objeví u již existujícího textu.

**/OVERSTRIKE** Přepne utilitu DosKey do režimu „overstrike". Již existující text bude nahrazen (přepsán) textem nově zadaným do příkazového řádku.

**/EXENAME=*aplikace*** Při tvorbě makra určuje název aplikace, která má být použita. Výsledné makro poběží v dané aplikaci. Aby tato volba fungovala, aplikace musí podporovat utilitu DosKey. Pokud tento přepínač příkazového řádku nezadáte, všechna vámi vytvořená nebo nainstalovaná makra poběží v příkazovém řádku, nikoli jako součást aplikace.

**/MACROFILE=*název_souboru*** Určuje soubor, který obsahuje makra, jež chcete nainstalovat.

***název_makra*** Určuje název makra, které vytváříte. Pokud zadáte název makra a za něj rovnítko a stisknete klávesu Enter, marko bude vymazáno ze seznamu.

***text*** Obsahuje text makra, které chcete nahrávat. Abyste mohli použít tento argument, musíte zadat název makra, za něj rovnítko a pak text makra. Například zadáním `DosKey MůjAdresář=Dir *.* /P` a stisknutím klávesy Enter můžete nadefinovat makro nazvané MůjAdresář, které zobrazuje adresář. Poté co jste toto makro vytvořili, můžete do příkazového řádku zadat `MůjAdresář` a stisknout Enter; utilita DosKey pak spustí makro MůjAdresář.

## Přístup k nápovědě v příkazovém řádku pomocí utility Help

Ve většině případů stačí zadat název utility, za který připíšete přepínač příkazového řádku `/?` – tak získáte více informací. Ale Microsoft se rozhodl to v některých případech trochu zkomplikovat. Některé utility například vyžadují, abyste použili přepínač příkazového řádku `/Help`, nebo budete muset použít utilitu Help, abyste se o daném příkazu nebo utilitě dozvěděli více. Pokud chcete znát seznam příkazů a utilit, které Help podporuje, napište Help a stiskněte klávesu Enter. Tato utilita používá následující syntaxi:

```
HELP [příkaz]
```

Dále je popsán argument příkazového řádku.

***příkaz*** Určuje jméno příkazu, o kterém chcete získat více informací. Pokud zadáte pouze **Help** a stisknete Enter, uvidíte seznam všech dostupných příkazů. Poté můžete zvolit konkrétní příkaz a utilita Help vám pomůže dozvědět se o něm více.

## Práce se systémovým časem pomocí příkazu Time

Pomocí příkazu Time lze zobrazovat a nastavovat systémové datum. Tento příkaz používá následující syntaxi:

```
TIME [{/T | čas}]
```

Níže jsou popsány jednotlivé argumenty příkazového řádku.

**/T** Zobrazí datum bez výzvy k zadání nového data. Tento přepínač příkazového řádku funguje jen tehdy, pokud povolíte rozšíření příkazu.

***čas*** Určuje nový systémový čas.

## Změna záhlaví příkazového okna pomocí příkazu Title

Záhlaví příkazového okna se vám možná zpočátku nebude zdát tak důležité, ale jsou dva důvody, kvůli kterým může mít svůj význam. Zaprvé, pokud máte otevřeno více příkazových oken, použití výstižného názvu může zjednodušit orientaci mezi jednotlivými okny v panelu nástrojů systému Windows. Zadruhé je třeba mít na paměti, že systém Windows spojuje provedené změny nastavení se záhlavím příkazového okna. Změna záhlaví má vliv na to, jakým způsobem systém Windows ukládá učiněné změny nastavení. Příkaz `Title` může změnit záhlaví příkazového okna. Tento příkaz používá následující syntaxi:

```
TITLE [řetězec]
```

Dále je popsán argument příkazového řádku.

***řetězec*** Obsahuje text pro příkazové okno.

# Vytváření dávkových souborů

Dávkové soubory jsou druhem jednoduchého programu, jehož prostřednictvím můžete uložit řadu příkazů, které chcete provádět opakovaně. Dávkový soubor je obvykle souborem s koncovkou BAT. Ve většině případů nepotřebujete k přípravě dávkového souboru žádné programování; jednoduše vytvořte soubor s příkazy, které chcete jeden po druhém spustit.

**Poznámka:** Systém Windows nepodporuje příkaz Break, který je součástí mnoha starších dávkových souborů. Původním účelem příkazu Break bylo zajištění kontroly nad kombinací kláves Ctrl+Break. Nastavení hodnoty `Break ON` umožňovalo stisknutím Ctrl+Break zastavit běh dávkového souboru. Systém Windows tento přepínač příkazového řádku ignoruje. Windows Vista kromě toho mění podporu pro některé dávkové pokyny ze starších verzí Windows a přidává nové příkazy. V důsledku toho se může stát, že dávkové soubory, které ve Windows XP fungovaly správně, ve Windows Vista nebudou pracovat.

Navzdory omezením dávkového souboru možná budete překvapeni, jakými způsoby je lidé využívají. Pokud například zjistíte, že neustále zapomínáte, jak provádět úkoly v příkazovém řádku, vytvořte si systém nabídek s vašimi oblíbenými příkazy. Takto budete muset příkazovou informaci vyhledat pouze jednou. Následující oddíly popisují programovací možnosti dávkových souborů a obsahují několik ukázek.

**Z praxe**

### Hledání kódu na Internetu

Jsem lovec kódů. Mám více odkazů na kódy, než kolik si troufnu představit. A mám je proto, že jsou nějakým způsobem užitečné. Tato kapitola obsahuje příklady dávkových souborů, s jejichž pomocí porozumíte základním principům a budete moci provádět některé důležité operace v příkazovém řádku. Ale pokud jste jako já, budete chtít vědět víc; budete chtít příklady, které vám ukážou, jak provádět tolik úloh, že by se to do jediné knihy vůbec nevešlo. Řadu dávkový souborů pro správu a další úkoly naleznete na internetových stránkách Roba van der Woudeho na adrese *http://www.robvanderwoude.com/batexamples_0.html*. Pokud jste zvědavi na některé velmi komplexní příklady, včetně použití matematiky v dávkovém souboru (o čemž většina lidí tvrdí, že toho nelze dosáhnout), pak navštivte stránky Toma Lavedase na adrese *http://members.cox.net/tglbatch/*, věnované aplikacím založených na dávkových souborech.

## Používání příkazu Call

Příkaz `Call` použijete pro volání jiné lokace pomocí aktuálního dávkového souboru nebo pro spuštění jiného dávkového souboru. Pokud chcete zavolat jinou lokaci v témže dávkovém souboru, použijete následující návěští.

```
Call :MéNávěští
```

Volání jiného dávkového souboru je podobné. Zadáte jednotku, cestu a název dávkového souboru. Kromě toho můžete zadat argumenty příkazového řádku pro externí dávkový soubor, jak vidíte níže.

```
Call C:\MéDávkovéSoubory\MůjSoubor.BAT Argumenty
```

Volání se liší od přesunu do jiné lokace. Když je volání dokončeno, dávkový soubor se vrátí do volající lokace a provede další instrukci. Naopak pokud použijete příkaz `Goto`, dávkový soubor přenese kontrolu na novou lokaci. Vlastnost návratu příkazu `Call` vám umožňuje naprogramovat pokročilou konstrukci, která se nazývá rekurze. K rekurzi dochází, když dávkový soubor volá sám sebe. Musíte ovšem zadat únikovou cestu, jinak se dávkový soubor dostane do nekonečné smyčky.

Nejjednodušším způsobem, jak pochopit účinek příkazu `Call`, je vytvoření dvou dávkových souborů s názvy `Davka1.BAT` a `Davka2.BAT`. Zde je obsah souboru `Davka1.BAT`.

```
@ECHO OFF
Call Davka2.BAT
Call Davka2.BAT Passed %1 %PATH%
ECHO V souboru Davka 1
GOTO :EOF
ECHO Nashledanou
```

A tady je obsah souboru `Davka2.BAT`.

```
ECHO V souboru Davka 2, nashledanou
IF NOT [%1]==[] ECHO String: %1
IF NOT [%2]==[] ECHO Davka 1 Input: %2
IF NOT [%3]==[] ECHO Environment Variable: %3
```

Podívejme se na obsah souboru `Davka1.BAT`. Příklad začíná vypnutím výpisu. Tento kód standardně vkládáte do vašich dávkových souborů, aby uživatel neviděl spoustu matoucího textu, který nemá nic společného s aktuálním úkolem. Symbol @, nacházející se před příkazem `ECHO`, říká systému, aby nezobrazoval ani příkaz `ECHO`. První volání souboru `Davka2.BAT` nepředává žádnou informaci, takže soubor `Davka2.BAT` zobrazí jen zprávu „V souboru Davka 2, nashledanou". Druhé volání souboru `Davka2.BAT` předává tři druhy informací, které lze do dávkového souboru vložit: řetězec, lokální proměnnou (argument) a globální proměnnou. Kód poté zobrazí „V souboru Davka 1" a ukončí se. Povel `GOTO :EOF` je speciální; říká dávkovému souboru, aby nyní skončil. V tomto případě nemusíte definovat návěští, protože EOF je součástí příkazového procesu (detaily naleznete v oddílu „Používání příkazu *Goto*" v této kapitole).

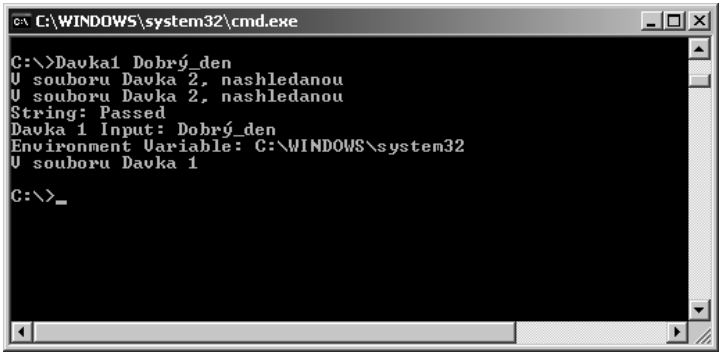
Soubor `Davka2.BAT` vždy napíše „V souboru Davka 2, nashledanou". V tomto případě povely `IF` potvrzují, že volající předal informaci dávkovému souboru. Když volající nepředá požadované informace, dávkový soubor pro daný výstup nezobrazí žádné informace. Konstrukce `[%1]==[]` je jednou z možností, jak ověřit, zda výstup je či není prázdný. Obrázek 7.5 zobrazuje výstup z této aplikace. Povšimněte si sledu událostí. První dávkový soubor zavolá druhý. Když je druhý dávkový soubor u konce, proces pokračuje dalším povelem v prvním dávkovém souboru.

Systém Windows nabízí rozšířené metody pro práci s proměnnými v dávkových souborech. Tato rozšíření vám pomohou předat specifické proměnné z vašich dávkových souborů volajícímu. Detaily o tomto postupu naleznete v oddílu „Používání nahrazení proměnných" v této kapitole.

## Používání příkazu Choice

VISTA

Pomocí příkazu `Choice` můžete přidat do dávkových souborů interaktivní procesy. To, zda tuto možnost využijete, záleží na typu automatizace, který chcete přidat do svých procesních úloh. Většina automatizací, které budete vytvářet pro optimalizační úlohy, nebude požadovat žádnou uživatelovu součinnost, protože už na základě zkušenosti s ručním provedením úlohy víte, jak má být daná úloha splněna. Ale přesto někdy budete potřebovat přidat interaktivitu. Můžete například spustit příkaz jed-




**Obrázek 7.5** Volání představuje způsob, jak provádět v dávkovém souboru podřazené úkoly a pak pokračovat hlavní úlohou

ním způsobem v pátek a jiným po zbytek týdne. Příkaz `Choice` vám také může pomoci přidat bezpečnostní prvky, které zajistí, že před provedením určité úlohy uživatel rozumí jejím následkům. V systému Windows Vista je příkaz `Choice` výrazně změněn, čímž narušuje mnoho dávkových souborů. Proto informace o tvaru příkazu `Choice` pro systém Vista naleznete dále v tomto oddílu. Pro příkaz `Choice` ve Windows XP a starších systémech jsou k dispozici následující volitelné argumenty.

***Text*** Obsahuje text, který zobrazí příkaz `Choice`, aby uživateli vysvětlil volbu. Shoduje se s kombinací příkazů `Choice` a `Echo`, ale vyžaduje jen jeden řádek kódu.

**/C:*Keys*** Určuje, že uživatel může zadat jako odpověď jeden znak. Výchozí hodnoty jsou Y (ano) a N (ne). Platné vstupní hodnoty jsou zobrazeny v závorkách. Pokud například použijete `/C:ABC`, příkaz `Choice` je v příkazovém řádku zobrazí jako `[A,B,C]`. Tuto volbu můžete vyřadit prostřednictvím přepínače `/N` – znaky zůstávají, ale `Choice` je nezobrazuje.

**/N** Říká příkazu `Choice`, aby nezobrazoval platné znaky v příkazovém řádku. Volby se objeví jako seznam písmen bez dalšího vysvětlení, což může v některých případech (seznam čísel) fungovat, ale v jiných (řada písmen) nikoli. Jedním z důvodů pro použití této volby může být zobrazení vlastního seznamu možností. Mnoho dávkových souborů vypíše znaky pro jednotlivé volby s dalším vysvětlením, například (U)končit. Uživatel stiskne U, aby ukončil činnost, ale podrobnosti snižují riziko případných nejasností.

**/S** Nastavuje vstup tak, že je citlivý na velikost písmen – standardně zachází `Choice` s velkými i malými písmeny stejně. Použití této volby zdvojnásobuje počet znaků, ze kterých lze vybírat, ale může také mást uživatele.

**/T:Character,NumberOfSeconds** Nabízí automatický výběr. Po uplynutí stanoveného počtu vteřin příkaz `Choice` automaticky zadá znak místo uživatele. Počet vteřin může být v rozsahu od 0 do 99, kdy hodnota 0 znamená, že volba bude provedena automaticky bez pauzy, takže uživatel nemůže zadat vstupní údaj. Některé dávkové soubory používají volbu 0 v situacích, kdy by uživatel standardně měl možnost volby, ale ve specifických případech nemá, například při instalaci aplikace.

Pokud použijete příkaz `Choice` bez argumentů, zobrazí jednoduchou výzvu `[Y, N]`. Z té se toho mnoho nedozvíte, pokud nepřidáte také příkaz `Echo`, aby bylo zřejmé, na co přesně má uživatel odpo-

vědět „ano“ nebo „ne“. Za běžných podmínek budete příkaz `Choice` kombinovat s jedním nebo více argumenty. Výpis 7.1 nabízí jednoduchý příklad fungování příkazu `Choice`.

**Výpis 7.1** Použití příkazu Choice

```
Echo Off

REM Opakovat, dokud uživatel nezadá E.
:Opakovat

REM Zobrazit volby.
Choice /C:ZKU /N /T:U,10 Zvolte možnost (Z)obrazit, (K)opírovat nebo (U)končit.

REM Jednat na základě uživatelovy volby.
If ErrorLevel 3 Goto Konec
If ErrorLevel 2 Goto Kopírovat
If ErrorLevel 1 Goto Zobrazit

REM Zkopírovat soubor.
:Kopírovat
Echo Rozhodli jste se zkopírovat soubor.
Goto Opakovat

REM Zobrazit soubor.
:Zobrazit
Echo Rozhodli jste se zobrazit soubor.
Goto Opakovat

REM Ukončit dávkový proces.
:Konec
Echo Nashledanou
Echo On
```

V úvodu kódu je vytvořeno návěští pro opakování, takže dávkový soubor pokračuje v činnosti, dokud jej uživatel nezastaví. Dále je použit příkaz Choice, jehož prostřednictvím vidí uživatel volby. Přepínač `/C` říká příkazu `Choice`, že platné volby jsou Z, K nebo U (namísto výchozích Y nebo N). Protože text definuje konkrétní znaky, které dávkový soubor očekává, v souboru je použit přepínač `/N`, aby se zabránilo zobrazení platných voleb znaků v příkazovém řádku. Přepínač příkazového řádku `/T` říká příkazu `Choice`, aby po 10 vteřinách automaticky zvolil U.

I když tento dávkový soubor ve skutečnosti nedělá nic se souborem, slouží jako ukázka toho, jak můžete nastavit soubor, aby postupoval podle uživatelské volby. Povšimněte si, že dávkový soubor používá pro příkaz `If` výjimku ErrorLevel, aby tak zjistil uživatelskou volbu. Tato výjimka detekuje každou volbu, která je nižší než volba zadaná uživatelem, takže musíte zadat hodnoty v opačném pořadí, jak je to znázorněno v příkladu. Navíc musíte dávkový soubor speciálně nastavit tak, aby přešel na další lokaci, protože po aktuální chybové úrovni zpracuje všechny ostatní povely.

VISTA

Procesní kód jednoduše zobrazuje řetězec, který vám říká, jakou volbu uživatel učinil. Za normálních okolností přidáváte úlohy, které by měl dávkový soubor provést na základě uživatelova výběru. Povšimněte si, že volby „Kopírovat“ a „Zobrazit“ říkají dávkovému souboru, aby se vrátil zpět na návěští „Opakovat“. Toto je nejběžnější postup, jak vytvořit smyčku v nabídce v dávkovém souboru. Dávkový soubor je ukončen tím, že se s uživatelem rozloučí a opět zapne výpis.

Příkaz Choice pro systém Vista se neliší argumenty, ale tím, jak jsou tyto argumenty kombinovány v příkazovém řádku. Následuje příkazový řádek pro systém Vista:

```
CHOICE [/C volby] [/N] [/CS] [/T vypršení /D volba] [/M text]
```

Následkem těchto změn je příkaz Choice čitelnější, ale současně je narušena funkčnost existujících dávkových souborů, a to způsobem, který nelze jednoduše opravit. Nový přepínač příkazového řádku /CS vám umožňuje nastavit volby tak, že jsou citlivé na velikost písmen, takže můžete mít k dispozici dalších dvacet šest voleb v nabídce. Ovšem povšimněte si, že přepínač /T už nezahrnuje výchozí volbu společně s hodnotou vypršení. Nový formát vyžaduje, abyste místo toho zadali volbu pomocí přepínače příkazového řádku /D. Pokud chcete doplnit volitelný text, musíte také zadat přepínač příkazového řádku /M. Následující ukázkový kód provádí stejnou úlohu, jen první část funguje v systému Windows XP (nebo starších systémech), zatímco druhá v systému Vista.

```
Starý formát příkazu Choice
CHOICE /C:N /N /T:N,15
Formát příkazu Choice v systému Vista
CHOICE /C N /N /T 15 /D N
```

**Poznámka:** Systém Vista nabízí k příkazu Choice alternativy. Utilita TimeOut poskytuje specifickou hodnotu vypršení času, aniž by uživatel musel učinit volbu. Více o této utilitě se můžete dozvědět v oddílu „Používání utility TimeOut" v této kapitole. Utilita WaitFor vám umožňuje použít signalizaci mezi systémy nebo mezi aplikacemi v témže systému. Jedna aplikace posílá signál a druhá po jeho přijetí reaguje. O této utilitě se můžete více dočíst v oddílu „Používání utility WaitFor" v této kapitole.

## Používání příkazu Echo

Příkazový řádek používá pojem *echo* (výpis) k tomu, aby popsal proces, při kterém systém vypisuje (opakuje) každý příkaz z dávkového souboru v příkazovém řádku. Tento výpis slouží jako prostředek ke sledování průběhu zpracování příkazů systémem. Ale pro uživatele, kteří nemají o běžících příkazech ponětí nebo se o ně nestarají, to může být matoucí. Kromě toho může výpis narušit vámi vytvořené vizuální efekty, například systémy nabídek. Příkaz `Echo` má dva tvary. První tvar

```
ECHO [{ON | OFF}]
```

zobrazí stav výpisu, pokud nezadáte žádné argumenty. Argument `ON` výpis zapne (takže můžete vidět příkazy), argument `OFF` výpis vypne (takže můžete vytvářet vizuální efekty). Před příkaz `Echo` můžete zadat znak @, takže se příkaz neobjeví jako jeden z vypisovaných příkazů. Zadání `@Echo OFF` vypne výpis a zároveň se tento příkaz nezobrazí v příkazovém řádku.

Pomocí druhého tvaru příkazu Echo

```
ECHO [zpráva]
```

můžete zobrazit zprávu. Jednoduše zadejte text, který za příkazem `Echo` chcete vidět. Systém pak nezobrazí příkaz `Echo`, ale jen zadanou zprávu. V tomto případě nepoužívejte znak @, jinak uživatel zprávu neuvidí.

## Používání příkazu Exit

Většina lidí si příkaz `Exit` spojuje se zavřením aktuálního příkazového okna. Použití samotného příkazu `Exit` zavře příkazové okno. Ale tento příkaz můžete také použít v dávkovém souboru k jeho ukončení. Aby se tato úloha provedla, musíte použít jeden nebo oba následující argumenty příkazu `Exit`.

**/B** Určuje, že chcete ukončit dávkový soubor, nikoli aktuální relaci příkazového řádku. Pokud tento přepínač příkazového řádku nezadáte, příkazové okno se zavře, a to i v případě, že příkaz `Exit` vyvoláte z dávkového souboru.

***ukončovací_kód*** Určuje ukončovací kód pro dávkový soubor. Výchozí ukončovací kód je 0, což standardně znamená úspěch. Ukončovací kódy můžete použít k tomu, abyste volajícího upozornili na chyby nebo zvláštní podmínky. Ukončovací kódy nejsou systémem definovány, takže můžete nastavit jakoukoli sadu těchto kódů, která je pro vaši aplikaci nezbytná.

## Používání utility ForFiles

V VISTA

Utilita ForFiles nabízí možnost, jak dokola procházet seznamem souborů a postupně s nimi pracovat. Můžete například chtít zpracovat všechny soubory, které se změnily po určitém datu. Ve většině případů tato metoda pracuje naprosto stejně jako příkaz For popsaný v oddílu „Používání příkazu *For*" v této kapitole. Příkaz ForFiles používá následující syntaxi:

```
FORFILES [/P cesta] [/M maska_hledání] [/S] [/C příkaz] [/D [+ | -] {MM/dd/yyyy | dd}]
```

Níže jsou popsány jednotlivé argumenty příkazového řádku.

**/P *cesta*** Určuje počáteční bod hledání. Cesta je výchozí složkou při hledání. Při výchozím nastavení je jako počáteční bod použit aktuální adresář.

**/M *maska_hledání*** Definuje masku hledání pro soubory. Můžete použít hvězdičku (*) a otazník (?) jako zástupné znaky, stejně jako byste to udělali při práci s příkazem Directory. Při výchozím nastavení jsou hledány všechny soubory v cílovém adresáři.

**/S** Prohledá všechny podadresáře zadaného adresáře.

**/C *příkaz*** Určuje příkaz, který chcete pro každý soubor provést. Příkaz vždy vložte do dvojitých uvozovek, abyste měli jistotu, že nebude interpretován jako součást příkazu ForFile. Výchozí příkaz je `"cmd /c echo @file"`. Před vnitřní procesový příkaz vždy napište cmd /c. Dále jsou popsány proměnné, které můžete použít jako součást příkazu.

**@file** Vrací název souboru včetně přípony.

**@fname** Vrací název souboru bez přípony.

**@ext** Vrací pouze příponu souboru.

**@path** Vrací celou cestu k souboru. Tato informace zahrnuje jednotku a skutečnou cestu.

**@relpath** Vrací relativní cestu k souboru. Relativní cesta začíná ve výchozí složce.

**@isdir** Určuje, zda je soubor typu adresář. Odpověď true značí adresář.

**@fsize** Udává velikost souboru v bajtech.

**@fdate** Udává datum, kdy byl soubor naposledy změněn.

**@ftime** Udává čas, kdy byl soubor naposledy změněn.

**Tip:** Pokud použijete formát 0xHH (kde HH je šestnáctkové číslo), můžete do příkazu vložit zvláštní znaky. Například tabulátor zadáte napsáním 0x09.

**/D *datum*** Vybere soubory, které byly naposledy změněny ve zvoleném rozmezí. Konkrétní datum zadáte pomocí formátu měsíc/den/rok (mm/dd/rrrr). Pokud přidáte před datum znaménko plus, vypíší se soubory změněné po zadaném datu, pokud minus, vypíší se soubory změněné před ním. Například při zadání /D -01/01/2001 budou vybrány všechny soubory, které byly změněny před

1. lednem 2008. Zadáním kladného nebo záporného čísla můžete také zadat relativní datum. Například při zadání /D -7 budou vybrány všechny soubory, které byly změněny během posledních sedmi dnů. Přepínač příkazové řádky /D toleruje jakoukoli hodnotu od 0 do -32,768.

## Používání příkazu For

Příkaz For zastává při programování dávkových souborů zvláštní místo. Na základě informací v rámečku „Práce se zástupnými znaky" ve 2. kapitole víte, že pokud potřebujete vybrat více souborů, můžete použít zástupné znaky. Tato možnost ale bohužel vždy nefunguje. Někdy potřebujete znát název souboru. Třeba proto, že utilita pro příkazový řádek zástupné znaky nepodporuje nebo argument souboru není s popisovací metodou zástupných znaků snadno použitelný. V takových situacích se u dávkových souborů dostává ke slovu povel For. Tento příkaz má následující podobu:

```
FOR %%proměnná IN (sada) DO příkaz [parametry_příkazového_řádku]
```

Tento příkaz můžete rovněž použít v příkazovém řádku pro ruční zpracování souborů. Místo jednoho znaku pro procento (%) použijete před proměnnou dva. Následuje ukázka, jak můžete příkaz For použít v dávkovém souboru.

```
Echo Off
For %%F In (*.BAT *.TXT) Do Dir %%F /B
Echo On
```

V tomto případě příkaz `For` zpracuje všechny soubory v aktuálním adresáři, které mají příponu BAT nebo TXT. Příkaz zpracuje soubory v tom pořadí, v jakém se nacházejí v adresáři; toto pořadí není garantováno. Proměnná `%%F` obsahuje název individuálního souboru. Příkaz `Dir` je volán jedenkrát pro každý soubor, vstupním údajem je proměnná `%%F`. V tomto případě má výstup příkazu podobu názvu souborů v holém formátu, takže tento dávkový soubor můžete použít k vytvoření textového souboru se seznamem souborů, které odpovídají zadaným kritériím. Další procesy mohou soubory archivovat nebo provést cokoli jiného, co potřebujete. Příkaz `For` používá následující argumenty.

**{%*proměnná* | %%*proměnná*}** Určuje nahraditelný parametr; argument, který obdrží jednotlivé členy sady. Nahraditelný parametr má dva tvary. Tvar `%proměnná` použijte, pokud chcete použít nahraditelný parametr jako výstup pro jiný příkaz nebo utilitu. Tvar `%%proměnná` použijte v případě, kdy chcete nahraditelný parametr použít pro práci v dávkovém souboru. Názvy prosměnných jsou citlivé na velká a malá písmena, tzn. `%f` není to samé jako `%F`. Navíc musíte používat jen abecední názvy proměnných, např. `%A`, `%B` nebo `%C`.

**(*Sada*)** Definuje sadu, která bude zpracována. Sada může obsahovat jeden nebo více souborů či adresářů, rozsah hodnot nebo textové řetězce, které chcete daným příkazem zpracovat. Jako sadu můžete použít například proměnné prostředí. Příkaz `For %%P In (%PATH%) Do Echo %%P` zobrazí členy systémového prostředí `PATH` jako jednotlivé řetězce.

***příkaz*** Určuje příkaz, kterým chcete zpracovat všechny záznamy v sadě.

***argumenty_příkazového_řádku*** Určuje argumenty příkazového řádku pro příkaz, kterým chcete zpracovat všechny záznamy v sadě. Argumenty příkazového řádku se vztahují k jednotlivým příkazům a utilitám; podrobnosti naleznete u dalších údajů v této knize.

### Provádění komplexní iterace souboru

Příkaz `For` můžete použít ke zpracování příkazového výstupu, řetězců a obsahu souborů. V takovém případě začíná příkaz `For` tím, že vstup rozdělí do jednotlivých řádků a vypustí všechny prázdné řádky. Dále rozdělí poskytnutá vstupní data na základě vámi stanovených pravidel do konkrétních

tokenů. Tokenem může být kontrolní znak, zvláštní slovo nebo cokoliv jiného, co zadáte jakou součást jednoduché syntaxe tohoto příkazu. Příkaz `For` předá token příkazu, který určíte jako výstupní. Následuje syntaxe příkazového řádku pro tento tvar příkazu `For`.

```
for /F ["klíčová_slova"] {%% | %}proměnná in (sada_názvů_souborů) do příkaz
   [argumenty_příkazového_řádku]
for /F ["klíčová_slova"] {%% | %}proměnná in ("řetězcový_literál") do příkaz
   [argumenty_příkazového_řádku]
for /F ["klíčová_slova"] {%% | %}proměnná in ('příkaz') do příkaz
   [argumenty_příkazového_řádku]
```

Povšimněte si, že pro každý typ výstupu je zapotřebí použít odlišnou syntaxi příkazu. Sada souborů je zadána bez uvozovek, zatímco řetězec je v dvojitých uvozovkách a příkaz v uvozovkách. Tyto malé rozdíly ve formátu příkazu určují, jak příkaz `For` nahlíží na vstupní údaje.

Vstup klíčová_slova je řetězec v uvozovkách, který stanoví pravidlo pro rozdělení vstupu do tokenů. Tato klíčová slova jsou vždy uvedena ve dvojitých uvozovkách, jak můžete vidět výše. Následující seznam popisuje klíčová slova, která můžete použít.

**eol=*c*** Určuje znak pro ukončení řádku. Příkaz `For` umožňuje určit jen jeden takový znak.

**skip=*N*** Určuje počet řádek, které mají být na začátku souboru přeskočeny.

**delims=*xxx*** Určuje sadu oddělovačů. Tato sada definuje, na které znaky příkaz `For` nahlíží jako na prvky mezi tokeny. Výchozí nastavení spoléhá na mezerník a tabulátor. To znamená, že příkaz `For` vygeneruje nový token pokaždé, když ve vstupních datech narazí na mezeru nebo tabulátor.

**tokens=*X,Y,M-N*** Stanoví, které tokeny mají být z každého řádku textu předány příkazu `For` pro jednotlivé iterace. Příkaz `For` přiřadí každému tokenu jednu proměnnou. Formát M-N definuje rozsah tokenů, které mají být použity na vstupu. Kdykoli je posledním znakem ve zpracovávaném řetězci hvězdička (*), příkaz `For` vytvoří další proměnnou, které bude přiřazen další text na řádce poté, co příkaz zanalyzuje poslední token.

**usebackq** Určuje použití nové sémantiky, kdy řetězec ve zpětných uvozovkách představuje příkaz a řetězec v jednoduchých uvozovkách představuje doslovný řetězec příkazu a umožňuje použít dvojité uvozovky pro názvy souborů v sadě názvů souborů.

Při využívání možnosti usebackq musíte použít mírně odlišnou syntaxi příkazu `For`. Následují tři příkazové řádky s touto syntaxí.

```
for /F ["usebackq_klíčová_slova"] {%% | %}proměnná in ("sada_názvů_souborů") do
   příkaz [argumenty_příkazového_řádku]
for /F ["usebackq_klíčová_slova"] {%% | %}proměnná in ('řetězcový_literál') do
   příkaz [argumenty_příkazového_řádku]
for /F ["usebackq_klíčová_slova"] {%% | %}proměnná in ('příkaz') do příkaz
   [argumenty_příkazového_řádku]
```

## Používání nahrazení proměnných

Nahrazením proměnných se rozumí záměna názvu proměnné za její obsah. Ovšem na rozdíl od rozšíření není nezbytně nutné použít celý obsah proměnné. Například místo zadání úplné cesty k souboru stačí použít jen písmeno jednotky, cestu nebo název souboru. V následujícím seznamu jsou popsány základní tvary nahrazení proměnných, které jsou dostupné pro příkaz `For` (předpokládá se, že používáte proměnou s názvem `I`, což je v příkazovém řádku přepsáno jako `%I`).

**%~I** Odstraní z obsahu proměnné všechny uvozovky.

**%~fI** Rozšíří proměnnou %I o úplný kvalifikovaný název cesty.

**%~dI** Rozšíří proměnnou %I pouze o písmeno jednotky.

**%~pI** Rozšíří proměnnou %I pouze o cestu.

**%~nI** Rozšíří proměnnou %I pouze o název souboru.

**%~xI** Rozšíří proměnnou %I pouze o příponu souboru.

**%~sI** Vytvoří proměnnou pro cestu a pak změní jakékoli dlouhé názvy adresářů na jejich zkrácené ekvivalenty.

**%~aI** Získá argumenty vstupního souboru.

**%~tI** Získá datum a čas vstupního souboru.

**%~zI** Získá velikost souboru.

**%~$PATH:*I*** Hledá v adresářích uvedených v systémové proměnné `PATH` soubor zadaný proměnnou I. Systém poté rozšíří první nalezený shodný výskyt na úplný kvalifikovaný název souboru, který obsahuje jednotku, cestu a název souboru. Pokud systémová proměnná `PATH` není definována nebo pokud systém nemůže nalézt shodný název souboru, toto nahrazení proměnné vrací prázdný řetězec.

Nahrazení proměnných můžete kombinovat, abyste dosáhli specifických výsledků. Dejme tomu, že chcete vytvořit výpis souborů, podobně jako v adresáři. V následujícím seznamu je uvedeno několik nápadů, jak kombinovat argumenty nahrazení proměnných.

**%~dpI** Výstupem je pouze písmeno jednotky a cesta k souboru, chybí název souboru.

**%~nxI** Výstupem je název souboru a přípona, chybí písmeno jednotky a cesta.

**%~fsI** Výstupem je informace o souboru, název souboru je pouze ve zkráceném tvaru (8.3).

**%~dp$PATH:*I*** Lokalizuje soubor pomocí systémové proměnné `PATH`. Výstupem ze systému je pouze písmeno jednotky a cesta k prvnímu nalezenému odpovídajícímu souboru.

**%~ftzaI** Vytvoří stejný výstup jako příkaz `Dir`. Tvar výstupu se ovšem od příkazu `Dir` liší, protože ve výpisu souborů může být zahrnuto více adresářů. Výstupy se liší v zaměření na výpis.

## Používání příkazu Goto

Příkaz `Goto` přenáší kontrolu z jedné části dávkového souboru do jiné. Tento příkaz nelze použít k přenesení kontroly mezi různými dávkovými soubory; v takovém případě použijete příkaz `Call`, který je popsán v oddílu „Používání příkazu *Call*“ v této kapitole. Příkaz `Goto` má jednoduchý tvar: `Goto Návěští`, kde `Návěští` je klíčové slovo, které slouží k určení přechodového bodu v dávkovém souboru. Před návěštím je vždy dvojtečka, např. `:MéNávěští`. Výpisy č 7.1 a 7.2 ukazují příkaz `Goto` v akci.

## Používání příkazu If

Abyste mohli napsat dostatečně komplexní dávkový soubor, potřebujete zajistit řízení toku – aktivní výběr kódu, který má za aktuálních podmínek běžet. Například můžete chtít před zahájením další úlohy vědět, zda předchozí úloha proběhla správně. V některých případech budete hledat specifický soubor nebo postupovat na základě vstupu do dávkového souboru zadaného uživatelem. Můžete také ověřit, zda uživatel zadal určitý vstupní řetězec. Smyslem je provádění kontroly nad tím, jak dávkové soubory reagují na podmínky systému a prostředí. Dávkové sobory neposkytují žádnou rozsáhlou podporu pro rozhodování, ale pro zlepšení jejich flexibility můžete použít následující tři tvary povelu `If`.

```
If [Not] ErrorLevel číslo příkaz
If [Not] řetězec1==řetězec2 příkaz
If [Not] Exist název_souboru příkaz
```

Ve všech třech případech můžete přidat slovo Not, aby kontrola proběhla v opačném pořadí. Například můžete provést úlohu s předpokladem, že soubor neexistuje, namísto předpokladu, že existuje. Kombinací obou verzí povelu `If` můžete vytvořit obdobu povelu `If...Else`, který se vyskytuje ve většině programovacích jazyků.

Argument `ErrorLevel` vyžaduje zvláštní rozvahu. Kdykoli spustíte aplikaci, dávkový soubor nebo skript, systém na výstupu oznámí číselnou chybovou úroveň. Standardem je, že chybová úroveň 0 vždy znamená úspěch. Ostatní čísla znamenají chybu nebo zvláštní podmínku. Zvláštní podmínka nemusí být vždy chybou, jen to zkrátka není úplný úspěch. Příklad příkazu, který se ukončí se zvláštními podmínkami, naleznete v oddílu „Používání příkazu *Choice*" v této kapitole. Chybové podmínky naznačují chybu uživatele, systému nebo aplikace. Vezměte v potaz například chybové úrovně utility Xcopy uvedené v tabulce 7.4.

**Tabulka 7.4** Chybové úrovně utility XCopy

| Chybová úroveň | Význam |
|---|---|
| 0 | Úspěch, nedošlo k žádné chybě. |
| 1 | Systém nenalezl žádné soubory ke zkopírování. |
| 2 | Uživatel zastavil XCopy stisknutím kláves Ctrl+C. |
| 4 | Při spuštění aplikace došlo k chybě. Systém nemá dostatek paměti nebo místa na disku. Možná jste zadali neplatné jméno jednotky nebo jste použili v příkazovém řádku neplatnou syntaxi. |
| 5 | Systém detekoval chybu při zápisu na disk. |

Jak můžete vidět, příčiny chyb se v závislosti na podmínkách značně liší. Ve všech případech je ale oprávněné tvrdit, že v aplikaci došlo k chybě. Ovšem povšimněte si, že chybová úroveň 2 se mohla ve skutečnosti vyskytnout kvůli zadání. Uživatel pozná chybu a stiskne Ctrl+C, aby proces kopírování zastavil ještě před jeho dokončením. V takovém případě je zapotřebí zvážit, zda chybová úroveň určuje zvláštní podmínku nebo chybu, a vyzvat uživatele k provedení odpovídajících kroků. Výpis 7.2 ukazuje příklady různých tvarů povelu `If` v akci.

**Výpis 7.2** Použití povelu If v dávkových souborech

```
Echo Off

REM Ověřuje, zda uživatel provedl akci.
If %1Err==Err GoTo ChybaProcesu

REM Simuluje chybu, pokud soubor neexistuje.
Copy MujSoubor.TXT MujSoubor2.TXT
If Not ErrorLevel 1 Goto KontrolaSouboru
   Echo Soubor neexistuje,
   Echo takže jej dávkový soubor nemůže zkopírovat.

REM Vyhledá zadaný soubor,
REM a pokud existuje, zpracuje jej.
```

```
:KontrolaSouboru
If Exist MujSoubor.TXT Goto ZpracovaniSouboru

REM Pokud soubor neexistuje, vytvořte jej.
REM Zobrazí zprávu s instrukcemi
REM a pak nechá uživatele zadat text.
Echo Zadejte text pro testovací soubor
Echo a pak stiskněte Ctrl+Z.
Pause
Copy CON MujSoubor.TXT

REM Toto je návěští pro zpracování souboru.
:ZpracovaniSouboru

REM Určuje, zda uživatel chce zobrazit soubor.
If Not %1==zobrazit Goto Zpracovani2
   Echo MujSoubor.TXT obsahuje:
   Type MujSoubor.TXT
   Goto Konec

REM Určuje, zda uživatel chce smazat soubor.
:Zpracovani2
If Not %1==smazat Goto ChybaProcesu
   Erase MujSoubor.TXT
   Echo MujSoubor.TXT smazán
   Goto Konec

REM Uživatel neurčil akci.
:ChybaProcesu
Echo Neurčili jste, co má dávkový soubor udělat!
Echo Pokud chcete soubor zobrazit, napište UseIf Zobrazit.
Echo Pokud chcete soubor smazat, napište UseIf Smazat.

:Konec
Echo On
```

První řádka tohoto příkladu znázorňuje princip, který byste měli vždy používat v dávkových souborech, jež bude používat někdo jiný – v rámci dávkového souboru provádějte kontrolu chyb. V tomto případě dávkový soubor očekává od uživatele zadání vstupní hodnoty `delete` nebo `display`. Pokud uživatel vstupní hodnotu nezadá, pak je první vstupní hodnota %1 prázdná, řetězec `Err` se tedy rovná `Err` a kód přejde na návěští nazvané `ChybaProcesu`. Dávkové soubory mohou při použití proměnných `%1` až `%9` pracovat s až devíti vstupními hodnotami současně. Povel `Goto` vždy říká kódu, aby přešel na dané návěští v dávkovém soboru. Návěští určíte tak, že před jeho název napíše dvojtečku, např. `:ChybaProcesu`.

Další část kódu se pokouší zkopírovat dočasný soubor do jiného souboru. Výsledkem této operace je chyba, kterou můžete zachytit použitím povelu `ErrorLevel`, když soubor neexistuje. Pokud hodnota `ErrorLevel` odpovídá vámi zadané hodnotě, pak povel `If` příkaz provede. Protože v tomto případě kód používá podmínku `Not`, je platnou hodnotou opak, tedy povel `If` provede příkaz `Goto` jen tehdy, pokud chybová úroveň není rovna 1. Povšimněte si, že v tomto případě kód používá příkaz `Echo` ke zobrazení chybové hlášky uživateli – `Echo` funguje nejen pro zapínání a vypínání zobrazo-

vání zpráv, ale také pro zobrazování nadefinovaných zpráv uživateli, které nastavení příkazu `Echo` rovněž neskryje.

Jakmile kód provede tyto úvodní kroky, určuje pomocí podmínky `Exit` příkazu `If`, zda `MujSoubor.TXT` existuje. Pokud soubor existuje, kód jej okamžitě zpracuje. V opačném případě zobrazí kód výzvu, aby uživatel zadal informace o souboru. Povšimněte si příkazu `Pause`, který přeruší běh dávkového souboru do doby, než uživatel stiskne klávesu. Příkaz `Copy` odesílá informace zadané uživatelem přes konzolu (`CON`) do `MujSoubor.TXT`, a to až do chvíle, kdy detekuje znak určující konec souboru; tento znak uživatel vytvoří stisknutím kláves Ctrl+Z.

Když je zřejmé, že soubor existuje, dávkový soubor jej může zpracovat. Tento dávkový soubor poskytuje dvě možnosti: zobrazení nebo smazání daného souboru. Problém u dávkových souborů je v tom, že porovnání řetězce je citlivé na velká a malá písmena – slovo `smazat` se liší od `Smazat`, takže zachycení chyby může být ve skutečnosti omylem. Někteří vývojáři tento problém řeší tak, že v dávkových souborech používají pro přepínače příkazového řádku jen jeden znak. Pak stačí provést jen dvě kontroly, jednu pro velké a druhou pro malé písmeno. Příklad výše používá kvůli názornosti celé slovo. Abyste viděli, jak to funguje, napište do příkazového řádku `Smazat` místo `smazat` – kód zobrazí chybovou hlášku. Pokud uživatel napíše `smazat`, dávkový soubor zadaný soubor smaže a zobrazí potvrzení. Podobné je to v případě, kdy uživatel napíše `zobrazit` – kód zobrazí obsah `MůjSoubor.TXT`. V obou případech kód přejde na `Konec`, kde dávkový soubor opět zapne `Echo`.

Zatím jsme se v této kapitole zabývali standardním tvarem příkazu `If`, který můžete spustit dokonce i v příkazové řádce v systému DOS. Příkaz `If` má následující dodatečné tvary syntaxe pro příkazová rozšíření.

```
if [/I] řetězec porovnávací_operátor řetězec2 příkaz [else výraz]
if CMDEXTVERSION číslo příkaz [else výraz]
if DEFINED proměnná příkaz [else výraz]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/I** Provádí porovnání dvou řetězců; porovnání rozlišuje velké a malé znaky. Tento prvek je užitečný, když chcete, aby uživatel zadal řetězec, ale nevíte, jakými písmeny jej napíše. Tato porovnání jsou generická v tom smyslu, že pokud je řetězec1 i řetězec2 tvořen čísly, systém převede řetězce na čísla a provede číselné porovnání.

**řetězec1** Určuje vstupní řetězec, první polovinu porovnání.

**porovnávací_operátor** Určuje porovnávacího operátora. Každý z níže uvedených operátorů provádí odlišené porovnání, jak je uvedeno v následujícím seznamu.

- **EQU** Je rovno
- **NEQ** Není rovno
- **LSS** Je menší než
- **LEQ** Je menší než nebo rovno
- **GTR** Je větší než
- **GEQ** Je větší než nebo rovno

**řetězec2** Určuje porovnávací řetězec, druhou polovinu porovnání.

***příkaz*** Určuje příkaz, který chcete provést, pokud je porovnání v pořádku.

**else *výraz*** Určuje výraz „else“ příkazu `If`. Pokud použijete tuto syntaxi, musíte části povelů `If` a `Else` napsat do kulatých závorek. Kromě toho musí být celý povel na jednom řádku. Různé elementy nelze oddělit (takže nelze vylepšit vzhled). Zde je příklad tohoto tvaru příkazu `If`.

```
IF [%1] EQU [] (ECHO Řetězec je prázdný) ELSE (ECHO Řetězec obsahuje data)
```

V tomto případě příkaz `If` provede kontrolu, zda vstup obsahuje řetězec pro první proměnnou. Pokud je k dispozici vstup, výstup sdělí uživateli, že řetězec obsahuje data. V opačném případě vstup zobrazí jako výstup hlášení „Řetězec je prázdný“.

**CMDEXTVERSION *Číslo*** Kontroluje specifickou verzi rozšíření příkazu. Pokud je číslo verze rozšíření příkazu rovno nebo větší než zadané číslo, podmínka je pravdivá. Tento tvar příkazu `If` při vypnutých rozšířeních příkazu nikdy podmínku nevyhodnotí jako pravdivou.

**DEFINED *Proměnná*** Kontroluje, zda máte zadánu specifickou proměnnou prostředí. Argument `DEFINED` funguje stejně jako argument `EXISTS`. Příkaz `If` vyhodnotí podmínku jako pravdivou, jestliže je proměnná nadefinována.

## Používání příkazu Pause

Příkaz `Pause` zastaví běh dávkového souboru, aby měl uživatel čas zareagovat na určitou situaci, například pokud kvůli dokončení běhu dávkového souboru potřebujete, aby uživatel vyměnil médium. V takovém případě můžete použít příkaz `Echo`, který uživatele na tuto potřebu upozorní, a pak příkaz `Pause`, abyste uživatele vyzvali ke stisknutí klávesy poté, co médium vymění.

## Používání příkazu Prompt

Příkaz `Prompt` mění podobu výzvy v příkazovém řádku. Například místo obvyklého písmena jednotky, adresáře a znaku „větší než“ (>) můžete použít jako výzvu čas a datum. Výzva může vlastně obsahovat jakýkoli text. Abyste výzvu změnili, jednoduše napište `Prompt` a dále text, který chcete zobrazit, a stiskněte Enter. V následujícím seznamu jsou uvedeny zvláštní znaky, které můžete použít jako součást příkazového řádku.

**$A** & (ampersand)

**$B** | (svislá čára)

**$C** ( (levá kulatá závorka)

**$D** Aktuální datum

**$E** Kód Escape (ASCII kód 27)

**$F** ) (pravá kulatá závorka)

**$G** > (znak „větší než“)

**$H** Backspace (maže předchozí znak)

**$L** < (znak „menší než“)

**$N** Aktuální jednotka

**$P** Aktuální jednotka a cesta

**$Q** = (rovnítko)

**$S** (mezera)

**$T** Aktuální čas

**$V** Číslo verze systému Windows

**$_** Znaky ENTER a LINEFEED

**$$** $ (symbol dolaru)

Pokud jsou povolena rozšíření příkazu, můžete použít dva další formátovací znaky, které jsou popsány v následujícím seznamu.

**$+** Žádný nebo více znaků plus (+) podle hloubky zásobníku adresářů PUSHD. Každý znak plus odpovídá jedné uložené úrovni.

**$M** Zobrazí vzdálený název spojený s písmenem aktuální jednotky. Pokud je to místní jednotka, pak systém zobrazí prázdný řetězec.

## Používání příkazu Rem

Příkaz `Rem` (`Remark`) vám umožní přidávat do vašich dávkových souborů komentáře. Jelikož jsou v dávkových souborech často použity sekvence kódů, které jsou obtížně srozumitelné a které budete pravděpodobně chtít časem změnit, je vhodné používat dostatečné množství komentářů. Vlastně byste měli ve vašem dávkovém souboru přidat aspoň jeden komentář ke každé komplexní řádce s kódem. Mnoho lidí ztratilo zájem o zajímavé a užitečné dávkové soubory, protože obsahují složitý kód, který se stane nesrozumitelným, jakmile jeho autor zapomene, co znamená.

## Používání příkazu Shift

Dávkový soubor podporuje nanejvýš 10 argumentů příkazového řádku, které jsou číslovány od `%0` do `%9`. Ovšem můžete se dostat do situace, kdy budete potřebovat více než deset argumentů. Příkaz `Shift` vám může tyto dodatečné argumenty pomoci posunout. Existující argumenty jsou nahrazeny stávajícími. Ve skutečnosti jsou všechny argumenty o jedno místo posunuty, takže argument `%1` se pak zobrazuje jako `%0`. Bohužel argument `%0` je takto vyřazen, takže není dostupný.

Pokud máte povolená rozšíření příkazu, můžete uchovat některé starší argumenty v paměti. Příkaz `Shift` akceptuje číselný argument, který určuje, na kterém místě má posun argumentů začít. Pokud například příkaz bude mít podobu `Shift /2`, hodnoty `%0` a `%1` nebudou ovlivněny. Ovšem argumenty od `%2` dále budou posunuty, takže `%2` bude obsahovat hodnotu z `%3`.

## Používání utility TimeOut

VISTA

Utilita TimeOut nabízí unikátní vlastnost; můžete nastavit délku čekací lhůty na určitou hodnotu, bez ohledu na to, co uživatel dělá. Takže na rozdíl od příkazu Command platí, že pokud řeknete utilitě TimeOut, aby čekala 30 vteřin, čeká celou dobu, i když uživatel stiskne klávesu. Kromě toho utilita TimeOut nezobrazuje zprávu o vypršení lhůty, takže ji můžete použít i tam, kde je zapotřebí tichý běh (například u úlohy na pozadí). Tento příkaz používá následující syntaxi:

```
TIMEOUT [/T] časový_limit [/NOBREAK]
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/T *časový_limit*** Určuje hodnotu časového limitu. Tento přepínač příkazového řádku je volitelný. Můžete zadat jakoukoli hodnotu od -1 do 99 999 vteřin. Hodnota -1 znamená, že utilita bude čekat po nekonečně dlouhou dobu na stisknutí klávesy. Utilita neumožňuje kombinovat hodnotu -1 s přepínačem příkazového řádku /NOBREAK, protože tak by došlo k účinnému zablokování systému.

**/NOBREAK** Brání utilitě rozpoznat stisknutí klávesy, takže čeká na uplynutí stanoveného limitu, než se ukončí.

## Používání utility WaitFor

VISTA

Utilita WaitFor umožňuje komunikaci mezi procesy. Můžete poslat signál z jedné aplikace do druhé. Tuto možnost můžete použít také pro signalizaci mezi dávkovými soubory. Pokud použijete tuto utilitu, spusťte nejprve přijímající aplikaci a nařiďte jí, ať čeká na signál. Odesílatel pošle signál, až je připraven. Tento příkaz používá následující syntaxi:

```
Pro odesílatele:
WAITFOR [/S systém [/U [doména\]uživatel [/P [heslo]]]] /SI signál
Pro příjemce:
WAITFOR [/T časový_limit] signál
```

V následujícím seznamu jsou popsány jednotlivé argumenty příkazového řádku.

**/S *systém*** Určuje vzdálený systém, který chcete zkontrolovat. Ve většině případů bude při použití tohoto přepínače zapotřebí zadat také přepínače `/U` a `/P`.

**/U [*doména*\]*uživatel*** Určuje jméno uživatele ve vzdáleném systému. Toto jméno se může lišit od jména v místním systému. Při práci s doménovým kontrolérem musíte zadat název domény.

**/P [*heslo*]** Určuje heslo pro daného uživatele. Tento přepínač příkazového řádku můžete do příkazového řádku zadat i bez upřesnění hesla v prostém textu; systém vás vyzve k jeho napsání. Tato možnost vám může pomoci při správě zabezpečení hesel používaných ve vašem systému.

**/SI** Posílá požadovaný signál přes síť.

***signál*** Určuje signál, který se má poslat nebo přijmout. Tento signál je jednoduchou hodnotou řetězce, např. Start-Setup. Systém může čekat na více unikátních signálů. Maximální délka názvu signálu je 255 znaků. Hodnotu řetězce můžete sestavit ze znaků a–z, A–Z, 0–9 a z jakéhokoli ASCII znaku v rozmezí od 128 do 255. Hodnota řetězce nemůže obsahovat zvláštní znaky nebo mezery.

**/T časový_limit** Určuje dobu, po kterou se má čekat na signál. Můžete zadat jakoukoli hodnotu od -1 do 99 999 vteřin. Při výchozím nastavení je doba čekání stanovena jako nekonečně dlouhá.

# Začínáme s úlohami v příkazovém řádku

V této kapitole jsme si ukázali, jaké funkce mohou nabídnout dávkové soubory. Pokud pracujete v prostředí příkazového řádku hodně často, měli byste mít řadu dávkových souborů, které rozšiřují sadu vašich nástrojů. Dávkové soubory jsou velmi jednoduchým typem programů, které mohou vést k pozoruhodným výsledkům, protože spoléhají na možnosti účinných příkazů, které systém Windows v příkazovém řádku nabízí. Řadu nástrojů, jejichž použitím se mohou dávkové soubory v příkazovém řádku stát ještě účinnějšími, naleznete v této knize ve dvanácté, třináctém a čtrnácté kapitole. Tato kapitola měla posloužit jako úvodní pojednání o tom, co může každý dělat s dávkovými soubory (neboť každý má k těmto prvkům přístup). Dávkové soubory mohou být celkem účinné.

Nyní je na vás, abyste sami zkusili dávkové soubory psát. Začněte jednoduše. Zkuste napsat dávkové soubory, s jejichž pomocí nastavíte příkazový řádek podle konkrétních potřeb. Můžete vyzkoušet něco jednoduchého, třeba změnit barvu příkazového řádku podle své aktuální nálady. Smyslem je vyzkoušet nejdříve jednoduchá schémata a postupně se propracovat ke složitějším úkolům. Značný

počet úkolů, které potřebujete provádět každý den kvůli správě sítě nebo přímo práci s jedním počítačem, můžete se správně zvolenými nástroji zautomatizovat.

Jakkoli účinné mohou dávkové soubory ve správných rukou být, nedokáží provést některé základní programovací úlohy. Dávkový soubor je omezen příkazy a utilitami, které jsou dostupné v příkazovém řádku. V osmé kapitole je nastíněn další krok v automatizaci použití příkazového řádku – použití skriptů. Skript může nabídnout lepší programátorskou podporu, lze dokonce docílit určité úrovně ladění chyb ve skriptech. Skript také přináší zásadní věc, kterou dávkový soubor nemá – pěkné uživatelské rozhraní. Ačkoliv jsou dávkové soubory dostačující pro zkušené uživatele a administrátory, nejsou tou nejlepší volbou pro nováčky. Skriptovací techniky popsané v osmé kapitole vám pomohou zlepšit dojem z práce s příkazovým řádkem u každého, dokonce – s těmi správnými doplňky – i pro začátečníky.