Další Předchozí Obsah

2. Práce se soubory a adresáři

Následující sekce popisují obecné příkazy pro správu souborů a vysvětlují jejich použití. Existuje řada dalších příkazů; ty zde uvedené by vám měly pomoci do začátku.

2.1 Výpis obsahu adresáře - příkaz ls

Pro zjištění toho, jaké soubory jsou uloženy v adresáři slouží příkaz `ls`. Základní syntaxe je následující:

```
$ ls [-volba]... [jméno souboru nebo adresáře]
```

Volby určují typ zobrazovaných informací a způsob jejich zobrazení. Pokud neuvedeme žádnou volbu zobrazí se tzv. krátký výpis. Jinak můžeme použít následujících voleb:

`-a`	Vypíše všechny soubory v adresářích včetně těch, jejichž jména začínají tečkou.
`-C`	Vypisovat soubory do sloupců tříděných svisle.
`-d`	Vypíše informace o zadaných adresářích nikoli o jejich obsahu.
`-F`	Za jména souborů připojit znak indikující typ souboru. Pro proveditelné soubory znak `*', `/' pro adresáře, `@' pro symbolické linky, `|' pro pojmenované roury (FIFO), `=' pro sokety, a nic pro normální soubory.
`-i`	Bude vypisovat čísla i-uzlů.
`-l`	Kromě jména souboru se bude vypisovat jeho typ, práva, počet pevných odkazů na soubor, jméno vlastníka, skupiny, velikost v bytech a časový údaj (čas poslední změny, pokud nebyl vybrán jiný údaj). U souborů, jejichž časový údaj je starší než 6 měsíců nebo novější než -1 hodina, bude místo hodiny a minuty vypsán rok.
`-r`	Výpis bude setříděn v obráceném pořadí.
`-R`	Vypíše obsah všech adresářů rekurzivně (včetně podadresářů).
`-t`	Setřídí výpis obsahu adresáře podle časového údaje a nikoli podle abecedy. Nejnovější soubory budou vypsány první.

2.2 Kopírování souborů - příkaz cp

Pro provádění kopií souborů slouží příkaz `cp`. Syntaxe je následující:

```
$ cp původní_soubor[y] cíl
```

Pokud je poslední parametr jméno existujícího adresáře, `cp` bude kopírovat zadané soubory pod původními jmény do zadaného adresáře. Jinak pokud jsou zadána pouze dvě jména, zkopíruje první soubor do druhého.

2.3 Přesun (přejmenování souboru) - příkaz mv

Pro přesun souboru na nové místo slouží příkaz `mv`. Syntaxe je následující:


```
$ mv původní_soubor[y] cíl
```

Pokud je poslední parametr jméno existujícího adresáře, `mv` přemístí zadané soubory pod původními jmény do zadaného adresáře. Jinak pokud jsou zadána pouze dvě jména, přemístí první soubor do druhého. Je chybou, pokud jsou zadána více než dvě jména a poslední parametr není adresář. Mezi různými svazky je možné přemísťovat pouze obyčejné soubory.

2.4 Mazání souboru - příkaz rm

K mazání souborů slouží příkaz `rm`.

```
$ rm [volba] soubor
```

Argument volba může nabývat následujících hodnot:

`-f`	Bude ignorovat neexistující soubory a nebude se ptát uživatele.
`-i`	Bude vyžadovat potvrzení před smazáním jednotlivých souborů. Pokud odpověď uživatele nezačíná znakem `y' nebo `Y', soubor nebude smazán.
`-r`, `-R`	Bude rušit zadané adresáře i s jejich obsahem.

2.5 Prohlížení obsahu souboru - příkazy more, less

Pro zobrazení obsahu souboru po obrazovkách slouží příkazy `more` a `less`. Syntaxe je následující:

```
$ more soubor[y]
$ less soubor[y]
```

2.6 Spojování souborů - příkaz cat

`cat` je zkratka pro *concatenate* (spojovat). Původně byl tento příkaz zamýšlen pro spojování více textových souborů do jednoho, ale používá se i k dalším účelům.

Ke spojení dvou souborů do jednoho jednoduše napište jejich názvy za `cat` a výstup přesměrujte do souboru. Příkaz `cat` pracuje se standardním vstupem a standardním výstupem, takže musíte použít shellovské znaky pro přesměrování. Například:

```
$ cat file1 file2 file3 > bigfile
```

Můžete `cat` používat i k prostému zobrazní obsahu textového souboru na obrazovce monitoru.

```
$ cat soubor
```

2.7 Zobrazení začátku/konce souboru - příkazy head, tail

V některých případech nechceme vidět celý soubor, ale zajímá nás pouze několik prvních resp. posledních řádků souboru. V takovém případě můžeme použít utility `head` a `tail`. Syntaxe je následující:


```
$ head [-n] soubor
$ tail [+-číslo] [volba] soubor
```

Argument `-n` udává, kolik řádků souboru se má zobrazit. Pokud není uveden, zobrazí se prvních 10 řádků.

V případě příkazu `tail` jsou argumenty následující:

`číslo`	Pokud není tento argument uveden, utilita zobrazuje posledních 10 řádků. Pokud je před číslem znaménko `+`, zobrazí zadaný počet jednotek od začátku souboru. Prokud je argumentem znaménko `-`, zobrazí zadaný počet jednotek před koncem souboru.
`volba`	Tento argument určuje, v jakých jednotkách se chápe argument `číslo`. Možné hodnoty jsou: `b` - počítáme po blocích `c` - počítáme po znacích `l` - počítáme po řádkách

2.8 Filtry: tee, sort, uniq, wc, tr

Programy označované jako filtry obvykle pracují se standardním vstupem a standardním výstupem. Nejčastěji se používají v kolonách s dalšími příkazy. Mezi filtry lze zařadit příkazy `tee` (uložení mezivýsledku do souboru), `sort` (setřídění vstupu), `uniq` (vyloučení duplicit), `wc` (počítání slov, řádků, znaků), `tr` (záměna znaků) a některé další.

```
$ tee soubor
```

Program **tee** kopíruje standardní vstup na standardní výstup a současně zapisuje do uvedeného souboru. Používá se pro uložení mezivýsledku ve vícenásobném propojení procesů přes roury.

```
$ sort [+číslo] [přepínače] [soubor]
```

Příkaz setřídí standardní vstup, popř. uvedený soubor. Standardně se třídí znakově, vzestupně. Třídění probíhá na základě rozčlenění řádků do sloupců (údaj z 1. sloupce je první klíč, z druhé sloupce je druhý klíč ...).

`-n`	setřídění numerické (pokud jsou číselné hodnoty)
`-r`	setřídění sestupné
`-f`	nebudou se rozlišovat malá a velká písmena (velká písmena se převádí na malá)
`+číslo`	určuje sloupec, který se má použít pro třídění. Sloupce se číslují od 0. Lze definovat více sloupců (např. zápis +3 +1 určuje, že primární klíč je ve 4. sloupci, sekundární klíč ve 2. sloupci). K číslu sloupce mohou být připojeny písmena `n`, `r` či `f`, které ovlivňují způsob třídění dle tohoto klíče
`-číslo`	určuje sloupec, před kterým klíče končí. Obvykle se neuvádí - klíče jsou až do posledního sloupce na řádce. Tento parametr se nejčastěji používá v kombinaci s vylučováním duplicit
`-t znak`	určení znaku, který odděluje jednotlivé sloupce
`-u`	z výstupního souboru budou vyloučeny duplicitní řádky (pro posouzení duplicit se porovnávají pouze klíče)

```
$ uniq [parametry] [soubor]
```


Tento příkaz odstraní ze vstupních dat všechny duplicitní řádky. Vstupní data musí být předem setříděna. Lze zadat následující parametry:

`-číslo`	přeskočí se zadaný počet sloupců od začátku
`-u`	budou se vypisovat pouze řádky, které nejsou duplicitní
`-d`	budou se vypisovat pouze řádky, které jsou duplicitní
`-c`	na začátku každého řádku se vypíše počet výskytů

```
$ wc [-lwc] [soubor]
```

Příkaz zjistí počet řádků (`-l`), slov (`-w`) či znaků (`-c`) ve vstupním souboru. Příkaz zjišťuje požadované počty i pro několik souborů najednou, na závěr uvádí celkový součet (total).

```
$ tr [-cds] staré-znaky nové-znaky
```

Příkaz nahradí znaky z prvního řetězce odpovídajícími znaky z druhého řetězce. Oktalové číslo znaku v ASCII tabulce lze zadat za zpětným lomítkem (např. \012 označuje znak konce řádky). Posloupnost znaků z ASCII tabulky lze vyjádřit pomocí prvního a posledního znaku odděleného pomlčkou (např. `a-z` označuje malá písmena). Pokud je druhý řetězec kratší, použije se poslední znak tohoto řetězce. Lze použít následující parametry:

`-d`	znaky specifikované v prvním řetězci se zruší
`-c`	první řetězec označuje znaky, které se nemají nahrazovat
`-s`	pokud je více stejných nových znaků za sebou, vypíše se pouze jeden

V příkazu `tr` nelze zadat jméno vstupního souboru na příkazové řádce - nutno pomocí přesměrování. Následující příklad umístí jednotlivá slova ze vstupního souboru na samostatné řádky s potlačením prázdných řádek:

```
  tr -cs 'a-zA-Z' '\012' < vstup > vystup
```

2.9 Hledání souborů - příkazy which, whereis, find, locate

which

Prvním je příkaz `which`. Obvykle je používán k rychlému zjištění umístění nějakého programu. Prohledává jen adresáře uvedené v cestě `PATH` a jako výsledek vrací první výskyt hledaného souboru a adresářovou cestu k němu. Příklad:

```
$ which bash
/bin/bash
```

Tady vidíte, že `bash` je v adresáři `/bin`. Toto je velmi omezený příkaz, protože prohledává jen adresáře v `PATH`.

whereis

Příkaz `whereis` pracuje podobně jako `which`, ale navíc může vyhledávat man stránky a zdrojové soubory.

`whereis` vyhledávání souborů `bash` by mělo dát tento výsledek:

```
$ whereis bash
bash: /bin/bash /usr/bin/bash /usr/man/man1/bash.1.gz
```

Tento příkaz nám neříká jen, kde je hledaný program, ale též kde je uložena online dokumetace k němu. Rovněž tento příkaz je omezený. Co kdybyste chtěli nalézt nějaký konfigurační soubor? Na to nemůžete použít `which` ani `whereis`.

find

Příkaz `find` umí hledat cokoliv. Syntaxe tohoto příkazu je následující:

```
$ find adresář výraz
```

Utilita find prochází zadané adresáře a podadresáře a vyhledává všechny soubory, které vyhovují danému výrazu. Kritéria, která se mohou ve výrazu vyskytnout jsou následující:

`-name jméno_souboru`	Hledají se soubory s daným jménem.
`-t typ_souboru`	Hledají se soubory uvedeného typu. Možné typy jsou: `b` - blokový speciální soubor `c` - znakový speciální soubor `p` - fifo (pojmenovaný pipe) `d` - adresář `f` - obyčejný soubor
`-user jméno`	Hledají se soubory, jejichž vlastníkem je uživatel jméno.
`-group jméno`	Hledají se soubory, jejichž vlastník je člen skupiny jméno.
`-size +-n[c]`	Hledají se soubory, jejichž velikost je určena hodnotou n. Je-li uveden znak c, velikost souboru se měří ve znacích.
`-atime n`	Hledají se soubory, k nimž byl proveden poslední přístup před n dny.
`-ntime n`	Hledají se soubory, jejichž poslední modifikace byla provedena před n dny.
`-print`	Akční kritérium. Po dosažení tohoto kritéria se zobrazí cesta k souboru.
`-exec příkaz\;`	Akční kritérium. Pro každý nalezený soubor se provede příkaz. Pokud se uvnitř příkazu vyskytne `{}`, reprezentuje jméno nalezeného souboru.
`-ok příkaz\;`	Akční kritérium. Funguje stejně jako `-exec`, jenom se musí každý příkaz potvrdit `y`.

Následující příkaz vypíše všechna jména souborů v aktuálním adresáři, jejichž vlastníky jsou uživatelé `jirka` a `lada`, a současně tyto soubory vymaže.

```
$ find . \( -user jirka -o -user lada\) -print -exec rm {}\;
```

Další příkaz v adesářích `/home/lada` a `/home/jirka` nalezne všechny soubory větší než 10000 bajtů a poslední přístup k nim byl před více než pěti dny.

```
$ find /home/jirka /home/lada -size +10000c -atime +5 -ok rm {}\;
```


locate

Příkaz `locate` prohledává celý souborový systém stejně, jako to dělá příkaz `find`, ale namísto skutečného souborového systému prohledává svoji databázi. Databáze je nastavena tak, aby se automaticky updatovala. Databázi pro `locate` můžete updatovat i ručně spuštěním příkazu `updatedb`. Příkaz `updatedb` nemůže spouštět normální uživatel. Tady je příklad příkazu locate v akci:

```
$ locate xinitrc
# nemusíme být přihlášeni jako root
/var/X11R6/lib/xinit/xinitrc
/var/X11R6/lib/xinit/xinitrc.fvwm2
/var/X11R6/lib/xinit/xinitrc.openwin
/var/X11R6/lib/xinit/xinitrc.twm
```

2.10 Archivátory - příkazy gzip, bzip2, tar, zip

UNIX nabízí několik programů, které můžete použít ke kompresi a archivování souborů. Tyto programy jsou obzvláště užitečné pro vytváření záloh a pro posílání kopií souborů mezi počítači prostřednictvím sítě. Tyto programy si poradí jak s Unixovými, tak i Windowsovými formáty archivů.

gzip

`gzip` je kompresní program z projektu GNU. Pracuje tak, že zkomprimuje jeden zadaný soubor. Základní použití vypadá takto:

```
$ gzip infile
```

Výsledný soubor bude pojmenován `infile.gz` a bude obvykle menší, než onen vstupní soubor. Všimněte si, že `infile` je nahrazen `infile.gz`. To znamená, že `infile` už nebude existovat. Pouze jeho gzipovaná forma.

Můžete rovněž vybalancovat optimalizaci velikosti komprimace, nebo její rychlosti. Maximální komprese dosáhnete takto:

```
$ gzip -9 infile
```

Takto bude komprimace trvat déle, ale výsledný soubor bude nejmenší, jaký umí `gzip` vyrobit. Zadáním nižších hodnot ve volbě v příkazové řádce dosáhnete kratšího času komprimace, ale komprese nebude taková.

Dekompresi gzipovaných programů můžete provést pomocí dvou příkazů, které jsou ale ve skutečnosti tím samým programem. Program `gzip` dekomprimuje jakýkoliv soubor s rozpoznanou příponou. Tou může být některá z těchto: `.gz`, `-gz`, `.z`, `-z`, `.Z`, nebo `-Z`. První metoda dekomprese je program `gunzip`, použitý takto:

```
$ gunzip infile.gz
```

To vytvoří dekomprimovanou verzi souboru `infile` v aktuálním adresáři a přípona `.gz` bude z názvu souboru odstraněna.

Druhým způsobem je použít opět `gzip`, ale s volbou `-d`:

```
$ gzip -d infile.gz
```

To způsobí přesně totéž, jako volání gunzip-u. Důvod je prostý: gunzip je pouze symbolickým linkem na `/bin/gzip`.

bzip2

`bzip2` je alternativní kompresní program. Používá jiný komprimační algoritmus než `gzip`. To vyúsťuje do několika výhod a několika nevýhod. Hlavní výhodou `bzip2` je velikost zkomprimovaného souboru. `bzip2` téměř vždy zkomprimuje soubor lépe, než `gzip`. V některých případech to může dávat podstatně menší soubory. To může být velká výhoda pro lidi, kteří mají pomalejší modemové připojení.

Nevýhodou programu `bzip2` je, že zatěžuje CPU mnohem více než `gzip`. To obecně znamená, že bzipování souboru bude trvat déle a bude více využívat CPU, než kolik by vyžadovalo gzipování. Když se rozhodujete, který kompresní program použít, musíte zvážit rychlost vs. velikost a rozhodnout, co je důležitější.

Používání programu `bzip2` je velmi podobné používání gzipu, takže se nebudeme zdržovat jeho popisováním. Jednoduše zadejte `bzip2` a název souboru, který chcete zkomprimovat:

```
$ bzip2 infile
```

Výsledný výstupní soubor bude obvykle menší, než vstupní a bude se jmenovat `infile.bz2`. Tak jako u gzipu nebude už vstupní soubor existovat - `bzip2` nahradí vstupní soubor jeho zkomprimovanou kopií.

Stejně jaku u programu gzip můžete používat číselné volby pro ladění kompresního poměru a rychlosti. Následující příklad ukazuje, jak dosáhnout maximální komprese (s odpovídajícím vytížením CPU):

```
$ bzip2 -9 infile
```

Pro dekompresi souborů jejichž název končí `.bz2` existují dva programy stejně, jako tomu bylo u programu `gzip`. Můžete použít `bzip2` nebo `bunzip2`.

tar

`tar` je páskový archivátor z projektu GNU. Z několika souborů a adresářů vytvoří jeden velký soubor. To pak umožňuje komprimovat celý adresářový strom, což by použitím samotného `gzip` nebo `bzip2` nebylo možné. Program tar má množství voleb zadávaných z příkazového řádku. Tato sekce pokrývá pouze nejobvyklejší způsoby jeho využití.

Tím úplně nejčastějším zůsobem použití programu tar je dekomprese a rozbalení (unarchive) balíčku, který jste si stáhli z webu nebo ftp. Většina těchto souborů má příponu `.tar.gz`. Obvykle se tomu říkává *tarball*. Znamená to, že několik souborů bylo zabaleno pomocí `tar`, a potom zkomprimováno gzip-em. Rovněž to někdy můžete vidět ve formě `.tar.Z`. To znamená to samé, jen to bylo nejspíš vytvořeno na starším unixovém systému.

Alternativně můžete někde objevit soubor s příponou `.tar.bz2`. Třeba zdrojové balíčky kernelu jsou obvykle distribuovány v této formě, protože je to menší pro download. Jak jste asi uhodli, je to několik souborů sbalených pomocí `tar` a posléze bzipovaných.

Ke všem souborům v archivu se můžete dostat použitím taru s jistými argumenty zadanými na příkazovém řádku. Při rozbalování tarballu používáme příznak `-z`, který říká, aby se před rozbalením použil `gunzip` a dekomprimoval jej. Obená forma dekomprese tarballu je následující:

```
$ tar -xvzf hejaz.tar.gz
```

Volba `-x` znamená *extract* - rozbalit. To je důležité, protože to říká taru, co má se vstupním souborem udělat. V tomto případě bude vstupní soubor rozbalen zpět do všech těch souborů a adresářů, ze kterých byl vytvořen.
Volba `-v` znamená, aby byl tar upovídaný (vebose). Takže tar bude vypisovat informace o všech souborech, které z archivu vybaluje. Je zcela přijatelné nechat tuto volbu vypnutou, když se ukáže nudnou. V opačném případě můžete zkusit `-vv` a dosáhnout tak mimořádné upovídanosti a získat výpis s mnoha dalšími informacemi o každém vybalovaném souboru.
Volba `-z` říká taru, aby `hejaz.tar.gz` nejprve prohnal gunzipem.
A konečně volba `-f` říká taru, že následující řetězec je jménem archivního souboru, o jehož rozbalení tu celou dobu běží. Je několik způsobů, jak zapsat tentýž příkaz. Na starších systémech postrádajících pořádnou kopii GNU taru, můžete zahlédnout tento zápis:

```
$ gzip -dc hejaz.tar.gz | tar -xvf -
```

Tento příkazový řádek unzipuje soubor a výstup pošle do programu `tar`. Volba `-c` říká, aby `gzip` poslal výsledek své práce do standardního výstupu (stdout), místo do souboru. Rourou se pak tento výstup předá programu `tar` k rozbalení. "-" na konci znamená, že místo souboru se mají data brát ze standardního vstupu (zde z roury). To rozbalí proud dat, který přichází z gzipu a zapíše je na disk.

Také můžete narazit na bzipované archivy. Místo volby `-z` zadejte `-j`.

Je důležité si zapamatovat, že tar ukládá vybalené soubory do aktuálního adresáře. Takže máte-li nějaký archiv v adresáři `/tmp`, a chcete jej rozbalit do svého domovského adresáře, máte dvě možnosti. Zaprvé můžete archiv přesunout do svého domovského adresáře a rozbalit ho tam. Nebo můžete zadat v příkazovém řádku cestu k archivu:

```
$ tar -xvzf /tmp/bar.tar.gz
```

Druhou nejobvyklejší činností s programem `tar` je vytváření vašich vlastních archivů. Vytvoření archivu není o mnoho komplikovanější, než jeho rozbalování. Jen se použije jiná sada voleb v příkazovém řádku. K vytvoření komprimovaeného archivu ze všech souborů v aktuálním adresáři včetně všech podadresářů a souborů v nich obsažených použijete tar v této formě:

```
$ tar -cvzf archive.tar.gz .
```

Volba `-c` na příkazovém řádku říká programu `tar`, aby vytvořil archiv. Volba `-z` prožene vytvořený archiv gzipem, aby se zkomprimoval. Ta tečka na konci označuje aktuální adresář (co se má archivovat). `archive.tar.gz` je jméno souboru v němž bude archiv uložen. Můžete si ho pojmenovat jak chcete a zdáte-li v názvu i cestu, bude archiv umístěn tam, kam cesta ukazuje. Uveďme si příklad:

```
$ tar -cvzf /tmp/archive.tar.gz .
```

Archiv pak bude vytvořen v `/tmp`. Co se má archivovat můžete zadat jako seznam souborů a adresářů na

konci příkazové řádky.

zip

Nakonec tu máme ještě dvě utility používané ke komprimaci souborů. Jsou velmi rozšířené ve světě Windows, takže i UNIX je - kvůli přenositelnosti souborů - obsahuje. Tím kompresním programem je `zip` a dekompresním protějškem program `unzip`. Komprese jednoho souboru je snadná:

```
$ zip foo *
```

Takto se vytvoří soubor `foo.zip`, který bude obsahovat všechny soubory, které jsou v aktuálním adresáři. Příponu `.zip` přidává zip automaticky, takže není nutné ji za jméno souboru psát. Zip může do archivu přidávat rekurzivně i adresáře a soubory, které jsou vnořené v aktuálním adresáři pomocí volby `-r`:

```
$ zip -r foo *
```

Dekomprese souborů je opět snadná:

```
$ unzip foo
```

Takto extrahujete všechny soubory ze souboru `foo.zip`, včetně veškerých adresářů, které by archiv obsahoval. Příponu `.zip` opět nemusíte zadávat.

2.11 Dokumentace - manuálové stránky, info

Ačkoli jsem na téma všudypřítomné podrobné dokumentace několikrát narazila, musím je zmínit ještě samostatně. Dokumentace k Unixu se dělí do několika částí. První představují manuálové stránky. Vyvoláte je v textové konzoli příkazem

```
$ man požadované_heslo
```

Manuálové stránky popisují všechny příkazy systému, důležité konfigurační soubory, funkce a mnoho dalších věcí. Obsahují podrobnou referenci i příklady.

Systém manuálových stránek je poměrně starý a zdá se, že je brzy nahradí tzv. `info`. Info je nápověda obsahem podobná manuálovým stránkám, umožňuje však odkazovat z jednoho tématu na druhé podobně jako na Internetu. Na rozdíl od manuálových stránek se dá zobrazit i obsah popisovaných témat. Vzhledem k tomu, že obliba info stále narůstá, může se stát, že novější věci najdete popsané jen v info.

Další Předchozí Obsah