Zrychlete práci s počítačem

Pavel Kameník

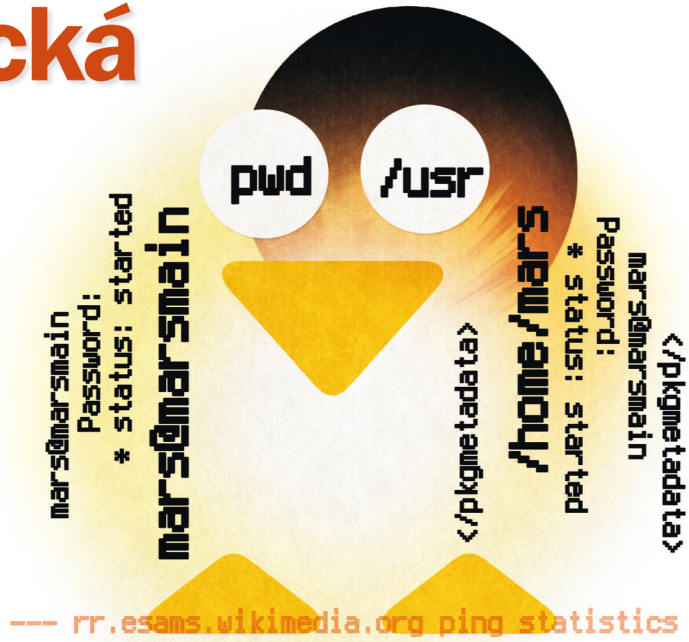
Příkazový řádek v Linuxu

Praktická řešení

Řešení nejčastějších operací bez myši

Praktické příklady pro každodenní použití

Rychlá změna nastavení systému, dávkové úlohy



computer press

Pavel Kameník

Příkazový řádek v Linuxu
Praktická řešení

Computer Press
Brno
2012

Příkazový řádek v Linuxu
Praktická řešení

Pavel Kameník

Obálka: Martin Sodomka
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN 978-80-251-2819-0

Vydalo nakladatelství Computer Press v Brně roku 2012 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 16 369.

Dotisk 1. vydání

ALBATROS MEDIA a.s.

Obsah

Kapitola 2

Práce se soubory a adresáři 33

Chtěl bych poděkovat manželce Radce a dcerám Elišce a Vendulce
za podporu při psaní této knihy.

Počítač slouží k tomu, aby nám pomáhal a práci ulehčoval.
V žádném případě nesmíme my sloužit počítači.

Úvod

Co je obsahem knihy

Co mne vedlo k napsání této knihy? Především fakt, že podobná kniha chybí v nabídce odborných publikací. Problematikou příkazového řádku se zabývá každá z knih věnujících se Linuxu, ale některá jen okrajově, některá téměř vůbec ne. Chtěl bych tuto knihu zaměřit jen a jen na příkazový řádek a teorii kolem funkčnosti, běhu operačního systému bych vypustil.

V některé z kapitol se sice setkáte se stručnou teorií, ale to spíše proto, abyste si uvědomili princip toho, co děláte. Nezávisle na verzi a stáří Linuxu bych chtěl probrat praktiky hodící se ke každodenní práci.

Nejsem profesionál, neustále se pohybuji ve Windows i v Linuxu; Windows, jelikož je to můj hlavní pracovní nástroj, používám častěji. Pokud používám systémy Linux a Unix, stává se mi stále častěji, že nemůžu najít ten správný příkaz a parametry, které potřebuji (samozřejmě příkazy, které používám neustále, si pamatuji, ale…). Chtěl jsem si vytvořit tahák, ale zjistil jsem, že by to byl tahák značně obsáhlý. Psaní poznámek na různá bezpečná místa, to se mi také neosvědčilo. Proto jsem přistoupil na variantu začít psát knihu a vše, co potřebuji, schovat do ní.

Pokud chcete srovnání s příkazovým řádkem ve Windows, dopředu musím prohlásit, že takové srovnání ani není možné, ve Windows se jedná o zcela jiný nástroj s omezenými možnostmi.

V Linuxu lze pomocí příkazového řádku provádět i složitější úkoly, které by těžko někdo zkoušel ve Windows. V Linuxu jste schopni spustit několik konzol a na každé provádět jinou operaci na základě psaní příkazů, ve Windows asi těžko někdo spustí několik příkazových řádků, aby v nich sledoval běžící procesy, řadil soubory v adresáři, nastavoval práva souborů a adresářů; některé z těchto příkladů ani nelze rozumně provádět. Prostě příkazový řádek v Linuxu je něco úchvatného, a kdo se neumí v tomto prostředí pohybovat, přichází o hodně. Je dobré si také uvědomit, že vykonání příkazů zadávaných přes příkazový řádek je rychlejší a úplnější.

Poznámka

Chtěl bych se omluvit za některé nepřesné nebo příliš stručné vysvětlení problematiky. Vedlo mne k tomu to, že jsem chtěl vytvořit co nejvíce praktický materiál k použití pro laiky i začínající uživatele stejně jako pro zkušené uživatele. Pro upřesnění musím ještě dodat, že v celé knize hovořím o programech, skriptech a příkazech. Z větší části budu brát programy, příkazy a skripty jako něco, co se spouští a něco vykonává. Proto mi prosím odpusťte nepřesnosti v definování toho nebo onoho balíku, do kterého program nebo skript patří. Spíše jsem se zaměřil na to, co se vykonává nejčastěji. Pokud bude někdo chtít zjistit, kam který příkaz nebo program patří, najde tyto informace v publikacích podrobně rozebírající jednotlivé distribuce.

Komu je kniha určena

Kniha je určena všem, kdo mají zájem proniknout do tajů tohoto mocného nástroje. Neklade žádné speciální požadavky na čtenáře, pouze bych upozornil, že pokud nemáte možnost si vyzkoušet probírané ukázky, nebude vám kniha asi velkým přínosem.

Doufám, že pro každého se zde něco najde, ať se jedná o laika, nebo profesionála. Věřím, že pokud si knihu koupíte, nebudou to zbytečně vyhozené peníze a najdete zde spoustu užitečných návodů a ukázek, které se vám budou hodit dnes a denně.

Jak tuto knihu číst

Rozhodně není nutné číst tuto knihu od začátku do konce. Kniha je dělena do kapitol; každá kapitola je věnována něčemu jinému, nezávisle na předchozí kapitole. Záleží jen na vás, jak vám bude čtení vyhovovat. Budu ale samozřejmě postupovat od jednoduchých věcí ke složitějším.

Omlouvám se, pokud některá část textu na vás bude působit dojmem, že něco tak jednoduchého není nutno rozebírat. Je to proto, že kniha je určena i laikům, kteří tyto základy nemusí znát. Není nic jednoduššího než část textu přeskočit a pokračovat na další stránce.

Kniha patří do kategorie „hotových řešení", to znamená, že nejprve nastíním problém a následně popíšu některé z variant. Rozhodně se nejedná o strohý překlad manuálových stránek. Proto zde nehledejte popis všech možných příkazů a jejich parametrů.

Co budete dále potřebovat

Potřebovat budete pouze tuto knihu, trochu času a hlavně jakýkoliv počítač s jakoukoliv verzí Linuxu. Verze operačního systému nemusí být nutně nejnovější, protože pro naše účely, které se zde chystám ukázat, není nutné mít žádné špičkové vybavení. Kouzlo práce s příkazovým řádkem spočívá i v tom, že nepotřebujete žádnou grafickou nadstavbu a výsledky budou stejné. Je jedno, jakou máte distribuci, všude si můžete zkoušet zde uvedené příklady.

Na jakých systémech jsou jednotlivé ukázky otestovány

Veškeré ukázky a příkazy jsem odzkoušel a testoval. K tomu mi posloužily tyto Linuxové distribuce: Fedora 12 a 13, OpenSuse 11.2 a 11.3, Debian 5, Slackware 13.1, okrajově Xubuntu a také Cygwin. Využíval jsem také virtuální počítače spouštěné na Sun VirtualBox. Záleží na vás, jestli máte svoji oblíbenou distribuci anebo použijete některou ze jmenovaných distribucí. V drtivé většině budou všechny příkazy fungovat. V těch zbývajících případech se musíte obrátit na manuálové stránky, stránky poskytovatele a na Internet.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu připravilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

redakce PC literatury
Computer Press
Spielberk Office Centre
Holandská 3
639 00 Brno

nebo

sefredaktor.pc@cpress.cz

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nedá. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji nahlásíte. Ostatní uživatele tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K1759* po klepnutí na odkaz Soubory ke stažení.

KAPITOLA 1

Začínáme

Něco lehkého na úvod

Předpokládám, že máte spuštěný Linux a umíte spustit příkazový řádek, nazývaný terminál v grafickém prostředí anebo konzola v prostředí bez oken. Můžete se přepínat mezi mnoha spuštěnými terminály – každý si žije vlastním životem a na každém můžete spouštět samostatné příkazy či skripty. (Jen na okraj: Příkaz je vlastně jeden úkol, který můžeme pomocí tzv. přepínačů rozvětvit, potom máme jeden příkaz i třeba na celý řádek. Skript je souhrn příkazů uložených někde na disku ve spustitelném souboru a příkazy se postupně vykonávají řádek po řádku.) Příkazem zde pro jednoduchost budeme rozumět spuštění programu nebo některé z utilit nainstalované v různých víceméně standartních balících, případně spuštění skriptu. Příkaz nebo skript se bude postupně vykonávat nezávisle na ostatních terminálech. Určitý problém ale může vzniknout, pokud systém bude přistupovat ke stejným souborům z více terminálových oken; v tom případě může vykonávání příkazu či skriptu skončit chybou. Mezi konzolami se můžete přepínat pomocí kláves Alt+F1 až F7 (pozor; zde už se počet konzol může lišit podle typu distribuce).

Můžete se i přepnout z grafického prostředí do konzolového pomocí kláves Ctrl+Alt+F1 až F7, zase záleží na distribuci; někde třeba tohle přepínání mezi konzolami a grafickým prostředím nebude fungovat anebo se bude muset nastavit, ale to zatím nebudeme potřebovat (jedna z kombinací, např. Alt+F7, vám opět poslouží k přepnutí do grafického prostředí; pokud běží, opět se může u různých distribucí lišit). Pro naše účely plně postačí terminál.

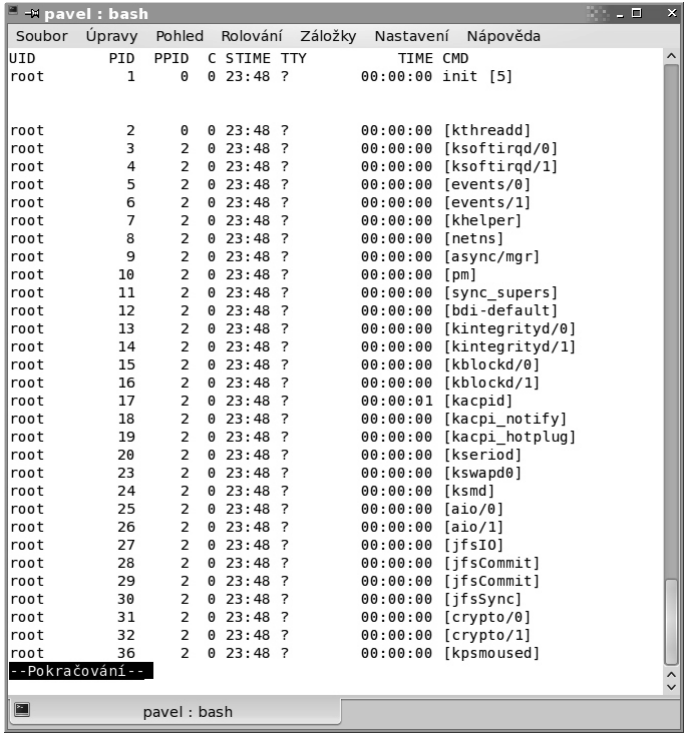
Nebudu zde rozebírat, co je součástí kterého programového balíku. Pokud by se jednalo o něco, co je nutné doinstalovat, co není součástí běžného Linuxového systému, tak to v úvodu konkrétní sekce knihy uvedu.

Poznámka

Ještě bych doplnil pro úplnost další užitečné funkce pro práci v příkazovém řádku nepostradatelné. Listování v naposled zadaných příkazech provádíte šipkami nahoru a dolů. Pokud napíšete začátek příkazu a systém jej zná, po stisknutí tabulátoru se příkaz dopíše; lze použít i na doplnění adresáře při procházení adresářovou strukturou.

První ukázka

Nejdříve zkuste napsat do příkazového řádku za blikající či jinak zvýrazněný kurzor `ps` a objeví se vám několikařádkový výpis. Tento příkaz zobrazuje běžící úlohy systému. A teď zkuste napsat příkaz `ps` a k němu další parametry `-ef` (celý příkaz bude vypadat `ps -ef |more`) a podívejte se na obrázek 1.1. Výpis je na několik stránek.



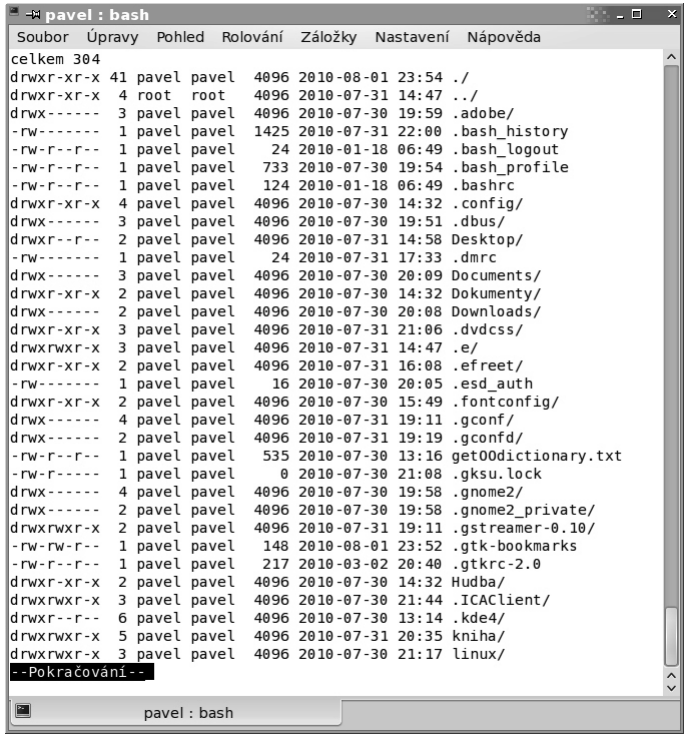
Obrázek 1.1:. Výpis pomocí dalších parametrů

Tip

Všechny příkazy a jejich parametry jsou tvořeny jako zkratka anglického slova a nebo skupiny slov - například `ps -ef` - „process status“ `-ef` e - „every“ f - „full“ . (Někomu může pomoci anglický výraz k zapamatování).

Každý příkaz může, ale nemusí mít další přídavné parametry, které píšeme za příkaz za pomlčku (to je to `-ef |more`). Tyto nepovinné doplňující informace jakýmsi způsobem daný příkaz rozvíjí a slouží k tomu, abychom měli možnost vykonat nebo zobrazit co nejpřesněji to, co právě potřebujeme.

Další příklad bude s adresářem. Použijte příkaz `ls` pro výpis obsahu adresáře a teď za něj doplňte další parametr `-a`. Výpis je o něco složitější; přidejte další parametr `-al` a zase výpis vypadá jinak.



Obrázek 1.2: Podrobný výpis adresáře

Ještě vám dlužím vysvětlení, co znamená ono `|more` – je to jednoduše zadání příkazu systému, aby výpis přizpůsobil obrazovce nebo oknu terminálu. Objeví se vám text výpisu a na konci stránky, pokud není výpis úplný, se zobrazí „pokračování“. Systém čeká na klávesu „mezerník“, aby zobrazil další stránku – tímto způsobem se stránkuje celý výpis.

Pokud je výpis kompletní, příkaz je ukončen a systém čeká na zadání dalšího příkazu. Jestliže stisknete klávesu „Q“, vykonávání příkazu se ukončí okamžitě.

Výpis běžících procesů podruhé

Chci si prohlédnout běžící procesy – příkaz ps

Jak jste viděli v předchozí kapitole, tento příkaz je velice užitečný. Použijete jej, pokud potřebujete zjistit, jaké procesy vám běží na pozadí, případně pod jakým uživatelem tyto procesy byly spuštěny. Dále jste potom schopni tyto úlohy i ukončit.

Pokud spustíte příkaz bez dalších parametrů, vypíší se vám pouze procesy, které byly spuštěny aktuálním uživatelem na aktuálním terminálu. Opět si můžete vyzkoušet příklad.

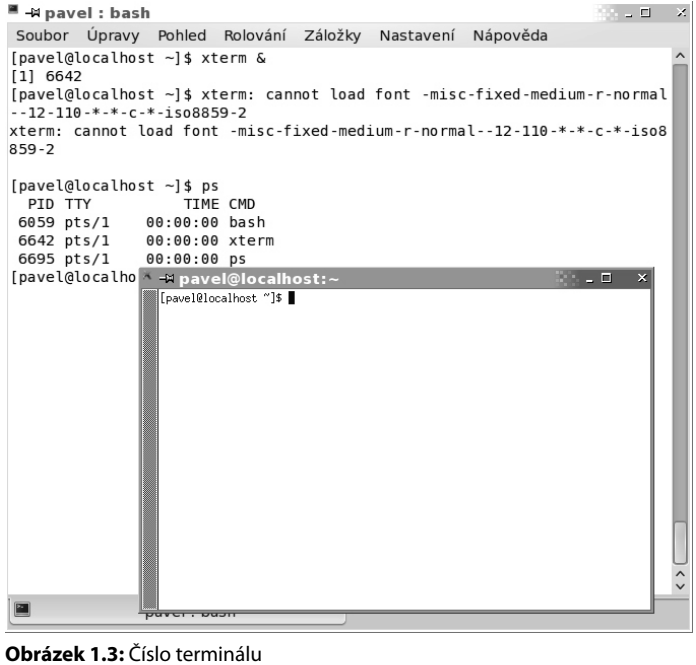
Máte otevřeno okno terminálu a do něj napište například

```
xterm &
```

(Musíte mít na vaší distribuci nainstalován program xterm – jednu z možných variant terminálů.) Nyní opět použijte příkaz `ps` bez parametrů a uvidíte, že vám přibyla další úloha ve výpisu. Pokud se vám spustí terminál xterm, zadejte v něm opět příkaz `xterm &`.

Přepněte se do původního okna terminálu, kde jste ověřovali běžící procesy, a zde opět spusťte výpis procesů příkazem `ps` bez parametrů. Ve výpisu se objeví xterm pouze jednou, protože druhý xterm jste spouštěli z jiného terminálu.

Tento fakt si můžete přečíst ve sloupci označeném TTY – to je číslo terminálu. Číslovat začíná systém od nuly a každý další spuštěný terminál čísluje vyšším číslem. Pokud tedy zkusíte vypsat procesy na posledním spuštěném terminálu, měli byste ve výpisu vidět například něco podobného jako na obrázku 1.3.



Obrázek 1.3: Číslo terminálu

Tímto jednoduchým příkladem jste se mohli přesvědčit o mém tvrzení uvedeném výše, že příkaz bez parametrů vám vypíše všechny procesy spouštěné přes aktuální terminál aktuálním uživatelem.

Ke zjištění, která úloha na kterém terminálu právě běží, dospějete také, pokud použijete příkaz s parametry `-ef`. Pouze budete muset složitě hledat ten řádek, který vás zajímá.

Poznámka

Tímto pokusem jsem vás chtěl upozornit na to, že pomocí parametrů se dostáváte k tomu, co opravdu chcete zjistit. Není důležité použít všechny možné parametry a následně složitě vyhledávat potřebné údaje. Pokud potřebujete, použijte jednoduchý výpis, a pokud potřebujete složitější výpis nebo další informace, použijte příkaz s parametry.

Musím vědět o procesech více

Nejprve se podívejte na výpis `ps -ef`, které informace jsou vám k dispozici. Výpis máte na obrázku 1.2.

Poznámka

Dlužím vám vysvětlení oněch dvou parametrů `e` a `f`. Parametr `e` je stejný jako parametr `a` – značí: vypiš údaje o všech procesech. Parametr `f` určuje, že ve výpisu bude více údajů, více sloupců – `f` jako full.

- V prvním sloupečku je UID – to značí, kdo proces spouštěl, komu patří. Většina systémových procesů bude patřit superuživateli – rootovi a další část přihlášenému uživateli. Tato informace se vám bude hodit, pokud budete chtít vyselektovat úlohy spouštěné konkrétním uživatelem.

Poznámka

Označení sloupců se může v různých distribucích lišit, ale význam je stejný.

- Sloupec označený *C* vám ukazuje, kolik času procesoru úloha využívá, ale pozor, tato hodnota je opravdu pouze orientační.
- Dále vás bude zajímat sloupec *PID* – ID číslo procesu. Pomocí tohoto čísla budete schopni například proces násilím ukončit. Každý proces je spouštěn pod svým číslem.
- Číslo *PPID* je číslo procesu rodiče, nadřízeného procesu; ve sloupci TTY můžete najít, z kterého terminálu proces běží.
- *STIME* zobrazí, kdy byl proces spuštěn.
- *TTY* – jak už víme, je to číslo terminálu, odkud byl proces spuštěn (můžeme tedy při použití parametru `-t` a čísla terminálu vyselektovat, co na kterém terminálu běží za proces).
- A poslední sloupec *CMD* zase ukazuje, odkud se úloha spouští. Výpis sloupce CMD je upraven tak, aby se vešel na řádek, proto někdy nemusí být celý. Celou cestu získáte použitím parametru `-w`.

Které procesy má spuštěny konkrétní uživatel

Zde použijete zcela jistě příkaz `ps -u pavel`. Opět můžete použít `|more`, protože výpis bude delší. alespoň u mne, jelikož jsem pod tímto účtem spouštěl grafické prostředí a další úlohy spouštěné po startu tohoto prostředí.

Poznámka

Příkaz `ps` umí opravdu hodně. Pokud bychom měli probírat všechny možnosti, zabralo by to hodně času a kniha by se stala nezajímavou. Spíše by se jednalo o opis manuálových stránek. Zkuste se podívat na manuálové stránky tohoto příkazu a zjistíte, kolik parametrů můžete použít. Záleží jen na vás a na praxi, se kterými parametry se naučíte pracovat a které budou pro vás ty důležité.

Proces najdu, ale potřebuji jej zrušit nebo zastavit

V další části se dostaneme k tomu, že hledáte určitý proces, který brzdí systém a nejde nijak vypnout. Musíte ho zrušit násilím, prostě ho „zabít" – odtud je i příkaz `kill`.

Příkaz kill

Tento příkaz vám poslouží v okamžiku, kdy určitý proces nereaguje, případně se nechová podle vašich představ. Opět má více možností použití. Zkuste si zadat příklad

`kill -l 3` (jedná se o písmeno malé L).

Systém vám odpoví:

```
QUIT
```

Při dalším pokusu

```
kill -l  9
```

systém odpoví

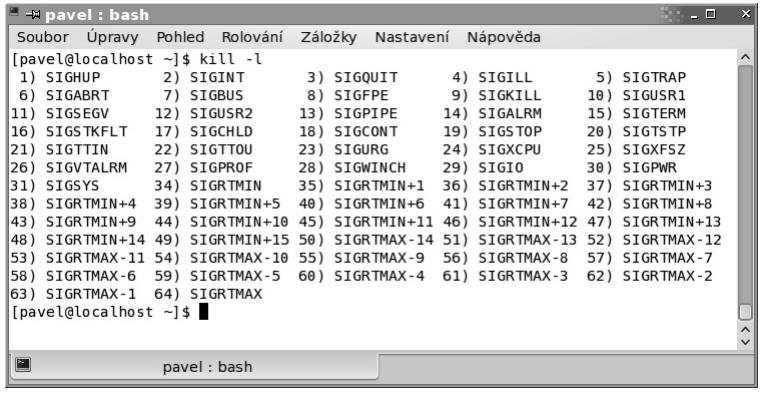
```
KILL
```

To značí, že každé číslo, které za příkazem kill použijete, má svůj význam. Pošlete úloze například příkaz QUIT nebo některý jiný a úloha bude pokračovat, doběhne do konce a pak se korektně ukončí.

Můžete se podívat, které signály můžete vyslat.

```
kill -l
```

Kill vám je všechny vypíše do tabulky, podobně jako na obrázku 1.4.



Obrázek 1.4: Tabulka signálů

Druhou možností je násilné ukončení. V tomto případě použijte hned možnost `kill -9` a číslo procesu, který potřebujete ukončit.

Tip

Číslo procesu už najít umíte z předchozích řádků, je to PID při použití příkazu `ps`.

Jak mám spustit program

Při spouštění jakéhokoliv nainstalovaného programu, „textového nebo grafického“, stačí začít psát název programu a následně pomocí tabulátoru se buď konec příkazu doplní, anebo po dalším stisknutí tabulátoru vám nabídne možnosti, jak by měl příkaz pokračovat. Pokud systém příkaz nezná, po druhém stisknutí tabulátoru se nic nestane a vy víte, že je něco špatně. Buď píšete příkaz chybně, anebo není program nainstalován.

Dále máte na výběr:

Zadáte parametry, se kterými se příkaz vykoná, zadáte výstup, pokud není na monitor.

Výstup můžete směrovat na vstup dalšího příkazu. K tomu slouží takzvaná „roura“ (pipe) – je to ono „svislítko“ |, které se často v příkazech objevuje. Podívejte se na jednoduchou ukázku:

```
ls -al |more
```

Zde předáváte výstup jednoho příkazu (`ls -al`) ne na monitor, ale na vstup druhého příkazu `more`. Tento příkaz provede zobrazení na monitor, a pokud je výpis na více stránek, čeká na příkaz, aby provedl zobrazení další stránky výpisu. To je nejjednodušší ukázka tohoto skvělého nástroje.

Potřebuji spustit program a dále pracovat na terminálu

Máte jednoduchý případ: Spustíte vícestránkový výpis obsahu adresáře `ls -al |more` a zároveň chcete spustit ještě jinou úlohu nebo více úloh a potřebujete, aby všechny úlohy běžely současně. Pro tento případ máte několik zkratek a příkazů. Jako první je spuštění příkazu na pozadí. Neuvažujte nyní o smysluplnosti tohoto příkladu, chci pouze demonstrovat na jednoduchém příkladu, jak to vlastně funguje.

Zadáte příkaz ve formátu:

```
ls -al |more &
```

Znak `&` na konci příkazu má za úkol spustit příkaz na pozadí. Vyzkoušejte si sami, že pokud zadáte tento příkaz, neuvidíte výpis adresáře, ale pouze něco takového:

```
$ ls -al |more &
 [1] 2935
$
```

Číslo 1 je identifikátor procesu v rámci aktuálního shellu. Druhé číslo (2935) je číslo procesu, tzv PID. Zkuste druhý příkaz, například `cat /etc/X11/Xmodmap | more`. Objeví se vám něco podobného tomuto:

```
$ ls -al |more &
[1] 2956
$ cat /etc/X11/Xmodmap | more
[2] 2958
[1]+  Pozastaven              ls --color=auto -al | more
$
```

Opět zde máte identifikátor procesu v rámci aktuálního shellu, první je jedna a druhý dva. Vidíte zde i čísla procesů. Obě tato čísla musí být jedinečná, nesmí se opakovat.

K manipulaci s těmito úlohami slouží několik příkazů.

Nejprve příkaz `jobs` – objeví se vám na obrazovce výpis procesů spuštěných v aktuálním shellu. V našem případě:

```
$ jobs -l
[1]-  3004 Dokonán                 ls --color=auto -al /etc
      3005 Pozastaven (výstup TTY) (SIGTTOU)          | more
[2]+  3006 Dokonán                cat /etc/X11/Xmodmap
      3007 Pozastaven (výstup TTY) (SIGTTOU)          | more
$
```

Zde vidíte, že úlohy jste spustili dvě, ale procesy jsou čtyři. Proč?

To je právě to přesměrování výstupu jednoho příkazu na vstup druhého. V první ukázce se provede příkaz `ls -al` a výsledek se přesune na vstup druhého příkazu |more. Příkaz `ls` se ukončí, ale jelikož je součástí druhého příkazu, je stále ve výpisu vidět. Obdobně je tomu i u druhé ukázky. Tomu odpovídá i číslo úlohy v aktuálním shellu – ty jsou pouze dvě.

Tohle by vám nebylo nic platné a pranic užitečné, pokud by se nedalo mezi takovými úlohami přepínat.

K tomuto účelu máte dva příkazy:

- `bg` – background – na pozadí
- `fg` – foreground – na popředí

Užití je jednoduché: zadáte `bg` nebo `fg` a číslo úlohy (ono číslo v hranaté závorce) a tím se přepínáte mezi úlohami. Úlohu, kterou momentálně nepotřebujete, přesunete na pozadí pomocí `bg` a naopak úlohu, kterou potřebujete, přesunete na popředí pomocí `fg`.

Abyste měli výčet úplný, ještě je potřeba si zapamatovat jednu klávesovou zkratku, a sice Ctrl+C. Pomocí této klávesové zkratky úlohu ukončíte (nemusí doběhnout – a ani nedoběhne – do konce).

Jak mám najít, odkud se příkaz spouští

Znáte příkaz a nevíte, odkud se spouští. Máte několik možností, jak zjistit, odkud se příkaz spouští.

Nejdříve se podívejte na příkaz:

```
whereis ping
```

Zadáním tohoto příkazu zjišťujete, z kterého adresáře se příkaz `ping` spouští. Třeba hledáte problematický program a potřebujete najít, kde všude je nakopírovaný. Nebo vám program nejde spustit. Pokud program najdete, můžete jej zkusit spustit zadáním celé cesty. Druhou možností je přepnutí do nalezeného adresáře a následně spustit. Budete ale nejspíš nuceni, pokud se přepnete do adresáře s programem, použít ke spuštění ještě `./prikaz` .

```
$ whereis java
java: /usr/bin/java /etc/java /usr/lib/java /usr/share/java /usr/share/man/man1/
java.1.gz
```

Na ukázce vidíte, jak asi bude vypadat váš výpis, pokud bude příkaz nalezen.

Problémem s tímto příkazem může být to, že vyhledávání probíhá jen v některých adresářích. Pokud víte, že spouštěcí soubor máte určitě někde v adresářové struktuře a příkaz `whereis` jej nenašel, budete muset použít jinou variantu.

Kde se tedy bude hledat při použití `whereis`:

- v adresářích určených systémem pro spouštění (/bin, /sbin, /usr/bin a /usr/sbin)
- v adresářích s manuálovými stránkami
- v adresářích uvedených v proměnné PATH
- ve zdrojových kódech (`/usr/src`)

Jaké další parametry může tento příkaz mít? Parametrem `-m` můžete hledat pouze manuálové stránky. Pokud zadáte parametr `-s`, budete hledat jen zdrojové soubory.

```
$ whereis -m umount
umount: /usr/share/man/man2/umount.2.gz /usr/share/man/man8/umount.8.gz
```

Zadal jsem špatný příkaz – jak smažu slovo

Pokud zadáte na konzole špatné slovo (tím se myslí směs znaků bez mezery), můžete postupně mazat jeden znak za druhým od konce.

Máte i druhou možnost, a sice klávesové zkratky, konkrétně:

Ctrl+W

Postupně budete mazat celá slova, tedy od mezery k mezeře. Mazání probíhá od konce.

Potřebuji smazat celý řádek

Případ podobný tomu předchozímu, opět použijete klávesovou zkratku, tentokrát:

Ctrl+U

Tím smažete celý řádek a kurzor čeká na další příkaz.

Hledání informací o příkazu

Pokud budete potřebovat najít všechny možné parametry k příkazu, musíte hledat v manuálových stránkách.

```
man ls
```

Procházíte po stránkách. Další stránku zobrazíte pomocí mezerníku. Ukončení prohlížení provedete stisknutím klávesy Q. Pohyb po stránkách funguje použitím kláves PageUp a PageDown.

Jak zobrazit všechny stránky manuálu pro daný příkaz

Někdy je k jednomu příkazu více manuálových stránek. Potřebujete-li hledat ve všech manuálových stránkách, použijete přepínač `-a`:

```
man -a ls
```

Ukončení jedné manuálové stránky a přechod na jinou provádíte opět pomocí klávesy Q.

Více o dalších parametrech se můžete dočíst pomocí

```
man --help
```

Chci informace o příkazu – příkaz info

Některé příkazy nemají manuálové stránky ve formátu `man`, ale v novějším formátu `info`:

```
info prikaz
```

tedy například

```
info man
```

Podívejte se na příklad:

```
$ man job
Žádný záznam pro job
$ info job
```

Ve výpisu dostanete informace o příkazu job. Opět se po stránkách pohybujete za použití kláves PageUp a PageDown.

Příkaz `info` má více možností, můžete například vyhledávat řetězce.

Stisknete klávesu S, zadáte text pro vyhledávání a najdete první výskyt slova. Pokud chcete další výskyt slova, stisknete opět S a následně Enter.

O dalších možnostech se můžete přesvědčit pomocí

```
info --help
```

Při použití příkazu `info` můžete také využívat křížové odkazy. Zkuste si následující:

Nejprve zadejte například:

```
info info
```

Prolistujte až na konec souboru; bude vypadat asi jako na následujícím výpisu

```
* tags tables in Info files:            Tags.           (line   6)
* u (Info mode):                        Help-FOO.       (line  13)
* unstructured documents:               Add.            (line  45)
* update Info tags table:               Tags.           (line  16)
* visible-mode:                         Help-Inv.       (line  17)


--zz-Info: (info.info.gz)Index, řádků: 125 --Bot----------------------------
```

Nyní si najeďte na některý řádek a stiskněte Enter. Dostanete se zpět do sekce uvedené klíčovými slovy na řádcích.

Potřebuji rychle možné parametry příkazu

Ve většině případů vám zcela jistě poslouží

```
prikaz --help
```

tedy ku příkladu

```
ps --help
```

Ve výpisu se vám objeví ve zkrácené verzi možnosti a přepínače, které můžete použít.

Někdy postačí zadat pouze příkaz; pokud tento očekává další parametry, tak vám je nabídne.

Kde můžu najít manuálové stránky v mé distribuci

Manuálové stránky jsou obvykle uloženy v adresáři

```
/usr/share/doc/jmeno_baliku
```

V jiných distribucích mohou být i jinde.

Jaké jsou základní klávesové zkratky

- Ctrl + C (někdy pomocí Delete) – zrušení provádění příkazu
- Ctrl + W – smaže slovo
- Ctrl + U – smaže celý řádek
- Ctrl + Q (někdy jenom q) – ukončení programu, výpisu apod.
- Ctrl + Z – pozastvení vykonávání příkazu
- Ctrl + D – ukončení zadávání (například pokud tvoříte soubor pomocí `cat`)
- Ctrl + Alt + číslo1–7 – přepínání mezi konzolami, pokud používáte grafické prostředí
- Alt + číslo1–7 – přepínání mezi konzolami, bez grafického prostředí
- šipka nahoru – pohyb v historii příkazů – směrem vzad (od posledního k prvnímu)
- šipka dolů – pohyb v historii příkazů – směr vpřed (od prvního k poslednímu)

Chci vyčistit (vymazat) okno terminálu

Na vyčištění (vymazání) okna terminálu použijte příkaz `clear`.

KAPITOLA 2

Práce se soubory a adresáři

Čtení souboru

Co je v souboru

Každodenní úkol: potřebuji si přečíst obsah souboru. Opět máte na výběr z několika možností.

Jednoduché použití příkazu more

V základní tvaru stačí pouze napsat příkaz ve tvaru

```
more /etc/fstab
```

A můžete si přečíst obsah souboru. Cestu k souboru opět zadáváte úplnou. Pokud jste v aktuálním adresáři, odkud chcete soubor číst, stačí napsat jen jméno souboru. Pokud je soubor delší, vypíše se pouze první obrazovka a dále se čeká na mezerník na zobrazení další stránky.

Tento příkaz má i některé volby navíc.

Chci soubor prohlížet jen po několika řádcích

V tomto případě se vám bude hodit zadání příkazu v tomto tvaru:

```
more -d5 /etc/X11/Xmodmap
!! ~/.Xmodmap
!!
!! Use this as a template for $HOME/.Xmodmap
!!
!! The leading `!' is the comment sign.
--Pokračování--(7%)[Stiskněte mezerník pro pokračování, 'q' pro ukončení.]
```

Za parmetr `-d` udáváte počet řádků, které se mají zobrazit. Následně po stisknutí mezerníku se opět zobrazí navolený počet řádků. To se vám může někdy hodit, pokud něco hledáte a víte, v které části souboru jsou údaje uloženy.

Lze prohlížet i více souborů po sobě

I tato volba je zde obsažena. Chcete-li prohlížet více souborů za sebou, napíšete je za sebe a oddělíte mezerou – stejně jako v ukázce:

```
more  /etc/X11/Xmodmap  /etc/fstab /etc/mtab
```

Nejprve se zobrazí obsah prvního souboru. Konec tohoto souboru bude označen a následně uvidíte jméno druhého souboru a jeho obsah.

```
!!
!! End of ~/.Xmodmap
::::::::::::::
/etc/fstab
::::::::::::::
/dev/disk/by-id/ata-VBOX_HARDDISK_VB1b044347-cba55693-part1 swap
```

Mezi prohlíženými soubory lze i přeskakovat. Použitím `:n` po skončení výpisu stránky skočíte na začátek dalšího souboru.

Použití příkazu less

Stačí, když se podíváte do manuálových stránek a porovnáte oba příkazy. Zjistíte, že příkaz `less` toho umí podstatně více.

Potřebuji se v souboru pohybovat vpřed i vzad

Pokud potřebujete tuto variantu, musíte použít příkaz less. Pomocí tohoto příkazu můžete opět stránkovat výpis, ale můžete se po souboru pohybovat vpřed i vzad.

Základní použití příkazu je opět jednoduché:

```
less /etc/X11/Xmodmap
```

..

```
/etc/X11/Xmodmap lines 1-23/40 62%
```

Na konci první stránky vidíte první změnu. Zobrazí se vám, kolik řádků bylo zobrazeno a kolik ještě chybí do konce.

Další užitečnou věcí je přesun na začátek anebo na konec souboru. Na konec se dostanete použitím velkého písmena G. Na začátek se dostanete použitím malého písmena g.

Potřebuji u výpisu vidět i čísla řádků

V tomto případě použijte příkaz:

```
less -N /etc/X11/Xmodmap
```

Nechci číst soubor od začátku

Pokud znáte číslo řádku, odkud chcete začít se čtením, bude se vám hodit příkaz s tímto parametrem:

```
less +2 /etc/X11/Xmodmap
```

Číslo za znaménkem plus udává, na kterém řádku se začne s výpisem. Tentokrát je to řádek druhý a pokračuje se dál.

Jak se pohybovat po souboru

Je to velice jednoduché: po souboru se můžete pohybovat pomocí šipek nahoru a dolů. Ve většině novějších distribucí tahle funkce funguje.

Napsání jednoduchého textu

Využití příkazu echo a přesměrování výstupu

Snad nejjednodušší variantou, jak můžete napsat krátký text do souboru, je tato:

```
echo "jak to jde" > pokus.txt
```

Nejedná se o žádný zázrak, pouze přesměrujeme výstup místo na obrazovku do souboru.

Příkaz `echo` má za úkol všechno, co se objeví za tímto příkazem, opět ukázat na obrazovce, případně na jiném výstupním zařízení. Můžete tedy psát na několik řádků, ale bez klávesy Enter. Klávesou Enter zadávání textu ukončíte a objeví se vám to stejné na obrazovce ještě jednou, ale bez zmiňovaného „echo".

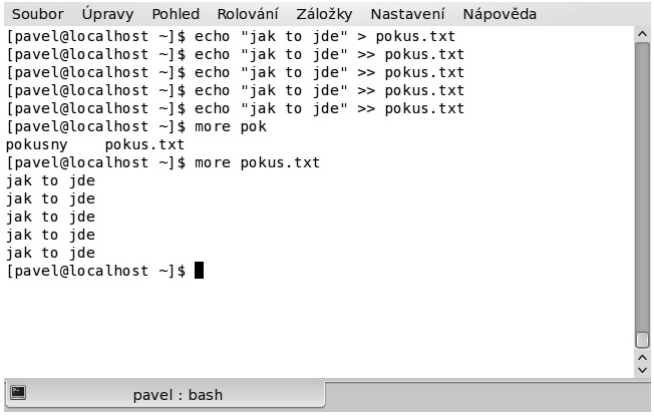
Zbytek už tak těžký není, pokud víte, že ostrá závorka > slouží k přesměrování výstupu. V tomto případě přesměrujeme výpis textu místo na obrazovku do souboru `text1.txt`.

Přesvědčit se můžete například takto:

```
more pokus.txt
```

Zadávaný text se vám objeví na obrazovce.

Nemůžete samozřejmě očekávat formátovaný text jako při použití některého z textových editorů. Tento způsob se může někdy hodit, pokud si chcete rychle poznamenat něco krátkého do souboru. Nemůžete soubor tímto způsobem editovat. Můžete pouze přidávat na konec souboru další text použitím dvou ostrých závorek >>. Text se dopíše na konec souboru – přesně jako na obrázku 2.1. K lepší práci se soubory a textem použijte některý z mnoha textových editorů, na který si zvyknete.



Obrázek 2.1: Jednoduché psaní

Tip

Tohoto lze využít i opačně – pokud chcete zadat vstup pro příkaz z nějakého souboru, použijete závorku opačnou, tedy:

```
prikaz  < soubor.
```

Elegantnější psaní textu pomocí cat

Jestliže chcete psát text rychle, zkuste tohle:

```
cat << 'EOF' >> soubor.txt
```

Použitím tohoto příkazu si velice jednoduše vytvoříte svůj editor. Do ostrých závorek mezi apostrofy napíšete, čím se má editor ukončit. Zvolte si kombinaci znaků, kterou ve svém textu nepoužijete. Teprve po napsání těchto znaků se vám vkládání znaků ukončí. Za závorkami už pouze napíšete jméno nového souboru.

```
cat << 'KIKLOP' >> soubor.txt
adassr
EOF
KIKLOP
```

Zde se můžete podívat na další příklad. Jako ukončení jsem použil znaky `KIKLOP`. Teprve po napsání těchto znaků se vkládání ukončí. Do souboru můžu psát písmena, čísla, používat Enter pro odřádkování. Pokud tento příkaz použijete na existující soubor, text se dopíše na konec souboru.

Jak vytvořit prázdný soubor, změnit čas přístupu k souboru

Prázdný soubor vytvoříte jednoduše použitím příkazu

```
touch soubor
```

Když spustíte příkaz znovu, nic se nestane, pokud už soubor existuje.

```
$ ls -al dopis2
-rw-r--r-- 1 pavel users 19 27. dub 11.22 dopis2
$ touch dopis2
$ ls -al dopis2
-rw-r--r-- 1 pavel users 19 27. dub 11.22 dopis2
$ touch dopis2
$ ls -al dopis2
-rw-r--r-- 1 pavel users 19 27. dub 11.23 dopis2
```

Opravdu se nic nestane? Ale stane – změní se datum a čas přístupu k souboru. Pokud zadáte příkaz bez parametrů, nastaví se aktuální systémové datum a čas.

```
$ touch -d 02:00 dopis2
$ ls -al dopis2
-rw-r--r-- 1 pavel users 19 27. dub 02.00 dopis2
```

Přidáním parametru `-d` a času ve tvaru 00:00 jste schopni nastavit čas přístupu k souboru.

Máte také možnost nastavit datum i čas přístupu k souboru:

```
touch -t 12310505 dopis2
```

Zadáváte ve formátu MMDDHHMM.

Údaje v souboru bych chtěl setřídit

Stačí, když zkombinujete dva příkazy, A sice `cat` na vytvoření a čtení souboru a příkaz `sort`, ten vám data setřídí.

Vytvoření souboru:

```
cat <<'EOF'>> soubor.txt
> kilo 22
> autobus 77
> vlak 4
> wesl 89
> EOF
```

Následné setřídění souboru:

```
cat soubor.txt |sort
> autobus 77
> kilo 22
> vlak 4
> wesl 89
```

Setřídění ale je pouze dočasné pro aktuální výpis. Jakmile použijete opět pouze příkaz `cat` bez `|sort`, bude dokument opět podle původního zadání. Ale můžete z původního souboru vytvořit nový setříděný, vyzkoušejte si.

V souboru jsou duplicitní záznamy, chci výpis bez duplicit.

```
$ cat > jmena
jana
petra
petr
petra
jirka
jindra
jirka
libor

$ sort jmena
jana
jindra
jirka
jirka
libor
petr
petr
petra
petra

$ sort -u jmena
jana
jindra
jirka
libor
petr
petra
```

Na další ukázce se opět můžete přesvědčit, že výpis je pouze dočasný, neprovádí se změny v originálním souboru.

```
$ cat jmena
jana
petra
petr
petra
jirka
jindra
jirka
libor
```

Pokud potřebujete tříděný seznam, lze to provézt třeba takto:

```
$ sort -u jmena > jmena2

$ cat jmena2
jana
jindra
jirka
libor
petr
petra
```

Jak vyhledávat řetězec v textovém souboru

```
$ cat > pouzivam_grep
Tady odzkousite pouziti prikazu grep
Tento chytry prikaz
vam umoznuje vyhledavat
jakykoliv retezec v textovem souboru.
Pouziti prikazu je velice jednoduche.
Vyzkousejte si sami.

$ grep 'prikazu' pouzivam_grep
Tady odzkousite pouziti prikazu grep
Pouziti prikazu je velice jednoduche.
```

Podívejte se na ukázku práce s příkazem grep. Napíšete nebo máte někde uložený nějaký jednoduchý textový soubor. Chcete vyhledat jen určitou část textu, v tomto případě 'prikazu'. Pomocí příkazu `grep` najdete výskyt řetězce v textu a vypíše se celý řádek, kde je řetězec obsažen. Pokud je obsažen vícekrát, vypíše se více řádků.

Chci se jen podívat na začátek souboru

V některých případech se vám může hodit příkaz `head`. Chcete se přesvědčit, co je v souboru, a postačí vám prohlédnou první dva řádky. Příkaz `head` s parametrem čísla značí počet řádků, které se mají zobrazit.

Vytvořte si nějaký soubor:

```
$ cat > dny
pondělí
úterý
středa
čtvrtek
pátek
sobota
neděle
```

Chcete vypsat jen první čtyři řádky.

```
$ head -4 dny
pondělí
úterý
středa
čtvrtek
```

Zajímá mne konec souboru

Pro tento případ můžete použít příkaz `tail`.

```
$ tail -4 dny
čtvrtek
pátek
sobota
neděle
```

Jak na konec souboru u stále rostoucích záznamů

Některé soubory, např. protokoly, logy a další, se neustále doplňují na konec řádků. I tyto konce souborů můžete číst použitím příkazu `tail` a přidáním parametru `-f`. Tedy:

```
tail -f protokol
```

Chci porovnat dva soubory

Další z běžných operací – potřebujete porovnat dva soubory. Opět můžete použít nově vytvořené soubory, třeba s ovocem (`ovoce` a `ovoce2`). Do souboru `ovoce` napište naše ovoce a třeba ananas, do souboru `ovoce2` napište exotické ovoce. Po porovnání souborů dostanete výsledek:

```
$ cat > ovoce
jahody
brusinky
meruňky
třešně
ananas

$ cat > ovoce2
ananas
kiwi
papaja
mango

$ diff ovoce ovoce2
1,4d0
< jahody
< brusinky
< meruňky
< třešně
5a2,4
> kiwi
> papaja
> mango
```

Jak vidíte, zobrazují se pouze rozdíly, stejné položky z obou souborů neuvidíte (v tomto případě ananas).

```
$ diff -u ovoce ovoce2
--- ovoce  2010-04-13 13:57:49.463976700 +0200
+++ ovoce2  2010-04-13 13:58:16.614975758 +0200
@@ -1,5 +1,4 @@
-jahody
-brusinky
-meruňky
-třešně
 ananas
+kiwi
+papaja
+mango
```

Jestliže přidáte k příkazu `diff` parametr `-u`, dostanete přehlednější výpis. Nejprve uvidíte, které soubory se porovnávají. Dále vidíte se znaménkem - záznamy prvního souboru neobsažené v druhém souboru. Bez znaménka jsou záznamy obsažené v obou souborech. Se znaménkem plus to jsou záznamy druhého souboru, které nejsou obsaženy v prvním souboru.

Můžu se přesvědčit, o jaký typ souboru se jedná

Pokud potřebujete zjistit, o jaký typ souboru se jedná, stačí, když použijete příkaz `file soubor`, například:

```
$ file ovoce
ovoce: UTF-8 Unicode text

$ file /mnt/win_c/pagefile.sys
/mnt/win_c/pagefile.sys: data
```

nebo třeba u souboru

```
$ file FootballArena.xls

FootballArena.xls: CDF V2 Document, Little Endian, Os: Windows, Version 5.1, Code
page: 1250, Author: Kl, Last Saved By: Kl, Name of Creating Application: Microsoft
Excel, Create Time/Date: Thu Mar 25 19:12:37 2010, Last Saved Time/Date: Tue Mar 30
20:18:53 2010, Security: 0
```

Můžu zjistit typ souboru u více souborů naráz

Jedním z možných řešení je vypsat všechny soubory za sebe. Druhým řešením je vytvořit soubor, ve kterém jsou jména souborů včetně úplné cesty, a následně použít příkaz `file` s parametrem `-f` a jménem souboru. Příkaz očekává soubor, který obsahuje jména dalších souborů, a u těchto určí jejich typ.

```
$ cat > soubory
ovoce
ovoce2
jmena
jmena2
/etc/fstab
/etc/mtab
/etc/hosts
```

a následně použijte příkaz

```
$ file -f soubory
ovoce:      UTF-8 Unicode text
ovoce2:     ASCII text
jmena:      ASCII text
jmena2:     ASCII text
/etc/fstab: ASCII text
/etc/mtab:  ASCII text
/etc/hosts: ASCII text
```

Další možností je vypsáním typu u všech souborů v adresáři:

```
$ ls
File_001.JPG  File_002.JPG  File_003.jpg  File_004.JPG  File_005.jpg

$ file *
File_001.JPG: JPEG image data, EXIF standard
File_002.JPG: JPEG image data, EXIF standard
File_003.jpg: JPEG image data, JFIF standard 1.01
File_004.jpg: JPEG image data, JFIF standard 1.01
File_005.JPG: JPEG image data, EXIF standard
```

Jak zabránit přepsání souboru

Je to vlastně jednoduchá vlastnost shellu, ve výchozím nastavení je vypnutá. Zapnutí i vypnutí provádíte nastavením proměnné `noclobber`.

```
$ date > a
$ date > a
```

Do souboru `a` jste zapsali aktuální čas, při opětovném použití stejného příkazu se soubor `a` přepíše.

```
$ set -o noclobber
```

Provedete nastavení proměnné noclobber.

```
$ date > a
bash: a: existující soubor nelze přepsat
```

Soubor nelze přepsat.

```
$ set +o noclobber
```

Zrušení ochrany:

```
$ date > a
```

Soubor je možné opět přepisovat.

Jak připojit k jednomu souboru další výstup

Přesměrování výstupu do souboru už umíte, nyní ještě potřebujete, aby se například text dopisoval na konec souboru. I to už jste viděli (u psaní jednoduchého textu pomocí příkazu `echo`); pouze pro zopakování: k tomuto účelu vám poslouží dvě ostré závorky vedle sebe >>.

Opět ukázka s datem:

```
$ date a
$ less a
Út dub 27 14:32:01 CEST 2010
a lines 1-1/1 (END)
$ date > a
$ less a
Út dub 27 14:32:32 CEST 2010
a lines 1-1/1 (END)
```

Po opětovném zadání příkazu se text v souboru přepíše.

```
$ date >> a
$ date >> a
$ date >> a
$ less a
Út dub 27 14:32:32 CEST 2010
Út dub 27 14:33:07 CEST 2010
Út dub 27 14:33:11 CEST 2010
Út dub 27 14:33:15 CEST 2010
a lines 1-4/4 (END)
```

A ještě něco na odlehčení

Existuje i možnost vypsat slova v souboru pozpátku.

```
rev soubor.txt
```

Následuje výpis opačný:

```
22 olik >
77 subotua >
4 kalv >
98 lsew >
```

Třeba najdete praktické využití…

Textový editor Vi

Stručně o editoru Vi

Nečekejte zde podrobné informace o tomto editoru. Pouze probereme základní, nejvíce používané operace při práci s textem. Podívejte se na nástin důvodů, proč si tento editor alespoň zkusit a seznámit se s ním:

- Tento editor existuje snad na všech typech Unix – Linux počítačů.
- Ke spuštění tohoto editoru není třeba mít grafické prostředí.
- Na drobné editace například konfiguračních souborů plně postačuje.
- Kdo si na něj zvykne, tomu dobře poslouží; umí i složitější operace.
- Existuje v několika variantách, ale princip a základní funkce zůstávají stejné.
- Jeho následníkem je editor Vim.

Editor se spouští velice jednoduše, stačí napsat:

```
vi jmeno_souboru
```

Pokud soubor existuje, otevře se a můžete dopisovat, přepisovat, mazat. Pokud soubor neexistuje, vytvoří se nový. Jméno souboru zadáváte buď jenom jako jméno, anebo je zadáte včetně cesty k souboru:

```
vi /home/pavel/text1.txt
```

Tip

Při zadávání cesty využívejte tabulátor, který umí doplňovat nejen zbytek příkazu. Napíšete jen první dvě písmena a zbytek se doplní po stisknutí této klávesy. Pokud je příkazů více, vypíšou se další možnosti po dalším stisknutí této klávesy k nezaplacení.

Stejným způsobem můžete procházet adresáři – zadáte první písmena, např. /ho, a stiskem tabulátoru se doplní zbytek, /home. Pokud nevíte, jaký podadresář následuje, stisknete dvakrát tabulátor a uvidíte, jaké podadresáře jsou ve složce /home.

Základní práce s editorem

Po otevření editoru si musíte uvědomit, že tento editor pracuje ve třech základních režimech:

- Příkazový – znaky, které píšete, jsou brány jako příkazy
- Vkládací – znaky jsou brány jako normální text
- Rozšířený, příkazový (ex mod) – zde doplňujete za znak : další znaky, které mají speciální funkce

Přepínání mezi režimy se provádí pomocí klávesy Esc.

Vkládání znaků provedete stisknutím klávesy Insert, dále vkládáte normální text. Po druhém stisknutí klávesy Insert se mód změní na přepisovací a text za kurzorem přepisujete.

Tento režim opustíte stisknutím klávesy Esc.

Pokud do souboru hodláte dopisovat, můžete použít i klávesu I a následně za kurzor dopisujete. Zrušíte opět pomocí klávesy Esc.

Po napsání dvojtečky (pozor, ne ve vkládacím režimu!) editor očekává pouze některé znaky, např. pro vykonání příkazů jako ulož, neukládej a ukonči apod…

Velice stručné seznámení máme za sebou a nyní začneme prakticky.

Chci napsat text do souboru

Napíšete do okna terminálu či na konzolu

```
vi text2
```

Poznámka

Jméno souboru si zvolte, jaké budete chtít. Pozor na malá a velká písmena – Linux dělá rozdíl mezi Text1 a text1, pokaždé to bude jiný soubor. Pokud soubor nebude existovat, vytvoří vám nový, jinak otevře existující. Přípony používat také nemusíte, Linux si poradí i bez přípon.

Obrázek 2.2: Vytvoření dokumentu