24. Práce s textem v Unixu

Obsah

- Příkazy pro práci s textem
- Regulární výrazy, rozšířené regulární výrazy
- Příkaz jako filtr
- Deskriptory
- Používání v kolonách
- Přesměrování
- Příkaz grep

Příkazy pro práci s textem

Unixové systémy poskytují bohatou sadu nástrojů pro práci s textovými daty. Tyto nástroje jsou navrženy tak, aby fungovaly samostatně nebo ve spojení s dalšími příkazy.

Základní příkazy

cat (concatenate) Spojuje a zobrazuje obsah souborů.

```
# Zobrazení obsahu souboru
cat soubor.txt

# Spojení více souborů
cat soubor1.txt soubor2.txt > vysledek.txt

# Zobrazení s čísly řádků
cat -n soubor.txt

# Zobrazení s označením netisknutelných znaků
cat -A soubor.txt
```

less Moderní zobrazovač obsahu souborů, který umožňuje procházení obsahu i zpět.

```
less soubor.txt
```

Navigace v `less`: - `q` - ukončení - `Šipky` - pohyb nahoru/dolů - `Mezerník` - stránka dolů - `b` - stránka nahoru - `/vzor` - hledání vzoru - `n` - další výskyt - `N` - předchozí výskyt

more Starší a jednodušší verze `less`. Zobrazuje obsah souboru po stránkách, ale neumožňuje pohyb zpět.

```
more soubor.txt
```

head Zobrazení začátku souboru (výchozí je 10 řádků).

```
# Zobrazení prvních 10 řádků
head soubor.txt

# Zobrazení prvních N řádků
head -n 20 soubor.txt

# Zobrazení prvních N bajtů
head -c 100 soubor.txt
```

tail Zobrazení konce souboru (výchozí je 10 řádků).

```
# Zobrazení posledních 10 řádků
tail soubor.txt

# Zobrazení posledních N řádků
tail -n 15 soubor.txt

# Průběžné sledování souboru (užitečné pro logy)
tail -f /var/log/syslog

# Zobrazení všech řádků od N-tého řádku
tail -n +5 soubor.txt  # od 5. řádku až do konce
```

nl Očíslování řádků souboru nebo vstupu.

```
# Očíslování všech řádků
nl soubor.txt

# Očíslování neprázdných řádků
nl -b t soubor.txt

# Vlastní formát číslování
nl -n rz -w 3 soubor.txt  # pravostranně zarovnané, 3 místa
```

cut Extrakce částí řádků ze souborů.

```
# Extrakce znaků podle pozice
cut -c1-5 soubor.txt  # znaky 1-5 z každého řádku

# Extrakce polí oddělených oddělovačem
cut -d':' -f1,3 /etc/passwd  # první a třetí pole (username a UID)

# Extrakce rozsahu polí
cut -d' ' -f1-3 soubor.txt  # první tři pole oddělená mezerami
```

tr (translate) Náhrada nebo mazání znaků.

```
# Náhrada znaků
echo "AHOJ" | tr "A-Z" "a-z"  # převod na malá písmena: ahoj

# Mazání znaků
echo "ahoj svete" | tr -d "aeiou"  # hj svt

# Komprese opakujících se znaků
echo "hezký   den" | tr -s " "  # hezký den
```

wc (word count) Počítání slov, řádků a znaků.

```
# Kompletní statistika (řádky, slova, znaky)
wc soubor.txt

# Pouze počet řádků
wc -l soubor.txt

# Pouze počet slov
wc -w soubor.txt

# Pouze počet znaků
wc -c soubor.txt
```

sort Řazení řádků podle různých kritérií.

```
# Základní řazení
sort soubor.txt

# Řazení podle čísel
sort -n cisla.txt

# Řazení podle měsíců
sort -M mesice.txt  # Jan, Feb, Mar, ...

# Řazení podle konkrétního sloupce (polí)
sort -k2,2 soubor.txt  # řazení podle druhého sloupce
sort -k2,2n soubor.txt  # řazení podle druhého sloupce číselně

# Řazení podle více kritérií
sort -k1,1 -k2,2n soubor.txt  # primárně první sloupec, pak druhý sloupec číselně

# Sestupné řazení
sort -r soubor.txt

# Definování oddělovače sloupců
sort -t':' -k3,3n /etc/passwd  # řazení podle UID
```

uniq Odstranění nebo počítání duplicitních řádků (vyžaduje seřazený vstup).

```
# Odstranění duplicitních řádků
sort soubor.txt | uniq

# Počet výskytů každého řádku
sort soubor.txt | uniq -c

# Pouze duplicitní řádky
sort soubor.txt | uniq -d

# Pouze unikátní řádky (vyskytující se jen jednou)
sort soubor.txt | uniq -u
```

join Spojení řádků ze dvou souborů na základě společného pole.

```
# Spojení na základě prvního sloupce
join soubor1.txt soubor2.txt

# Spojení na základě konkrétních sloupců
join -1 2 -2 3 soubor1.txt soubor2.txt  # 2. sloupec z 1. souboru, 3. z 2. souboru
```

paste Sloučení souborů horizontálně (řádek po řádku).

```
# Sloučení dvou souborů
paste soubor1.txt soubor2.txt

# Definování oddělovače výstupu
paste -d':' soubor1.txt soubor2.txt
```

expand a unexpand Konverze tabulátorů na mezery a naopak.

```
# Převod tabulátorů na mezery
expand soubor.txt
```

```
# Převod mezer na tabulátory
unexpand -a soubor.txt
```

Pokročilé textové procesory

sed (Stream Editor) Streamový editor pro transformaci textu.

```
# Substituce textu (nahrazení prvního výskytu na každém řádku)
sed 's/original/nahrazeni/' soubor.txt

# Globální substituce (nahrazení všech výskytů)
sed 's/original/nahrazeni/g' soubor.txt

# Substituce s case insensitive
sed 's/original/nahrazeni/gi' soubor.txt

# Substituce pouze na určitých řádcích
sed '3,5s/original/nahrazeni/g' soubor.txt  # pouze na řádcích 3-5

# Odstranění řádků
sed '/vzor/d' soubor.txt  # řádky obsahující vzor
sed '1,3d' soubor.txt     # řádky 1-3

# Výpis konkrétních řádků
sed -n '1,5p' soubor.txt  # pouze řádky 1-5

# Přidání textu před/za řádek
sed '/vzor/a\Nový text za řádkem' soubor.txt
sed '/vzor/i\Nový text před řádkem' soubor.txt

# Použití více příkazů
sed -e 's/a/A/g' -e 's/b/B/g' soubor.txt
```

awk Jazyk pro zpracování a analýzu textu.

```
# Základní formátování výstupu
awk '{print $1, $3}' soubor.txt  # první a třetí sloupec

# Použití oddělovače
awk -F: '{print $1, $3}' /etc/passwd  # username a UID

# Filtrace řádků podle podmínky
awk '$3 > 1000 {print $1}' soubor.txt  # jména s UID > 1000

# Sumarizace
awk '{sum += $1} END {print "Součet:", sum}' cisla.txt

# Počet řádků splňujících podmínku
awk '/vzor/ {count++} END {print "Počet:", count}' soubor.txt

# Komplexnější skript s více akcemi
awk 'BEGIN {print "Start"} {print $1} END {print "Konec"}' soubor.txt

# Formátovaný výstup
awk '{printf "%-10s %5d\n", $1, $2}' soubor.txt
```

Regulární výrazy, rozšířené regulární výrazy

Regulární výrazy (regex) jsou mocným nástrojem pro popis a vyhledávání vzorů v textu.

Základní regulární výrazy (BRE - Basic Regular Expressions)

Symbol	Význam
`.`	Libovolný jeden znak (kromě nového řádku)
`^`	Začátek řádku
`$`	Konec řádku
`*`	0 nebo více opakování předchozího znaku
`[abc]`	Jeden ze znaků a, b nebo c
`[a-z]`	Jeden ze znaků v rozsahu a až z
`[^abc]`	Jakýkoliv znak kromě a, b, c
`\< \>`	Hranice slova (začátek/konec)
`\(abc\)`	Skupinování
`\n`	Zpětná reference na n-tou skupinu

Rozšířené regulární výrazy (ERE - Extended Regular Expressions)

Rozšířené regulární výrazy přidávají další funkce a operátory pro větší flexibilitu. Nejsou zcela zpětně kompatibilní s BRE, proto vyžadují přepínač `-E` pro `grep` (nebo příkaz `egrep`).

Symbol	Význam
`+`	1 nebo více opakování předchozího znaku
`?`	0 nebo 1 výskyt předchozího znaku
`{n}`	Přesně n opakování předchozího znaku
`{n,}`	Minimálně n opakování předchozího znaku
`{n,m}`	Mezi n a m opakováními předchozího znaku
`|`	Alternativa (nebo)
`(abc)`	Skupinování (bez zpětného lomítka)

Speciální třídy znaků

Třída	Význam
`[:alnum:]`	Alfanumerické znaky
`[:alpha:]`	Písmena
`[:digit:]`	Číslice
`[:lower:]`	Malá písmena
`[:upper:]`	Velká písmena
`[:space:]`	Bílé znaky (mezera, tabulátor, nový řádek, ...)
`[:punct:]`	Interpunkce

Příklady použití regulárních výrazů

```
# Základní vzory
grep "^a" soubor.txt          # řádky začínající na 'a'
grep "a$" soubor.txt          # řádky končící na 'a'
grep "^$" soubor.txt          # prázdné řádky
grep "a.*b" soubor.txt        # řádky obsahující 'a' následované 'b'
grep "[aeiou]" soubor.txt     # řádky obsahující samohlásku
grep "[^0-9]" soubor.txt      # řádky obsahující něco jiného než číslice

# Rozšířené regulární výrazy
grep -E "ab+" soubor.txt      # 'a' následované 1+ 'b'
grep -E "a|b" soubor.txt      # řádky obsahující 'a' nebo 'b'
grep -E "a{2,3}" soubor.txt   # řádky obsahující 2-3 'a' za sebou
grep -E "(ab){2}" soubor.txt # řádky obsahující 'abab'
```

```
# Praktické příklady
grep -E "^[0-9]{3}-[0-9]{3}-[0-9]{4}$" soubor.txt  # telefonní čísla ve formátu XXX-XXX-XXXX
grep -E "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$" soubor.txt  # základní validace emailu
```

Příkaz jako filtr

V unixové filozofii platí, že příkazy by měly být navrženy tak, aby fungovaly jako filtry, což znamená: - Číst data ze standardního vstupu - Provádět jednu konkrétní operaci - Zapisovat výsledky na standardní výstup

Tento přístup umožňuje kombinovat příkazy do komplexnějších operací pomocí rour (pipes).

Důležité vlastnosti unixových filtrů

1. **Zpracování řádek po řádce** - většina příkazů zpracovává vstupní data po řádcích
2. **Bez stavovosti** - filtr nemá paměť předchozích spuštění
3. **Atomárnost** - filtr by měl dělat jednu věc, ale dělat ji dobře
4. **Textová orientace** - práce s textem jako univerzálním rozhraním

Příklady příkazů jako filtrů

```
# cat jako filtr
echo "ahoj" | cat -n  # číslování řádků ze standardního vstupu

# grep jako filtr
ps aux | grep firefox  # filtrování výstupu ps pro procesy obsahující 'firefox'

# sort jako filtr
find /var/log -type f | sort  # seřazení výstupu příkazu find

# head/tail jako filtr
cat /var/log/syslog | head -n 5  # zobrazení prvních 5 řádků

# tr jako filtr
echo "ABCDEF" | tr '[:upper:]' '[:lower:]'  # převod na malá písmena

# sed jako filtr
echo "ahoj svete" | sed 's/ahoj/nazdar/'  # substituce

# awk jako filtr
cat /etc/passwd | awk -F: '{print $1 " má UID " $3}'  # formátovaný výstup
```

Deskriptory

Deskriptor souboru je abstrakce, která umožňuje programům číst ze zdrojů dat a zapisovat do nich. V Unixových systémech je vše reprezentováno jako soubor, včetně zařízení, síťových soketů a rour.

Standardní deskriptory

Každý proces má standardně přiřazené tři deskriptory:

1. **Standardní vstup (stdin)** - deskriptor 0
   - Výchozí zdroj vstupu dat pro proces
   - Obvykle připojen ke klávesnici terminálu
2. **Standardní výstup (stdout)** - deskriptor 1
   - Výchozí cíl výstupních dat procesu
   - Obvykle připojen k displeji terminálu
3. **Standardní chybový výstup (stderr)** - deskriptor 2

- Výchozí cíl chybových hlášení procesu
- Také obvykle připojen k displeji terminálu, ale může být přesměrován nezávisle na stdout

Operace s deskriptory

```
# Otevření souboru pro čtení
exec 3< soubor.txt  # otevře soubor na deskriptoru 3 pro čtení

# Otevření souboru pro zápis
exec 4> soubor.txt  # otevře soubor na deskriptoru 4 pro zápis

# Připojení k existujícímu souboru
exec 5>> soubor.txt  # otevře soubor na deskriptoru 5 pro připojení

# Čtení z deskriptoru
read -u 3 promenna  # čtení z deskriptoru 3 do proměnné

# Zápis na deskriptor
echo "test" >&4  # zápis na deskriptor 4

# Uzavření deskriptoru
exec 3>&-  # uzavření deskriptoru 3
```

Používání v kolonách

Kolona (pipeline) je mechanismus, který umožňuje propojit výstup jednoho příkazu se vstupem dalšího příkazu. Vytvoří se pomocí svislé čáry |.

Princip fungování

- Výstup příkazu vlevo od | se stává vstupem příkazu vpravo
- Lze řetězit neomezený počet příkazů
- Každý příkaz běží paralelně s ostatními

Příklady kolonových konstrukcí

```
# Základní kolona
ls -l | grep ".txt"  # výpis všech souborů txt

# Více příkazů v koloně
ps aux | grep firefox | awk '{print $2}'  # získání PID procesů Firefox

# Komplexní zpracování textu
cat /etc/passwd | grep -v "^#" | sort -t: -k3,3n | cut -d: -f1,3 | head -n 5

# Počítání výskytů
cat logfile.txt | grep "ERROR" | wc -l  # počet řádků s 'ERROR'

# Zpracování a formátování dat
ps aux | grep -v USER | sort -nrk 3,3 | head -n 5 | awk '{print $2, $3 "%", $11}'
# výsledek: PID, %CPU, COMMAND pro 5 procesů s nejvyšším využitím CPU

# Filtrování a transformace dat
cat data.csv | cut -d',' -f2,4 | sort | uniq -c | sort -nr
# výsledek: četnost unikátních kombinací 2. a 4. sloupce, seřazeno sestupně

# Vyhledání největších souborů
```

```
find . -type f -name "*.log" | xargs du -h | sort -rh | head -n 5
# výsledek: 5 největších log souborů
```

Přesměrování

Přesměrování umožňuje změnit zdroj nebo cíl standardních vstupů a výstupů příkazů.

Základní operátory přesměrování

Operátor	Popis
>	Přesměrování stdout do souboru (přepíše soubor)
>>	Přesměrování stdout, připojení na konec souboru
<	Přesměrování stdin ze souboru
2>	Přesměrování stderr do souboru
2>>	Přesměrování stderr, připojení na konec souboru
&>	Přesměrování stdout i stderr do souboru
>&2	Přesměrování stdout na stderr
2>&1	Přesměrování stderr na stdout
<<	Here document (zadání více řádků textu)
<<<	Here string (zadání jednořádkového textu)

Příklady přesměrování

```
# Přesměrování výstupu
ls -l > seznam.txt  # zápis do souboru (přepíše existující)
echo "nový řádek" >> seznam.txt  # připojení na konec souboru

# Přesměrování vstupu
sort < seznam.txt  # čtení ze souboru
grep "vzor" < soubor.txt > výsledky.txt  # kombinace přesměrování

# Přesměrování chybového výstupu
find / -name "*.tmp" 2> chyby.log  # chyby do souboru
find / -name "*.tmp" 2> /dev/null  # zahození chyb

# Kombinované přesměrování
ls -l /neexistuje > výstup.txt 2> chyby.txt  # oddělené soubory
ls -l /neexistuje > výstup.txt 2>&1  # stderr i stdout do stejného souboru
ls -l /neexistuje &> vše.txt  # zkrácený zápis pro obojí

# Here document (vícesložkový vstup)
cat << EOF > soubor.txt
Řádek 1
Řádek 2
Řádek 3
EOF

# Here string (jednořádkový vstup)
grep "vzor" <<< "hledej vzor v tomto textu"

# Přesměrování mezi deskriptory
echo "Toto je chyba" >&2  # výpis na stderr

# Null device (/dev/null)
find / -type f -name "*.tmp" -delete 2> /dev/null  # potlačení chybových hlášení
```

Příkaz grep

Grep je nástroj pro vyhledávání řádků, které odpovídají zadanému vzoru. Název je akronymem pro "Global Regular Expression Print".

Základní použití

```
# Základní vyhledávání
grep "vzor" soubor.txt  # najde řádky obsahující "vzor"

# Vyhledávání ve více souborech
grep "vzor" soubor1.txt soubor2.txt
grep "vzor" *.txt  # všechny soubory s příponou .txt

# Rekurzivní vyhledávání v adresářích
grep -r "vzor" /cesta/k/adresari/

# Vyhledávání z stdin
echo "testovací text" | grep "test"
ps aux | grep "firefox"
```

Užitečné přepínače

Přepínač	Význam
`-i`	Ignorovat velikost písmen
`-v`	Inverzní hledání - řádky, které NEOBSAHUJÍ vzor
`-w`	Hledat pouze celá slova
`-n`	Zobrazit čísla řádků
`-c`	Zobrazit pouze počet odpovídajících řádků
`-l`	Zobrazit pouze názvy souborů s odpovídajícími řádky
`-L`	Zobrazit pouze názvy souborů BEZ odpovídajících řádků
`-A n`	Zobrazit n řádků po každém nalezeném řádku
`-B n`	Zobrazit n řádků před každým nalezeným řádkem
`-C n`	Zobrazit n řádků před a po každém nalezeném řádku (kontext)
`-E`	Použít rozšířené regulární výrazy (stejné jako `egrep`)
`-F`	Použít vzor jako doslovný řetězec, ne jako regex (stejné jako `fgrep`)
`-o`	Zobrazit pouze odpovídající části řádku
`-q`	Tichý režim - nevypisuje nic, pouze nastaví návratový kód
`--color=auto`	Zvýraznění nalezeného textu barvou

Příklady pokročilého použití grep

```
# Kombinace přepínačů
grep -in "error" logfile.txt  # case-insensitive s čísly řádků

# Inverzní vyhledávání
grep -v "^#" /etc/apt/sources.list  # řádky, které nejsou komentáře
```

```bash
# Kontext vyhledávání
grep -A2 -B1 "error" logfile.txt  # 1 řádek před a 2 po každém "error"

# Použití regulárních výrazů
grep "^[A-Z].*[0-9]$" soubor.txt  # řádky začínající velkým písmenem a končící číslicí

# Rekurzivní hledání s filtrací typů souborů
grep -r --include="*.py" "def main" /path/to/project/

# Vyhledávání celých slov
grep -w "is" soubor.txt  # najde "is", ale ne "island" nebo "this"

# Výstup pouze odpovídajících částí
grep -o "[0-9]\+" soubor.txt  # extrakce všech čísel

# Tichý režim pro testování
if grep -q "pattern" file; then echo "Nalezeno"; fi

# Komplexní filtrování s grep
ps aux | grep "[f]irefox"  # vynechá samotný grep proces z výsledků

# Počítání výskytů s výpisem
grep -c "error" *.log | grep -v ":0$"  # soubory obsahující "error"

# Extrakce IP adres
grep -E -o "([0-9]{1,3}\.){3}[0-9]{1,3}" logfile.txt

# Vyhledávání více vzorů
grep -E "error|warning|critical" logfile.txt  # jakýkoli z těchto výrazů
grep -E "fatal.*error|error.*fatal" logfile.txt  # složitější kombinace
```

Varianty grep

- **grep** - základní verze, používá základní regulární výrazy (BRE)
- **egrep** - podporuje rozšířené regulární výrazy (ERE), ekvivalent `grep -E`
- **fgrep** - rychlejší vyhledávání bez podpory regulárních výrazů, ekvivalent `grep -F`
- **rgrep** - rekurzivní grep, ekvivalent `grep -r`
- **pgrep** - vyhledává procesy podle jména a vrací jejich PID
- **zgrep** - vyhledává ve zkomprimovaných souborech

Praktické ukázky

Extrakce a zpracování dat z logů

```bash
# Nalezení a počítání chyb podle typu
grep "ERROR" application.log | cut -d: -f4 | sort | uniq -c | sort -nr

# Analýza webového logu
grep -E "404|500" access.log | awk '{print $7}' | sort | uniq -c | sort -nr | head -n 10

# Sledování živého logu
tail -f access.log | grep --color=auto -E "ERROR|WARN"

# Extrakce IP adres z logu
grep -E -o "([0-9]{1,3}\.){3}[0-9]{1,3}" access.log | sort | uniq -c | sort -nr | head
```

Zpracování CSV dat

```
# Extrakce konkrétního sloupce
cat data.csv | cut -d, -f3 | tail -n +2  # třetí sloupec, bez hlavičky

# Filtrování řádků podle hodnoty
cat data.csv | grep ",New York," | cut -d, -f1,4  # jména a věk lidí z New Yorku

# Výpočet průměru
cat data.csv | tail -n +2 | cut -d, -f4 | awk '{ sum += $1 } END { print "Průměr:", sum/NR }'

# Nahrazení hodnot
cat data.csv | sed 's/New York/NY/g' > new_data.csv
```

Manipulace se soubory

```
# Hromadné přejmenování souborů
ls *.txt | while read file; do mv "$file" "${file%.txt}.bak"; done

# Vytvoření archivu starých logů
find /var/log -name "*.log" -mtime +30 | xargs tar -czf old_logs.tar.gz

# Kopírování pouze určitých typů souborů
find . -name "*.jpg" | xargs -I{} cp {} /cesta/k/cíli/

# Změna formátu souboru (DOS/Unix)
find . -name "*.txt" | xargs dos2unix
```

Vyhledávání v kódu

```
# Nalezení všech funkcí v C souborech
grep -r "^[a-zA-Z_][a-zA-Z0-9_]* *(" --include="*.c" .

# Nalezení použití určité funkce
grep -r "použití_funkce\(" --include="*.php" /var/www/

# Počítání řádků kódu podle typu souboru
find . -name "*.py" | xargs wc -l | sort -nr

# Vyhledání TODO komentářů
grep -r "TODO\|FIXME" --include="*.java" .
```

Diagnostika systému

```
# Sledování procesů s nejvyšším zatížením CPU
ps aux | sort -nrk 3,3 | head -n 5

# Nalezení největších souborů a adresářů
du -h /var | sort -rh | head -n 10

# Monitoring síťových připojení
netstat -tuln | grep LISTEN

# Sledování změn v adresáři
watch -n 1 "ls -la /tmp"
```

Komplexní manipulace s textem

```
# Extrakce a výpočet statistik
cat access.log |
    grep "GET /api" |
    awk '{print $4}' |
    cut -d: -f1 |
    sort |
    uniq -c |
    sort -nr |
    head -n 10

# Generování CSV souboru z dat
echo "jméno,věk,město" > osoby.csv
grep -r "Person" --include="*.txt" . |
    sed -E 's/.*Person\("([^"]+)", ([0-9]+), "([^"]+)"\).*/\1,\2,\3/' >> osoby.csv

# Vyhledávání a počítání obrazových souborů
find . -type f -name "*.jpg" -o -name "*.png" | wc -l

# Zjištění rozložení typů souborů
find . -type f |
    sed -E 's/.*\.([^.]+)$/\1/' |
    sort |
    uniq -c |
    sort -nr |
    head -n 10

# Vytvoření přehledu o velikostech souborů
find . -type f -name "*.log" -exec ls -lh {} \; |
    awk '{print $5, $9}' |
    sort -hr
```

Závěr

Unixové nástroje pro práci s textem poskytují výkonný a flexibilní způsob, jak zpracovávat a manipulovat s textovými daty. Jejich síla spočívá v možnosti je kombinovat pomocí roury (pipe) a vytvářet tak komplexní řetězce operací.

Klíčové principy: 1. **Každý nástroj dělá jednu věc a dělá ji dobře** 2. **Textové rozhraní je univerzálním formátem pro komunikaci** 3. **Nástroje fungují jako filtry, které čtou ze standardního vstupu a zapisují na standardní výstup** 4. **Komplexní operace lze vytvářet kombinací jednoduchých nástrojů**