UNIXový cheat sheet

Martin "mhb" Böhm, Pavel "PeeJay" John

4. června 2008

Obsah

1 control structures **2**
- 1.1 Příklad: . . . 3
- 1.2 while . . . 3
- 1.3 Příklad: . . . 3
- 1.4 for . . . 3
- 1.5 Příklad: . . . 3
- 1.6 case . . . 3
- 1.7 Příklad: . . . 4

2 /etc/passwd **4**

3 id **4**
- 3.1 Příklad: . . . 4
- 3.2 Příklad: . . . 5

4 /etc/group **5**

5 Zamykani **5**
- 5.1 Příklad: . . . 5

6 /.forward **6**

7 ls -l **6**

8 rights **7**

9 filesystem **7**

10 ps **7**

11 date **7**
- 11.1 Příklad . . . 7

12 shell **8**
12.1 wildcardy 8

13 regexp **8**
13.1 zakladni struktury 8
13.2 grep 9

14 find **9**

15 sort **9**

16 test **9**

17 expr **10**

18 stat **10**
18.1 Příklad: 10
18.2 Nahrada: 10
18.3 Náhrada: 11

19 mail **11**
19.1 Příklad: 11

20 ed **11**

21 sed **12**

22 awk **12**

23 awk **12**
23.1 funkce 12
23.2 examples 13

24 mail structure **13**

1 control structures

```
if command; then
        : # commands
elif command; then

        : # commands
else
        : # commands
fi
```

1.1 Příklad:

```
myvariable="konec"
if [ $myvariable = "konec" ]; then
  echo je to konec
else echo jeste nekoncime
fi
```

Podmínka musí být příkaz - true pokud skončí s 0 nebo false při čemkoliv jiném. Příkaz test condition nebo [condition]

1.2 while

```
{while | until} [command]; do
        # prikazy
done
```

1.3 Příklad:

```
while read -r var1 var2
do
 echo "$var1" "$var2"
done < inputFile
```

1.4 for

```
for [var] in [text]; do
        # commands
done
```

1.5 Příklad:

```
RESTRICTED="root admin mailer Root"
# zkontroluj, jestli LOGIN neni v RESTRICTED
# pro dost kratka RESTRICTED
# pozor - nepouzivejte uvozovky! Pak by REST
# nabyl jen jedne hodnoty - celeho "$RESTRICTED"
for REST in $RESTRICTED; do
        if [ "$REST" = "$LOGIN" ]; then
                echo "Alert!"
        fi
done
```

1.6 case

```
case [text] in
        pattern1|pattern2)
                # commands
        ;;

        *)
```

```
                # default commands
        ;;
esac
```

1.7 Příklad:

```
# velmi casty priklad - parsovani argumentu na prikazove radce:

#!/bin/sh
# je rozumne nastavit vychozi cinnosti primo jako vychozi hodnoty promennych
# predpokladame, ze jen jedno NAME je dovoleno nastavit v parametrech
NAMESET=0
NAME=pepa
NEWER='0'

while [ "$#" -gt 0 ]; do
        case "$1" in
                '-name' )
                if [ $NAMESET -eq 0 ]; then
                        NAME=$2
                        NAMESET=1
                else
                        echo 'Too many -name arguments!'
                fi

                shift 2
                ;;
                '-newer' )
                NEWER=$2
                shift 2
                ;;
                * )
                echo "$1" >> '/tmp/'$$'targets'
                shift 1
                ;;
        esac
done

echo 'name='"$NAME"
echo 'newer='"$NEWER"
```

2 /etc/passwd

```
Name:(Password):UserID:PrimaryGroupID:ExtraInfo:HomeDirectory:Shell
root:x:0:0:root:/root:/bin/bash
```

3 id

```
id [login name]
        zobrazí info o aktuálním uzivateli nebo zadaném uzivateli.
```

3.1 Příklad:

```
id martin
```

```
uid=1000(oicw) gid=100(users) groups=16(dialout),33(video),100(users)
```

3.2 Příklad:

```
vypis vsechny GID skupin, ve kterych martin je

id -G martin
1000 4 20 24 25 29 30 44 46 107 109 115
```

4 /etc/group

Obsahuje jenom uzivatele, pro které není grupa jejich primární, přes příkaz id se dá zjistit do kterých skupin uzivatel patří.

```
Name:(Password):GroupID:UserList
root:x:0:
```

5 Zamykani

```
viz Forstovy slidy, je obtizne vytvorit
```

5.1 Příklad:

```
tenhle postup funguje (testovano - ale na vlastni riziko)

je trochu moc slozity, ale v zasade funguje az na pripad
ze nas program sleti zrovna na kontrole zamku, coz je
nepravdepodobne

proc je to tak reseno:
a) operaci mkdir bereme jako atomickou
b) potrebujeme zkontrolovat, ze nam skript nezuchnul
a ostatni hladovi - proto /tmp/appcrlock/pid
c) potrebujeme zajistit, aby kontrola, jestli
proces pristupujici do kriticke sekce nezuchnul,
byla atomicka - druhy zamek

kdy je to potreba:

skripty ktere pousti nekdo jiny a vickrat, zvlaste pak, pokud checkuji MAILBOX

#!/bin/sh


# always enter the cycle at least once
RV=1
CHK=0

while [ "$RV" -ne 0 ]; do
        sleep 10
        # make sure nobody else is checking
        mkdir /tmp/appcrcheck 2> /dev/null
        CHK=$?

        # lock for check
```

```
        while [ "$CHK" -ne 0 ]; do
                sleep 10
                mkdir /tmp/appcrcheck 2> /dev/null
                CHK=$?
        done

        # begin checking section
        mkdir /tmp/appcrlock 2> /dev/null
        RV=$?

        # if we had no luck, the process might be dead
        if [ "$RV" -ne 0 ]; then
                LID=`cat /tmp/appcrlock/pid`
                DEAD=`ps -p $LID | wc -l`
                if [ "$DEAD" -lt 2 ]; then
                        # the process is dead, retake the lock
                        rm /tmp/appcrlock/pid
                        RV=0
                fi
        fi

        if [ "$RV" -eq 0 ]; then
                echo "$$" > /tmp/appcrlock/pid
        fi

        # end checking section
        rmdir /tmp/appcrcheck
done

# begin critical section

# end critical section
rm -rf /tmp/appcrlock
```

6 /.forward

```
\kadlp7am
| /afs/ms.mff.cuni.cz/u/k/kadlp7am/mail_dispatcher

        první řádek posle e-mail do normální schránky
        druhý řádek ho posle rourou do skriptu
```

7 ls -l

```
typ, prava, pocet linku, vlastnik, skupina, delka souboru v bytech, datum+cas posl. modifikace, jmeno
-rwxr-x--x 2 forst users 274 Jan 5 17:11 test
file types:
        - ... plain text
        d ... directory
        b,c ... device
        l ... symbolic link
        p ... name
        d ... pipes
        s ... sockets
options:
        -l ... dlouhý výpis
        -1 ... krátký do jednoho sloupce
        -a ... zobrazit skryté
        -g ... potlačit skupiny
```

```
-t ... třídit dle času
-r ... třídit pozpátku
-F ... označit typ souboru
-R ... rekurze
-L ... sledovat linky
-i ... vypis souboru s cisly I-nodu
```

8 rights

user (u), group (g), other (o); read (r), write (w), execute (x), sticky bit (t) (/tmp)

9 filesystem

/etc/fstab: <file system> <mount point> <type> <options> <dump> <pass>
/dev/hda1 / reiserfs notail 0 1
I-node: pocet linku, vlastnik, skupina, prava, typ, velikost, casy, *data
ln -s

10 ps

ps aux:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      4131  0.0  0.1   1716   404 tty4     Ss+  00:41   0:00 /sbin/getty 38400 tty4
root      4827  0.0  0.2  13416   768 ?        Ss   00:41   0:00 /usr/sbin/gdm
```

11 date

vypise aktualni datum

11.1 Příklad

jak tedy porovnavat cas?

Podle specifikace? Nijak - v Single UNIX Specification v2
neni zadny zpusob, jak toto primo z shellu nebo AWK udelat. Forst
Te fakt nemuze penalizovat, kdyz to do komentare napises.

tedy nelegalni reseni:

nelegalni a)

```
date -d

date -d "Jun 2 15:51" "+%s"
1212414660

date -d "Aug 6 2007" "+%s"
1186351200
```

```
nelegalni b)

krmit ls -l --full-time

LC_ALL=C ls -l --full-time
total 4
-rw-r--r-- 1 martin martin    0 2008-06-02 00:00:00.000000000 +0200 18:19:03.000000000
-rw-r--r-- 1 martin martin    0 2008-06-02 18:19:03.000000000 +0200 file
drwxr-xr-x 2 martin martin 4096 2008-06-02 18:19:03.000000000 +0200 one

Vyhodou je, ze tento format se:
a) da tridit lexikograficky, vetsi cislo -> mladsi soubor
b) da pouzit pro touch:

touch -d "2008-06-02 18:19:03.000000000 +0200" file funguje

nelegalni c)

krmit ls --time-style, viz stat example
```

12 shell

```
$#        - počet parametrů scriptu
$0        - název scriptu
$n        - n-tý parametr scriptu
shift [n]         - posun číslování parametrů
set - text        - nastavení nových parametrů (set - a + b) $1="a", $2="+", $3="b", $#=3
$*        - vsechny parametry scriptu
$@        - dtto, ale "$@" je "$1" "$2" ...
$?        - návratový kód posledního příkazu
$$        - PID tohoto shellu
$!        - PID posledního procesu na pozadí
```

12.1 wildcardy

```
*              zadny nebo vice znaku
?              prave jeden znak
[abcde]        prave jeden znak z vyctu
[a-e]          prave jeden znak z rozsahu
[!abcde]       libovonlny znak, ktery neni na vyctu
[!a-e]         libovolny znak mimo rozsah
{debian,linux} prave jedno kompletni slovo z danych moznosti
```

13 regexp

13.1 zakladni struktury

```
.       - prave jeden libovolny znak ( [a.] znamena a nebo tecka )
[cdef]  - prave jeden ze znaku c, d, e, f
[^cd]   - libovolny znak krome c, d
^       - na zacatku symbolizuje, ze pattern musi byt na zacatku stringu
$       - totez co ^ ale na konci
\{m,n\} - predchozi element se opakuje m- az n-krat
```

```
\{m,\}  - alespon m-krat
\{,n}   - nejvyse n-krat
*       - predchozi element bude nula nebo vicekrat opakovan

\(neco\) - oznacuje, ze na neco se pujde odkazat
\n       - pri pouziti v druhe casti substituce rika, ze zde bude n-te neco
```

13.2 grep

```
 grep [options] PATTERN [FILE...]
 grep [options] [-e PATTERN | -f FILE] [FILE...]
       pokud clovek nezada posledni FILE cte ze standardniho vstupu
       -f - file s PATTERNs, pokud jich chceme vic naraz
       -c - vypise jen zadany pocet radek ktery najde
       -v - vypisuje radky ktere nevyhovuji regexpu
       -e PATTERN - vyhodne pouze k tomu, kdyz PATTERN ma zacinat -
       -H - vypise jmeno souboru ve kterem nasel PATTERN
       -i - ignore case
       -m NUM - skonci pote co najde NUM patternu
       -n - vypisovat cisla radek
       -o - vypisovat pouze PATTERN a ne celej radek
       -w - vybere lajny kde PATTERN odpovida celemu slovu
       -x - vybere jen ty lajny, kde PATTERN odpovida cely lajne
```

14 find

```
find cesta podminka akce
       -mindepth - minimální hloubka rekurze
       -maxdepth - maximální hloubka rekurze
       -regex porovnava cestu s regexpem
       -newerXY - porovnava casy (specifikovane misto pismen XY) souboru
       -P - nesledovat symlinky
       -L - sledovat symlinky
       -o - or na podmínky
       -a - and na podmínky
```

15 sort

```
       -kPOS1[,POS2] ... POS1-ty [az POS2-ty] sloupec je klic
       -n ... numeric (klic je cislo)
       -r ... reverse
       -t ... field separator
       -u ... unique
```

16 test

```
vhodné přesměrovat stdout do řiti, občas blije nesmysly do konzole
       -f file        - soubor file existuje
       -d file        - soubor file je adresář
       -L file        - soubor file je symbolický link
       -r file        - uzivatel má k souboru file právo r
       -w file        - uzivatel má k souboru file právo w
       -x file        - uzivatel má k souboru file právo x
       -s file        - soubor file má nenulovou délku
```

```
        -z str        - řetězec str je prázdný
        -n str        - řetězec str je neprázdný
        str1 = str2        - rovnost řetězců
        str1 != str2        - nerovnost řetězců
        int1 -eq int2        - rovnost čísel (-ne, -lt, -le, -gt, -ge)
        ! - negace, píse se před vlastní podmínku
        -a - and na podmínky
        -o - or na podmínky
```

17 expr

```
operators:
        a1 | a2 - vraci a1 pokud to neni 0, nebo null
        a1 & a2 - vraci a1 pokud jsou oba rozdilny od 0 a null, jinak vraci 0
        a1 < a2
        a1 <= a2
        a1 = a2
        a1 != a2
        a1 >= a2
        a1 > a2
        a1 + a2
        a1 - a2
        a1 * a2
        a1 / a2
        a1 % a2 - zbytek z a1 po deleni a2
        string : regexp - vraci nalezeny patter toho regexpu
        substr string position length - klasickej substr ale pozice znaku zacina 1kou
        index string chars - vrati prvni index ve stringu kde se nachazi jeden z chars
        length string
```

18 stat

```
stat bohuzel neni ve specifikaci - nicmene mne je to ukradeny,
ve specifikaci neni zpusob, jak zaridit praci s daty
```

18.1 Příklad:

```
inode cislo:

stat -c %i '.'
137227

access time, resp. mtime, resp. ctime od Epochy

stat -c "%X %Y %Z" '.'
1212445546 1212445544 1212445544
```

18.2 Nahrada:

```
vypis inode cislo pro '.':
 ls -ai '.' |head -n 1
 843653 .
```

18.3 Náhrada:

```
# nalezni cas v epochalnim tvaru:
# taky neni ve specifikaci - tam vlastne neni zadny zpusob
# jak tohle vyresit

ls -l --time-style "+%s"
-rw-r--r--  1 martin martin     14 1212414134 a
drwx------  2 martin martin   4096 1212439036 Desktop
drwxr-xr-x  2 martin martin   4096 1209666175 Dokumenty
```

19 mail

```
mail        [-eIinv] [-b bcc-adress] [-c copy-adress] [-s subject] [-a attachement] komu < Zprava
       -a odesle prilohu
       -b odeslat bcc
       -c odeslat cc
       -e pokud neprijde zadna zprava, tak se neodesle nic (jinak posila i prazdny mail)
       -I bezi v interaktivnim modu, ikdyz vstup neni z terminalu - muze clovek pouzivat commandy cely ty aplikace mail
       -i ignoruje interrupt signaly z terminalu
       -n nacte /etc/mail.rc driv nez se spusti
       -v verbose...
```

19.1 Příklad:

```
# posli mail administratorovi
EMAIL=root@forst.cz
cat mailcontent | miail -s "Information" $EMAIL
```

20 ed

```
  ed [options] file
  ed [options] file < script_file
       dávkový editor přístupný z commandlajny, edituje kopii souboru, nutné ulozit před skončením

options:
       -s - vypne diagnostické zprávy

addressing:
       .        - současná řádka
       $        - poslední řádka
       n        - n-tá řádka
       n,m        - řádky mezi n a m včetně
       -n, ^n        - n-tá předchozí řádka
       /re/        - první následující řádka odpovídající regexu
       ?re?        - první předchozí řádka odpovídající regexu
       lc        - řádek označený písmenkem 'c' (viz příkaz k)

commands:
       (.)a                        - appenduje za adresovanou lajnu text (input mode)
       (.,.)c                        - změní adresované lajny na text (input mode)
       (.,.)d                        - smaze adresované lajny z bufferu
       g/re/command-list        - aplikuje command list na lajny odpovídající regexu,
                                  v command listu musí být kazdý command na nové lajně a kromě posledního
                                  oddělený backslashem.
       v/re/command-list        - ditto akorát na lajny neodpovídající regexu
       (...)i                        - vlozí text před adresovanou lajnu
```

```
(.,.+1)j               - spojí adresované lajny
(.,.)m(.)               - přesune adresované lajny za adresu vpravo
q                       - ukončí ed
(.)r file               - za adresovanou lajnu načte soubor
f file                       - nastaví default filename
(.)kc                       - označí lajnu písmenkem 'c'
(.,.)l
(.,.)p                       - tiskne vybrané lajny
(.,.)n                       - ditto ale i s jejich čísly
(.,.)s/re/replace/       - nahrazuje na lajnách regexy replacementem, přípona g - způsobí na celé lajně
                          n jenom n-tý nález. Lze se odkazovat na regex pomocí \n, nebo & - první nahradí
                          n-tou závorku, druhé nahradí namačovaným textem
(1,$)w file               - ulozí soubor
```

21 sed

```
  sed [options] commnads file
  sed [options] -f script_file file
        streamový editor, který edituje řádek po řádce zadaný soubor a vypisuje na stdout

options:
        -e - zadá příkazy, které se mají provést
        -f - soubor se skriptem pro sed
        -i - úprava souboru "in-place"

commands:
        prakticky totozné s edem, pokud je adresován, provede se jenom na řádcích odpovídajících adresám
        pozor na problémy s \n, typicky není načten, ale při hraní si s pattern a hold space to dělá bordel.

        d           - smaze pattern space a spustí dalsí cyklus
        x           - prohodí pattern space a hold space
        h H           - kopíruje/přidá pattern space na hold space
        g G           - kopíruje/přidá hold space na pattern space
        p           - vytiskne pattern space
        t T label   - pokud byl nějaký/nebyl zádný subst. příkaz úspěsný, skoč na label
        b label     - skoč na label

        y/abcd/defg/ vsude kde je písmena ze vzoru budou nahrazena odpovídajícími písmeny z obrazu
```

22 awk

23 awk

23.1 funkce

```
getline       ...nacte novou radku z aktualniho souboru
getline <var  ...nacte novou radku ze souboru var
getline vraci 0, pokud EOF, 1, pokud OK, -1, pokud chybka
```

23.2 examples

```
awk '
 BEGIN {
  i=1;
  for(i=1; i<=4; i++) {
   getline < "soubora"
   letter = $1
   getline < "soubor1"
   number = $1
   printf("%s-%s", letter, number)
  }
 }

 END {}
'

*join
 -1 2 field   - bude se spojovat podle pole field v prvnim/druhem souboru
 -i           - ignorovat velikost pismen
 -e empty     - tam kde neni vstup, dej empty
 -o 1.2, 1.1  - format vystupu je 2. pole z 1. soubotu a 1. pole z 1. souboru

*cut
*paste
```

24 mail structure

```
~/.forward

/cesta/k/adresaru/soubor_postove_schranky
jina.emailova@adresa.com
\vaslogin
| /cesta/k/adresaru/se/skriptem/jmeno_skriptu parametry pro skript

==================================

From zaskodnik@matfyz.cz  Thu Jun  9 09:06:05 2005
Return-Path: <zaskodnik@matfyz.cz>
X-Original-To: honzik@ss1000.ms.mff.cuni.cz
Delivered-To: honzik@ss1000.ms.mff.cuni.cz
Received: from localhost (localhost [127.0.0.1])
        by ss1000.ms.mff.cuni.cz (Postfix) with ESMTP id E44F12C024
        for <honzik@ss1000.ms.mff.cuni.cz>; Thu,  9 Jun 2005 09:06:04 +0200 (CEST)
Received: from ss1000.ms.mff.cuni.cz ([127.0.0.1])
 by localhost (ss1000 [127.0.0.1]) (amavisd-new, port 10024) with ESMTP
 id 14106-02 for <honzik@ss1000.ms.mff.cuni.cz>;
 Thu,  9 Jun 2005 09:06:03 +0200 (CEST)
Received: from u-pl20 (u-pl20.ms.mff.cuni.cz [195.113.21.150])
        by ss1000.ms.mff.cuni.cz (Postfix) with ESMTP id DFEB52C021
        for <honzik@ss1000.ms.mff.cuni.cz>; Thu,  9 Jun 2005 09:05:37 +0200 (CEST)
From: anicka.dusicka@matfyz.sk
Message-Id: <20050609070537.DFEB52C021@ss1000.ms.mff.cuni.cz>
Date: Thu,  9 Jun 2005 09:05:37 +0200 (CEST)
To: honzik@ss1000.ms.mff.cuni.cz
Subject: pranicko

Preji Ti hezky den! :-)
```