Bourne-Again SHell and Linux CLI

© 2019 Martin Bruchanov, bruxy@regnet.cz

Set interpreter: #!/bin/bash   Remarks: # this is comment

1 Interactive control

Action	set -o vi	set -o emacs
vi-command mode (C)	Esc	—
Previous/next command in history	j / k	Ctrl+p / Ctrl+n PageUp / PageDown
Automatic fill of file name	Esc Esc	Tab
List of all matches	Esc =	Tab Tab
Horizontal move in command line	h / l	Ctrl+b / Ctrl+f, ← / →
Jump to line begin/end	^ / $	Ctrl+a / Ctrl+e
Backward/forward search in history	/ / ?	Ctrl+r / Ctrl+s
Delete word to the end/begin	dw / db	Esc d / Esc h
Delete text from cursor to the line end/begin	d$ / d^	Ctrl+k / Ctrl+u

1.1. Command line history

- history, fc -l – display numbered history of commands
- !*n* – run command number *n*
- !*p* – run last command beginning by *p*
- !! – repeat last entered command
- !!:*n* – expand *n*-th parameter of last command
- !$ – expand the last parameter of last command
- fc – run defined $EDITOR wit last command
- fc -e vim *z k* – open vim editor with commands from *z* to *k*
- ^old^new – substitute *old* with *new* in last command
- *program* \`!!\` – use output of last command as input

1.2. Help and manuals

- type -a *command* – information about command
- help *command* – brief help on bash command
- man *command*, info *command* – detailed help
- man -k *key*, apropos *key*, whatis *key* – find command

2 Debugging

Run a script as: bash *option script and its parameters*

- bash -x – print commands before execution
- bash -u – stop with error if undefined variable is used
- bash -v – print script lines before execution
- bash -n – do not execute commands

3 Variables, arrays and hashes

- *NAME*=10 – set value to variable $NAME, ${NAME}
- export NAME=10, typedef -x NAME – set as environment variable
- D=$(date); D=\`date\` – variable contains output of command date
- env, printenv – list all environment variables
- set – list env. variables, can set bash options and flags shopt
- unset *name* – destroy variable of function
- typeset, declare – set type of variable
- readonly *variable* – set as read only
- local *variable* – set local variable inside function
- ${!var}, eval \$$var – indirect reference
- ${*parameter-word*} – if *parameter* has value, then it is used, else *word* is used
- ${*parameter=word*} – if *parameter* has no value assing *word*. Doesn't work with $1, $2, ets.
- ${*parameter:-word*} – works with $1, $2, etc.
- ${*parameter?word*} – if *parameter* has value, use it; if no display *word* and exit script.
- ${*parameter+word*} – if *parameter* has value, use *word*, else use empty string
- array=(a b c); echo ${array[1]} – print ‚b'
- array+=(d e f) – append new item/array at the end
- ${array[*]}, ${array[@]} – all items of array
- ${#array[*]}, ${#array[@]} – number of array items
- declare -A hash – create associative array (from version)
- hash=([key1]=value ["other key2"]="other value") – store items
- ${hash["other key2"]}, ${hash[other key2]} – access
- ${hash[@]}, ${hash[*]} – all items
- ${!hash[@]}, ${!hash[*]} – all keys

3.1. Strings

- STRING="Hello" – indexing: H_0 e_1 l_2 l_3 o_4
- STRING+=" world!" – concatenate strings
- ${#string}, expr length $string – string length
- ${string:position} – extract substring from position
- ${string:position:length} – extract substr. of length from position
- ${string/substring/substitution} – substitute first occurrence
- ${string//substring/substitution} – substitute all
- ${string/%substring/substitution} – substitute last occurrence
- ${string#substring} – erase shortest substring
- ${string##substring} – erase longest substring

3.2. Embedded variables

- ~, $HOME – home directory of current user
- $PS1, $PS2 – primary, secundary user prompt
- $PWD, ~+ / $OLDPWD, ~- – actual/previous directory
- $RANDOM – random number generator, 0 – 32,767
- $? – return value of last command
- $$ – process id. of current process
- $! – process id. of last background command
- $PPID – process id. of parent process
- $- – display of bash flags
- $LINENO – current line number in executed script
- $PATH – list of paths to executable commands
- $IFS – Internal field separator. List of chars, that delimiter words from input, usually space, tabulator $'\t' and new line $'\n'.

4 Script command line parameters

- $0, ${0} – name of script/executable
- $1 to $9, ${1} to ${255} – positional command line parameters
- PAR=${1:?"Missing parameter"} – error when ${1} is not set
- $# – number of command line parameters (argc)
- ${!#} – the last command line parameter
- $* – expand all parameters, "$*" = "$1 $2 $3..."
- $@ – expand all parameters, "$@" = "$1" "$2" "$3"...
- $_ – last parameter of previous command
- shift – rename arguments, $2 to $1, $3 to $2, etc.; lower counter $#
- xargs *command* – read stdin and put it as parameters of *command*

4.1. Read options from command line

```
while getopts "a:b" opt; do case $opt in
    a) echo a = $OPTARG ;;
    b) echo b ;;
    \?) echo "Unknown parameter!" ;;
esac; done
shift $(($OPTIND - 1)); echo "Last: $1"
```

5 Control expressions

- (*commands*), $(*commands*), \`*commands*\`, {*commands;*} – run in subshell
- $(*program*), \`*program*\` – output of program replaces command
- test, [ ] – condition evaluation:
  - numeric comparison: a -eq *b* ...$a = b$, a -ge *b* ...$a \geq b$, a -gt *b* ...$a > b$, a -le *b* ...$a \leq b$, a -lt *b* ...$a < b$
  - file system: -d *file* is directory, -f *file* exists and is not dir., -r *file* exists and is readable, -w *file* exists and is writable, -s *file* is non-zero size, -a *file* exists
  - logical: -a and, -o or, ! negation
- [[ ]] – comparison of strings, equal =, non-equal !=, -z *string* is zero sized, -n *string* is non-zero sized, <, > lexical comparison
- [*condition*] && [*condition*]
- true – returns 0 value
- false – returns 1 value
- break – terminates executed cycle
- continue – starts new iteration of cycle
- eval *parameters* – executes parameters as command
- exit *value* – terminates script with return value
- . *script*, source *script* – reads and interprets another script
- : *argument* – just expand argument or do redirect
- alias *name='commands'* – expand *name* to commands
- unalias *name* – cancel alias
- if [*condition*]; then *commands*;
  elif [*condition*]; then *commands*;
  else *commands*; fi
- for *variable* in *arguments*; do *commands;* done
  - {a..z} – expands to a b c ...z
  - {*i..n..s*} – sequence from *i* to *n* with step *s*
  - \"{a,b,c}\" – expands to "a" "b" "c"
  - {1,2}{a,b} – expands to 1a 1b 2a 2b
  - seq *start step end* – number sequence
- for((i=1; i<10; i++)); do *commands;* done
- while *returns true*; do *commands;* done
- until [*test returns true*]; do *commands;* done
- case $prom in value$_1$) commands ;;
  value$_2$) commands ;; *) implicit. commands ;;
  esac
- Function definition: function *name* () {*commands*; }
- return *value* – return value of the function
- declare -f *function* – print function declaration

6 Redirections and pipes

- 0 stdin/input, 1 stdout/output, 2 stderr/error output
- > *file* – redirection, create new file or truncate it to zero size
- >> *file* – append new data at the end of file
- *command$_1$*<<<*command$_2$* – ouput from 2nd to stdin of 1st
- *command* < *file* – read stdin from file
- tee *file* – read stdin, writes to file and to stdout
- *command* 2> *file* – redirect error messages to file
- exec 1> >(tee -a log.txt) – redirect stdout also to file
- 2>&1 – merge stderr and stdout
- exec 3<>/dev/tcp/*addr*/*port* – create descriptor for network read/write
- exec 3>&- – close descriptor
- *command* > /dev/null 2>&1 – suppress all output
- n> n>> n>&m – operation redirect for descriptors *n*, *m*
- mkfifo *name* – make a named pipe, it can be written and read as file
- *command$_1$* | *command$_2$* – pipe, connection between processes
- *command* 2>&1 | ... – can be shortened to *command* |& ...
- ${PIPESTATUS[0]}, ${PIPESTATUS[1]} – retvals before and after pipe
- read *parameters* – read input line and separate it into parameters

6.1. Input for interactive programs (here documents)

```
./program <<  EOF       ./program <<-'EOF' # suppress tabulators
Input1                      Input1
Input2                      Input2
EOF                         EOF
```

6.2. Process file line by line

```
cat file.txt | (while read L; do echo "$L"; done)
```

7 Evaluating mathematical expressions

- let *expression*, expr *expression*, $((*expression*)), $((*expression1*, *expression2*)), $[*expression*]
- Numeric systems: *base*#*number*; hexa 0xABC, octal 0253, binary 2#10101011
- Operators: i++, ++i, i--, --i, +, -; ** power, *, /, % remainder; logical: ! neg., && and, || or; binary: ~, &, |; <<, >> shifts; assignment: = *= /= %= += -= <>= &= ^= |= >>= <<=; relations: < <= > >=
- factor *n* – factorize *n* into primes
- Floating point operations: echo "scale=10; 22/7" | bc

8 Screen output

- echo "text" – print text, echo * print all files in current dir
- echo -e "text" – interpret escape-sequences (\t tab., \a beep, \f new page, \n new line), -n, \c suppressing \n, \x*HH* hex-byte, *nnn* oct. byte, \u03B1 „α" (U+03B1) in UTF-8
- stty – change and print terminal line settings
- tty – print name of terminal connected to stdout
- printf *format values* – format output
- printf -v *variable form. val.* – form. output into variable
  - % [flags][width][.precision][length]specifier
  - Specifier: %u, %d, %i decimal; %E, %f float, %x, %X hex; %o octal, %s string, %% char %
  - Width: n prints at least n chars, spaces from right, $0n$ print at least n chars, zeros from left, * width specified in preceding parameter
  - Precision: min. number of digits, digits after decimal point, number of printed chars, * number of chars given by preceding parameter
  - Flags: - left-justify, + prints number with sign +/-
- printf "%d" \'A – display ASCII code of char "A" (65)
- printf \\$(printf '%03o' 65) – print char given by ASCII code
- tput *action* – terminal dependent action
- reset, tput sgr0, tset – reset terminal, cancel attributes
- clear, tput clear – clear screen

9 Process management

- *command* & – run *command* in background
- prog$_1$ && prog$_2$ – run prog$_2$, if prog$_1$ ends with success
- prog$_1$ || prog$_2$ – rub prog$_2$, if prog$_1$ ends with error
- Ctrl+z – stop process (SIGSTOP)
- bg / fg – run last stopped process in background/foreground
- jobs – list processes running in background
- exec *command* – shell is replaced by *command*
- wait – wait for end of background tasks
- top – watch CPU, memory, system utilization
- ps -xau – list processes and users, ps -xaf, pstree tree listing
- pgrep *process*, pidof *process* – get PID by name of process
- nice -n *p command* – priority *p* od −20 (max.) to 19 (min.)
- renice -n *p* -p *pid* – change priority of running process
- kill -s *k n* – send signal *k* to proces id. *n*, 0, 1 SIGHUP; 2 SIGINT Ctrl+c; 3 SIGQUIT; 9 SIGKILL; 15 SIGTERM; 24 SIGSTOP
- trap '*command*' signals – run command when signal received
- killall *name* – send signals to process by name
- nohup *command* & – command will continue after logout
- time *command* – print time of process execution
- times – print user and system time utilization in current shell
- watch -n *s command* – every *s* seconds run command

10 Time and process planning

- date – print date, date --date=@*unix_time*
- date +"%Y%m%d %H:%M:%S %Z" – format to 20130610 13:39:02 CEST
- cal – display calendar
- crontab -e – edit crontab, -l list, format *min hour date month day command*, * * * * * command run every minute, 1 * * * * command 1st min of every hour
- at, batch, atq, atrm – queue, examine or delete jobs for later execution

11 File operations

File name wildchars: ? a char; * zero or more chars; [*set*] one or more given chars, interval [0-9] [a-z], [A-Z]; [!*set*], [^*set*] none of chars.

- ls – list directory, ls -la, vdir all files with info
- tree – display hierarchy tree of directories
- file *file* – determine file by its magic number
- lsattr, chattr – list and change file attributes for ext2,3
- umask – define permission mask for new file
- pwd (-P) – logical (physical) path to current directory
- cd directory – change directory, cd jump to $HOME, cd - to $OLDPWD
- dirs – list stack of directories
- pushd *directory* – store *directory* to stack
- popd – set top stack directory as actual directory
- cp *source target* – copy file
- ln -s *source link* – create a symbolic link
- mkdir, rmdir – create, remove directory
- rm *file*, rm -r -f *directory*, unlink – delete
- touch *file* – create file, set actual time to existing file
- du -h – display space usage of directories
- stat *file* – file statistics, stat --format=%s size
- basename *name suffix* – remove path or suffix
- dirname */path/to/file* – print only path
- repquota – summarize quotas for a filesystem
- mktemp – create file with unique name in /tmp

12 Work with file content

- cat – concatenate files and print them to stdout
- cat > file – create file, end with Ctrl+d
- mapfile A < file – store stdin into array $A
- tac – like cat, but from bottom to top line
- more, less – print by pages, scrollable
- od, hexdump -C, xxd – print in octal, hex dump
- wc – get number of lines -l, chars -n, bytes -c, words -w
- head/tail – print begin/end, tailf, tail -f wait for new lines
- split, csplit – split file by size, content
- sort – -n numerical, -r reverse, -f ignore case
- uniq – omit repeated lines, -d show only duplicates
- sed -e '*script*' – stream editor, script y/ABC/abc/ replaces A, B, C for a, b, c; s/regexp/substitution/
- tr *a b* – replace char *a* for *b*
- tr '[a-z]' '[A-Z]' < file.txt – change lowercase to uppercase
- awk '/pattern/ {action }' *file* – process lines containing pattern
- cut -d *delimiter* -f *field* – print column(s)
- cmp *file1 file2* – compare files and print first difference
- diff, diff3, sdiff, vimdiff – compare whole files
- dd if=*in* of=*out* bs=*k* count=*n* – read *n* blocks of *k* bytes
- strings – show printable strings in binary file
- paste *file$_1$ file$_2$* – merge lines of files
- rev – reverse every line

13 Search

- whereis, which – find path to command
- grep – -i ignore case, -n print line number, -v display everything except pattern, -E extended regexp
- locate *file* – find file
- find *path* -name 'file*' – search for *file**
- find *path* -exec grep *text* -H {}\; – find file containing *text*

14 Users and permissions

- whoami, who am i – tell who I am :)
- w, who, users, finger – list connected users
- last / lastb – history successful / unsuccessful logins
- logout, Ctrl+d – exit shell
- su *login* – change user to *login*
- sudo – run command as other user
- id *login*, groups *login* – show user details
- useradd, userdel, usermod – create, delete, edit user
- groupadd, groupdel, groupmod – create, delete, edit group
- passwd – change password
- pwck – check integrity of /etc/passwd
- chown *user:group file* – change owner, -R recursion
- chgrp *group file* – change group of file
- chmod *permissions file* – change permissions in octal of user, group, others; 444=-r--r--r--, 700=-rwx------, 550=-r-xr-x---
- runuser *login* -c *"command"* – run command as user

15 System utilities

- uname -a – name and version of operating system
- uptime – how long the system has been running
- fuser – identify processes using files or sockets
- lsof – list open files
- sync – flush file system buffers
- chroot *dir command* – run command with special root directory
- strace,ltrace *program* – show used system/library calls
- ldd *binary* – show library dependencies

15.1. Disk partitions

- df – display free space
- mount – print mounted partitions
- mount -o remount -r -n / – change mount read only
- mount -o remount -w -n / – change mount writeable
- mount -t iso9660 cdrom.iso /mnt/dir -o loop – mount image
- mount -t cifs \\\\server\\ftp /mnt/adr -o user=a,passwd=b
- umount *partition* – unmount partition
- fdisk -l – list disk devices and partitions

- `blkid` – display attributes of block devices
- `tune2fs` – change ext2/3/4 filesystem parameters
- `mkfs.ext2`, `mkfs.ext3` – build file-system
- `hdparm` – set/read parameters of SATA/IDE devices

15.2. System utilization

- `ulimit -l` – print limits of system resources
- `free`, `vmstat` – display usage of physical, virt. memory
- `lspci`, `lsusb` – list PCI, USB devices
- `dmesg` – display messages from kernel
- `sysctl` – configure kernel parameters at runtime
- `dmidecode` – decoder for BIOS data (DMI table)
- `init`, `telinit` – command `init` to change runlevel
- `runlevel`, `who -r` – display current runlevel

16 Networking

- `hostname` – display computer hostname
- `ping` *`host`* – send ICMP ECHO_REQUEST
- `dhclient eth0` – dynamically set `eth0` configuration
- `host, nslookup` *`host/adr`* – DNS query
- `dig` – get record from DNS
- `whois` *`domain`* – finds owner of domain or network range
- `ethtool eth0` – change HW parameters of network interface `eth0`
- `ifconfig` – display network devices, device configuration
- `ifconfig eth0 add 10.0.0.1 netmask 255.255.255.0`
- `ifconfig eth0 hw ether 01:02:03:04:05:06` – change MAC address
- `route add default gw 10.0.0.138` – set network gateway
- `route -n`, `netstat -rn` – display route table
- `netstat -tlnp` – display processes listening on ports
- `arp` – display ARP table
- `iptables -L` – display firewall rules
- `tcpdump -i eth0 'tcp port 80'` – display HTTP communication
- `tcpdump -i eth0 'not port ssh'` – all communication except SSH
- `ssh user@hostname` *`command`* – run command remotely
- `mail -s "subject" address` – send email to address
- `wget -e robots=off -r -L http://`*`path`* – mirror given page