System & Terminal

SYSTEM & TERMINAL	
uname	[**-s**] prints the kernel name, **-r** kernel version, **-v** order and date of kernel compilation, **-p** processor type (architecture), **-o** operating system name, **-n** computer network name (nodename), **-a** all information
lsb_release	prints LSB (Linux Standard Base) version, **-i** Linux distribution, **-r** version of the Linux distribution, **-d** Linux distribution name and version, **-c** codename of the distribution version, **-s** short output (without header), **-a** all information `$ lsb_release -si` (prints the Linux distribution name)
cat /etc/*release | **lsb_release -d**	prints the Linux distribution and its version

dmidecode	displays information about the system's hardware components and BIOS, **-s** <keyword> information about a specified item from the list of available keywords, **-t** <type> information about a specified item from the list of available types
	`# dmidecode -s system-product-name`
	(prints the machine type (first 4 characters) and model)
	`# dmidecode -s system-serial-number`
	(prints the system serial number)
	`# dmidecode -s system-uuid`
	(prints the system ID)
	`# dmidecode -s system-product-name | egrep -i 'virtual|vm|bochs'`
	(prints possible information about the system virtualization)
	`# dmidecode -t processor`
	(prints information about the processor)
	`# dmidecode -t memory`
	(prints information about physical memory)
	`# dmidecode -t baseboard`
	(prints information about the base board)
	`# dmidecode -t bios`
	(prints information about the BIOS)

lscpu	displays information about the processor
cat /proc/cpuinfo	displays detailed information about the particular processor cores
mpstat [<interval> [<count>]]	displays the processor usage statistics, **-P** {<CPU_number[,...]> | ALL} specifies the processor `$ mpstat -P ALL 2 5` (prints five statistics reports on processor usage at two second intervals)

SYSTEM & TERMINAL	
sar [<interval> [<count>]]	displays the processor usage statistics, **-P** {<CPU_number[,...]> | ALL} specifies the processor, **-d** disk load statistics, **-n** {<keyword[,...]> | ALL} network statistics, **-r** memory usage statistics, **-S** swap usage statistics, **-f** <file> reads data from a file, **-o** <file> writes data to a file, **-h** in human readable format `$ sar -P ALL 2 5` (prints five statistics reports on processor usage at two second intervals)
free	displays the total amount and usage of physical memory and swap in the system and the amount of buffers and cache used by the kernel in kB (data from */proc/meminfo*), **-m** in MB, **-h** in human readable format, **-t** total
cat /proc/meminfo	displays detailed information about physical memory and swap usage in kB
vmstat [<interval> [<count>]]	displays statistics on virtual memory, swap, block devices and processor usage, **-d** disk load statistics, **-p** <partition> partition load statistics, **-s** output in a table format, **-w** output in a wider format `$ vmstat -w 2 5` (prints five statistics reports on virtual memory, swap, disks and processor usage at two second intervals)
nmon	displays information about the processor, memory, swap, network, disks, kernel, file systems, NFS and top processes in an interactive and dynamic way
ulimit	[**-f**] prints or sets the resource usage limits of the shell and any processes started by it, **-a** prints all current limits, **-H** sets a "hard" resource limit, **-S** sets a "soft" resource limit, **-f** <limit> the maximum size of files created by the shell in MB, **-n** <limit> the maximum number of open file descriptors, **-t** <limit> the maximum amount of the CPU time in seconds, **-u** <limit> the maximum number of processes available to a single user; (permanent settings are configured in */etc/security/limits.conf* or */etc/security/limits.d/*.conf*)
setup	starts a configuration tool for the system setup

getconf [<variable>]	displays the specified system configuration values, **-a** all values
	`$ getconf LONG_BIT`
	(displays the system architecture)
	`$ getconf _NPROCESSORS_ONLN`
	(displays the number of processors)
	`$ getconf -a | grep ^NAME_MAX`
	(displays the maximum file name length in bytes)

getent <database> [<key ...>]	displays entries from databases (files) supported by the Name Service Switch libraries, configured in */etc/nsswitch.conf*
	`$ getent passwd`
	(lists all local and remote (LDAP or NIS) users on the system)
	`$ getent services smtp`
	(prints the service name and its port and protocol)

locale	displays the current locale settings for each locale category, **-a** lists all available locales

SYSTEM & TERMINAL	
localectl (implemented from RHEL 7)	[**status**] displays the system locale and keyboard layout settings (data from */etc/locale.conf*), **set-locale LANG=**<locale> sets the system locale and keyboard layout settings `# localectl set-locale LANG=en_US.utf8` (sets the English localization of the system)

cal [[[<DD>] <MM>] <YYYY>]	displays a calendar (the current month or the specified year or the specified month of the year), **-m** Monday as the first day of the week, **-y** the entire current year, **-j** ordinal days in the year, **-w** week numbers, **-v** vertical layout
	`$ cal 12 2005`
	(displays December 2005)
	`$ for year in {2000..2100}; do { cal -mv 04 $year; echo;} | grep -E '^Mo.+[ ]+3[ ]+' -C 7; done`
	(displays years between 2000 and 2100 in which April 3 falls on a Monday)

date [<MMDDhhmmYY>]	displays or sets the current system date and time, **+**<format> displays the date and time in a specified format (e.g., **%H** prints hour in a 2-digit 24-hour format, **%M** minute in a 2-digit format, **%S** second in a 2-digit format, **%d** day in month in a 2-digit format, **%m** month in a 2-digit format, **%Y** year in a 4-digit format, **%F** YYYY-MM-DD format, **%s** number of seconds since January 1st, 1970), **-d** <specification> prints the date as specified
	`$ date +%H:%M-%d.%m-%Y`
	(displays the current system date and time in hh:mm-DD.MM-RRRR format)
	`$ date -d @123456`
	(converts the time in seconds since January 1st, 1970 to human readable format)
	`$ date -d "+90 days" +%F`
	(prints the date in 90 days in YYYY-MM-DD format)
	`$ TZ='America/New_York' date`
	(displays the current date and time in a specified time zone)

timedatectl (implemented from RHEL 7)	displays the current system date and time, including the time zone, **set-time** <YYYY-MM-DD hh:mm:ss> sets the current system date and time, **list-timezones** lists all available time zones, **set-timezone** <time_zone> sets a time zone, **set-ntp** {true | false} enables or disables NTP synchronization
tzselect	selects a time zone in interactive mode
ntpdate <server>	sets the system date and time via a specified NTP server (permanent settings are configured in */etc/ntp.conf*) `# ntpdate tak.cesnet.cz`
hwclock	**-r** prints hardware date and time, **-s** sets the system time according to the hardware clock, **-w** sets the hardware time according to the system clock, **--set --date=**<YYMMDD hh:mm:ss> sets the hardware time to the specified value `# hwclock --set --date='780321 22:06:56'`

SYSTEM & TERMINAL	
uptime	displays the system time, how long the system has been running since the last boot, how many users are currently logged in and the system load averages for the past 1, 5 and 15 minutes, **-s** the time of the last system boot (equivalent to the "who -b" command)
who -b | **uptime -s**	prints the time of the last system boot
dmesg	prints messages from the kernel ring buffer, especially about the hardware devices that the kernel detects during the boot process (data from */var/log/dmesg*), **-L** colorizes important messages, **-T** displays the date and time in human readable format `$ dmesg | grep -i usb` (prints information about USB devices recognized by the kernel)
cat /var/log/messages	prints generic system messages, including those that are logged during system startup
cat /var/log/secure	prints messages related to security and authentication events

journalctl
(implemented from RHEL 7)
	prints all systemd logs stored by the journal daemon by default in */run/log/journal* or */var/log/journal*, **-b** since the last reboot (by default), **-S** <YYYY-MM-DD [<HH:mm:ss>]> since a specified period (a persistent log storage may be required), **-U** <YYYY-MM-DD [<HH:mm:ss>]> until a specified period (a persistent log storage may be required), **-n** [<n>] last *n* entries (10 by default), **-f** continuously prints new entries as they are appended to the journal, **-k** kernel messages only, **-u** <unit> messages for a specified systemd unit (service), **--user-unit** <unit> messages for a specified logged-in user's unit (service), **-p** <priority> messages of a specified priority or higher, **-o** <output> specifies the output format, **-r** reverse output (the newest entries are displayed first), **--disk-usage** prints the total disk usage of all archived and active journal files, **q** quits the program
	`# journalctl -b -p err`
	(displays all systemd logs since the last reboot with a priority of "error" or higher)
	`# journalctl -u httpd -S yesterday`
	(displays logs related to the web server since yesterday)
	`# journalctl -ko verbose`
	(displays kernel messages in detailed output)
	`# journalctl --disk-usage`
	(displays the total disk usage of all archived and active journal files)

SYSTEM & TERMINAL	

auditctl	manages audit daemon rules for logging system event information to */var/log/audit/audit.log*, **-a** <list,action> adds a system call rule, **-F** <field_specification> specifies a filter based on audit record fields, **-C** <field_comparison> specifies fields to be compared, **-S** <system_call> specifies a system call, **-w** <file | directory> adds a watch rule for a file or directory (recursively, within one file system), **-p** {r | w | x | a} specifies the permission access type on which the watch rule is triggered ("r" = read, "w" = write, "x" = execute, "a" = attribute change), **-k** <keyword> specifies a keyword for an audit rule, **-l** lists all rules, **-d** <list,action> <other_specification> removes a system call rule, **-W** <file | directory> removes a watch rule for a file or directory, **-D** removes all rules; (permanent settings are configured in */etc/audit/rules.d/audit.rules*)
	`# auditctl -a always,exit -S all -F pid=1011`
	(creates a rule to audit all system calls made by a specified process)
	`# auditctl -a always,exit -S openat -F auid=530`
	(creates a rule to audit files opened by a specified user)
	`# auditctl -w /etc -p wa -k etc_change`
	(creates a rule to recursively audit changes to the /etc directory contents and attributes and associates the records with the keyword "etc_change")

aureport	creates a summary report of audit daemon logs, **-a** reports SELinux messages, **-au** reports authentication attempts, **-l** reports login attempts, **--failed** reports failed events only, **--success** reports successful events only, **-x** reports selected executables, **-ts** [<date>] [<time>] specifies the start date/time (without the date/time specified today/midnight is assumed), **-te** [<date>] [<time>] specifies the end date/time (without the date/time specified today/now is assumed), **-i** converts numeric values to names, **--summary** reports a total of each element, **-if** <file> specifies the input file instead of the default */var/log/audit/audit.log*
	`# aureport -ts today`
	(creates a summary report of today)
	`# aureport -i -l --summary`
	(creates a summary report of all login attempts)
	`# aureport -x --summary`
	(creates a summary report of the execution of selected executables)

SYSTEM & TERMINAL	

ausearch	searches audit daemon logs based on specified criteria, **-a** <ID> specifies an event ID, **-f** <file | directory> specifies a file or directory name, **-x** <executable> specifies an executable name, **-k** <keyword> specifies a previously defined keyword, **-m** <message_type> specifies a message type, **-n** <computer> specifies a remote computer, **-p** <PID> specifies a process, **-ts** [<date>] [<time>] specifies the start date/time (without the date/time specified today/midnight is assumed), **-te** [<date>] [<time>] specifies the end date/time (without the date/time specified today/now is assumed), **-i** converts numeric values to names, **--format** <format> specifies the output format, **-if** <file> specifies the input file instead of the default */var/log/audit/audit.log*
	`# ausearch -m AVC -ts boot`
	(searches for SELinux records since the last system boot)
	`# ausearch -m AVC -ts recent`
	(searches for SELinux records within the last 10 minutes)
	`# ausearch -k etc_change`
	(searches for records based on the previously defined keyword)
	`# ausearch -m USER_LOGIN --format csv > report.csv`
	(searches for records related to user login events and exports the results in CSV format)

logger [<message>]	creates an entry in the system log, **-p** <priority> specifies the priority; with no argument the contents of STDIN are logged `$ logger -p local3.info test` (creates a log containing the string "test" by the facility "local3" with priority "info")
cat /etc/grub.conf | **cat /etc/lilo.conf**	prints the boot loader setup for starting the OS
cat /etc/default/grub (implemented from RHEL 7 – GRUB 2)	prints the boot loader setup for starting the OS, the related scripts are found in */etc/grub.d*
grub-mkconfig	generates a configuration file for GRUB based on data in */etc/grub.conf*, **-o** <file> writes the generated output to a file instead of STDOUT `# grub-mkconfig -o /boot/grub/grub.cfg`

grub2-mkconfig
(implemented from RHEL 7 – GRUB 2)
	generates a configuration file for GRUB 2 based on data in */etc/default/grub*, **-o** <file> writes the generated output to a file instead of STDOUT
	`# grub2-mkconfig -o /boot/grub2/grub.cfg`
	(generates a GRUB 2 configuration file on a system with the BIOS firmware)
	`# grub2-mkconfig -o /boot/efi/EFI/${ID}/grub.cfg`
	(generates a GRUB 2 configuration file on a system with the UEFI firmware)

grub-install <device>	installs GRUB on a specified device `# grub-install /dev/sda`
grub2-install <device> (implemented from RHEL 7 – GRUB 2)	installs GRUB 2 on a specified device `# grub2-install /dev/sda`

SYSTEM & TERMINAL	

efibootmgr	prints the UEFI Boot Manager configuration, **-a** activates a boot option, **-A** deactivates a boot option, **-b** <bootnum> specifies a boot option, **-B** removes a boot option, **-o** <bootnum,bootnum...> sets the boot order, **-v** detailed output
	`# efibootmgr`
	(prints the current boot option, boot order and all available boot options)
	`# efibootmgr -b 0006 -B`
	(removes the specified boot option)

mkinitrd [<RAM_disk_image>] <kernel_version>	creates an initial RAM disk image containing basic modules needed for mounting the root file system "/" (only then it is possible to load the modules contained in */lib/modules/<kernel_version>*), **-f** overwrites an existing "initrd-$(uname -r).img" file, **--with=**<module> adds a kernel module into the initial RAM disk image, **-v** detailed output
	`# mkinitrd /boot/initrd-$(uname -r).img $(uname -r)`
	(recreates the initial RAM disk image)
	`# mkinitrd --with=raid1 /boot/initrd-raid1-$(uname -r).img $(uname -r)`
	(the new initial RAM disk image with the particular module has to be added to */boot/grub/grub.conf* as a new paragraph – i.e., copy the previous part from the line beginning with "title..." to "initrd..." and correct the new RAM disk image title on the last line; at system startup it is possible to choose to boot with this image)

dracut [<RAM_file_system_image> [<kernel_version>]]
(implemented from RHEL 6)
	creates an initial RAM file system image containing basic modules needed for mounting the root file system "/" (only then it is possible to load the modules contained in */lib/modules/<kernel_version>*), **-f** overwrites an existing "initramfs-$(uname -r).img" file, **-a** <module> adds a dracut module into the initramfs image, **-d** <module> adds a kernel module into the initramfs image, **--regenerate-all** regenerates all initramfs images at the default location with the kernel versions found on the system, **-v** detailed output
	`# dracut -f`
	(recreates the initial RAM file system image for the current kernel version)
	`# dracut -f /boot/initramfs-2.6.32-220.7.1.el6.x86_64.img 2.6.32-220.7.1.el6.x86_64`
	(recreates the initial RAM file system image for the specified kernel version)

chroot <root_directory>	changes the root directory to that specified by the argument and starts a new shell (e.g., when working in a rescue mode from an external medium) `# chroot /mnt/sysimage` (connects directly to the hard disk with the original installation from where it is possible to repair a damaged system)
lsmod	lists all modules (drivers) loaded in the kernel (data from */proc/modules*)

SYSTEM & TERMINAL	
modinfo <module ...>	prints information about a module, **-d** short description, **-p** lists the supported module parameters
insmod <module ...>	inserts a module into the kernel (provided no other dependencies are required)
rmmod <module>	removes a module from the kernel (provided no other dependencies are required), **-v** detailed output
depmod <module ...>	generates a list of dependencies of the specified module into */lib/modules/<kernel_version>/modules.dep*, **-a** all modules
modprobe <module ...>	inserts a module into the kernel including all its dependencies, **-r** remove a module, **-v** detailed output `# modprobe usbnet`

sysctl [<parameter ...>]	prints a specified kernel parameter and its current value, **-n** prints only the value of the parameter, **-a** prints all kernel parameters and their current values, **-w** <parameter>=<value> modifies kernel parameters at runtime (permanent settings are configured in */etc/sysctl.conf* or */etc/sysctl.d/**), **-p** <file> loads the settings from the specified file (*/etc/sysctl.conf* by default)
	`$ sysctl vm.swappiness`
	(prints the value of the "vm.swappiness" parameter)
	`# sysctl -w net.ipv4.icmp_echo_ignore_all=1` | `# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all`
	(ignores ping requests)
	`# sysctl -w net.ipv4.ip_forward=1` | `# echo 1 > /proc/sys/net/ipv4/ip_forward`
	(enables IP forwarding)
	`# sysctl -w net.ipv6.conf.all.disable_ipv6=1` | `# echo 1 > /proc/sys/net/ipv6/conf/all/disable_ipv6`
	(disables IPv6)

systemctl [<command>] (implemented from RHEL 7)	prints the state of all systemd objects ("units"), **show** [<unit ... | job ...>] displays properties of the specified unit, job or the systemd manager (if no argument is specified)
systemctl list-dependencies [<unit>] (implemented from RHEL 7)	displays a tree structure of all units that are required by the specified unit; with no argument the required units for "default.target" are displayed `$ systemctl list-dependencies graphical.target | grep "target"` (lists all "target units" required by the "graphical.target" unit)
systemd-analyze (implemented from RHEL 7)	**verify** <file ...> verifies that a systemd unit file contains no errors, **security** <unit ...> analyzes the security settings of a systemd service unit, **--user** manages user systemd files (units)
systemctl daemon-reload (implemented from RHEL 7)	reloads the systemd manager configuration, **--user** manages user systemd files (units)
runlevel	prints the previous and current system runlevel; if a runlevel cannot be determined, "N" is printed instead
who -r	prints the current system runlevel and the time it started

SYSTEM & TERMINAL	
systemctl list-units -t target (implemented from RHEL 7)	prints the current system runlevel (currently loaded "target units"), **--all** all "target units"
init | **telinit** <runlevel>	changes the system runlevel, **0** = halt, **1** = single-user mode without network, **2** = multi-user mode without network, **3** = multi-user mode with network, **4** = not used (user-definable), **5** = runlevel 3 + GUI, **6** = reboot; the configuration file of the init process is */etc/inittab*, if the file is changed, its reloading is triggered by the "init q" command `# init 5`

systemctl isolate <runlevel> (implemented from RHEL 7)	changes the system runlevel (current "target unit"), **runlevel0.target** | **poweroff.target** = power off, **runlevel1.target** | **rescue.target** = single-user mode without network, **runlevel2.target** | **multi-user.target** = multi-user mode with network, **runlevel3.target** | **multi-user.target** = multi-user mode with network, **runlevel4.target** | **multi-user.target** = multi-user mode with network, **runlevel5.target** | **graphical.target** = multi-user mode with network + GUI, **runlevel6.target** | **reboot.target** = reboot
	`# systemctl isolate multi-user.target`
	(changes the system runlevel to multi-user mode with network)
	`# systemctl isolate runlevel5.target`
	(changes the system runlevel to multi-user mode with network, including the graphical environment)

systemctl rescue (implemented from RHEL 7)	changes the system runlevel to single-user mode without network, mounts all local file systems and starts important system services
systemctl emergency (implemented from RHEL 7)	changes the system runlevel to single-user mode without network, mounts only the root file system for reading and starts the most important system services (used when the system is unable to enter rescue mode)
systemctl get-default (implemented from RHEL 7)	prints the default system runlevel (default "target unit")
systemctl set-default <runlevel> (implemented from RHEL 7)	changes the default system runlevel (default "target unit") `# systemctl set-default graphical.target` (changes the default system runlevel to multi-user mode with network, including the graphical environment)
reboot | **init 6**	stops all processes, saves pending data to the disk, unmounts file systems and reboots the system
systemctl reboot (implemented from RHEL 7)	stops all processes, saves pending data to the disk, unmounts file systems and reboots the system
halt | **init 0**	stops all processes, saves pending data to the disk, unmounts file systems and stops the processor (stand-by mode, the system can be safely shut down manually)
systemctl halt (implemented from RHEL 7)	stops all processes, saves pending data to the disk, unmounts file systems and stops the processor (stand-by mode, the system can be safely shut down manually)
poweroff	stops all processes, saves pending data to the disk, unmounts file systems and powers off the system

SYSTEM & TERMINAL	
systemctl poweroff (implemented from RHEL 7)	stops all processes, saves pending data to the disk, unmounts file systems and powers off the system

shutdown	stops all processes, saves pending data to the disk, unmounts file systems and stops the processor, **-h** <time> powers off the system, **-r** <time> reboots the system, **-c** cancels a scheduled shutdown, **-k** <time> false shutdown (only sends out warning messages to users and disables new logins to the system)
	`# shutdown -h now` | `# shutdown -h 22:00`
	(powers off the system immediately / at the specified time)
	`# shutdown -r +30 "quick reboot"`
	(reboots the system in 30 min and displays a warning message)

startx	starts an X window system
xterm	runs a terminal in an X window system, **-e** <command> specifies a command to be run in the xterm window `# xterm -e "tail -f /var/log/secure" &` (runs a terminal in which it displays newly incoming logs)
tty	prints the terminal name of the logged-in user
stty [<argument>]	prints or changes the terminal's settings for the logged-in user, **-a** prints the current settings in human readable format
reset	clears the terminal screen and sets its default values
clear	clears the terminal screen
chvt <n>	changes a virtual terminal in the range 1–12 (similar to the key combination (Ctrl-)LeftAlt-F<n>)
tmux [<command>]	creates and manages multiple terminal sessions within a single terminal window, **Ctrl**+**b** followed by **"** splits the current pane horizontally, **Ctrl**+**b** followed by **%** splits the current pane vertically, **Ctrl**+**b** followed by arrow keys switches between the panes, **Ctrl**+**b** followed by **{** moves the current pane left, **Ctrl**+**b** followed by **}** moves the current pane right, **Ctrl**+**b** followed by **!** converts the current pane into a window, **Ctrl**+**b** followed by **x** closes the current pane, **Ctrl**+**b** followed by **c** creates a new window, **Ctrl**+**b** followed by **w** lists windows, **Ctrl**+**b** followed by the window number switches between the windows, **Ctrl**+**b** followed by **&** closes the current window, **Ctrl**+**b** followed by **:** switches to the command mode, **q** quits the command mode; the status line at the bottom of the screen shows information on the current session and is used to enter interactive commands

SYSTEM & TERMINAL	

declare | **typeset** [<variable>[=<value>] ...]	defines a local variable and/or sets its attributes (in most cases it is enough with an implicit declaration <variable>=<value>), **-p** displays the attributes and values of each variable, **-f** displays shell functions only, **-a** treats the variable as an indexed array (keys are ordered integers starting from 0), **-A** treats the variable as an associative array (keys are arbitrary strings), **-i** treats the variable as an integer (arithmetic evaluation is performed when the variable is assigned a value), **-r** sets the variable "read-only" (equivalent to the "readonly" command), **-x** exports the variable to all child shell processes (equivalent to the "export" command), **+**<option> turns off the variable attribute (except for "+a" and "+r"); with no argument all variables (global + local) and shell functions are displayed
	`$ IndexedArray=(zero one two)` | `$ declare -a IndexedArray=(zero one two)`
	(defines an indexed array)
	`$ declare -A AssociativeArray=([system]="linux" [shell]="bash")`
	(defines an associative array)

readonly | **declare -r** | **typeset -r** [<variable>[=<value>] ...]	defines a "read-only" local variable whose values cannot be changed by subsequent assignment and which cannot be removed by the "unset" command, **-f** [<function>] sets a function "read-only" or lists "read-only" functions, **-a** treats the variable as an indexed array (keys are ordered integers), **-A** treats the variable as an associative array (keys are arbitrary strings); with no argument all "read-only" variables are displayed
export | **declare -x** | **typeset -x** [<variable>[=<value>] ...]	exports a local variable to all child shell processes (a local variable is permanently set by defining and exporting in *~/.bash_profile*, a global variable in */etc/profile*), **-f** [<function>] exports a function or lists exported functions, **-n** <variable | function> unexports a variable, with option "-f" unexports a function; with no argument all exported variables are displayed `$ x=y; export x` | `$ export x=y` (exports a local variable "x" with value "y")
set	prints all variables (global + local) and shell functions, **-o** prints the current shell option settings, **-/+o** <option> changes the shell option settings ("**-**" sets the option, "**+**" cancels the option), **-x** depending on the place of occurrence in the script, runs the script or part of it in debug mode, **+x** exits debug mode at the place of occurrence in the script `$ set -o vi` (sets the "vi" editor environment to the shell)
unset <variable ...>	removes a local variable, **-f** <function> removes a function `$ unset AssociativeArray[system]` (removes the key "system" and its value from the associative array variable)
printenv [<variable> ...]	prints all environment (global) variables and their values or values of the specified variables

SYSTEM & TERMINAL	
env [<variable>=<value> <command>]	runs a process in an environment temporarily modified by its own variable settings that are either added to or removed from the current environment, **-u** <variable> removes a variable, **-i** removes all variables (loads an empty environment); with no argument all environment (global) variables and their values are displayed (equivalent to the "printenv" command) `$ env LANG=en_EN.UTF-8 date` (displays the date in English)

echo [<text> | $<variable>]	prints the specified text or the value of the variable, **-e** enables interpretation of escape sequences (e.g, "\n" = newline, "\t" = horizontal tab, etc.), **-n** suppresses a newline at the end of the output; with no argument a blank line is printed
	`$ echo "End of file" >> ~/file.txt`
	(appends the specified text to the end of the file)
	`$ echo $LANG`
	(displays the value of the variable "LANG")
	`$ echo ${AssociativeArray[shell]}`
	(displays the value assigned to the key "shell" of the associative array)
	`$ echo ${AssociativeArray[@]}`
	(displays all values of the associative array)
	`$ for val in "${AssociativeArray[@]}"; do echo $val; done`
	(displays all values of the associative array, each on a separate line)
	`$ for key in "${!AssociativeArray[@]}"; do echo $key ${AssociativeArray[$key]}; done`
	(displays all keys and their values of the associative array, each on a separate line)

printf <format> [<argument>]	prints a formatted string; format specification e.g: **%s** string, **%d** decadic integer, **%%** %, **\n** new line, **\t** tabulator
	`$ printf "Hello, I am %s.\n" $LOGNAME`
	(prints the specified text and login name of the user)
	`$ printf "%s\t%s\t%s\n" Buy Sale Profit`
	(prints the specified text delimited by tabs)
	`$ for i in $(seq 1 10); do printf "%03d\n" "$i"; done`
	(prints numbers 1–10 in a 3-digit format starting with zero)
	`$ printf "#%.0s" {1..10}`
	(prints a specified string 10 times)
	`$ printf "\U0001f600\n"`
	(displays an emoji icon)

eval <argument ...>	concatenates the arguments and executes them as a shell command `$ eval $(ssh-agent)` (executes the "ssh-agent" command and sets the environment variables for the shell session)

SYSTEM & TERMINAL	

xargs [<command>]	reads and then divides a large number of arguments from STDIN into smaller parts so that they can be assigned to the next command that will be executed (commands cannot be assigned an unlimited number of arguments), **-0** arguments are terminated by a null character instead of whitespace and quotes and backslashes in their names have no special meaning (this option must also be supported by the command providing the input for "xargs"), **-t** prints the command line on STDERR before executing it; with no argument the command "echo" is executed
	`# find . -name "*core" -print0 | xargs -0 -t rm`
	(deletes files named "core")
	`$ find . -type d | xargs chmod 750`
	(sets the specified permissions to directories)
	`$ find . -atime -1 -type f | xargs ls -lutr | tail -1`
	(prints the last used file)
	`$ pgrep ssh | xargs`
	(prints PIDs of the ssh processes next to each other instead of under each other)

expr <expression>	evaluates mathematical and other expressions; individual arguments are separated by spaces and special characters must be suppressed against the shell expansion
	`$ expr 7 + 3`
	`$ expr "(" 6 - 3 ")" "*" 10`
	`$ expr length ABCD`

let <argument ...>	evaluates each argument as an arithmetic expression `$ let a=1+2`
bc [<file>]	evaluates mathematical and other expressions from a specified file; with no argument it starts in interactive mode

units [<source_unit> <target_unit>]	converts units from one scale to another (and additionally displays the conversion in the reverse direction); with no argument it starts in interactive mode
	`$ units 'mile' 'km'`
	(converts a mile to kilometers)
	`$ units '27 inch' 'cm'`
	(converts 27 inches to centimeters)

SYSTEM & TERMINAL	

seq <specification>	prints a sequence of numbers (one on each line) defined by an upper and possibly lower limit or an increment, **-f** <format> in a specified format, **-s** <string> uses a string to separate the numbers, **-w** equalizes width by padding with leading zeroes in front of a single digit
	`$ seq 2 5`
	(a lower and upper limit defined)
	`$ seq 1 .2 3`
	(a lower and upper limit defined including an increment)
	`$ seq -w 15`
	(digits 01–15)
	`$ seq -s ':' 10`
	(digits 1–10 separated by ":"
	`$ seq -f '***%g***' 3`
	(digits 1–3 between three stars)
	`$ for i in $(seq 1 100); do expr $i "*" $i; done`
	(prints the square of numbers 1–100, each on a separate line)

shuf [<file>]	generates random permutations (each on a separate line), **-e** treats each argument as an input line, **-i** <m-n> treats each number of the specified range as an input line, **-n** <n> specifies the number of output lines, **-r** repeats the output lines, **-o** <file> specifies the output file; if no file is specified, reads from STDIN
	`$ shuf -e a b c`
	(generates random permutations of the specified characters)
	`$ shuf -n 1 -e Tom Dan Jan`
	(randomly selects one of the names)
	`$ shuf -i 1-9`
	(generates random numbers within a specified range)
	`$ for n in {01..10}; do echo "$n:" "$(shuf -e {01..49} -n 6 | tr '\n' ' ')"; done`
	(generates 10 times 6 random numbers in the range 1–49)
	`$ shuf -e "${IndexedArray[@]}"`
	(generates random permutations of an indexed array values)

yes [<string ...>]	repeats printing "y" or a specified string (each on a separate line)
timeout <duration> <command>	executes a command with a time limit, **-s** <signal> specifies a signal to be sent on timeout ("SIGTERM" by default); the duration is specified by a number and a suffix **s** in seconds (by default), **m** in minutes, **h** in hours, **d** in days `$ timeout 5 ping prompt.cz` (runs the "ping" command for 5 seconds)

time <command>	measures the duration of a command execution
	`$ time wget prompt.cz`
	`$ time timeout 5s yes`

alias [<custom_command>[=<command>]]	creates an alias for the specified command (permanent settings are configured in *~/.bashrc*); with no argument all current aliases are printed `$ alias nopaste='curl -F file=@- nopaste.com/a'`

SYSTEM & TERMINAL	
unalias [<custom_command ...>]	removes an alias, **-a** removes all aliases
script [<file>]	saves everything that is displayed after executing the program on STDOUT into the specified file, **-a** appends the output into the existing file, **-c** <command> saves only the output of the specified command, **Ctrl**+**d** exits the program; with no argument the dialogue is saved in the file "typescript"
history [<n>]	prints the history of last *n* commands (last 1000 commands by default), **-c** clears the history, **-d** <n> clears the history entry at the specified position (line), **-a** appends the new command history entered since the beginning of the current shell session to the command history file (*.bash_history*), **-r** appends the contents of the command history file to the current command history list

fc [<n> | <m n>]	edits and executes last or selected command(s) from the history, **-l** prints a numbered list of last executed or selected commands from the history, **-r** reverses the order of the commands, **-s** executes the last or selected command from the history
	`$ fc -l -20 | grep cat`
	(prints last 20 commands containing "cat")
	`$ fc -l 210 250`
	(prints a list of commands in the specified range)
	`$ fc -s | $ !!`
	(executes the previous command)
	`$ fc -s 560 | $ !560`
	(executes a selected command)

setenforce <mode>	sets a SELinux mode if it is enabled (**0** = permissive – the security policy is not applied, only possible warning messages are being logged into */var/log/messages*, **1** = enforcing – the security policy defining access to files, ports and processes is applied); permanent settings are configured in */etc/selinux/config*; when changing the SELinux status from "disabled" to "enforcing" or "permissive" or vice versa, the system must be restarted
getenforce	prints the current SELinux mode ("enforcing", "permissive", or "disabled")
sestatus	prints the current SELinux status, possibly its mode and policy being used, **-v** detailed output based on data from */etc/sestatus.conf*

SYSTEM & TERMINAL	

semanage <object> <definition>	defines SELinux policy rules, object **fcontext** defines the rules that the "restorecon" command uses to set the default SELinux security context on a file or directory, **boolean** defines rules for booleans, **port** defines rules for ports, **login** defines rules for Linux users, **user** defines rules for SELinux users, **-a** adds (defines) a security context, **-d** deletes a security context, **-m** modifies a security context, **-t** specifies the context type, **-l** lists a security context, **-C** with "-l" option lists only local modifications from the default policy
	`# semanage fcontext -l`
	(prints the default SELinux security context for files and directories)
	`# semanage fcontext -at httpd_sys_content_t '/web(/.*)?'`
	(defines the default SELinux security context for the directory and its contents)
	`# semanage fcontext -d '/web(/.*)?'`
	(deletes the SELinux security context from the directory and its contents)
	`# semanage boolean -l | grep httpd_enable_homedirs`
	(prints whether a boolean is enabled permanently)
	`# semanage port -l`
	(prints the port types including their assigned numbers)
	`# semanage port -at http_port_t -p tcp 81`
	(allows the web server to listen on TCP port 81)
	`# semanage login -l`
	(prints the mappings between Linux users and SELinux users)
	`# semanage user -l`
	(prints the mappings between SELinux users and SELinux roles restricting the commands that the user can execute)
	`# semanage login -m -s user_u -r s0 __default__`
	(changes the default mapping to map the Linux users to the "user_u" SELinux user)
	`# semanage login -a -s sysadm_u admin`
	(maps the system administrator to the "sysadm_u" SELinux user)

restorecon <file ... | directory ...>	sets (applies) the default SELinux security context on the file or directory, **-R** recursively, **-v** detailed output `# restorecon -Rv /web` (recursively sets the default SELinux security context on the "/web" directory)

SYSTEM & TERMINAL	

chcon <context> <file ... | directory ...>	temporarily changes the SELinux security context of a file or directory (the change will be preserved until the system is rebooted or until the "restorecon" command is run; the security context consists of 5 colon separated parts – user:role:type:level:category, 2 last of which may not be defined in the system; when copying a file its security context changes according to the new, parent directory, when moving a file its security context persists), **-u** <user> for a specified user ("root" is used for the root user, "user_u" for other users and "system_u" for processes), **-r** <role> for a specified role (defines the purpose of a particular file, process or user; "object_r" is used for files and "system_r" for processes and users), **-t** <type> for a specified type (defines which types of processes can access specific types of files; "unconfined_t" means that it is not restricted by SELinux policy), **--reference** <source> sets a file the security context according to the source file, **-R** recursively, **-v** detailed output
	`# chcon -Rt httpd_sys_content_t /web`
	(temporarily changes the security context of the directory so that its contents are accessible only to the web server)
	`# chcon --reference /etc/hosts.allow /etc/hosts.deny`
	(sets the second file a security context according to the first file)

setsebool <boolean> <state>	sets the state of the SELinux boolean, **-P** permanent settings `# setsebool -P httpd_enable_homedirs on` (activates the specified SELinux boolean)
getsebool [<boolean>]	prints the current state of the SELinux boolean, **-a** prints all booleans and their states
seinfo [<expression>] [<policy>]	prints SELinux policy statistics, **-b** list of booleans, **-r** list of roles, **-t** list of types (contexts), **-u** list of users
sealert	displays useful information that can help with SELinux troubleshooting, **-a** <file> prints all incidents in the file, **-l** <ID> prints information about a specified incident `# sealert -a /var/log/audit/audit.log` (prints all incidents in the file related to SELinux)

SYSTEM & TERMINAL	

virsh <command> [<virtual_machine>] [<argument ...>]	**nodeinfo** prints the processor and physical memory information of the host (KVM hypervisor) running the virtual machines (domains), **list** prints active virtual machines, **domstate** <VM> prints the status of a virtual machine, **start** <VM> starts a virtual machine, **shutdown** <VM> shuts down a virtual machine, **reboot** <VM> reboots a virtual machine, **dominfo** <VM> prints basic information about a virtual machine, **domblklist** <VM> prints block devices of a virtual machine, **domifaddr** <VM> prints network interfaces of a virtual machine with their IP and MAC addresses, **console** <VM> opens the console of a virtual machine, **net-list** prints active virtual networks, **net-start** <network> starts a virtual network
	`# virsh list --all`
	(prints the status of all virtual machines)
	`# virsh net-start default`
	(starts a "default" virtual network)

exit | **Ctrl**+**d**	exits a shell (logs a user out)
logout	exits a login shell

Bash Keyboard Shortcuts	
Commands / File Names Completions:	
Tab	allows to quickly complete commands or file names after they have been typed enough at the prompt to make them unique; if the characters typed are not unique, pressing the Tab key twice displays all available commands or file names that begin with the characters already typed
Command History:	
Ctrl+**p** | **Up Arrow**	prints the previous command in the history
Ctrl+**n** | **Down Arrow**	prints the next command in the history
Ctrl+**r**	searches the command history for a pattern typed (press it again for the next search)
Ctrl+**o**	executes the command found with "Ctrl+r"
Ctrl+**g**	exits the command history search
Process Management:	
Ctrl+**c**	terminates a running process in the foreground
Ctrl+**d**	terminates a running process that reads data from STDIN / exits the current shell
Ctrl+**z**	suspends a running job in the foreground
Ctrl+**s**	suspends all output to the screen (useful when running a command with a long verbose output, but it is not necessary to terminate the process)
Ctrl+**q**	resumes the output to the screen after stopping it with "Ctrl+s"
Moving the Cursor:	
Ctrl+**b** | **Left Arrow**	moves the cursor one character to the left
Ctrl+**f** | **Right Arrow**	moves the cursor one character to the right
Alt+**b**	moves the cursor one word to the left

Bash Keyboard Shortcuts	
Alt+**f**	moves the cursor one word to the right
Ctrl+**a**	moves the cursor to the beginning of the command line
Ctrl+**e**	moves the cursor to the end of the command line
Editing the Characters:	
Ctrl+**d** | **Delete**	clears the character under the cursor
Ctrl+**h** | **Backspace**	clears the character before the cursor
Ctrl+**u**	clears all characters before the cursor, adding them to the clipboard
Ctrl+**k**	clears all characters after the cursor, adding them to the clipboard
Ctrl+**l**	clears the entire screen
Ctrl+**y**	pastes the last characters added to the clipboard
Alt+**.** | **Esc**+**.**	inserts the last word of the previous command at the cursor's current position
Alt+**u**	converts the characters from the cursor to the end of the current word to upper case
Alt+**l**	converts the characters from the cursor to the end of the current word to lower case
Ctrl+**_**	undoes the change before the last key press