Files & Directories

FILES & DIRECTORIES	
pwd	prints the path to the current (working) directory
cd | **cd ~** | **cd $HOME**	changes the current directory to the logged-in user's home directory
cd <directory>	changes the current directory to the specified directory; the path is either absolute (beginning with "/") or relative (specified by the directory name – if it is a subdirectory of the working directory, or by ".." or "-", which represents a parent or previous directory)
tree [<directory ...>]	lists the contents of the current or specified directory in a tree-like format, **-a** including hidden files, **-d** directories only, **-L** <n> up to the specified maximum directory tree level, **-p** file permissions, **-u** owner of the file, **-g** group of the file, **-s** file size in B `$ tree -a | tail -1` (prints the total number of directories and files in the working directory)
ls | **dir** [<file ... | directory ...>]	prints a file name or the contents of a directory in alphabetical order, **-a** including hidden files, **-d** directory itself without its contents, **-i** i-node number, **-l** detailed output (file type ("-" = file, "d" = directory, "l" = symbolic link, "p" = named pipe, "b" = block device, "c" = character device), permissions, number of hard links, owner, group, size in B, time of last change, name; the "+" character at the end of the permissions means that ACL permissions are set, "." indicates that special attributes are set), **-n** equivalent to the "-l" option, but prints numeric user and group IDs, **-h** with the "-l" option prints sizes in human readable format, **-r** reverse order, **-c** sorts by last change of file attributes (permissions, number of hard links, owner, group, size in B, time of last change, name), **-t** sorts by last change of file contents, **-u** sorts by last access time, **-A** does not list implied "." and "..", **-F** indicates an item type ("*" = executable file, "@" = symbolic link, "/" = directory, "=" = socket, "|" = named pipe), **-R** recursively, **-S** sorts by size, **-U** does not sort alphabetically, **-X** sorts by item extension, **-Z** prints the SELinux security context, **-1** prints one file per line; with no argument the contents of the working directory are displayed `$ ls -latr` (lists the contents of the working directory with the most recently changed files and directories at the end of the listing)
stat <file ... | directory ...>	prints detailed file attributes (name, size in B, number of blocks, file type, i-node number, permissions, owner, group, SELinux security context, time of last access, time of last change of file attributes and time of last change of file contents), **-c** <format> displays the output in the specified format (**%a** octal permissions, **%A** symbolic permissions, **%b** number of blocks, **%F** file type, **%g** group ID, **%G** group name, **%i** i-node number, **%n** file name, **%s** size in B, **%u** owner ID, **%U** owner name, **%x** time of last access, **%y** time of last change of file attributes, **%z** time of last change of file contents), **-f** displays file system information `$ stat -c "%U %G" file` (prints the owner and group of the file)
du [<file ... | directory ...>]	displays the amount of disk space in kB occupied by the specified file or directory, **-a** displays the size of all files and directories recursively, **-b** displays the size in B, **-m** displays the size in MB, **-h** displays the size in human readable format, **-s** total summary without details; with no argument the size of the working directory including all subdirectories is displayed `# du -sh /home/*` (prints the total size of home directories) `# du -am /var/log | sort -rn | head -50` (displays 50 biggest files and directories in the specified directory)
touch [<file ... | directory ...>]	creates an empty file; if it is an existing file or directory, it changes time of last access and time of last change of file attributes to the current time, **-a** changes only time of last access, **-m** changes only time of last change of file attributes, **-d** "<YYYY-MM-DD hh:mm:ss>" sets the specified time instead of the current time, **-c** does not create any file
mkdir <directory ...>	creates a directory, **-m** <permissions> sets access permissions, **-p** creates parent directories (if they do not already exist) `$ mkdir -p /dir1/dir2/dir3` (creates directories in the specified path)

FILES & DIRECTORIES	
mktemp	creates a temporary file with a unique name in the /tmp directory (especially useful in scripts where temporary data is required during processing), **-d** creates a temporary directory, **-p** <directory> specifies a target directory instead of "/tmp" `$ datasets_temp=$(mktemp)`
mknod <file> <type> [<major> <minor>]	creates a special file of the specified type (**b** for a block device, **c** for a character device, **p** for a pipe); major and minor numbers must be specified for a block or character device, **-m** <permissions> sets access permissions `# mknod /dev/sdb b 8 16` (creates a special file for a block device "/dev/sdb" with major number 8 and minor number 16)
mkfifo <file ...>	creates a named pipe (FIFO), **-m** <permissions> sets access permissions
ln <source> [<target>]	creates a hard link to the file specified by the first argument; if no target is specified, it creates a link in the working directory to the source with the same name, **-s** symbolic link to a file or directory
rm <file ... | directory ...>	removes a file, **-d** empty directory, **-R** directory with its contents, **-i** asks for confirmation, **-f** without confirmation and ignores nonexistent files or directories, **-v** detailed output
rmdir <directory ...>	removes an empty directory, **-p** including parent directories
cp <source> <target>	copies a source file to an existing target directory; if there are two files, and the second one does not exist, it will be created, if it exists, the contents of the second file will be automatically overwritten, **-i** asks for confirmation, **-f** without confirmation, **-b** creates a backup of the target files that might be possibly overwritten, **-R** copies a directory with its contents (if the target directory does not exist, it creates it and copies only the contents of the source directory into it), **-p** preserves the source attributes, **-u** copies only when the source is newer than the target or the target is missing, **-v** detailed output, **-Z** sets the SELinux security context for the target file according to the default policy for that location `$ cp file1 ./file2` (copies a file under a different name to the same directory) `# cp -pR /home/tom/{*,.[^.]*} /mnt/extdisk/home/tom` (copies the contents of user "tom" home directory to an external disk, including hidden files)
install <source> <target>	copies a source file to an existing target directory; if there are two files, and the second one does not exist, it will be created, if it exists, the contents of the second file will be automatically overwritten, **-b** creates a backup of the target files that might be possibly overwritten, **-m** <permissions> sets access permissions, **-o** <user> sets the user ownership, **-g** <group> sets the group ownership, **-p** preserves the source attributes, **-d** <directory> creates a directory (including parent directories), **-v** detailed output, **-Z** sets the SELinux security context for the target file according to the default policy for that location `# install -m 0755 /bin/chmod.broken /bin/chmod` (copies the contents of one file to another file and sets its access permissions)

FILES & DIRECTORIES	
rsync [[<user>@]<host>:]<source ...> [[<user>@]<host>:]<target>	copies a source file to an existing target directory; if there are two files, and the second one does not exist, it will be created, if it exists, the contents of the second file will be automatically overwritten; if the target already contains some data identical to the source, only their difference is copied; data can also be transferred between remote computers, **-r** copies a directory with its contents (if the target directory does not exist, it creates it), **-l** preserves symbolic links, **-p** preserves the permissions of the file, **-o** preserves the owner of the file, **-g** preserves the group of the file, **-t** preserves time of last change of file contents, **-D** preserves special files, **-a** includes options "-rlptgoD", **-z** compresses the data during the transfer, **--delete** removes redundant data in the target directory that no longer exists in the source, **--ignore-errors** removes redundant data even in case of I/O errors, **--exclude=**<pattern> excludes files matching the specified pattern, **--exclude-from=**<file> excludes files matching the patterns contained in the specified file, **--progress** prints the progress of synchronization, **--rsh** <command> specifies an alternative program for the data transfer, **-v** detailed output `# rsync -a --delete --exclude ".gvfs" /home /mnt/extdisk` (synchronizes data on the external disk with data in the home directory) `# rsync -avz /var/log/ 192.168.0.20:/logs` (copies compressed logs to a remote computer) `$ rsync --rsh="ssh -l root -p 22022" file1 file2 192.168.152.47:/tmp` (copies files to a remote computer using ssh)
dd <parameter>**=**<value>	copies data between devices, **if=**<source>, **of=**<target>, **bs=**<block_size_in_bytes> (512 B by default), **count=**<number_of_blocks> `$ dd if=/dev/cdrom bs=2048 of=/tmp/image.iso` (copies the CD image to a file) `$ dd if=/dev/zero of=test bs=5M count=10` (creates an empty file of 50 MB size) `$ for file in {1..50}; do dd if=/dev/zero of=$file bs=1M count=1; done` (creates 50 empty files of 1 MB size) `# dd if=/dev/urandom of=/dev/sdb1 bs=1G` (overwrites the partition with random data)
truncate [<file ...>]	sets, shrinks or extends the size of a file, **-s** specifies the size in B or in units – "K", "M", "G", "T" (powers of 1024) or "KB", "MB", "GB", "TB" (powers of 1000), the "+" character extends the size, "-" shrinks the size, "<" sets the maximum size, ">" sets the minimum size, "/" rounds down to multiple of, "%" rounds up to multiple of; if the file does not exist, it will be created `$ truncate -s 1M data1` (creates an empty file of 1 MB size) `$ truncate -s "<5M" *` (sets the maximum size of all files in the working directory to 5 MB)
cdrecord <track ...>	writes data to CD/DVD media, **-scanbus** finds the SCSI address of the CD/DVD recorder on the system, **dev=**<target> specifies the SCSI address of the CD/DVD recorder, **speed=**<n> specifies the speed factor of the writing process, **-v** detailed output `$ cdrecord -v dev="6,0,0" speed=8 cd.iso` (writes the contents of the "cd.iso" file to a CD/DVD)
mv <source> <target>	moves a source file or directory to an existing target directory; if there are two files or two directories, and the second one does not exist, the first one will be renamed to the second one, if the second file exists, its contents will be automatically overwritten, **-i** asks for confirmation, **-f** without confirmation, **-b** creates a backup of the target files that might be possibly overwritten, **-v** detailed output, **-Z** sets the SELinux security context for the target file according to the default policy for that location
rename <source> <target> <file | directory>	renames a specified file or directory from the source name to the target name according to the specified patterns `$ rename .htm .html *.htm` (renames files with the extension "htm" to "html") `$ find / -name "jack*" | xargs rename jack john` (renames files and directories named "jack*" to "john*")
basename <file | directory>	prints the file or directory name whose path was passed as an argument

FILES & DIRECTORIES	
dirname <file ... | directory ...>	prints the path to the directory that contains the file or directory whose path was passed as an argument `# read a; while [[ $a != "/" ]]; do ls -ld $a; a=$(dirname $a); done` (prints permissions to the directories in the path specified as an argument)
namei <file ... | directory ...>	follows the specified path, resolving symbolic links and reporting the file type of each component encountered ("-" = file, "d" = directory, "l" = symbolic link, "p" = named pipe, "b" = block device, "c" = character device), **-o** prints the owner and group, **-m** prints permissions, **-l** prints permissions, including the owner and group

FILES & DIRECTORIES	
find [<path ...>] [<option>] [<test>] [<action>]	searches for a file or directory in the specified path, if the path is not specified, searches in the working directory; using the option **-maxdepth** <level> searches up to the specified maximum directory tree level, **-mindepth** <level> searches minimally from the specified directory tree level; for testing is used e.g., **-name** <pattern> according to the pattern, **-iname** <pattern> according to the pattern (the match is not case sensitive), **-regex** <pattern> according to the name which matches a specified regular expression, **-iregex** <pattern> according to the name which matches a specified regular expression (the match is not case sensitive), **-inum** <-n | n | +n> according to the i-node, **-atime** / **-amin** <-n | n | +n> according to the last access in the order of days/minutes, **-ctime** / **-cmin** <-n | n | +n> according to the attributes change in the order of days/minutes, **-mtime** / **-mmin** <-n | n | +n> according to the contents change in the order of days/minutes, **-size** <-n | n | +n><unit> according to the size ("c" = B, "k" = kB, "M" = MB, "G" = GB), **-empty** empty file or directory, **-type** according to the type ("d" = directory, "f" = file, "l" = symbolic link, "b" = block device, "c" = character device), **-context** <context> according to the SELinux security context, **-user** <user | UID> according to the owner, **-group** <group | GID> according to the group, **-perm** <permissions> exactly according to the specified permissions, **-perm -**<permissions> according to the set permissions whose all bits include at least the specified permissions bits, **-perm /**<permissions> according to the set permissions whose any bit also includes the specified permissions bit, **-nouser** searches for files of non-existent users, **-nogroup** searches for files of non-existent groups; for the action is used e.g., **-exec** <command> [<argument ...>] **{}** {**\;** | **\+**} executes a command (the string "{}" is replaced by the path to the file found, ";" indicates that the specified command is run individually for each matched file, while "+" processes multiple file names at once as arguments to the command; all of these constructions should be quoted or escaped to protect them from expansion by the shell), **-ok** <command> [<argument ...>] **{} \;** interactive version of the "-exec" option (asks for confirmation), **-print** prints the result (by default), **-print0** prints also file names containing a white space; extended testing is provided by operators **-a** = "and" (both expressions are true), **-o** = "or" (at least one expression is true), **!** = negation (expression is false); with no argument the contents of the working directory are displayed recursively `# find /home -name "*.jpg" -o -name "*.png"` (searches the home directory for files with ".jpg" or ".png" extension) `# find . -type f -o -type d | wc -l` (counts all files and directories in the working directory) `# find ! -type f` (searches the working directory for everything except regular files) `# find /tmp -mmin -120` (searches for all files that have been changed during last 2 hours) `# find / -user tom -exec rm -Rf '{}' \;` (deletes all files and directories of user "tom") `# find . -maxdepth 2 -type l` (searches for symbolic links to the second level of the directory tree) `# find / -nouser -o -nogroup` (searches for files of non-existent users and groups) `# find / -perm -4000` (searches for all files with SUID bits set) `# find / -perm -444 -perm /222 ! -perm /111` (searches for files that are readable for everybody, have at least one write bit set, but are not executable for anybody) `# find / -context "*tmp_t*"` (searches for files according to the specified SELinux security context) `# find /var/log -type f -size +100M -exec du -h '{}' \+ | sort -rh | head -10` (searches for 10 files bigger than 100 MB and sorts them by size in descending order) `# find /etc/rc.d -type f \( -perm -g=w -o -perm -o=w \)` (searches for writable files for group or others excluding symbolic links) `# find / -type d \( -perm -o=w -a ! -perm -o=t \) -exec ls -ld '{}' \;` (searches for writable directories for others without a sticky bit set) `# find / -type f -regex ".*\.\(sh|pl|py\)" -perm -o=x` (searches for executable scripts for others)

FILES & DIRECTORIES	
which <command ...>	displays the absolute path of the specified command or script (or its alias) in $PATH, **-a** prints all occurrences in $PATH
whereis <command ...>	displays the absolute path to a binary, source and manual file of the specified command or script in $PATH
type <command ...>	prints whether a command is a program or a shell builtin or keyword, the absolute path to the binary file is printed for a program, **-a** shows all the occurrences including aliases and functions
apropos | **man -k** <keyword ...>	searches for commands containing the specified string in the name or description
whatis | **man -f** <keyword ...>	searches for commands whose name matches the specified string
man | **info** <command>	prints a manual page for the specified command, **q** quits the program
help [<pattern>]	prints help for shell builtins and keywords matching the pattern; with no argument all shell builtins and keywords are displayed
ldd <program ...>	prints shared libraries required by the specified program
lsof [<file ...>]	prints information about the specified open file – e.g., its name, the program that uses the file, PID, username, file descriptor, file type, major and minor number of the device, file size in B and the i-node number, **+D** <directory> prints open files in the specified directory recursively, **-l** prints UID, **-p** <PID> files opened by the specified process, **-u** <user> files opened by processes owned by the specified user, **-i** lists network files, **-n** does not resolve hostnames via DNS, **-P** does not convert port numbers to port names, **-r** <n> refreshes the list every *n* seconds; with no argument all open files belonging to all active processes are displayed; an open file may be a regular file, directory, block special file, character special file, library or network file (Internet socket, NFS file or UNIX domain socket) `$ lsof /dev/cdrom` (prints detailed information about processes using the CD/DVD drive) `$ lsof /dev/snd/*` (prints detailed information about processes using the sound card) `# lsof +D /opt/app | awk 'NR > 1 {print $9}' > /tmp/open_files.txt` (saves a list of open files in the "/opt/app" directory to "/tmp/open_files.txt") `$ lsof -p $$` (prints open files from the current shell) `# lsof -i -n -P` (prints all open ports including protocol types, IP addresses of the hosts, users and processes using a particular port)

The Contents of a File	
file <file ...>	detects the data format type of a regular file (ASCII, PDF, HTML etc.), **-s** reads block or character special files, **-z** reads the contents of a compressed file `$ file * | grep ASCII` (lists files in the working directory containing plain text)
cat [<file ...>]	prints the contents of a text file, **-n** numbers all lines, **-b** numbers lines containing text, **-s** suppresses repeating blank lines; if no file is specified, reads from STDIN `$ cat a b c > abc` (writes the contents of the three files to the file "abc")
tac [<file ...>]	prints the contents of a text file in reverse order; if no file is specified, reads from STDIN
more <file ...>	prints the contents of a larger text file one screenful at a time, **space** moves the page forwards, **b** moves the page backwards, **/**<string> searches forwards for the specified string, **n** continues in further searching, **q** quits the program
less <file ...>	prints the contents of a larger text file one screenful at a time, **space** / **page down** moves the page forwards, **b** / **page up** moves the page backwards, **<** to the beginning of the file, **>** to the end of the file, /<string> searches forwards for the specified string, **n**/**N** continues in further searching / searching in the reverse direction, **-N** numbers lines, **:n** prints the contents of the next file, **:p** prints the contents of the previous file, **q** quits the program
head [<file ...>]	[**-n** [-]<n>] prints the first *n* lines of a file (10 by default), with the leading '-', prints all but the last *n* lines; if no file is specified, reads from STDIN
tail [<file ...>]	[**-n** [+]<n>] prints the last *n* lines of a file (10 by default), with the leading '+', prints lines starting with line *n*, **-f** updates the output with newly appended data; if no file is specified, reads from STDIN `$ head -25 access.log | tail -1` (prints the 25th line of the file)
wc [<file ...>]	prints the number of lines, words and bytes in a text file, including white spaces, **-l** counts lines, **-w** words, **-m** characters, **-c** bytes; if no file is specified, reads from STDIN

The Contents of a File	
nl [<file ...>]	numbers lines of a file, **-v** <n> sets a number that starts the numbering (1 by default), **-i** <n> sets an increment by which the number of the following line is increased (1 by default), **-b** <option> numbers lines corresponding to the specified option (**a** all lines, **t** non-empty lines (by default), **n** no lines, **p**<expression> lines containing a specified regular expression), **-s** <string> sets a separator between the number of the line and its contents (tab by default), **-w** <n> sets the number of characters (spaces) to align particular figures to the right (6 by default); if no file is specified, reads from STDIN `$ nl style.css | more` (numbers the lines of the file the contents of which are displayed one screenful at a time) `$ nl -s ": " -b p^# script.sh` (numbers the lines of the file starting with the "#" character)
sort [<file ...>]	sorts the lines of a text file in ascending order (alphabetically), **-n** by numeric values, **-h** by numeric values specified in human readable format, **-k** <n> by column, **-u** ignores duplicate lines, **-r** in reverse order; if no file is specified, reads from STDIN `$ sort file.txt | uniq` (sorts and prints unique lines of the file)
uniq [<file>]	prints the contents of a text file, discarding all but one of adjacent identical lines, **-d** prints duplicate adjacent lines (to list all duplicate lines the contents of the file need to be sorted), **-u** unique lines, **-c** prefixes each line by the number of occurrences of identical adjacent lines, **-i** ignores case distinctions; if no file is specified, reads from STDIN
grep <pattern> [<file ... | directory ...>]	searches a text file for strings containing the specified pattern using basic regular expressions and prints the matching line, **-v** prints lines that do not contain the specified pattern, **-n** prints the line number, **-c** prints the number of lines containing the specified pattern, **-l** prints the names of files containing the specified pattern, **-L** prints the names of files that do not contain the specified pattern, **-h** suppresses file names printing on output, **-i** ignores case distinctions, **-f** <file> reads patterns from the specified file (one per line), **-o** prints only the part of the line matching the pattern, **-A** <n> prints *n* more lines after matching lines, **-B** <n> prints *n* more lines before matching lines, **-C** <n> prints *n* more lines before and after matching lines, **-E** interprets patterns as extended regular expressions (equivalent to the "egrep" command), **-F** interprets patterns as fixed strings (equivalent to the "fgrep" command), **-R** recursively, **-w** finds the whole words that match the pattern, **--color** displays the matched string in color; if no file is specified, reads from STDIN `# grep -i linux /etc/httpd/logs/access_log` (prints the lines of the file containing the pattern "linux" in a combination of lowercase and uppercase letters) `$ ls -l | grep '^d'` (lists subdirectories in the working directory) `$ ps -ef | grep -w "httpd"` (lists web server processes) `$ grep '^P.*r$' notes` (prints the lines of the file starting with "P" and ending with "r") `$ grep -Ei 'red hat|centos|fedora' /etc/*release` (prints whether the system matches one of the specified Linux distributions) `$ grep -R "header" /var/www/html` (prints all lines of all files in the specified directory containing the pattern "header")
egrep | **grep -E** <pattern> [<file ... | directory ...>]	searches a text file for strings containing the specified pattern using extended regular expressions and prints the matching line; if no file is specified, reads from STDIN `# egrep -v '^[#;]|^$' *conf*` (prints all lines of the configuration files that are not commented or empty) `$ egrep '^[^#]*authpriv\.\*' /etc/rsyslog.conf` (prints the lines of the file starting with the pattern "authpriv.*") `$ egrep -o '^[^:]+' /etc/passwd` (displays all users on the system)
fgrep | **grep -F** <pattern> [<file ... | directory ...>]	searches a text file for strings containing the specified pattern without using regular expressions and prints the matching line; if no file is specified, reads from STDIN `$ fgrep '/*' style.css` (prints the lines of the file containing a comment)

The Contents of a File	
sed [<address>] <command> [<file ...>]	filters and edits text in a non-interactive way (often used in scripts), **-E** allows the use of extended regular expressions, **-f** <file> executes a script included in the specified file, **-i** saves the changes directly in the file (otherwise changes need to be saved by using redirections), **-n** does not print the output on STDOUT; the address represents either a sequential number from the beginning of the input (e.g., 1,10 = range of lines), "$" character (last line of the input) or a pattern (regular expression) bounded from the both sides by a separator – normally by "/"; command **s** replaces the pattern by the following expression bounded by separators (only the first matching pattern), *n* exactly *n*-th pattern occurence, **g** replaces the pattern globally, in each occurence, **a** appends an expression below the line with the specified pattern, <n>**p** prints a particular line of the file (with the "-n" option), **d** removes the whole line containing the specified expression, **!**<command> executes a command for all addresses except for those specified; if no file is specified, reads from STDIN `$ echo "11 x, 22 x, 33 x" | sed 's/x/y/2'` (replaces only the second occurence of "x" pattern by "y") `$ sed -i 's/string1/string2/g' file` (replaces the first string with the second one within the whole file) `# sed -i '/shared/s/ro/rw/' /etc/fstab` (replaces "ro" string with "rw" string on the line with the first occurrence of the word "shared") `$ sed 's/ \+/\t/g' test1 > test2` (replaces multiple spaces by one tab) `$ echo "Log retention:" $(sed -n '3p;6p' /etc/logrotate.conf)` (prints the 3rd and 6th line of the file) `$ sed '/dev/!d' file` (prints all lines containing a "dev" expression) `$ sed -i '1,10d' file` (removes the first 10 lines of the file) `$ sed -i '5,$d' file` (removes all from the 5th line to the end of the file) `$ sed '1,/START/d' file` (removes all up to and including the word "START") `$ sed '1,/STOP/!d' file` (removes all from the word "STOP") `$ sed -n 'p;n' file` (prints odd lines of the file) `$ sed -n 'n;p' file` (prints even lines of the file) `$ sed '/^#/d; /^$/d' file1 > file2 | $ sed -E '/^(#|$)/d' file1 > file2` (removes all comments and empty lines in the file) `# sed -i '$s/^/from="'$(awk -F "|" '{print $9 "," $11}' hosts.csv)'" /' .ssh/authorized_keys` (inserts the command output before the text on the first line) `# sed -i '$s/$/ !!Ansible user!!/' .ssh/authorized_keys` (inserts a string after the text on the last line) `# sed -i '/auth[[:blank:]]\+required[[:blank:]]\+pam_env.so/a auth\trequired\tpam_tally2.so deny=5' /etc/pam.d/system-auth` (appends an expression below the line with the specified pattern) `$ sed '/^[^#]*\(sha[25]|md5\)/!d' /etc/pam.d/system-auth /etc/pam.d/common-password 2> /dev/null | egrep -o "sha[1256]{3}|md5" || sed '/^[^#]*ENCRYPT_METHOD/!d' /etc/login.defs` (displays encryption algorithm for newly created passwords)
awk '<program>' [<file ...>]	filters text in a non-interactive way (often used in scripts); the program is a set of rules containing a pattern, action, or both, the action is enclosed in "{}"; **-f** <file> reads a program from the specified file, **-F** <string> specifies the input item separator, **-v** <variable>=<value> sets a specified variable within the program; built-in variable **NR** represents the number of records (lines) separated by a new line, **NF** represents the number of fields in the record separated by a separator (white space by default), individual fields are represented by $1, $2 etc., $0 represents the whole record, **RS** defines records separator, **FS** defines fields separator, **ORS** defines output records separator, **OFS** defines output fields separator; operators used to compare strings include **==** is equal to, **!=** is not equal to, **<** less than, **>** more than, **<=** less than or equal to, **>=** greater than or equal to; regular expressions are enclosed between "//", for comparison use **~** (the pattern matches the expression), **!~** (the pattern does not match the expression); **\|\|** logical OR, **&&** logical AND; if no file is specified, reads from STDIN `$ awk '/nameserver/ {print}' /etc/resolv.conf` (prints all lines containing "nameserver") `$ awk -F ":" '{print $1, $7}' /etc/passwd` (prints all users and their login shells) `$ awk '{print $1 > "file1"; print $2 > "file2"}' test` (copies the first and second word of the file to the specified files) `$ awk 'length > 10' list.txt` (prints lines longer than 10 characters) `$ awk 'NF < 5' list.txt` (prints lines shorter than 5 words) `$ awk 'NR == 5' error.log` (prints the 5th line of the file) `$ awk 'NR == 5, NR == 10' album` (prints the 5th up to the 10th line of the file) `$ awk '{print NR,"->",NF}' list.txt` (prints the number of fields for each record of the file) `$ ls -l | grep "^-" | awk '{sum += $5} END {print sum/1024/1024}'` (prints the size of all files in the working directory in MB) `# ne=$(awk -F ":" '{if (length($2) > 2 && ($5 == "" || $5 >= 99999)) print $1}' /etc/shadow); [[ -z "$ne" ]] && echo "NONE" || echo "$ne" \| tr " " "\n"` (prints users with never expiring passwords) `$ sed '/^[^:]/!d' /etc/inittab | awk -F "#" '{print $1}' | awk -F ":" '{if (($4 !~ /^\// && $4 !~ /^$/) || ($4 ~ /[|]+.?[^/]/ && $4 !~ /^$/)) print $4}'` (finds commands in the file that are not specified in the absolute path)

The Contents of a File	
tr <string1> [<string2>]	replaces individual characters from the first string with new ones from the other string (does not overwrite the original file), **-d** removes characters, **-s** replaces repeated characters with a single occurrence `$ echo "FIBRE CHANNEL STATISTICS REPORT" | tr 'A-Z' 'a-z'` (converts uppercase to lowercase) `$ cat file1 | tr 'a-z' 'A-Z' > file2` (converts lowercase to uppercase in the file and redirects the content to a new file) `$ cat text1 | tr ' ' '\t' > text2` (replaces each space in the file with a tab) `$ awk -F ":" '{print $1}' /etc/passwd \| tr '\n' ' '` (replaces the new line with a space) `$ tr -d ';' < text1.txt > text2.txt` (removes the specified character in the file and redirects the content to a new file)
rev [<file ...>]	reverses the order of characters on each line of a file (does not overwrite the original file); if no file is specified, reads from STDIN `$ echo madam | rev`
tee [<file ...>]	reads from STDIN and writes to STDOUT and optionally to a specified file, **-a** appends the command output to the end of the specified file; if the file does not exist, it will be created `$ ls | tee /tmp/test | wc -l` (the output of the "ls" command is saved to the "/tmp/test" file, only the number of lines is displayed on STDOUT) `$ who | sort | tee -a log1 log2` (the sorted output of the "who" command is written both to STDOUT and to files "log1" and "log2")
cat > <file>	inserts the contents of STDIN into a file `$ cat > /dev/pts/2 <-'` `<text> <-'` **Ctrl**+**d** (redirects the particular text to the specified terminal)
> | **cat /dev/null >** <file>	removes the contents of a file

The Contents of a File	
vi | **vim** <file>	starts a "vi"/"vim" text editor and creates or opens a file
v	starts/quits visual character mode
Shift+**v**	starts/quits visual line mode
Ctrl+**v**	starts/quits visual block mode
Esc	switches from the insert mode to the command mode
:<n>	moves the cursor at the beginning of a specified line
:q	quits the editor (assuming the file has not been changed)
:q!	quits the editor without saving any changes in the file
:w / **:w** <file>	saves the changes in the file / saves the file as a new one
:wq / **:x** / **ZZ**	saves the changes and quits the editor
:w! / **:wq!**	saves the changes / saves the changes and quits the editor for a read-only file
:set number / **nonumber**	numbers the lines / removes line numbering
:set ignorecase / **noignorecase**	ignores/respects case sensitivity
:set all	prints option settings
moving the cursor:	
left:	
h / **Left Arrow**	one character to the left
b	at the beginning of the previous word
0	at the beginning of the line
right:	
l / **Right Arrow**	one character to the right
w	at the beginning of the next word
$	at the end of the line
up:	
k / **Up Arrow**	on the previous line
H	on the first line of the screen
1G	on the first line of the file
down:	
j / **Down Arrow**	on the next line
L	on the last line of the screen
G	on the last line of the file
adding text:	
i	at the current cursor position
a	after the current cursor position
I	at the beginning of the line
A	at the end of the line
o	at the beginning of a new line bellow the current line
O	at the beginning of a new line above the current line
removing / **extracting text:**	
x	one character in the current cursor position
db	characters before the current cursor position to the beginning of the word
dw	characters from the current cursor position to the end of the word
d0	characters before the current cursor position to the beginning of the line
d$	characters from the current cursor position to the end of the line
dd / <n>**dd**	the whole line in the current cursor position / next *n* lines
d1G	the whole line in the current cursor position to the beginning of the file
dG	the whole line in the current cursor position to the end of the file
:1,$d	the contents of the whole file
modifying text:	
yw / <n>**yw**	copies characters from the current cursor position to the end of the word / next *n* words
yy / <n>**yy**	copies the current line / next *n* lines
P / **p**	pastes the extracted or copied text before/after the cursor or above/below the current line (in case of extract/copy of whole lines)
r	replaces a character in the current cursor position
R	replaces characters in the current cursor positions
:[<address>]**s/**<searched_string>**/**<replacing_string>(**/**<options>)	replaces the searched expression with a new one; address **1,$** specifies the range of the whole document, if the address is not specified, the change affects the current line only, option **g** replaces all occurrences of the searched string, option **c** requires confirmation to perform the operation `:1,$s/hundred/100/g` (replaces all occurrences of the word "hundred" with the number "100" in the whole file)
u	undoes the last change made
U	undoes all changes made on the current line
.	repeats the last command
searching in text:	
/<string>	searches the file forwards for the specified string
?<string>	searches the file backwards for the specified string
n / **N**	continues in further searching / searching in the reverse direction
column [<file ...>]	formats the lines of a file into columns, **-s** <separator> specifies the input item delimiter (whitespace by default), **-t** creates a table; if no file is specified, reads from STDIN `$ mount | column -t` (formats the command output into a table)
cut [<file ...>]	**-c** <m-n> cuts a specified range of characters on each line of the file (from the beginning of the line), **-f** <m-n> cuts a specified range of columns (fields) separated by a tab, **-d** <character> defines a field separator instead of a tab; list items are separated by a comma; if no file is specified, reads from STDIN `$ cut -f 2,4 -d ' ' data` (cuts columns 2 and 4 from the file and separates them by a space) `# cut -d : -f 1 < /etc/passwd | sort` (lists all users on the system in alphabetical order)
paste [<file1> <file2>]	joins the contents of files one next to the other (1 column = 1 file), **-d** <string> specifies a field separator instead of a tab, **-s** transposition of lines and columns (1 line = 1 file); if no file is specified, reads from STDIN `$ paste -s file1 file2 > file3` (joins the contents of two files into one – 1 line = 1 file)
split [<file>]	splits a file into pieces of the same number of lines (1000 by default), **-l** <n> specifies the number of lines, **-b** <n> splits a file by the specified number of bytes; if no file is specified, reads from STDIN

The Contents of a File	
join <file1> [<file2>]	joins lines of two sorted text files according to the first identical field delimited by whitespace, **-i** ignores case distinctions; if no file is specified, reads from STDIN
diff <file1> <file2> | <dir1> <dir2>	compares the contents of two text files or directories and prints differences, **-i** ignores case distinctions, **-q** reports when the files differ, **-s** reports when the files are identical, **-r** recursively, **-u** prints the output in a unified format `$ diff <(echo "Welcome to prompt.cz") <(cat /etc/motd)` (compares the output of both commands)
cmp <file1> <file2>	compares the contents of two files of any type and prints the first different byte and line number, **-l** prints all differences byte by byte
sum [<file ...>]	prints checksum and block counts of a file; if no file is specified, reads from STDIN
cksum [<file ...>]	prints CRC checksum and byte counts of a file; if no file is specified, reads from STDIN
sha1sum | **sha512sum** | **md5sum** [<file ...>]	prints or checks SHA1 (160-bit), SHA512 (512-bit) or MD5 (128-bit) checksum of a file, **-c** <file> compares the checksum stored in the file with the original file; if no file is specified, reads from STDIN `$ sha512sum ./F-7-x86_64-DVD.iso` (prints the checksum of the file) `$ md5sum page.txt > page.md5; md5sum -c page.md5` (saves the checksum of the "page.txt" file to "page.md5" and then compares it with the original file)
ksvalidator <file ...>	verifies the syntax of a kickstart file (automatically created by the Anaconda installer and saved as */root/anaconda-ks.cfg*)
enca <file ...>	detects the encoding type of a text file, **-L** <language> specifies the language of an input file `$ enca -L cs input.srt` (detects the encoding type of the text file if the program did not automatically recognize the language used)
convmv <file ... | directory ...>	converts the encoding type of a file or directory name, **-f** <source_code> specifies an original code, **-t** <target_code> specifies a new code, **-i** interactive mode, **-r** recursively, **--list** lists all available encodings, **--notest** performs the operation `$ convmv --notest -f cp1250 -t UTF-8 *` (converts the encoding type of the working directory content names from "cp1250" to "UTF-8")
iconv [<file ...>]	converts the encoding type of a text file content, **-f** <source_code> specifies an original code, **-t** <target_code> specifies a new code, **-l** lists all available encodings; if no file is specified, recodes STDIN to STDOUT `$ iconv -f WINDOWS-1250 -t UTF-8 < input.srt > output.srt` (converts the encoding type of a text file content from "WINDOWS-1250" to "UTF-8")
recode [<source_code>]**..**<target_code> [<file ...>]	converts the encoding type of a text file content, **-l** prints known types of encoding, **-f** suppresses error output, **-v** detailed output; if no file is specified, recodes STDIN to STDOUT `$ echo 'Výpis jmen všech souborů' | recode -f UTF-8..flat` (prints the text without diacritics) `$ recode ..HTML < page.txt > page.html` (converts the encoding type of the "page.txt" file content to "html" and writes it to "page.html") `$ find . -name "*.txt" -exec recode cp1250..UTF-8 '{}' \;` (converts the encoding type of text files contents from "cp1250" to "UTF-8")
od [<file ...>]	[**-o**] prints the contents of a binary file in octal format, **-d** in decimal format, **-x** in hexadecimal format, **-j** <byte> prints data from the specified byte (offset), **-w**<number_of_bytes> displays the specified number of bytes on one line; the first column of the output represents an offset – the sequence of the first byte of the particular line from the beginning of the file; if no file is specified, reads from STDIN `$ od -w8 /usr/bin/who` `$ od -j 0474360 /usr/bin/find`
strings <file ...>	prints text strings contained especially inside a binary file with a minimum length of four characters `# strings $(which vsftpd) | egrep "libwrap|hosts"` (checks if the program is related to tcp wrappers)
msgfmt <file>.po	converts a text file to a binary format, **-o** <file>.mo specifies the output file `$ msgfmt -o woocommerce-cs_CZ.mo woocommerce-cs_CZ.po`
convert <source> <target>	converts the file format, possibly also its size `$ convert image.tif image.jpg` (converts "tif" file format to "jpg") `$ IFS=','; cmds="hostname -s,date,ifconfig -a"; for cmd in $cmds; do { echo "[$USER@${HOSTNAME} ${PWD/#$HOME/~}]" "${cmd}";}; eval "${cmd}"; done > info.txt && convert -background black -fill white info.txt info.png` (creates a printscreen of the terminal with the specified commands)
lame <file>.wav <file>.mp3	converts audio format from "wav" to "mp3" or vice versa, **-b** <bitrate> specifies the minimum bitrate, **-v** variable bitrate, **--verbose** detailed output
aplay <file ...>	plays a specified audio file
arecord <file>.wav	creates a "wav" audio file

Archiving & Compression	
tar [-f <archive>] [<file ... | directory ...>]	archives or restores data (the source is preserved), **-c** creates an archive, **-f** <file> specifies an archive name, **-v** detailed output, **--remove-files** removes the source, **--acls** preserves ACL permissions, **--selinux** preserves SELinux settings, **-M** multi-volume archive (data requires more than one medium), **-V** <name> specifies a volume name, **-X** <file> excludes files matching patterns listed in the specified file, **-d** compares data in an archive with data on the disk, **-t** prints the contents of an archive (even compressed), **-r** appends files to the end of an archive, **-x** extracts files from an archive, **-C** changes to a directory (specifies the target directory for extracting the archive), **-p** preserves the original ownership and permissions during the restore, **-Z** (de)compresses data with the "compress" tool, **-z** (de)compresses data with the "gzip" tool, **-j** (de)compresses data with the "bzip2" tool, **-J** (de)compresses data with the "xz" tool, **-w** interactive mode `# tar -cvf ssh.tar /etc/ssh` (creates an archive from the specified directory) `# tar -cvf scripts.tar backup.sh restore.sh` (creates an archive from the specified files) `# tar -cMf /dev/st0 / -V system_bckp` (creates a system backup to the tape) `# tar -tvf ssh.tar` (prints the contents of the archive) `# tar -xvf ssh.tar` (extracts files from the archive) for archiving and (de)compression by the "compress" tool: `# tar -cvZf ssh.tar.Z /etc/ssh` `# tar -xvZf ssh.tar.Z` for archiving and (de)compression by the "gzip" tool: `# tar -cvzf ssh.tar.gz /etc/ssh` `# tar -xvzf ssh.tar.gz` for archiving and (de)compression by the "bzip2" tool: `# tar -cvjf ssh.tar.bz2 /etc/ssh` `# tar -xvjf ssh.tar.bz2` for archiving and (de)compression by the "xz" tool: `# tar -cvJf ssh.tar.xz /etc/ssh` `# tar --remove-files -cvJf maillog_old.tar.xz maillog-20161225 maillog-20170102` `# tar -xvJf ssh.tar.xz`

Archiving & Compression	
[<source> |] **cpio** > <target>	archives or restores data, **-o** creates an archive, **-d** creates a directory tree, **-i** extracts an archive, **-m** preserves previous file modification time, **-t** prints the contents of an archive, **-v** detailed output `$ ls | cpio -ov > archive.cpio` (creates an archive from the contents of the working directory) `# find / -mtime -1 -type f -print | cpio -ov > /dev/rmt0` (creates a tape archive of all files that have been changed during the last day) `$ cpio -o > /dev/fd0 <-'` `<file1> <-'` `<file2> <-'` **Ctrl**+**d** (creates an archive to a floppy disk in an interactive way) `# cpio -ivmd /\* < /dev/rmt0` (extracts the archive from the tape) `$ rpm2cpio setup-2.12.2-6.el8.noarch.rpm | cpio -tv` (lists the files in the package) `$ rpm2cpio setup-2.12.2-6.el8.noarch.rpm | cpio -idv` (extracts the files from the package)
compress <file ...>	compresses a file (a directory must first be archived into a single file with the "tar" tool; the source is removed), **-c** prints the output on STDOUT and preserves the source, **-d** decompresses a file (equivalent to the "uncompress" command), **-v** detailed output
uncompress | **compress -d** <file ...>.Z	decompresses a file (the source is removed), **-c** prints the output on STDOUT and preserves the source (equivalent to the "zcat" command), **-v** detailed output `$ zcat ssh.tar.Z | tar -x`
gzip <file ...>	compresses a file (a directory must first be archived into a single file with the "tar" tool; the source is removed), **-**<n> compression level (the value range is from 1–9, the higher the value, the higher the compression, the default value is 6), **-c** prints the output on STDOUT and preserves the source, **-d** decompresses a file (equivalent to the "gunzip" command), **-v** detailed output `# gzip -c access.log > access.log_2017-02-15.gz && > access.log` (compresses a file which is saved under a new name and removes the contents of the original file)
gunzip | **gzip -d** <file ...>.gz	decompresses a file (the source is removed), **-c** prints the output on STDOUT and preserves the source, **-l** prints an archive name, its size in B before and after the compression and the compression ratio in percent, **-t** tests the integrity of a compressed file, **-v** detailed output `$ gzip -cd ssh.tar.gz | tar -x`

Archiving & Compression	
bzip2 <file ...>	compresses a file (higher compression than "gzip"; a directory must first be archived into a single file with the "tar" tool; the source is removed), **-**<n> compression level (the value range is from 1–9, the higher the value the higher the compression, the default value is 6), **-k** preserves the source, **-d** decompresses a file (equivalent to the "bunzip2" command), **-v** detailed output
bunzip2 | **bzip2 -d** <file ...>.bz2	decompresses a file (the source is removed), **-c** prints the output on STDOUT and preserves the source, **-k** preserves the source, **-t** tests the integrity of a compressed file, **-v** detailed output `$ bzip2 -cd ssh.tar.bz2 | tar -x`
xz <file ...>	compresses a file (higher compression than "bzip2"; a directory must first be archived into a single file with the "tar" tool; the source is removed), **-**<n> compression level (the value range is from 0–9, the higher the value the higher the compression, the default value is 6), **-k** preserves the source, **-d** decompresses a file (equivalent to the "unxz" command), **-v** detailed output
unxz | **xz -d** <file ...>.xz	decompresses a file (the source is removed), **-c** prints the output on STDOUT and preserves the source, **-k** preserves the source, **-l** prints an archive name, its size before and after the compression and the compression ratio, **-t** tests the integrity of a compressed file, **-v** detailed output `$ xz -cd ssh.tar.xz | tar -x`
zip <target> <source ...>	compresses a file or directory (the source is preserved), **-r** recursively, **-e** under a password `$ zip -r ssh.zip /etc/ssh`
unzip <file>.zip	decompresses a file (the source is preserved), **-d** <directory> specifies the target directory for extracting the archive, **-l** lists an archive contents, **-t** tests the integrity of a compressed file

From:
https://prompt.cz/en/ - **Prompt.cz**

Permanent link:
https://prompt.cz/en/files-and-directories

Last update: **2025/07/08 11:38**