Devices & File Systems

DEVICES & PARTITIONS	
cat /proc/devices	lists character and block devices sorted by their major numbers

lshw	prints detailed information about the recognized hardware and its configuration, **-C** <class> displays only the specified class of hardware, **-short** displays hardware paths, devices, classes and hardware descriptions
	`# lshw -C system`
	(displays information about the system)
	`# lshw -C bus`
	(displays information about the motherboard)
	`# lshw -C processor`
	(displays information about the processor)
	`# lshw -C disk`
	(displays information about disks)
	`# lshw -C memory`
	(displays information about physical memory)
	`# lshw -C network`
	(displays information about network interfaces)
	`# lshw -C display`
	(displays information about the graphics card)
	`# lshw -C multimedia`
	(displays information about the sound card)

lspci	prints information about PCI buses and devices connected to them, **-k** prints kernel drivers and modules used for a particular device, **-v** detailed output
	`$ lspci | grep -i ethernet`
	(displays network interfaces)
	`$ lspci | grep -i fibre`
	(displays fibre channel adapters)
	`# lspci -v`
	(displays detailed information about all devices connected to the PCI buses)

lsusb	prints information about USB buses and devices connected to them, **-v** detailed output
lsscsi	prints information about SCSI buses and devices connected to them, **-s** disk size, **-u** LUN (logical unit identifier), **-w** WWN (device identifier – equivalent to a MAC address), **-v** detailed output `$ lsscsi | grep tape` (lists all SCSI tape devices)
rescan-scsi-bus.sh	rescans all SCSI buses, **-a** checks also newly added devices

DEVICES & PARTITIONS	

lsblk [<device ... | partition ... | LV ...>]	prints information about all or specified block devices in a tree-like format (device name, major and minor number of the device, whether or not the device is removable, size of the device, whether or not the device is read-only, type of device and mount point), **-a** including empty devices, **-f** file system type, volume label and UUID, **-m** owner, group and permissions, **-p** full device paths, **-S** SCSI devices only, **-o** <list> specifies information (columns) to be displayed, **-l** output in the form of a list
	`$ lsblk -o NAME,SIZE`
	(prints the name and size of all block devices)
	`$ lsblk | grep mpath | sort -u`
	(prints a sorted list of multipath devices)

blkid [<device ... | partition ... | LV ...>]	prints attributes of all or specified block devices (device name, UUID, volume label, block size and file system type), **-L** <LABEL> prints a partition or logical volume that uses the specified label, **-U** <UUID> prints a partition or logical volume that uses the specified UUID
multipath [<device>]	detects multiple paths to remote storage devices, **-ll** prints the current multipath topology, **-r** reloads all multipath device maps, **-t** lists the default configuration values, **-f** removes a specified unused multipath device map, **-F** removes all unused multipath device maps
mpathconf [<command ...>]	displays device mapper multipath configuration, **--enable** enables multipathing, **--disable** disables multipathing
hdparm <device ... | partition ...>	prints or sets parameters of a specified SATA/IDE device, **-I** prints detailed information about a disk `# hdparm -r0 /dev/sdb*` (removes the write protection from the USB device)
blockdev <command ...> <device ... | partition ...>	prints or sets device properties, **-v** detailed output, **--setro** sets "read-only" permissions, **--setrw** sets "read-write" permissions, **--getsize** prints the number of sectors per device
badblocks <device | partition>	checks the device for bad blocks, **-b** <n> specifies the size of blocks in B (1024 by default), **-i** <file> skips testing a list of already existing known bad blocks contained in the file, **-v** detailed output `# badblocks -vb 4096 /dev/sda2 > bad_blocks` (searches the specified device for bad blocks and saves the result to a file)
smartctl <device>	**-a** prints detailed information about the status of the specified device (supporting SMART technology) in order to prevent its damage, **-s** {on | off} activates/deactivates SMART, **-t** {short | long} executes short/long test `# smartctl -s on -t long /dev/sda` (activates SMART and executes a long test on the specified device)
iotop	displays disk I/O loads by running processes and data write and read speeds in an interactive and dynamic way, **-o** active processes only, **-u** <user> processes of a specified user, **-p** <PID> a specified process only; interactive options: **q** quits the program

DEVICES & PARTITIONS	

iostat [<device ...>] [<interval> [<count>]]	displays the processor usage and disk I/O load statistics, **-d** disk I/O loads only, **-p** including partitions, **-x** extended statistics, **-h** in human readable format
	`$ iostat -d 5`
	(continuously prints information about the disk load at five second intervals)
	`$ iostat -dpxh sda sdb 1 3`
	(prints three reports of extended statistics at one second intervals for devices "sda" and "sdb" and their partitions)

xxd [<file | device>]	prints the contents of the specified file in hexadecimal format; if no file is specified, reads from STDIN, **-l** <n> specifies the number of octets `# xxd -l 512 /dev/sda` (displays the contents of the master boot record (MBR) in hexadecimal format)
cat /proc/partitions	displays partitions, their major and minor numbers and the number of occupied blocks
fdisk <device | partition>	prints the number of cylinders on a disk; using the subsequent option **m** displays a menu for working with the partition table: **p** prints the partition table, **n** creates a new partition, **l** prints all available types of partitions, **t** sets the type of partition, **d** removes a partition, **q** quits the program, **w** writes changes to the disk and quits the program; **-l** lists the partition table for the specified device, if not specified, lists the partition tables for all currently mounted disks `# fdisk -l /dev/sdb` (prints the partition table on the specified device)

DEVICES & PARTITIONS	

parted [<device>] [<command>]	**print** displays the partition table, **unit** specifies the units to display partition sizes (e.g., "s", "B", "kB", "MB", "MiB", "GB", "GiB", "TB", "TiB"), **mklabel** creates a new disk label (e.g., "msdos" or "gpt"), **mkpart** <partition_type | partition_name> [<FS_type>] <start> <end> creates a new partition, **rm** <partition_number> removes a partition, **select** <device> selects a device to use, **set** <partition_number> <flag> <state> sets the state of the flag on the partition, **resizepart** <partition_number> <end> resizes a partition to the required value, **quit** quits the program, **-l** prints partition layout on all block devices; if no command is specified, it starts in interactive mode
	`# parted /dev/vda print`
	(displays the partition table on the "/dev/vda" disk)
	`# parted /dev/vda unit s print`
	(displays the partition table by sectors)
	`# parted /dev/vdb mklabel msdos`
	(writes an MBR disk label to a disk)
	`# parted /dev/vdb mkpart primary xfs 2048s 1001MB`
	(creates a primary partition that is 1000 MB in size)
	`# parted /dev/vdb resizepart 1 100%`
	(extends the partition to occupy the remaining free space on the disk)
	`# parted /dev/vdc mklabel gpt`
	(writes a GPT disk label to a disk)
	`# parted /dev/vdc mkpart primary 1MiB 1001MiB; parted /dev/vdc set 1 lvm on`
	(creates a 1000 MiB partition with type Linux LVM)
	`# parted /dev/vdc mkpart swap1 linux-swap 1001MiB 1501MiB`
	(creates a swap partition named "swap1")

partprobe [<device ...>]	informs the OS of partition table changes, **-s** prints a summary of block devices and their partitions
kpartx <device>	**-a** creates device maps (in */dev/mapper* directory) for partitions detected on a device, **-d** removes device maps, **-u** updates device maps `# kpartx -a /dev/mapper/mpatha` (creates device mapper devices for the partitions on the specified device)
dmsetup <command> [<VG/LV ... | device ...>]	**ls** lists mapped devices (in */dev/mapper* directory) and their major and minor numbers, **table** displays mappings of the physical block devices to logical (mapped) devices – logical block device, start of logical sector, number of sectors, target type (e.g., "linear", "striped", "mirror", "snapshot" or "crypt"), physical block device (major and minor numbers) and start of physical sector, **info** displays information about a mapped device (e.g., its name, state, major and minor number or UUID), **remove** removes a mapped device, **--target** specifies the target type `# dmsetup ls --target crypt` (lists mapped devices of encrypted block devices)

DEVICES & PARTITIONS	
cryptsetup <command> <partition | LV> [<options> <arguments>]	**luksFormat** encrypts the device using LUKS and sets a passphrase, **luksOpen** decrypts the encrypted device and maps it to the specified logical device mapper device, which it creates at the same time, **luksAddKey** adds a passphrase to the encrypted device, **luksRemoveKey** removes a passphrase from the encrypted device, **luksDump** prints encryption information of the encrypted device (the cipher, encryption algorithm and key slots for the passphrase), **luksClose** unmaps the encrypted device from the logical device mapper device `# cryptsetup luksOpen /dev/vdb1 secure` (decrypts the encrypted device "/dev/vdb1" and maps it to the logical device mapper device "/dev/mapper/secure")

mdadm [<mode>] <raid_device> [<option>] [<component_device>]	manages software RAID (virtual) devices; useful modes are **--create** creates software RAID, **--grow** changes the size or shape of an active array, **--add** adds a physical device, **--fail** marks a physical device as faulty and deactivates it (useful when testing RAID 1 or RAID 5, but not RAID 0), **--remove** removes a physical device (must not be active), **--detail** prints detailed information about a RAID device, **--stop** deactivates a raid device, **--assemble** activates a pre-existing raid device, **--scan** applies a specified operation to all arrays listed in */etc/mdadm.conf*; **-v** detailed output
	`# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sd{b,c}1`
	(creates software RAID 1 including "/dev/sdb1" and "/dev/sdc1")
	`# mdadm --grow /dev/md1 --raid-devices=3`
	(changes the size of the raid device to three active physical devices)
	`# mdadm --add /dev/md1 /dev/sdd1`
	(adds the physical device "/dev/sdd1" to the raid device "/dev/md1")
	`# mdadm --stop /dev/md1`
	(deactivates the raid device)
	`# mdadm --assemble --scan`
	(activates all defined raid devices)

cat /etc/mdadm.conf	prints configuration information about mdX RAID devices
cat /proc/mdstat	prints the current status of mdX RAID devices
sync	synchronizes the data stored in the disk buffer with the disk (usually the system saves data with a time delay)
eject [<device>]	ejects a removable device (CD/DVD by default), **-t** inserts a removable device `$ eject` | `$ eject /dev/cdrom`
xinput [<option>] [<device>]	configures and tests X input devices such as mice, keyboards, and touchpads, **list** prints information about the specified input devices, **test** tests a device, in case of error an endless loop of received events is displayed; with no argument all input devices are listed `$ xinput test 11` (tests the device with ID 11)

FILE SYSTEM & SWAP	
cat /proc/filesystems	displays a list of the file system types currently supported by the kernel
cat /etc/fstab	lists persistently defined file systems – a device/partition name, UUID or LABEL, mount point, FS type, mount options (see the "mount" command), backup by the "dump" program (0 = no, 1 = yes) and the order of the FS check by the "fsck" program at system startup (0 = no check, 1 = first order (for the "/" FS), 2–9 other orders)

mke2fs | **mkfs.ext2** | **mkfs.ext3** | **mkfs.ext4** <partition | LV>	creates an ext2, ext3 or ext4 file system on the specified partition or logical volume, **-b** <n> specifies the block size in B, **-c** checks the device for bad blocks, **-i** <n> specifies the number of bytes per i-node, **-t** <FS> specifies the type of FS (when using "mke2fs"), **-L** <LABEL> sets a volume label of the FS, **-v** detailed output
	`# mke2fs -t ext4 /dev/sdb2`
	(creates an "ext4" file system on the specified partition)
	`# mkfs.ext4 /dev/mapper/centosvg/weblv`
	(creates an "ext4" file system on the specified logical volume)

mkfs.xfs <partition | LV>	creates an xfs file system on the specified partition or logical volume, **-b size=**<n> specifies the block size in B (4096 B by default), **-L** <LABEL> sets a volume label of the FS `# mkfs.xfs /dev/vg00/opt_lv` (creates an "xfs" file system on the specified logical volume)
mkisofs <directory>	creates an ISO 9660 file system (.iso file) from the specified directory, **-J** Joliet extension (the contents are readable in windows), **-r** RockRidge extension (the contents are readable in Linux/Unix), **-o** <file> defines an output file `$ mkisofs -Jro cd.iso /data` (creates a "cd.iso" file from the specified directory)

resize2fs <partition | LV> [<size>]	changes (enlarges or shrinks) an ext2, ext3 or ext4 file system size to the required value in kB (K), MB (M), GB (G) or TB (T), the FS can be mounted while being enlarged; if the size is not specified when enlarging the FS, the partition's maximum size is automatically used (if the FS occupies the entire partition that needs to be enlarged, the partition must be expanded prior the FS; if the partition needs to be shrunk, it must be unmounted first, the FS should be checked, the size of the FS reduced, and only then the size of the partition should be shrunk so that it is not smaller than the new size of the FS), **-p** prints a percentage completion bars
	`# resize2fs /dev/sdb3 500M`
	(sets the file system size to 500 MB)
	`# resize2fs /dev/vg00/homelv`
	(sets the file system size according to the logical volume size)

xfs_growfs <FS | partition | LV>	enlarges an xfs file system size (the FS must be mounted), **-D** <size> specifies the FS size (if the size is not specified, the partition maximum size is automatically used)
dumpe2fs <FS | partition | LV>	displays an ext2, ext3 or ext4 file system information
xfs_info <FS | partition | LV>	displays an xfs file system information

FILE SYSTEM & SWAP	

tune2fs <partition | LV>	changes parameters of an ext2, ext3 or ext4 file system, **-l** lists the contents of the FS superblock, **-L** <LABEL> sets a volume label of the FS, **-j** adds an ext3 journal to the ext2 FS (it must be unmounted first; for permanent settings the entry "ext2" must also be modified to "ext3" in */etc/fstab*), **-O** (^)<feature> sets or when using "^" character clears a particular feature of the FS, **-o** (^)<mount_option> sets or when using "^" character clears a particular mount option of the FS
	`# tune2fs -j /dev/sdb1`
	(changes the file system type "ext2" to journaled "ext3")
	`# tune2fs -O ^has_journal /dev/sdb1`
	(changes the file system type "ext3" back to "ext2")
	`# tune2fs -o acl /dev/sdb1`
	(enables ACL on the file system)

xfs_admin <partition | LV>	changes parameters of an xfs file system (the FS must be unmounted, "/" in "read-only" mode), **-L** <LABEL> sets a volume label of the FS
debugfs [<partition | LV>]	interactively examines and potentially alters an ext2, ext3 or ext4 file system (the FS must be unmounted, "/" in "read-only" mode), **-w** opens the FS in "read-write" mode; **open** <partition> opens the FS for a check or editing, **close** leaves the FS, **icheck** <block> prints an i-node number using a specified block, **ncheck** <i-node> prints the path to the specified i-node (file), **quit** quits the program
xfs_db <partition | LV>	interactively examines and potentially alters an xfs file system (the FS must be unmounted, "/" in "read-only" mode), **quit** quits the program

e2fsck | **fsck.ext2** | **fsck.ext3** | **fsck.ext4** <partition | LV>	checks and optionally repairs an ext2, ext3 or ext4 file system (the FS must be unmounted, "/" in "read-only" mode), **-b** <superblock> uses the specified superblock, **-c** checks the device for bad blocks, **-f** forces checking even if the FS seems to be clean, **-n** checks the FS in a non-interactive way and answers 'no' to all questions, **-p** automatically repairs the FS, **-v** detailed output
	`# e2fsck -n /dev/sdb1`
	(checks a file system)
	`# e2fsck /dev/sdb1 -b 97608`
	(repairs a file system using an alternate superblock)

xfs_repair <partition | LV>	checks and repairs an xfs file system (the FS must be unmounted, "/" in "read-only" mode), **-n** performs a check without altering the FS, **-v** detailed output
	`# xfs_repair -n /dev/sdb1`
	(checks a file system)
	`# xfs_repair /dev/sdb1`
	(checks and automatically repairs a file system)

FILE SYSTEM & SWAP	

dump [<source>]	backs up an ext2, ext3 or ext4 file system (the FS must be unmounted, "/" in "read-only" mode), **-**<n> specifies an incremental backup level (values of 0–9, 0 = full backup), **-a** automatic size (writes data to the end of the tape), **-f** <target> specifies a target file or device, **-z** gzip compression, **-j** bzip2 compression, **-u** updates */etc/dumpdates* (overview of performed backups), **-v** detailed output, **-L** <label> labels an archive, **-M** "multi-volume" backup (for archives bigger than 2 GB), **-S** determines the amount of space that is needed to perform the dump, **-W** prints the FS, which needs to be dumped (data from */etc/dumpdates* and */etc/fstab*)
	`# dump -0uf /tmp/home.bckp /home`
	(backs up the home directory to the specified file)
	`# dump -0auf /dev/st0 -L Full_bckp_11-07-08 /`
	(backs up the entire system to the tape under the specified name)

xfsdump [<source>]	backs up an xfs file system (the FS must be unmounted, "/" in "read-only" mode), **-l** <n> specifies an incremental backup level (values of 0–9, 0 = full backup), **-f** <target> specifies a target file or device, **-v** detailed output, **-I** displays the xfsdump inventory */var/lib/xfsdump/inventory/** (overview of performed backups), **-L** <label> labels an archive `# xfsdump -f /dev/st0 /` (backs up the entire system to the tape)

restore	restores files or file systems from backups made by the "dump" program to the working directory, **-f** <source> specifies a source file or device, **-r** newly formatted FS, **-x** existing FS, **-s** <n> number of a backup (file), **-i** interactive mode, **-t** prints the contents of an archive, **-v** detailed output
	`# restore -rvf /dev/rmt0`
	(performs a full file system recovery from the tape)
	`# restore -is 3 -f /dev/rmt0`
	(performs an interactive data recovery – individual files can also be recovered)

xfsrestore [<target>]	restores files or file systems from backups made by the "xfsdump" program, **-f** <source> specifies a source file or device, **-i** interactive mode, **-r** performs a cumulative restore of full and incremental backups, **-t** prints the contents of an archive, **-v** detailed output, **-L** <label> specifies the label of an archive to restore, **-R** resumes a previously interrupted restore `# xfsrestore -f /dev/st0 /` (performs a full file system recovery from the tape)
quota	displays the logged-in user's disk usage and limits (quotas) in kB on all mounted ext2, ext3 or ext4 file systems, **-f** <FS> for a specified file system, **-u** <user> for a specified user, **-g** <group> for a specified group, **-s** in human readable format, **-v** detailed output

FILE SYSTEM & SWAP	

xfs_quota	runs in a basic mode and interactively displays disk usage and limits (quotas) on all mounted xfs file systems, **-x** runs in an expert mode and interactively displays or sets quotas on xfs file systems, **-c** <command> executes a subcommand directly from the command line (non-interactive mode)
	`# xfs_quota -xc 'report -h' /home`
	(displays a quota report for the "/home" file system)
	`# xfs_quota -xc 'limit bsoft=1000m bhard=1500m marek' /home`
	(sets a soft and hard block limit of 1000 MB and 1500 MB for user "marek" on the "/home" file system)

repquota [<FS ...>]	prints a summary of the disk usage and quotas for the specified FS, **-a** for all mounted FS, **-u** reports quotas for users (by default), **-g** reports quotas for groups, **-s** in human readable format, **-v** detailed output

quotacheck [<FS>]	checks quota information (maximum size of available disk space in i-nodes or in 1-kB blocks for users or groups) on the specified FS with "usrquota" or "grpquota" attribute in */etc/fstab* (before the scan the FS is mounted in "read-only" mode, then again in "read-write" mode), **-a** checks all mounted FS except for NFS, **-c** performs a new scan and creates *aquota.user* or *aquota.group* files (ignores existing quota files), **-u** checks user quotas (by default), **-g** checks group quotas, **-v** detailed output, **-i** interactive mode
	`# quotacheck -cvug /home`
	(activates quotas on a specified FS)
	`# quotacheck -avug`
	(prints the usage of all mounted FS with quotas enabled)

edquota	sets quotas by editing *aquota.user* or *aquota.group* files, **-u** <user> for a specified user, **-g** <group> for a specified group, **-p** <pattern> according to the pattern (user, group or project), **-f** <FS> for the specified FS, **-t** sets a maximum period (in units of seconds, minutes, hours or days), during which a "soft limit" can be exceeded up to the value called a "hard limit" (if this period is not specified, the maximum value = "soft limit") `# edquota -p rob $(awk -F: '$3 > 499 {print $1}' /etc/passwd)` (sets quotas for all users whose UID is greater than 499 according to user "rob")
setquota [<FS ...>]	sets quotas directly from the command line, **-u** <user> for a specified user, **-g** <group> for a specified group, **-p** <pattern> according to the pattern (user, group or project), **-a** for all mounted FS with a "usrquota" / "grpquota" parameter set, **-t** sets a maximum period (in units of seconds, minutes, hours or days), during which a "soft limit" can be exceeded up to the value called a "hard limit" (if this period is not specified, the maximum value = "soft limit") `# setquota -u tim 512 1024 0 0 /home` (sets user "tim" quotas on the specified FS in the order "block-softlimit block-hardlimit inode-softlimit inode-hardlimit")

FILE SYSTEM & SWAP	
quotaon [<FS ...>]	turns quotas on on a specified FS, **-a** on all mounted FS, **-u** user quotas (by default), **-g** group quotas
quotaoff [<FS ...>]	turns quotas off on a specified FS, **-a** on all mounted FS, **-u** user quotas (by default), **-g** group quotas
cat /etc/mtab | **cat /proc/mounts** | **cat /proc/self/mounts**	lists currently mounted file systems
mount [<device | partition | LV | directory | NFS_server:FS [directory]>]	prints currently mounted file systems (data from */etc/mtab*) or mounts a specified FS, **-a** all FS in */etc/fstab* with the "defaults" or "auto" option, **-L** <LABEL> according to the volume label, **-U** <UUID> according to the UUID, **-t** <FS> specifies the FS type, **-n** without writing in */etc/mtab* (e.g., if /etc should be mounted as "read-only" FS), **-o** <attribute> specifies comma separated mount options (e.g., **remount** remounts an already mounted FS – **ro** "read-only", **rw** "read-write", (**no**)**dev** allows (not) character and block devices to access the FS, (**a**)**sync** allows (not) a synchronous read / write access to the FS, (**no**)**auto** mounts (not) the FS automatically, **users** any user is allowed to mount and unmount the FS, (**no**)**user** allows (not) a regular user to mount the FS (and only this user is allowed to unmount it again), **usrquota**/**grpquota** allows disk usage and limits on the FS for particular users/groups (the attribute must exist in */etc/fstab*), (**no**)**acl** allows (not) extended ACL permissions for an ext2, ext3 or ext4 FS, (**no**)**atime** updates (not) i-node access time for each access to the FS, (**no**)**suid** allows (not) setuid and setgid permissions for executable files, (**no**)**exec** allows (not) execution of binary files or scripts on the FS, **defaults** = rw, suid, dev, exec, auto, nouser, async) `# mount /dev/fd0	mount /mnt/floppy` (mounts a floppy disk – mount options for the device/partition must be specified in */etc/fstab*) `# mount -t iso9660 /dev/cdrom /mnt/cdrom` (mounts the contents of the CD/DVD on the specified directory) `# mount -t iso9660 -o loop /tmp/image.iso /mnt/iso` (mounts the contents of the ISO file on the specified directory) `$ mount -t vfat /dev/sdc1 /mnt/flash` (mounts the contents of the USB flash disk on the specified directory) `# mount -no remount,ro /` (mounts the root file system as "read-only") `# mount server_01:/data /mnt/nfs` (mounts "/data" from a remote server on the local directory via NFS) `# mount LABEL="Test Label" /test` (mounts the file system on the specified directory according to its volume label) `# mount UUID="b3cb79b8-a8c6-f188-064b-82236deb1604" /test` (mounts the file system on the specified directory according to its UUID)

FILE SYSTEM & SWAP	
umount <device ... | partition ... | LV ... | directory ...>	unmounts a file system (must not be used by any process), **-a** all mounted FS, **-f** forces the unmount of the FS `$ umount /mnt/flash` | `$ umount /dev/sdc1` (unmounts the USB flash disk mounted on the specified directory or associated with the specified device)
mountpoint <directory>	checks whether a specified directory is a mount point (data from */proc/self/mountinfo*), **-d** prints the major and minor number of the device that is mounted on the directory
showmount [<host>]	prints a list of all hosts (clients) which are allowed to mount a local directory via NFS from the local or specified server (data from */etc/exports(.d/*)*), **-a** prints hostnames or IP addresses of the clients and the directories that are currently exported to them, **-e** prints a list of all exported directories and clients which are allowed to mount them

exportfs [<options>] [<client>**:**<directory>]	defines remote hosts (clients) and local directories that will be exported to them via NFS (permanent settings are configured in */etc/exports(.d/*)*); with no argument or with the **-s** option a list of all exported directories is displayed, **-a** exports all directories listed in */etc/exports(.d/*)*, **-r** repeats exporting of all items from */etc/exports(.d/*)* (synchronizes */var/lib/nfs/etab* with */etc/exports(.d/*)*), **-i** ignores information in */etc/exports(.d/*)* and uses only options specified directly on the command line, **-u** <client>**:**<directory> cancels a specified export, with the "-a" option cancels the export of all items in */etc/exports(.d/*)*, **-o** specifies other options for exporting a local directory (the default options are **sync,ro,root_squash,wdelay**), **-v** detailed output
	`# exportfs -av`
	(exports all directories listed in */etc/exports(.d/*)*)
	`# exportfs -o rw client1:/usr/tmp`
	(allows the remote computer to write to the specified directory)
	`# exportfs -au`
	(cancels the export of all directories listed in */etc/exports(.d/*)*)

nfsstat	displays statistics about NFS client and server activities, **-c** the NFS client statistics only, **-s** the NFS server statistic only, **-m** displays detailed information about mounted NFS file systems
nfsiostat [<mount_point ...>] [<interval> [<count>]]	displays NFS client I/O statistics on specified or all NFS mount points (data from */proc/self/mountstats*) `$ nfsiostat /mnt/nfs 10 5` (displays five I/O statistics on a specified mount point at ten second intervals)

FILE SYSTEM & SWAP	

df [<file ... | directory ... | partition ... | LV ...>]	displays information about the disk space usage on the specified or all currently mounted file systems in kB (the file system's device name, total size, used space, available space, percentage of used space, and the mount point), **-m** in MB, **-i** lists inode information instead of block usage, **-h** in human readable format, **-l** only local FS, **-T** prints the type of all FS, **-t** <FS> only FS of the specified type, **-P** uses the POSIX output format
	`$ df -ml | sed '1 d' | awk '{sum+=$3} END {printf "%.f\n",sum}'`
	(prints the total used space in MB, which is occupied by all local file systems)
	`$ df -P | awk '+$5 >= 90 {print}'`
	(prints file systems with a used capacity equal to or greater than 90 %)

mkswap <partition | LV>	creates a swap space for virtual memory, **-L** <LABEL> sets a volume label
swapon <partition ... | LV ...>	activates a swap space (for permanent activation the swap partition must be specified in */etc/fstab* with the "auto" option), **-a** activates all swap partitions specified in */etc/fstab*, **-s** displays active swap partitions (data from */proc/swaps*), **-v** detailed output (partition name, size, usage and priority)
swapoff <partition ... | LV ...>	deactivates a swap space, **-a** deactivates all swap partitions specified in */etc/fstab*, **-v** detailed output
cat /proc/swaps	displays active swap partitions
photorec	recovers lost data from a HDD or portable media of various types of file systems; it starts in interactive mode and automatically detects the FS; a specific data format can also be recovered

LOGICAL VOLUME MANAGER (LVM)	**Virtualization And Dynamic Partition Management**
Physical Volume (PV)	**Physical Block Device Or Partition**
pvcreate <device ... | partition ...>	creates a physical volume (PV) from the specified disks or partitions `# pvcreate /dev/sda2 /dev/sdb` (creates a physical volume from the specified partition and disk)
pvs	lists all physical volumes (PV) in the system, their volume group (VG), attributes, size and free space
pvdisplay [<PV ...>]	prints detailed information about all or specified physical volumes (PV), **-m** displays the mapping of physical extents (PE) to logical volumes (LV) and logical extents (LE)
pvchange <PV ...>	changes attributes of a physical volume (PV), **-x** {y | n} enables/disables allocation of physical extents (PE) on the physical volume (PV)

LOGICAL VOLUME MANAGER (LVM)	**Virtualization And Dynamic Partition Management**

pvresize <PV ...>	resizes a physical volume (PV)
	`# pvresize --setphysicalvolumesize 40G /dev/sda1`
	(sets the size of the physical volume)
	`# pvresize /dev/sda4`
	(extends the physical volume with all free space on /dev/sda4)

pvmove <source_PV> [<target_PV>]	moves data from one physical volume (PV) to another, if the target physical volume (PV) is not specified, the data is moved to any free space within the particular volume group (VG)
pvremove <PV ...>	removes a physical volume (PV) from LVM
Volume Group (VG)	**Storage Pool from One Or More Physical Volumes**
vgcreate <VG> <PV ...>	creates a volume group (VG) from the specified physical volumes (PV), **-s** <size> sets the physical extent (PE) size on the physical volumes (PV) in this volume group (VG) (4MB by default) `# vgcreate root_vg /dev/sda2 /dev/sdb` (creates a volume group "vg00" from the specified physical volumes)
vgs	lists all volume groups (VG) in the system, the number of their physical volumes (PV), logical volumes (LV) and copies, attributes, size, and free space
vgdisplay [<VG ...>]	prints detailed information about all or specified volume groups (VG), **-v** including their physical volumes (PV) and logical volumes (LV)
vgchange <VG ...>	changes attributes of a volume group (VG), **-a** {y | n} activates/deactivates the volume group (VG), **-l** <n> sets the maximum number of logical volumes (LV) in the volume group (VG), **-p** <n> sets the maximum number of physical volumes (PV) in the volume group (VG), **-x** {y | n} enables/disables the addition or removal of physical volumes (PV) to/from the volume group (VG) `# vgchange -a n data_vg` (deactivates a volume group)
vgrename <old_VG> <new_VG>	renames a volume group (VG) `# vgrename data_vg datavg` (renames a volume group "data_vg" to "datavg")
vgextend <VG> <PV ...>	adds a physical volume (PV) to the volume group (VG) `# vgextend vg00 /dev/sda4` (adds a physical volume "/dev/sda4" to the volume group "vg00")
vgreduce <VG> <PV ...>	removes a physical volume (PV) from the volume group (VG) (the physical volume (PV) must be empty)
vgexport <VG ...>	exports a volume group (VG) (makes it unknown to the system)
vgimport <VG ...>	imports a volume group (VG) (makes an exported volume group (VG) known to the system)

LOGICAL VOLUME MANAGER (LVM)	**Virtualization And Dynamic Partition Management**
vgsplit <source_VG> <target_VG> [<PV ...>]	moves a physical volume (PV) to a new or existing volume group (VG), **-n** <LV> moves only physical volumes (PV) used by the specified logical volume (LV), **-l** <n> sets the maximum number of logical volumes (LV) in the volume group (VG), **-p** <n> sets the maximum number of physical volumes (PV) in the volume group (VG) `# vgsplit vg01 vg02 /dev/sde1` (moves the physical volume "/dev/sde1" from the volume group "vg01" to "vg02")
vgmerge <target_VG> <source_VG>	merges an inactive source volume group (VG) into the target volume group (VG) (the physical extent (PE) sizes must be equal and physical volume (PV) and logical volume (LV) summaries of both the volume groups (VG) must fit into the target volume group's limits) `# vgmerge vg01 vg02` (merges "vg02" into "vg01", keeping all the properties of the "vg01" volume group)
vgremove <VG ...>	removes a volume group (VG)
vgcfgrestore <VG>	restores a volume group (VG) metadata from the last backup, **-l** prints a list of all archived metadata (archive files) for the specified volume group (VG), **-f** <archive_file> restores the volume group (VG) configuration from the specified file `# vgcfgrestore -f /etc/lvm/archive/vg00_00031-1442492648.vg vg00` (restores the "vg00" volume group configuration from the specified file)
Logical Volume (LV)	**Virtual Partition Within a Volume Group**

lvcreate <LV> <VG>	creates a logical volume (LV) in the specified volume group (VG), **-n** <LV> specifies the name of the logical volume (LV), **-L** <size> specifies the size of the logical volume (LV), **-l** <n> specifies the size of the logical volume (LV) by allocating the number of logical extents (LE), **-m** <n> specifies the number of logical volume (LV) copies (creates a mirror of the linear logical volume (LV)), **-p** {r | rw} sets read or read-write permissions
	`# lvcreate -n web_lv -L 10G datavg`
	(creates a 10 GB "web_lv" logical volume in the "datavg" volume group)
	`# lvcreate -n db_lv -l 25%VG datavg`
	(creates a "db_lv" logical volume that occupies 25% of the "datavg" volume group)

lvs	lists all logical volumes (LV) in the system, their volume group (VG), attributes, size, and other information `# lvs -a -o segtype,devices,lv_name,vg_name` (prints the type of logical volume (linear, striped, mirrored, snapshot), physical volume, name of the logical volume and volume group)

LOGICAL VOLUME MANAGER (LVM)	Virtualization And Dynamic Partition Management
lvdisplay [<VG/LV ... | VG ...>]	prints detailed information about all or specified logical volumes (LV) or all logical volumes (LV) in the specified volume group (VG), **-m** displays the mapping of logical extents (LE) to physical volumes (PV) and physical extents (PE) and the type of logical volume (LV) (linear, striped, mirrored, snapshot)
lvchange <VG/LV ...>	changes attributes of a logical volume (LV), **-a** {y | n} activates/deactivates the logical volume (LV), **-p** {r | rw} sets read or read-write permissions `# lvchange -a n datavg/web_lv` (deactivates a logical volume)
lvrename <old_VG/LV> <new_VG/LV>	renames a logical volume (LV) `# lvrename datavg/web_lv datavg/weblv` (renames a logical volume "web_lv" to "weblv")

lvresize <VG/LV ...>	changes the size of a logical volume (LV), **-L** [**+** | **-**]<size> sets, enlarges or shrinks the size of the logical volume (LV) to the required value in MB (M), GB (G), TB (T), PB (P), EB (E), **-l** [**+** | **-**]<n> specifies the size of the logical volume (LV) by allocating the number of logical extents (LE) (if the file system occupies the entire logical volume (LV) that needs to be enlarged, the logical volume (LV) must be expanded prior the file system; if the logical volume (LV) needs to be shrunk, it must be unmounted first, the file system should be checked, the size of the file system reduced, and only then the size of the logical volume (LV) should be shrunk so that it is not smaller than the new size of the file system), **-r** resizes the associated file system
	`# lvresize -rL +200M vg00/rootlv`
	(increases the size of the logical volume including the file system by 200 MB)
	`# lvresize -l +100%FREE /dev/VolGroup00/LogVol00`
	(extends the logical volume by the remaining free space in the volume group)

lvconvert <VG/LV>	changes the type of logical volume (LV), **-m** <n> specifies the number of logical volume (LV) copies (creates a mirror of the linear logical volume (LV)), **-s** <source_LV> <target_LV> creates a snapshot of the logical volume (LV) `# lvconvert -s vg00/lvol1 vg00/lvol2` (creates a snapshot of the logical volume)
lvremove <VG/LV ... | VG ...>	removes a logical volume (LV) or all logical volumes (LV) in the specified volume group (VG)

ADVANCED STORAGE FEATURES	
STRATIS (implemented from RHEL 8)	**manages pools of physical storage devices and expands file systems when needed**
stratis pool create <pool> <device ...>	creates a pool of one or more block devices `# stratis pool create pool1 /dev/vdb` (creates the pool "pool1" of the specified disk)
stratis pool list	lists all pools on the system
stratis pool rename <old_pool> <new_pool>	renames a pool

ADVANCED STORAGE FEATURES	
stratis pool destroy <pool ...>	removes a pool
stratis pool add-data <pool> <device ...>	adds a block device to the pool `# stratis pool add-data pool1 /dev/vdc` (adds a disk to the pool "pool1")
stratis blockdev list [<pool>]	lists block devices in the specified pool; with no argument block devices in all pools are displayed
stratis fs create <pool> <FS ...>	creates a dynamic and flexible file system in the pool (in */stratis/<pool>/<fs>*) `# stratis fs create pool1 fs1` (creates the file system "fs1" in the pool "pool1")
stratis fs snapshot <pool> <FS> <snapshot>	creates a snapshot of the specified file system in the pool (in */stratis/<pool>/<fs_snap>*) `# stratis fs snapshot pool1 fs1 fs1_snap` (creates the snapshot "fs1_snap" of the file system "fs1" in the pool "pool1")
stratis fs list [<pool>]	lists file systems and their snapshots in the specified pool; with no argument file systems and their snapshots in all pools are displayed
stratis fs rename <pool> <old_FS> <new_FS>	renames a file system
stratis fs destroy <pool> <FS ... | snapshot ...>	removes a file system or snapshot from the specified pool `# stratis fs destroy pool1 fs1` | `# stratis fs destroy pool1 fs1_snap` (removes the file system "fs1" / snapshot "fs1_snap" from the pool "pool1")
VDO (Virtual Data Optimizer) (implemented from RHEL 8)	**reduces storage consumption on block devices and minimizes data replication**
vdo create --name=<volume> **--device=**<device> **--vdoLogicalSize=**<size>	creates and starts a VDO volume of the specified device and size `# vdo create --name=vdo1 --device=/dev/vdd --vdoLogicalSize=500G` (creates the VDO volume "vdo1" from the disk "/dev/vdd" 500 GB in size)
vdo list	lists all running VDO volumes
vdo start --name=<volume>	starts a stopped, activated VDO volume, **--all** all stopped and activated VDO volumes
vdo stop --name=<volume>	stops a running VDO volume, **--all** all running VDO volumes
vdo activate --name=<volume>	activates a specified VDO volume, **--all** all VDO volumes
vdo deactivate --name=<volume>	deactivates a specified VDO volume, **--all** all VDO volumes
vdo status --name=<volume>	prints settings for the specified VDO volume; if not specified, settings for all VDO volumes are displayed
vdostats	prints the usage of all running VDO volumes, **--human-readable** in human readable format
vdo remove --name=<volume>	removes a VDO volume, **--all** all VDO volumes

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

Permanent link:
https://prompt.cz/en/devices-and-file-system

Last update: **2024/11/25 23:46**