Accounts & Permissions

ACCOUNTS	
whoami | **echo $USER**	prints the username corresponding with the effective UID
who am i | **who -m**	prints the login name of the logged-in user, terminal name and login time
logname	prints the login name of the logged-in user
id [<user ... | UID ...>]	prints the UID and GID of the logged-in or specified user, including all of their groups, **-u** the effective UID, **-g** the effective GID, **-G** the GID of all the user's groups, **-n** with the "-u", "-g" or "-G" option prints the user or group name instead of the numeric designation
lid [<user>]	prints the groups to which the logged-in or specified user is assigned, **-g** <group> prints users in the specified group; the command can only be used with root privileges
finger [<user ...>]	prints the login and real name of the user, their home directory, login shell, last login time and inbox information; with no argument the login and real names of the logged-in users are displayed, including their terminal, idle time, login time and connection method
lslogins [<user>]	prints information about all or specified users in the system – their UID, name, number of running processes, account lock information, time of last login and comment (GECOS); in addition, for the specified users their home directory, login shell, primary group, GID, terminal and computer is printed, **-a** prints information about the password last change and expiration, **-s** lists system accounts, **-u** lists user accounts (with UID 1000 and higher)
users	prints currently logged-in usernames, their number corresponds with the current open sessions (data from */var/run/utmp*)
who	prints currently logged-in usernames, their terminal names and login times (data from */var/run/utmp*), **-u** including PID of their current process, **-q** usernames and their number only, **-m** login name of the user, terminal name and login time (equivalent to the "who am i" command)
w [<user>]	prints the names of all currently logged-in users or the name of the specified logged-in user, their terminal, connection method, login time, idle time, processor usage and current process name; in addition it displays the system time, the time since the computer was last started, number of currently logged-in users and the system load averages for the past 1, 5 and 15 minutes (equivalent to the "uptime" command)

ACCOUNTS	
last [<user ...>]	prints the login time of all or specified users to the system during the last period (since */var/log/wtmp* was created) including the terminal name and connection method, **-f** <file> reads data from the specified file, **-n** <n> last *n* logins only, **-s** <time> since the specified time, **-t** <time> until the specified time, **-d** prints hostnames for remote logins, **-i** prints IP addresses for remote logins, **-x** prints the system shutdown entries and run level changes `$ last -s -3days` (displays user logins during last three days) `$ last -s 2021-03-01 -t 2021-03-31` (displays user logins in the specified period)
lastb [<user ...>]	prints the time of unsuccessful login attempts of all or specified users in to the system during the last period (since */var/log/btmp* was created) including the terminal name and connection method, **-f** <file> reads data from the specified file, **-n** <n> last *n* logins only, **-s** <time> since the specified time, **-t** <time> until the specified time, **-d** prints hostnames for remote logins, **-i** prints IP addresses for remote logins
lastlog	prints a list of all users in the system and their last login times including the terminal names (data from */var/log/lastlog*), **-u** <user> records of the specified user, **-t** <n> information about users logged in during last *n* days
faillog	prints authentication failure records of all users (data from */var/log/faillog*), **-a** all data, **-u** <user> records of the specified user, **-m** <n> sets the maximum number of unsuccessful login attempts, **-r** resets the unsuccessful login attempts counter; uses the "pam_tally" module `# faillog -u kuba -r` (resets the unsuccessful login attempts counter for the specified user)
pam_tally2 (implemented from RHEL 6)	prints authentication failure records of all users (data from */var/log/tallylog*), **-u** <user> records of the specified user, **--reset** resets the unsuccessful login attempts counter; uses the "pam_tally2" module `# pam_tally2 --reset -u jan` (resets the unsuccessful login attempts counter for the specified user)
faillock (implemented from RHEL 6)	prints authentication failure records of all users (data from */var/run/faillock/**), **--user** <user> records of the specified user, **--reset** resets the unsuccessful login attempts counter; uses the "pam_faillock" module `# faillock --user tom --reset` (resets the unsuccessful login attempts counter for the specified user)

ACCOUNTS	
authconfig <options>	**--update** updates the PAM configuration according to the specified options (modifies the */etc/pam.d/*-ac* configuration files), **--updateall** restores the PAM configuration according to the settings in */etc/sysconfig/authconfig*, **--test** prints the current PAM configuration, **--savebackup=**<directory> backs up the PAM configuration files, **--restorebackup=**<directory> restores the PAM configuration files `# authconfig --enablefaillock --faillockargs="deny=5 fail_interval=900 unlock_time=3600" --update` (enables and configures the "pam_faillock" module with a limit of 5 failed login attempts in a 15-minute interval and automatically unlocks locked accounts after 60 minutes)
useradd <user>	creates a user account including their home directory */home/<user>* (copies the contents of the */etc/skel* directory inside), email spool */var/spool/mail/<user>* and primary group of the same name; creating a new account is based on the settings configured in */etc/default/useradd* and */etc/login.defs*, **-D** prints default values, **-d** <directory> specifies a home directory instead of the default, **-g** <group | GID> assigns an existing group as a primary group, **-G** <group> assigns a user into other, comma separated, supplementary groups, **-u** <UID> assigns a specified UID (otherwise the first available one is used), **-o** assigns a duplicate UID (available with "-u" option only), **-r** creates a system account (with UID in the range 201–999, never expiring password and without a home directory), **-s** <shell> assigns a login shell, **-e** <YYYY-MM-DD> sets an account expiration date, **-f** <DD> sets the number of days after a password expires until the account is permanently disabled, **-c** <comment> provides any information about a user (GECOS field in */etc/passwd*), **-Z** <SELinux_user> maps the user to the specified SELinux user `# useradd -c "Jan Novak" -G admins jan` (creates user "jan" with the comment "Jan Novak" and the supplementary group "admins")
userdel <user>	removes a user account, **-r** including a home directory and email spool, **-f** including a home directory and email spool, even if the user is logged in, **-Z** including the mapping to an SELinux user

ACCOUNTS	
usermod <user>	modifies a user account; the same options as for the "useradd" command are used, in addition the **-a** with "-G" option assigns a user into other, comma separated, supplementary groups without the need to specify all the previously defined groups (because the "-G" option itself always defines all valid supplementary groups from the beginning, which overwrites the former settings), **-l** <new_user> renames a user account, **-L** locks a user's password (inserts "!" in front of the encrypted password), **-U** unlocks a user's password (removes "!" in front of the encrypted password) `# usermod -c "" jan` (removes the comment for the specified user) `# usermod -l honza -d /home/honza jan` (renames the user "jan" to "honza" and changes his home directory to "/home/honza")
chfn [<user>]	changes GECOS field in */etc/passwd* of the logged-in or specified user, **-f** <name> a real name, **-p** <number> office phone number, **-h** <number> private phone number; with no argument it starts in interactive mode (none = empty field)
chsh [<user>]	[**-s** <shell>] changes a login shell of the logged-in or specified user, **-l** prints a list of available shells from */etc/shells*; with no argument it starts in interactive mode
chage <user>	changes a user's account and password lifetime settings, **-d** <DD> sets the number of days since January 1st, 1970 when the password was last changed, **-E** <YYYY-MM-DD> sets an account expiration date ("-1" = unlimited account expiration), **-I** <DD> sets the number of days of inactivity after a password has expired before the account is locked, **-l** prints information about an account and password expiration settings, **-m** <DD> sets the minimum number of days between password changes ("0" = the user may change the password at any time), **-M** <DD> sets the maximum number of days during which a password is valid ("-1" = unlimited password expiration), **-W** <DD> sets the number of days of warning before a password change is required; if no option is specified, it starts in interactive mode; the default password expiration settings are configured in */etc/login.defs* `# chage -d 0 james` (changes the user's password expiration date and prompts them to change it on the first login)

ACCOUNTS	
passwd [<user>]	sets or changes the password of the logged-in or specified user, **--stdin** reads the password from STDIN (pipe), **-d** sets no password for an account, **-n** <DD> sets the minimum password lifetime in days, **-x** <DD> sets the maximum password lifetime in days, **-w** <DD> sets the number of days in advance the user is warned of the password expiration, **-l** locks a user's password (inserts "!!" in front of the encrypted password), **-u** unlocks a user's password, **-S** <user> prints information about the settings of the user's password (password status: "PS" = password set, "NP" = no password, "LK" = password locked, the date of the last password's change, minimum and maximum lifetime in days, a warning period before the password's expiration and a period between the password's expiration and the account being locked in days); the default password expiration settings are configured in */etc/login.defs* `# for user in $(awk -F : '{print $1}' /etc/passwd); do passwd -S $user | grep LK; done` (prints users with locked accounts)
mkpasswd	creates a random password, **-l** <n> sets the password's length (9 characters by default), **-C** <n> sets the minimum number of capital letters (2 by default), **-c** <n> sets the minimum number of small letters (2 by default), **-d** <n> sets the minimum number of digits (2 by default), **-s** <n> sets the minimum number of special characters (1 by default)
chpasswd <user>**:**<password>	modifies a specified user's password and encrypts it by algorithm defined in */etc/login.defs*, **-c** {NONE|DES|MD5|SHA256|SHA512} specifies a different encryption algorithm, **-e** indicates the newly submitted passwords are in encrypted form (they are specified in clear text by default) `# for user in $(awk -F ":" '{if (length($2) > 2 && $2 !~ /^(!!)?(\$[1256]\$)/) print $1":"$2 }' /etc/shadow); do echo "$user" | chpasswd -c SHA512; done` (encrypts clear passwords of all users)
cat /etc/passwd	prints existing local users, their encrypted password (character "*" means that an account is locked) or an "x" character (the password is in */etc/shadow*), UID, primary GID, comment field (GECOS), home directory and login shell `$ awk -F ":" '{if ($7 ~ /.+(sh\|bash\|ksh\|zsh)$/ && ($3 >= 500)) print $1}' /etc/passwd` (prints user accounts in the system – with UID 500 and higher) `$ awk -F ":" '{if ($4 == 3000) print $1}' /etc/passwd` (lists all users whose primary group has GID 3000)

ACCOUNTS	
cat /etc/shadow	prints existing local users, their encrypted password (if the field is empty, the account has no password; inserting the character "*", "!" or "!!" before the password locks the account; by default, the "useradd" command creates a locked user account – i.e., only "!!" characters are present instead of a password), last password change in days since January 1st, 1970, the minimum number of days between password changes ("0" = the user may change the password at any time), the maximum number of days during which a password is valid ("-1" = unlimited password expiration), the number of days of warning before a password change is required, the number of days of inactivity after a password has expired before the account is locked and the number of days since January 1st, 1970 the account has been locked
groupadd <group>	creates a group account, **-g** <GID> assigns a specified GID (otherwise the first available one is used), **-o** assigns a duplicate GID (available with "-g" option only), **-r** creates a system group (with GID in the range 201–999) `# groupadd -g 350 -r nexus` (creates a system group "nexus" with GID 350)
groupdel <group>	removes a group account (it is not possible to remove an existing user's primary group, the user has to be removed as first)
groupmod <group>	modifies a group account, the same options as for the "groupadd" command, besides these exist: **-n** <new_group> renames a group account
groups | **id -nG** [<user>]	prints the groups to which the logged-in or specified user is assigned
newgrp <group>	logs a user into one of the groups available in */etc/group*; with no argument the primary GID is assigned (used especially when creating new files)
cat /etc/group	prints existing local groups, their encrypted password (character "*" means that an account is locked) or an "x" character (the password is in */etc/gshadow*), GID and a list of comma separated explicit members `$ awk -F ":" '/^admin/{print $3}' /etc/group` (prints the GID of the "admin" group) `$ awk -F ":" '{if ($3 == 3000) print $4}' /etc/group` (lists all explicit users of a group whose GID is 3000)
cat /etc/gshadow	prints existing local groups, their encrypted password (character "*" means that an account is locked) or "!" character (a passwordless account), a list of comma separated administrators and a list of comma separated secondary mebers
vipw	edits */etc/passwd*
vigr	edits */etc/group*
pwconv	creates */etc/shadow* file based on data from */etc/passwd* and */etc/login.defs*, which ensures a safe storage of users' passwords
pwunconv	removes */etc/shadow* (opposite of the "pwconv" command)

ACCOUNTS	
grpconv	creates */etc/gshadow* based on data from */etc/group* and */etc/login.defs*, which ensures a safe storage of groups' passwords
grpunconv	removes /etc/gshadow file (opposite of the "grpconv" command)
pwck	verifies the integrity of */etc/passwd* and */etc/shadow*, the user is prompted to correct possible errors, **-r** prints errors only, **-s** sorts the output by UID
grpck	verifies the integrity of */etc/group* and */etc/gshadow*, the user is prompted to correct possible errors, **-r** prints errors only, **-s** sorts the output by GID

PERMISSIONS	
umask [<permissions>]	prints or sets default permissions for newly created files and directories in the following order: user (owner) - group - others in a numeric (octal) expression, however the digits stand for the permissions that are to be taken from the system value 666 for files and 777 for directories, **-S** symbolic expression; (permanent settings are configured in *~/.bashrc* and *~/.bash_profile*, the default global value is 002 for regular users and 022 for root in */etc/profile* and */etc/bashrc*) `$ umask 0027` | `$ umask 027` | `$ umask 27` (the owner has all permissions, the group has read permissions and access to directories and others have no permissions)

PERMISSIONS	
chmod <permissions> <file ... | directory ...>	changes a file or directory access permissions 1) in symbolic expression in the following order – user definition (**u** = user (owner), **g** = group, **o** = others, **a** = all), operator (**+** adds permissions, **-** removes permissions and **=** sets permissions) and permission specification (**r** = read a file / list the contents of a directory (file or directory names only), **w** = write to a file / write to a directory (creating, deleting and renaming any files or directories), **x** = execute a file / access a directory and make its contents available for reading and writing, **X** = access a directory and make its contents available for reading and writing, **s** = SUID or SGID bit, **S** = "s" and missing "x", **t** = sticky bit, **T** = "t" and missing "x") `$ chmod +x script.sh` (sets the file execute bits for all users) `$ chmod o= .` (removes all permissions for others on the working directory) `$ chmod ug=rw,o-w text.txt` (sets read and write permissions for the owner and group, and removes write permissions for others) 2) in numeric (octal) expression in the following order – (special attribute) - user (owner) - group - others (**4** = read a file / list the contents of a directory (file or directory names only), **2** = write to a file / write to a directory (creating, deleting and renaming any files or directories), **1** = execute a file / access a directory and make its contents available for reading and writing); the values are summed `# chmod 640 /etc/crontab` (sets read and write permissions for the owner and read permissions for the group) with both the expressions it is possible to use option **-R** for recursive mode and **-c** to see the files whose permissions are being changed; a directory must always have an access permission set `# chmod -R go-w /var/www/html` (sets permissions recursively for all files and directories in the specified path) special attributes concern mostly executable files (programs and scripts) or directories and have these values: **4** = SUID bit (an executed process runs with the permissions of the owner of the file, not with the permissions of the user who executed it), **2** = SGID bit (an executed process runs with the permissions of the group of the file, not with the permissions of the user who executed it; if the SGID bit is set for a directory, it ensures that its new contents will be owned by the same group of owners who own the directory), **1** = sticky bit (used for directories whose contents can be deleted or renamed only by the owner of the file or directory, not by any user with write and access permissions for the directory) `# chmod 4755 /usr/bin/passwd` (sets the SUID bit for the specified file) `# chmod 2770 /web` (sets the SGID bit for the specified directory) `# chmod +t /usr/local/tmp` (sets the sticky bit for the specified directory)

PERMISSIONS	
setfacl <option> [[<user>]**:**[<permissions>]] <file ... | directory ...>	**-m** sets ACL permissions to a file or directory depending on the specified options (**u:**[<user | UID>] for a specified user, if it is not specified, the settings are valid for the owner of the file or directory, **g:**[<group | GID>] for a specified group, if it is not specified, the settings are valid for the group owner of the file or directory, **o** for others, **d:** ensures inheriting of the ACL permissions from a directory to its newly created contents, **m:** sets the mask – specifies maximum permissions possible for all named users and groups), **-x** removes ACL permissions from a file or directory depending on the specified options (**u:**<user | UID> for a specified user, **g:**<group | GID> for a specified group), **-b** removes all ACL permissions from a file or directory, **-R** recursively, **--set-file** <file | directory> sets ACL permissions based on the specified file or directory `$ setfacl -m d:u::rwx,g::rx,o:000 ./projects` (sets ACL permissions to the specified directory and newly created content) `$ setfacl -m u:kuba:rw ./projects/kuba.txt` (sets ACL permissions to the file for the specified user) `$ setfacl -x u:kuba ./projects/kuba.txt` (removes ACL permissions from the file for the specified user) `$ setfacl -bR ./projects` (removes all ACL permissions in the specified path recursively) `# setfacl -m u::rwx,g::rx,o::rx /bin/chmod` (sets ACL permissions to the specified file for the owner, group, and others) `$ getfacl file1 | setfacl --set-file - file2` (sets the file "file2" the same ACL permissions as "file1")
getfacl <file ... | directory ...>	prints ACL permissions to a file or directory for specified single users or groups (provided they are set up), **-n** prints UID and GID instead of an account name, **-R** recursively, **-s** skips files with basic permission entries
chattr <operator><attribute> <file ... | directory ...>	sets attributes of a specified file or directory on ext2, ext3 or ext4 file system; operator **+** adds, **-** removes and **=** sets an attribute; attribute **a** prevents from removing and modifying a file (applicable even for root), permits appending new data to the file only, **d** prevents from the backup by the "dump" program, **i** prevents from removing and any kind of modifying a file (applicable even for root); **-R** recursively `# chattr +i /etc/inittab` (adds an attribute to the specified file)
lsattr [<file ... | directory ...>]	prints attributes of the contents of the working directory or a specified file or the contents of a specified directory on ext2, ext3 or ext4 file system, **-a** prints hidden files, **-d** directory itself without its contents, **-R** recursively

PERMISSIONS	
chown [<owner>][:[<group>]] <file ... | directory ...>	changes the user and/or group ownership of a file or directory, **-R** recursively, **-c** prints the files whose ownership is being changed; if the username or UID is followed by a colon or dot and a group name or GID, the group ownership of the file is changed as well; if no group follows a colon or dot (*chown user: /tmp /var/tmp*), the user's primary group is considered; if a colon or dot and group are specified, but the user is omitted (*chown :group /tmp /var/tmp*), only the group ownership of the file is changed (equivalent to the "chgrp" command)
chgrp <group> <file ... | directory ...>	changes the group ownership of a file or directory; the group is specified by its name or GID, **-R** recursively, **-c** prints the files whose ownership is being changed
su [<user>]	switches to root (system administrator) or a specified user account (changes the effective UID and GID), **-** | **-l** including the user's environment settings (initializes "HOME", "SHELL", "USER", "LOGNAME" and "PATH" variables), **-c** <command> only executes the command under another user
sudo [<command>]	allows an authorized user to execute a command with root or another user privileges using their own password; the user must be specified in */etc/sudoers(.d/*)* in the following order: <user> <host> = [([<effective_user>][:<effective_group>])] [<tag>:] <command> (in absolute form); at the beginning of the file it is possible to define by capital letters aliases representing the authorized users, effective users, hosts and commands, considering that "ALL" expression represents any value in the mentioned items: *kuba ALL = (root) /bin/mount -t iso9660 /dev/cdrom /mnt/cdrom, NOPASSWD: /bin/umount /mnt/cdrom* (kuba is allowed to mount the CD-ROM drive as root and unmount it without a password requirement) *miro localhost = /bin/su [!-]*, !/bin/su *root** (miro is allowed to switch to any user except root on the local host without loading the user's environment) *%admin ALL = SERVICES, PROCESSES, STORAGE* (the members of "admin" group are allowed to execute the commands represented by the specified aliases on any host) *%osadmin ALL=(ALL) ALL* (the members of "osadmin" group are allowed to execute any command on any host) **-b** runs a specified command in the background, **-l** prints information whether the logged-in user is allowed to use "sudo" and lists possible commands that can be executed, **-i** switches to root account, **-u** <user> runs a command as a user other than root, **-g** <group> runs a command with the specified primary group privileges; only root is allowed to edit */etc/sudoers* by the "visudo" command; the usage of "sudo" is logged to */var/log/secure* `$ sudo vi /etc/fstab` (edits the file with root privileges) `$ sudo bash -c "cd /home; du -s * | sort -rn > usage"` (runs the commands in a subshell with root privileges to make the "cd" command and file redirection work) `$ sudo su - root -c /bin/bash` (starts the shell with root privileges)

PERMISSIONS	
sudoedit <file>	allows an authorized user to edit a file with root or another user privileges using their own password, **-u** <user> edits a file as a user other than root, **-g** <group> edits a file with the specified primary group privileges; if the file does not exist, it will be created
visudo	edits */etc/sudoers*, **-c** verifies the integrity of the file, **-f** <file> specifies an alternative sudoers file instead of */etc/sudoers*