

Unix Fundamentals – Users and Groups

Marek Kozłowski

Faculty of Mathematics and Information Sciences

Warsaw University of Technology

*Never use copy&paste for the following exercises. Retype all commands manually!
Don't just read those exercises and examples. Do them, please!*

1. Users and groups database

- (a) Check your UID, GID and secondary group membership:
`$ id`
- (b) Display your local users' database:
`$ less /etc/passwd`
What is root's home directory?
What shells are defined for system accounts?
- (c) List all valid login shells installed on your system:
`$ cat /etc/shells`
- (d) Check the syntax of `/etc/shadow`:
`$ man 5 shadow`
What happens if an encrypted password starts with `!` or `*`?
What password change policies can be applied in Linux?
- (e) The same plain passwords for a few users don't result in the same encrypted passwords. Why? Read about the *salt* parameter:
`$ man 3 crypt`
- (f) Take a look at the `fail2ban` project web page: <https://www.fail2ban.org>. Nowadays it is a standard solution.
- (g) Generate a set of strong random passwords:
`$ pwgen -s`
or just a single one:
`$ pwgen -sN1`
Read the man page:
`$ man pwgen`
- (h) Read about the command for changing passwords interactively:
`$ man passwd`
Don't use it for changing **your** password! Your account details are obtained from a remote database with a different encryption so this won't work (or may result in your password being invalid)!
- (i) Read about the command for changing passwords in batch mode:
`$ man chpasswd`
Administrators like it ;-)
- (j) Take a look at your local groups' database:
`$ less /etc/group`
Some important groups include `'wheel'`, `'wireshark'` or `'uucp'`.
- (k) Take a look at the man page on PAM:
`$ man 8 pam`
What are the authentication tasks?

- (l) List some files in `/etc/pam.d` (most include `/etc/pam.d/system-auth`):
- ```
$ ls /etc/pam.d
$ less /etc/pam.d/system-auth
```
- Note that our computers are configured to use: `'pam_ldap'` and `'pam_group'` for most tasks. The first one allows authentication against remote servers compatible with the LDAP protocol. The latter one allows dynamic group membership granting during logging in.
- Lightweight Directory Access Protocol is out of scope of this course. `'pam_group'` configuration is quite simple:
- ```
$ less /etc/security/group.conf
```
- or for better clarity:
- ```
$ grep -vP '^[\t]*(#|$)' /etc/security/group.conf
```
- (note: numerous PAM modules have `/etc/security/*.conf` configuration files)
- (m) Execute:
- ```
$ id
```
- and:
- ```
$ id your_neighbour
```
- What is the reason for the difference? The module `'pam_group'` which works for currently logged in users.
- (n) Arch Linux is a *systemd* distro (whatever it is). Linux servers based on *openrc*, *sysvinit* etc (whatever those are) should undoubtedly use <https://man.archlinux.org/man/limits.conf.5.en>
- (o) Examine the file `/etc/nsswitch.conf`:
- ```
$ less /etc/nsswitch.conf
```
- and note that there are two databases set for user and group queries – local files and remote LDAP directories. For listing users and groups from both sources use:
- ```
$ getent passwd
$ getent group
```
- (p) Re-login as a different user (*uszatekm*):
- ```
$ su - uszatekm
$ id
$ env # switched thanks to '-'
$ exit
```
- (q) We've mentioned that the `'wheel'` group membership may be important in some cases. 'What cases?' - you may ask. Well, for explanation take a look at:
- ```
$ less /etc/pam.d/su
```
- (r) Check the default settings for the `'useradd'` command:
- ```
$ cat /etc/default/useradd
```
- BTW: why don't you take a quick look at the man page:
- ```
$ man useradd
```

## 2. File system rights

*Note that our lab workstations use some remote file system for home directories. Bi-directional synchronization may take a while, so any changes applied by you may be visible to your neighbor after several seconds (up to 20-30)!*

- (a) Create some file:
- ```
$ touch file2a
```
- List all groups you belong to:
- ```
$ id -Gn
```
- Change group of this file to any other group you belong to (replace *new\_group* with a group you really belong to):
- ```
$ chgrp new_group file2a
```
- or
- ```
$ chown :new_group file2a
```
- Try the same with any group you don't belong to.

- (b) Check permissions of publicly readable and secure files used in the first exercise:
- ```
$ ls -l /etc/passwd
$ ls -l /etc/shadow
```
- (c) List your home directory:
- ```
$ ls -l ~
```
- Remember that directories need the 'x' permission for being readable (consult the slides if in doubt).
- (d) Create a file containing bash commands (you are free to use any editor for this purpose) in your home directory:
- ```
$ cat exe_file
    date
    uname
    id -un
```
- Make the file executable:
- ```
$ chmod +x exe_file
```
- and run it:
- ```
$ ./exe_file
```
- (note: working directory is not in your \$PATH so you must precede program file name with its full path; we can use wildcards and denote it as: './')
- Congratulations! You've just written your first bash script. Honestly it's still a little bit imperfect, so don't add *Bash scripting* to the *Skills* section of your CV yet.
- (e) Make your script publicly readable:
- ```
$ chmod ugo+r exe_file # or just a+r
```
- Can your neighbours display it?
- ```
$ cat ~/.../your_name/exe_file
```
- Grant file access right to your home directory to all users:
- ```
$ chmod +x ~
```
- and ask your neighbours to retry.
- Can they list your home directory or any other files in it?
- Finally reset your home directory rights to restrictive ones:
- ```
$ chmod 700 ~
```
- or
- ```
$ chmod go-x ~
```
- (f) Set file permissions to 'exe\_file' to none:
- ```
$ chmod 000 exe_file
```
- Can you rename this file?
- Can you change its group?
- Can you delete the file?
- (g) Create a file with permissions set to 640:
- ```
$ touch file2g
$ chmod 640 file2g
$ ls -l file2g
```
- Change those permissions to 777:
- ```
$ chmod 777 file2g
$ ls -l file2g
```
- and reset them again to 640 using `chmod` in symbolic mode:
- ```
$ chmod u=rw,g=r,o-rwx file2g
```
- (h) List all executables in '/usr/bin' with *setuid*:
- ```
$ ls -l /usr/bin/ | grep -E '^.{3}s'
```
- or alternatively:
- ```
$ find /usr/bin/ -perm -u=s | sort
```
- (i) Modify foregoing examples: list all executables in '/usr/bin' with *setgid* and then – all executables with both *setuid* and *setgid* bits set.

- (j) Check if `/usr/bin/su` has the *setuid* bit set:  
`$ ls -l /usr/bin/su`  
 Copy this file to your working directory:  
`$ cp /usr/bin/su .`  
 and check if copying preserves it:  
`$ ls -l su`  
 Unfortunately, it does not!  
 Cleanup:  
`$ rm su`
- (k) Create a file in the `/tmp` directory:  
`$ echo whatever > /tmp/`whoami``  
 Let your neighbor connect to your computer via SSH (assuming your computer's name is `p21801`, see the top cover or check your prompt):  
`$ ssh p21801`  
 Can he/she delete this file? No, they can't; it's due to the *sticky bit* set for `/tmp`.
- (l) Check your umask:  
`$ umask`  
 Set it to 000, create a new file and check its permissions:  
`$ umask 000; touch file2l1; ls -l file2l1`  
 then set it to 777, create another file and check its permissions:  
`$ umask 777; touch file2l2; ls -l file2l2`  
 Reset umask to its default value:  
`$ umask 022`
- (m) Perform some final cleanup:  
`$ rm file2*`

### 3. Access Control Lists

- (a) Find files in the `/dev` directory with ACL permissions set:  
`$ ls -l /dev/ | grep -P '^[^ ]*\+'`  
 Those ACLs allow desktop environment users to access some hardware/multimedia settings.
- (b) Display the ACL permissions (let's assume `/dev/rfkill` has been listed in the previous exercise):  
`$ getfacl /dev/rfkill`

*The following exercise may not work in our labs due to some NFS (network file system) bugs. Ask the teacher before proceeding.*

- (c) Grant file access right to your home directory to all users:  
`$ chmod +x ~`  
 Create a file in it and allow your neighbour to read and write to it:  
`$ touch file3c`  
`$ setfacl -m u:your_neighbour:rw file3c`  
 Display its permissions (mind the '+') and ACL permissions:  
`$ ls -l`  
`$ getfacl file3c`  
 Ask your neighbour to modify its contents:  
`$ echo "some text" > ~/../your_name/file3c`  
 Check if it's been changed:  
`$ cat file3c`  
 Remove ACL permissions to this file:  
`$ setfacl --remove-all file3c`  
 and reset your home directory rights to restrictive ones (see 2e.).