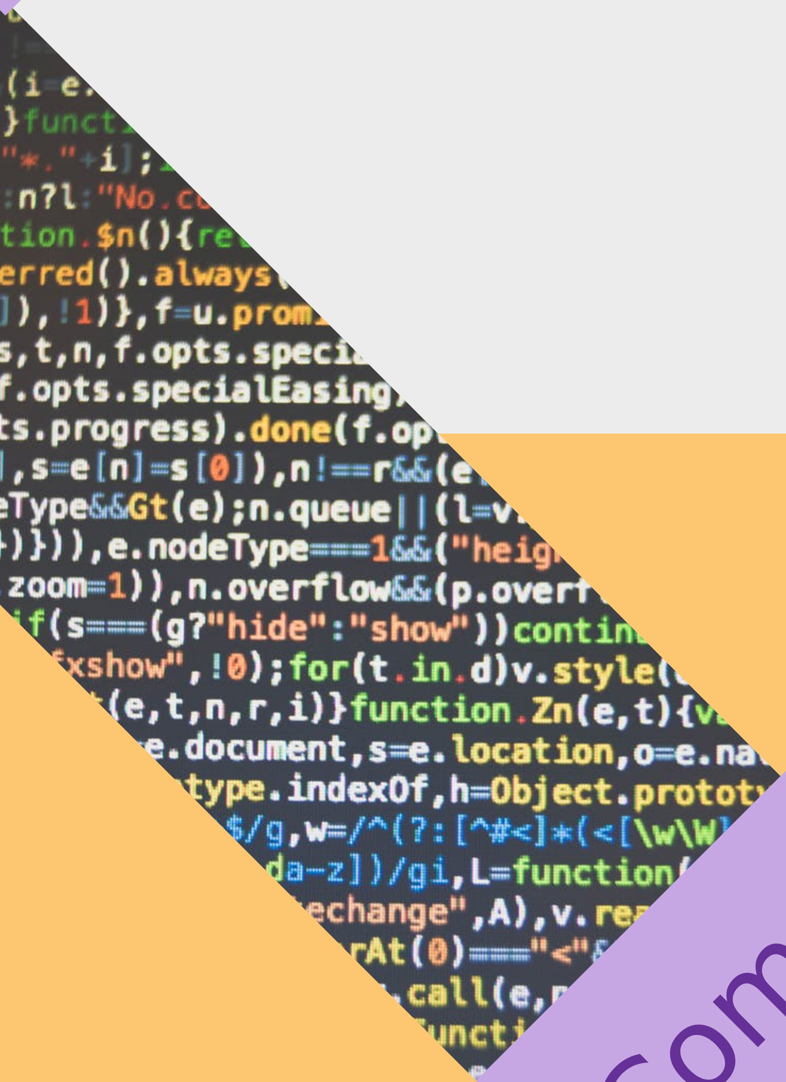
Curs 2020-2021



Computer Systems
Linux Admin.

IES Jaume II El Just
Tavernes de la Valldigna

Contents

1 Users and groups

In Linux, users are identified by a user number called UID (User ID). That number is different from any other UID number or other user. It is like the ID card number. That number identifies just one person and it can't be used again for another live person. Besides, each user should belong to a group, called primary groups. Moreover, the same user can belong to other groups called secondary groups:

- The primary group in most cases, has the same name as your login name. The primary group is used by default when creating new files (or directories), modifying files or executing commands.
- Secondary groups: these are groups that you are a member of outside of your primary group. It means that if some user belongs to some group as a secondary, that user will have the same (permission or accessibility) applied to the group of the file or directory.

A group is a set of users, but it is possible that a group has just one user. In the same way that user has a UID number, groups are identified by GID (Group ID). As you might have guessed, it is a different number for each group.

1.1 Kind of users

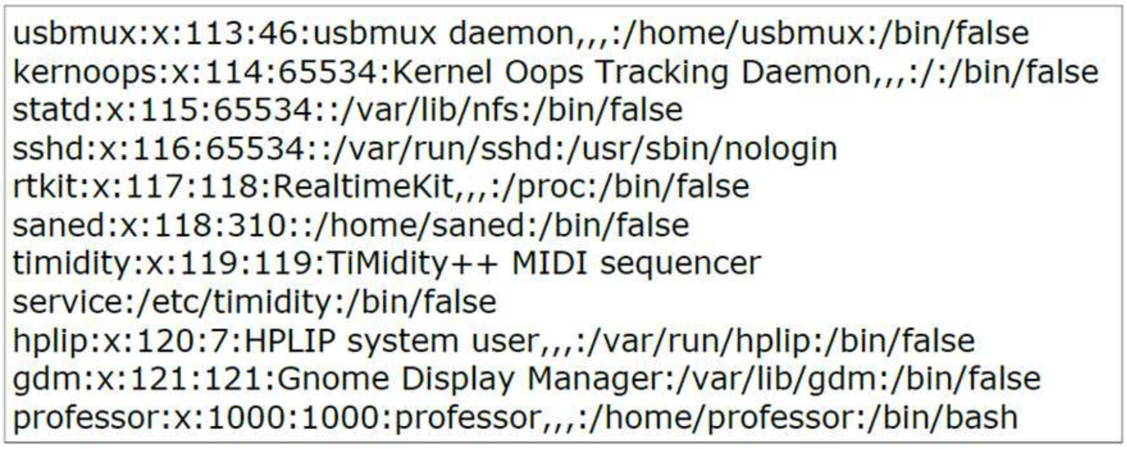
There are three different kinds of users in the Linux system:

- **Root user**: also known as superuser or administrator. It is the user that has all of the privileges, and it can do almost everything. It has total access of the system, and it is in charge of the administration, updating and maintenance; Its UID number is 0.
- **System users**: They are generated when the operating systems are installed or when some service is installed. Although they do not have all the privileges, they have some of them depending on the user. They aren't used in order to validate in a session, but they are necessary in order to execute some services or processes in the operating system. The number of UID is greater than 1 and lower than 1000, except the nobody user. The UID for that user is 65534 which is the last.
- **Regular users**: They are normal users which will connect to the system and they can start a session in the operating system. They have a work directory in the /home directory. They have all the permission in that directory. The UID assigned to these kinds of users is greater or equal to 1000, although it is possible to change it. When the operating system is installed, one of these kinds of user is created by default. A regular user has just access to the files of its property or those files or directories with the permission which somebody has given to it.

1.2 Configuration files

The configuration files of system administration are the files which are read or modified by the system when users or group are managed.

- **/etc/passwd**: it contains information about the user in each of its lines. The information is organized in fields. The delimiter of each field is the character ":". The structure of each line is the following information:



```
usbmux:x:113:46:usbmux daemon,,,:/home/usbmux:/bin/false
kernoops:x:114:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
statd:x:115:65534::/var/lib/nfs:/bin/false
sshd:x:116:65534::/var/run/sshd:/usr/sbin/nologin
rtkit:x:117:118:RealtimeKit,,,:/proc:/bin/false
saned:x:118:310::/home/saned:/bin/false
timidity:x:119:119:TiMidity++ MIDI sequencer
service:/etc/timidity:/bin/false
hplip:x:120:7:HPLIP system user,,,:/var/run/hplip:/bin/false
gdm:x:121:121:Gnome Display Manager:/var/lib/gdm:/bin/false
professor:x:1000:1000:professor,,,:/home/professor:/bin/bash
```

- **username**: it is the login of the user of that line. It is the first field.
- **password**: That field is the password of the user, but the ”x” means that the password is encrypted in the /etc/shadow file.
- **UID**: it is the UID number of the user
- **GID**: it is the GID number of the primary group of the user. That number is equal or greater than 0. Number 0 is the root GID number group.
- **User info**: field which contains information about the user. The delimiter of the information is “,” but it is possible that the field is empty. That field is called GECOS.
- **User home directory**: it is the absolute path of the personal directory of the user(personal home directory)
- **Command/Shell**: it is the absolute path of the login Shell used by the user by default. The most used is shell bash, although there are a lot.

• **/etc/shadow**: This file contains the encrypted passwords of each user. It is a very important and sensitive file. Therefore, root or other user with the same privileges can read it. Each time some user accesses the system, Linux checks the password in that file.

Some years ago, the encrypted password was used in order to be saved in the field password in the file /etc/passwd. That password was encrypted with DES (Data Encryption Standard) encrypted algorithm. That algorithm uses Linux crypt function. Due to safety reasons, it started to use that file (/etc/shadow) and more complex algorithm encryptions. The most used algorithms are: DES, MD-5, SHA-256 and SHA-512, Blowfish. Usually, the password format is set to $id$salt$hashed. The $id is the algorithm used on GNU/Linux as follows:

- DES if they are 13 characters
- 1 is MD5
- $2a$ is Blowfish
- $2y$ is Blowfish
- 5 is SHA-256
- 6 is SHA-512

The structure of each line is:

- **Username** : It is your login name.
- **Password**: encrypted password of each user.
- **lastchanged password** : Days since Jan 1, 1970 that password was last changed
- **Minimum** : The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
- **Maximum** : The maximum number of days the password is valid (after that user is forced to change his/her password)
- **Warn** : The number of days before password is to expire that user is warned that his/her password must be changed
- **Inactive** : The number of days after password expires that account is disabled
- **Expire** : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used. If that field is empty, account never expires

- **/etc/group**: File which contains the registered groups in the system. The structure if the file is:

  group:x:GID:users:

  Where each field contains the following information:

  - **Group**: it is the name of the group.
  - **X**: it is mostly used in the Unix system in order to save the group password. Nowadays, it is not used except in some cases, such as protecting some groups in order to prevent some user from gaining access.
  - **GID**: it is the number of the group.
  - **Users**: it is the list of the users with this as a secondary group. Each user will be delimited by "," In order to know the primary group of each user, it is necessary to check /etc/passwd file.

- **/etc/gshadow**: It is the file which group passwords are saved- Although passwords are not used, it is necessary to have the file.
- **/etc/default/useradd**: in this file, it is possible to find the default values in order to add some user with useradd command.
- **/etc/adduser.conf**: it contains the default values in order for the user to be added to the system with adduser command.
- **/etc/deluser.conf**: it contains the values by default when some user is erased with the deluser command.
- **/etc/login.defs**: File which contains information with some values about the encryption of the password and other parameters of the user creation.
- **/etc/shells**: it is a list of the valid shells.

1.3 Users and group managing

File editor

In order to watch the settings files, it is possible to use commands like cat, more, less, head or tail. But in order to modify them, it is necessary to use an editor. In text mode (through terminal) nano is used, vi or some other one.

1.3.1 Commands in order to change the user or to run commands with administrator privileges

During the installation of Linux (Ubuntu distributions or Mint), a user is created, that user is not root, but it is in the list called "sudoers". In this list it is possible to give privileges to regular users, and the user created during the installation is one of them. This way it is possible to execute some instruction as root was running them.

On the other hand, the root account is deactivated, which is to say, it is not possible to login as root, unless a password is given to root by the user with privileges.

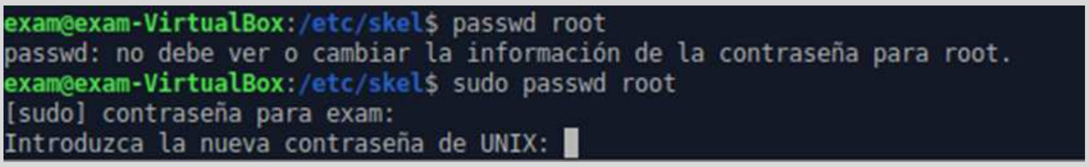
So, in order to run commands with privileges, it is mandatory to use the command "sudo" before the command.

Example

Change the password to root user from your own user.

sudo passwd root

This way password command (command in order to change password) is run as root by the user logged into the system. It is presumed that the user logged in is on the sudoers list



```
exam@exam-VirtualBox:/etc/skel$ passwd root
passwd: no debe ver o cambiar la información de la contraseña para root.
exam@exam-VirtualBox:/etc/skel$ sudo passwd root
[sudo] contraseña para exam:
Introduzca la nueva contraseña de UNIX:
```

In the last picture, password command is run without privileges, being an exam user. Then an error reported. So, in the second chance, sudo is written before the command and the command works.

Awhile after introducing a password in order to execute the command with privileges, the following commands will be executed as a root for five minutes.

Change time to user sudo

In order to change the time that the user can execute commands with that privilege, the /etc/sudoers file should be modified.

sudo /etc/sudoers

And then, this line should be added into it

Defaults timestamp_timeout=15

If establishing a different time is desired for a specified user, it should be added to the name of the user

Defaults:user_name timestamp_timeout=15

Where user_name is the name of the specified user .

On the other hand, the command to change the user is su:

- Command “su” is an abbreviation of “Switch User”.
- Two options are left:
  - Without parameters: in this case, it tries to log in as root. It works even if root account is disabled.
  - With parameter: it has a parameter that is the username that you want to log in.

If you run the command as root, it automatically logs you in as the user. If you are a normal user, it asks you for the user password.

1.3.2 Commands to Manage user and groups

- **adduser** ⇒ Add users to the system. Also, it is used in order to add some user to an existing group.

  Syntax:

  ```
  adduser [options] user
  adduser [options] user group
  ```

  Options:

  - --ingroup namegroup ⇒ User will be created with the primary group specified.
  - --gid num_gid ⇒ It is the same option as before but this time, GID number is used instead of the name group.
  - --home directoryname ⇒ in order to change the user home directory instead of the default name of it. It is normally /home/user_name.

- **useradd** ⇒ that command does the same although the password should be indicated, otherwise use an exclamtion mark (!) and is added in /etc/shadow file. That means that user is in locked state; so, it is necessary to set a password for that account with passwd command. And the same with the personal directory; it is required to create a personal directory. It is possible to create it with the command options.

  Syntax:

  ```
  adduser [options] username
  ```

  Options:

  - -d ⇒ specify the directory for the user
  - -m ⇒ create the personal directory.
  - -e ⇒ the date when the user account expires. The format of the date should be YYYYMMDD
  - -g ⇒ name of the primary group

- -G ⇒ name of secondary groups of the new account. The delimiter for each secondary group is ","

It is possible to check for more information in useradd --help.

Example:

```
useradd -d /home/mydirectory -m newuser
```

- **deluser** ⇒ to erase a user. The userdel command exists. Bear in mind that it is not possible to erase a user who is connected at that time.

  Syntax:

  ```
  deluser [options] user
  ```

  Options:

  - --remove-home ⇒ it deletes the personal directory of the user (with userdel command the option which does the same would be -r)
  - --remove-all-files ⇒ All the files which belong to the user and the personal directory will be deleted.

- **addgroup** ⇒ that command adds a group to the system. The same result would be achieved with the command adduser and the –group option:

  Syntax:

  ```
  addgroup grupo
  ```

- **delgroup** ⇒Delete a group. The same result would be obtained with the command deluser --group grupo. Bear in mind that the group which is supposed to be deleted, should not be a primary group of any user.

  Sintax:

  ```
  delgroup grupo
  ```

- **usermod** ⇒ that command modifies an existing user.

  Sintax:

  ```
  usermod [options] user
  ```

  Options:

  - -d directory [-m] ⇒ change the personal directory
  - -m ⇒ move the contents of the old personal directory to the new directory. It is possible to use it just with –d option.
  - -g group ⇒change the primary group of the user.
  - -u uid ⇒ change the uid number of the user.
  - -l name ⇒ change the name of the user.

  - -s shell ⇒ change the shell

- **groupmod** ⇒ in order to modify an existing group

  Sintax:

  ```
  groupmod [options] group
  ```

  Options:

  - -n name directory [-m] ⇒ change the name group.
  - -g gid ⇒change the GID of the group

- **gpasswd** ⇒ it is used in order to add or delete a user to some group. It is used to establish a password to a group.

  Sintax:

  ```
  gpasswd [options] user group
  gpasswd group
  ```

  Options:

  - -a ⇒ add user to the group
  - -d ⇒delete user from the group.
  - -M ⇒ add several users to the group.

- **id** ⇒ information of the user is displayed; for example, UID, name, groups to which it belongs and GID of them.

  Sintax:

  ```
  id [options] [users]
  ```

  Options:

  - -u ⇒ --user ⇒ display just the UID of the user.
  - -n ⇒ --name⇒ display just the name of the user instead of the UID

- **groups** ⇒ display the name of the groups to which the user belongs.

  Sintax:

  ```
  groups [user]
  ```

- **finger** ⇒ It displays the information about the specified user. Without no specified user, it shows information about the connected users.

  Sintax:

  ```
  finger [user]
  ```

- **chfn** ⇒ That command is used in order to change the information of the GECOS field of the /etc/passwd file.

  Sintax:

  ```
  chfn [options] user
  ```

  Options:

  - -f name ⇒ change the name
  - -r ⇒ office_number ⇒ change the office number
  - -w phone ⇒ change the work phone number
  - -h phone ⇒ change the home phone number
  - -o other ⇒ change some other additional information.

- **newgrp** ⇒It is used in order to change the primary group of a user. That command works till the session is finished or the primary group is changed again. If the task involves changing to some other group which the user doesn't belong, the group password will be asked.

  Sintax:

  ```
  newgrp grupo
  ```

- **chsh** ⇒ That command is used in order to change the user Shell. If no user is specified, the command is executed about the user who runs the command.

  Sintax:

  ```
  chsh [options] [user]
  ```

  Options:

  - -l ⇒ used to specify your login shell.
  - -u ⇒ prints the list of shells
  - -s ⇒ used to set the shell as your login shell.

Try it

From your terminal:

1. Create a new group, called newgroup.
2. Afterwards, add a user called newuser.
3. The primary group of it is newgroup.
4. Check the result.
5. Create another group called group2.
6. Add newuser to this group as a secondary group
7. Check the result

8. Delete the created groups and the users

```
sudo addgroup newgroup
sudo adduser --ingroup newgroup newuser
id newuser
su newuser (log in as a newuser in order to check if it has been created)
whoami
exit
sudo adduser --group group2
sudo adduser usunuevo group2
cat /etc/group (check the proper file)
groups newuser (check the groups that newuser belongs)
su newuser
exit
sudo deluser --remove-home newuser
sudo delgroup group2
sudo deluser --group newgroup
```

1.3.3 Commands in order to change files and directories of user and group

As it is known, each file and directory have an owner and group owner. So, the following commands are used in order to change the owner or group owner of the directories and files.

- **chown** ⇒ Change the owner of the file. It is possible to use it in order to change the group as well.

  Sintax:

  ```
  chown [options] [owner[:grupo]] file/s
  ```

  Options:

  – -r -recursive ⇒ that option is used in order to change the owner and group owner to a tree of directories.

- **chgrp** ⇒ change the group owner of the file.

  Sintax:

  ```
  chgrp [options] group file/s
  ```

  Options:

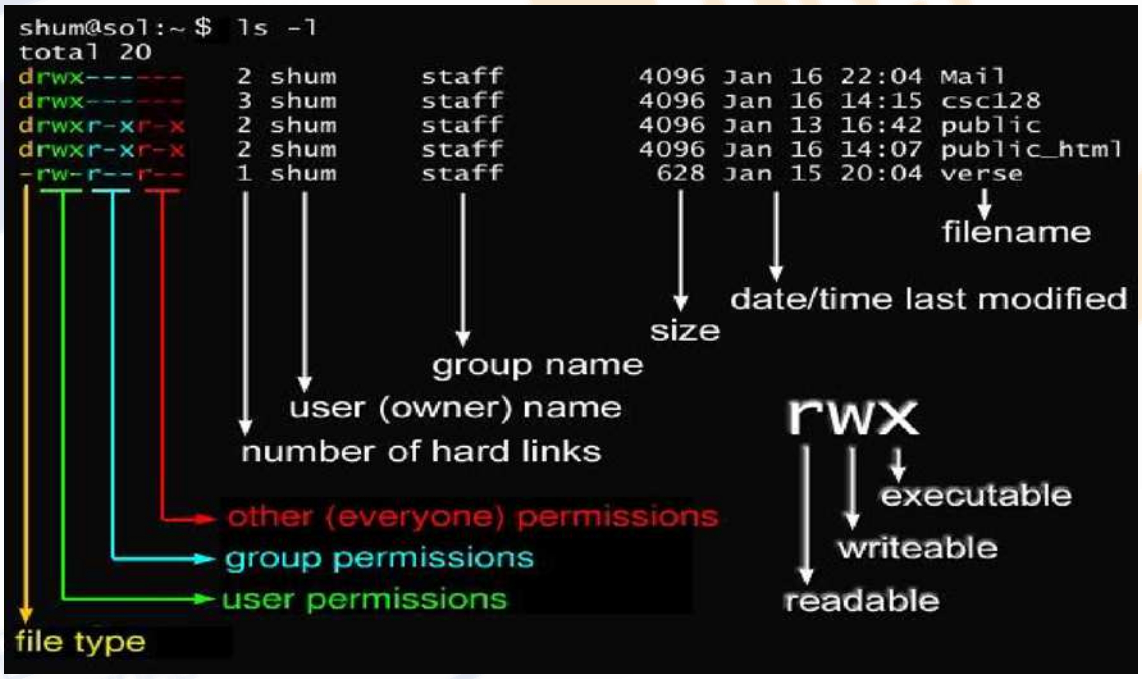
  – -r --recursive ⇒ change the group owner to a tree of directories

2 File and directory Permissions

In Linux there are different users and groups, and each file or directory should belong to some user and group. Thus, it is necessary to establish permissions. The permissions will be established in 3 groups of 3 kinds of permissions. The 3 kinds of permissions will be read, write and execute. Depending on the permissions which are applied to a file or directory it will have a different meaning:

- Read:
  - In a file: allows to read its content.
  - In a directory: allows to list its files, directories names and attributes (command ls).
- Write:
  - In a file: you can modify content of the file.
  - In a directory: you can delete or create files and directories in that directory.
- Execute:
  - In a file: you can run the file (like Windows ".exe").
  - In a directory: you can enter the directory (cd command).

As it is known, command ls with the option –l displays several pieces of information about the directories and files contained in the current directory. In the first field it shows the permissions of each files and directories. The first character refers to what kind of files (regular, link, etc. . .) or directory it is. The following 9 characters refers to the permissions:



```
shum@sol:~$ ls -l
total 20
drwx------  2 shum    staff     4096 Jan 16 22:04 Mail
drwx------  3 shum    staff     4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum    staff     4096 Jan 13 16:42 public
drwxr-xr-x  2 shum    staff     4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum    staff      628 Jan 15 20:04 verse
```

The three groups refer to user permissions (the first three), group permissions (the second group of three) and others (the third group of three). Others are the rest of the user of the system.

When the permission is indicated with the letter (r (read), w (write), x(execution)) means that this permission is given. Otherwise "–" is shown in order to point out that this permission is forbidden.

- **chmod** ⇒ Change the permission of the file

  Sintax:

```
chmod [permissions [,permissions]...] file/s
```

Permissions:

The permission can be added or deleted with the following expressions:

- chmod {a,u,g,o}{+ | - | =}{r,w,x,X,s,t} files
  - {a,u,g,o} ⇒ The first letter refers to who is going to change the permission:
    - a ⇒ all, option by default.
    - u ⇒ user
    - g⇒ group
    - o ⇒ others.
  - {+ | - | =}
    - ⇒ + in order to add permissions
    - - ⇒ in order to remove permissions-
    - = ⇒ in order to set the permission. It takes into account who sets the permission and overrides the permissions set earlier.
  - = ⇒ in order to set the permission. It takes into account who sets the permission and overrides the permissions set earlier.
  - {r,w,x,X,s,t} ⇒ In order to set what permissions is stablished:
    - r ⇒ read
    - w ⇒ write
    - x⇒ execution
    - X ⇒ execution in case of directories
    - s ⇒ special permission Seguid bit or segid bit.
    - t⇒ sticky bit

On the other hand, the permissions can be established as a numeric way in octal base. The values will be added with each of the 3 digits and each permission has a value: permissions:

- 0 = any permission
- 1 = execute permission (x)
- 2 = write permission (w)
- 4 = read permission (r)

In order to establish permission, the values will be added.

Exemple

If it is wanted to establish the following permissions:

rwx-w-r-x ⇒ 111 010 101 (binary) ⇒ 725 (octal)
rwx-w-r-x ⇒

- first group of permissions ⇒ r+w+x = 4+2+1 = 7
- second group of permissions ⇒ w = 2
- third group of permission ⇒ r+x = 4+1 = 5

- **umask** ⇒ change or show the permissions which will be assigned by default when a directory or file is created. This is temporary.

  Sintax:

  ```
  umask [mask]
  ```

  Arguments:

  That mask of permissions will have different consequences depending on a file or directory that is created.

  There are some base permissions for the files (666) and base permissions for the directories (777)

  Due to that:

  - when a directory is created, the established mask will be subtracted from the base.
  - When a file is created, the following operation will be applied:

    Base And not (mask)

    That means that the last bit never will be activated

Exemple

if umask is 033, ¿With what permission would files and directories be created?

Solutions:

- Directories

  033 octal binary ⇒ 000 011 011 Subtrack 777-033 ⇒ 111 111 111 -000 011 011 ⇒ 111 100 100 = rwx r– r–

  In octal 744.

  Or simply: 777 − 033 = 744

- Files:

  base AND (NOT umask) ⇒ 666 AND (NOT 033) ⇒ 666 AND 744 110 110 110 AND 111 100 100 = 110 100 100 = rw- r– r–

  In octal 644.

Exemple

1. Check the umask value
2. Check the permissions of the directories and files of your personal directory.
3. Change umask to 000
4. Create a directory called newdirect and an empty file called newfile
5. Check the permissions
6. Change to have all of your permissions (rwx), the group member can read and execute, and others can have any permission.

Solutions:

```
umask
ls -l
umask 000
mkdir newdirect
ls > newfile
ls -ld new*
chmod 750 newdirect
chmod u+x,g-w,o-rw newfile
```