

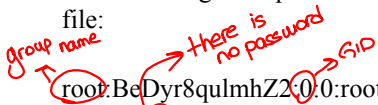
User Management

Any multi-user system must maintain a database of users, in this database, generally of a relational type, there is a record for each user.

Each user has several properties associated with him, to manage the users and the whole system in general there is a "super-user" or administrator, in Unix systems there is a single administrator who has the name "root".

In Unix-type systems, users are defined in the `/etc/passwd` file, in this text file, each line corresponds to a user, consisting of a sequence of fields, separated by colons, all fields are mandatory, but some may be empty.

The following example illustrates an 8-line (8 users) extract from a `/etc/passwd` file:



```
root:BeDyr8qulmhZ2:0:0:root:/root:/bin/bash
daemon:*:2:2:daemon:/sbin:/bin/bash bin:*:1:1:bin:/bin:/bin/bash
postgres:*:26:2:Postgres Database Admin:/var/lib/pgsql:/bin/bash
wwwrun:*:30:65534:Daemon user for apache:/tmp:/bin/bash
empres:*:35:2:Empress Database Admin:/usr/empres:/bin/bash
guest:a28HqK3Yamh7t:1001:102:Utilizador nosso
convidado:/home/guest:/bin/csh user:uHr5fg6RtEw23:1002:102:Utilizador
local:/home/user:/bin/bash +
```

username → salt + hash → UID → GID → User full name → working directory location → initial shell name

The fields are, in order, as follows:

- "Login Name" or "Username" - is the name that the user uses to identify himself to the system, so it must be unique in the system.
- SALT+HASH(SALT+PASSWORD) - is the result of applying a "hash" function to the user's password (together with SALT which introduces a random factor). It is used for user authentication, for this purpose the user provides his "Username" and respective PASSWORD, the system then calculates HASH(SALT+PASSWORD) and checks if it matches what is in the /etc/passwd file. In this field, separated by commas, there can still be several values referring to the validity of the password, see the file /etc/shadow.
- User Identifier (UID) - is an integer used to identify the user within the system, as the "Username" must be unique, however unlike the "Username" which only appears in the /etc/passwd file and is used just for initial identification to the system, the UID is used extensively in internal post-login operations. As you can see the UID of the "root" user is zero, and should exist on all Unix-like systems.
- Group Identifier (GID) - integer that identifies the primary group to which the user belongs, in Unix user groups are defined in the /etc/group file, with each group having a unique identification number, the GID.
- User's full name
- Working directory location - this is the name of the working directory to be assigned upon entering the system, typically it is a home directory to which only that user has permissions, usually called HOME.
- Initial shell name - when the user logs into the system (after authentication) control of the session has to be transferred to a program that will be run already with that user's permissions, typically this is an interactive text-based command interpreter program, which has the designation of "shell".

User groups

The definition of user groups is very advantageous from the point of view of administration because when you want to assign many users a given right (permission) it is possible to assign that right to the group, with the effect that all users who are members of the group they also begin to enjoy this right.

On Unix systems the groups are defined in the `/etc/group` file, this file has a similar structure to `/etc/passwd`, the fields are as follows:

↳ where the users are defined

- The 1st field defines the **group name**, as for users this field must have a unique value.
- Usually no passwords for groups are used, so the 2nd field has a "*" or an "x".
- The 3rd field contains the **GID**, as in `/etc/passwd` for UIDs, GIDs must be unique in the system.
- The 4th and last field has a list of users (separated by comma) that belong to the group, this is only necessary for users who do not have the group mentioned as the primary group in `/etc/passwd`.

The following is an extract from a `/etc/group` file:

group name → password
root:x:0:root → list of users
bin:x:1:root,bin,daemon
daemon:x:2:
users:x:102:
nogroup:x:65534:root

It can be seen that although the group "users" has no members defined, in reality, by crossing the previous example `/etc/passwd`, the users "guest" and "user" are members.

Another aspect to highlight is the existence of a group called "root", if users are placed in this group, they will have many of the rights that the administrator has over system objects, since this is the primary group of the administrator.

→ primary of group of administrator

Shadow passwd

One of the problems with the `/etc/passwd` file is that it has to be readable by all users. Since the HASH of the password is in the file, it is possible to try to "guess" a "password" that generates that HASH, possibly it will not be the same "password" that the user uses, but it will serve as authentication. Breaking the authentication mechanism in this way is extremely difficult and time-consuming, the approach that is used in practice is to use a "password generator" program, based, for example, on a dictionary and try those "passwords".

Regardless of how access to the user's password hash is clearly a weak point, the authentication attempt process that should be controlled by the system (controlling the maximum number of attempts and the delay between attempts) can be simulated in On another system, just copy the "hash" in `/etc/passwd`.

To solve this security problem, there is the possibility to use a shadow file to store the hash of the "passwords", this file is `/etc/shadow`, but unlike the file `/etc/passwd` it is only readable by the "root" user. The designation "shadow" (shadow) is due to the fact that it contains exactly the same records as `/etc/passwd`.

The `/etc/shadow` file has a similar structure to `/etc/passwd`, but the fields are those provided for the password field in the `/etc/passwd` file. The fields, in order, are as follows:

- "Login Name" or "Username" - is the name that the user has exactly the same as the one found in the `/etc/passwd` file.
- SALT+HASH(SALT+PASSWORD) - is the result of applying a hash function to the user's password, as described for the `/etc/passwd` file, its entry in the `/etc/passwd` file is now passed to contain an "x".
- Date of last password change - specified in number of days since 01/01/1970.
- Number of days left for the user to change the password.
- Number of days left before the user is required to change the password (password expires).
- Number of days before being forced to change the password in which the user starts to be warned.
- Number of days after the password expires that the account is deactivated.
- Date the account was deactivated - specified in number of days since 01/01/1970.
- Unspecified field, reserved for future use.

Unix user database management

There are several programs that can be used to manage users, here I will just refer to the basic programs, which should be available on all types of Unix systems, then there are many other programs that use the first ones and that are specific to each particular system. , many of them with graphical interface.

In fact, to manage users in Unix all you need is a text editor, preferably "vi", the only field that cannot be edited directly is the "hash" of the password, for that it is necessary to use the "passwd" program.

On systems where "shadow passwords" are used, after manually modifying /etc/passwd it is still necessary to invoke the "pwconv" program.

The "pwconv" creates in the current directory a file "npasswd" and an "nshadow" which correspond to the new versions of the files /etc/passwd and /etc/shadow after the merging of the information.

Users can change some aspects of their account, the administrator can change any account, for this purpose the following programs check if whoever is running them is the administrator or not:

- passwd - allows changing the user's password, it also allows the administrator to manage the supplementary parameters of the /etc/shadow file.
- chfn ("Change full-name") - allows you to change the user's full name.
- chsh ("Change shell") - allows changing the user's initial program.

Generally all systems have more sophisticated programs to manipulate users, the use of these programs is generally reserved for the administrator:

- useradd - allows adding new users, automatically searches for an unused UID and creates the "home directory", users are created according to a "template", or by specifying the different features on the command line.
- userdel - allows you to remove a user and can also remove the "home directory".
- usermod - allows you to change any data related to a user, including the "username" and even the UID, as well as groups to which it belongs.
- groupadd - creation of new user groups.
- groupmod - modification of user groups (group name and GID).
- groupdel - deletion of user groups.

The use of specific programs is more reasonable than manual editing, these specific programs guarantee mutual exclusion for writing, that is, they guarantee that there are never two programs simultaneously changing the users' databases, with the unpredictable results that can result from this, on the other hand, manual editing can, due to the administrator's mistake, lead to the files not respecting the formatting or not being coherent.

As far as the users' database is concerned, these errors can be very serious.

Centralized User Databases – NIS

With the rapid evolution of computer networks, largely driven by Unix operating systems, it became absolutely necessary to create centralized systems for these databases that would avoid independent management for each machine on a network, the most traditional in a Unix environment is the system known as "Yellow Pages" (yp) or NIS ("Network Information Service").

This system consists of a "ypserv" server, installed on a central machine by "ypbind" clients installed on other machines. The various clients (ypbind) contact the server to obtain maps, which are no more than equivalent to the usual contents of /etc/passwd, /etc/group and others, such as /etc/hosts or /etc/services.

The server (ypserv) does not directly use local files to respond to clients, there are programs (usually invoked via the "make" command in the /var/yp directory) that convert information residing in local files (here seen as source files) to a format suitable for the server.

The source files may or may not be the versions in use locally on the server (located in the /etc directory), if they are used as a source of local versions there is a great advantage that the above mentioned user management programs can be used, it is only necessary don't forget to call the "make" command in the /var/yp directory after making the changes to update the NIS database. The disadvantage of this option is that system users such as root, daemon, field, ... will also be placed in the database.

The option for separate files as NIS source (in a directory other than /etc) implies the need to use specific programs for user management that allow the manipulation of files in other directories, these programs do not abound.

NIS clients use the ybind program directed to the NIS server, so that when authenticating users the NIS server is consulted, the last line of the local files (/etc/passwd, /etc/group and /etc/shadow) must contain the "+" sign, which means to add all the information from the NIS server (it is also possible to add individual NIS users, using lines containing "+username". On NIS clients it is even possible to use various commands such as "ypcat" to query the various maps provided by the NIS server, or "ypmatch" to search the maps.

The NIS server (ypserv) is read-only, so that users can change their data "password/shell/full-name" uses a separate service, as well as "ypserv" installs another server, the "yppasswdd" on the NIS client side now uses the yppasswd, ypchsh and ypchfn commands which contact yppasswdd on the NIS server.

Support for multiple data sources – nsswitch

With the introduction of NIS, customers now have two sources for user definitions, local databases (in the /etc directory) and NIS databases.

The way in which this is signaled is by placing "+" signs in the local versions, which means that at that point in the local search, a search must be carried out on the NIS server.

For the case of hostname resolution we can even have 3 different sources for the data: local (/etc/hosts), NIS or DNS, which systems are used and in what order they are specified in the /etc/host.conf file .

More recent Linux systems significantly expand this idea, generalizing support for multiple sources for system information, namely regarding users. The NSS system ("Name Service Switch"), uses the information contained in the /etc/nsswitch.conf file to define for each type of information, which sources to search and in what order.

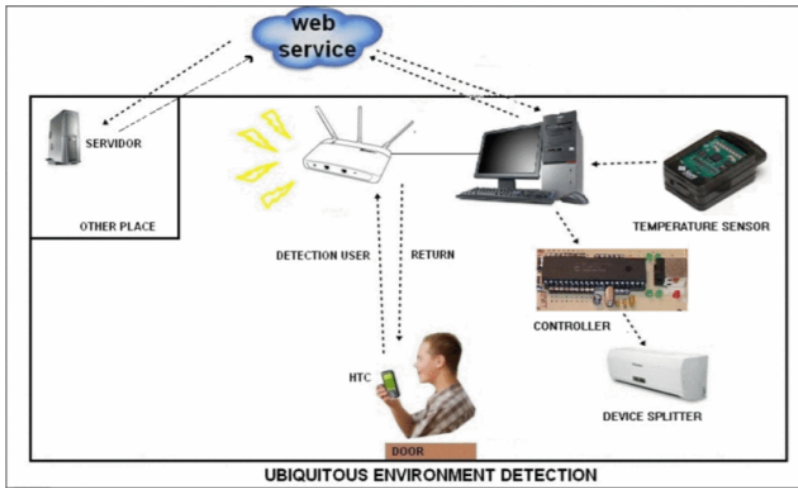
NSS allows distinguishing the following types of system information: aliases, ethers, group, hosts, netgroup, network, passwd, protocols, publickey, rpc, services and shadow. For each of the types of information, the sources to be searched and the order in which they are searched can be defined.

The possible sources for the search are defined in a modular way, for each type of source the appropriate library must be installed, usually with the type designation `libnss_SERVICE.so`, where `SERVICE` is replaced by the designation of the database type that can later be used in `/etc/nsswitch.conf`. Some examples are:

- `libnss_files.so` - local databases (`/etc`)
- `libnss_compat.so` - local or NIS databases (compatibility with previous systems)
- `libnss_nis.so` - NIS servers
- `libnss_dns.so` - DNS servers
- `libnss_ldap.so` - LDAP servers (X.500)
- `libnss_winbind.so` - Microsoft servers (NT/2000) - Samba suite

The NSS allows transparent access to various sources of system information, however when it comes to information regarding authentication everything gets complicated, in the case of NIS, passwords are stored in exactly the same way as in local versions, but if we want to generalize to other types of system databases will certainly need to support other formats. Remember that even for NIS it was necessary to create a separate service (`yppasswdd`) to allow changing passwords.

To solve the problem of authentication with different types of databases, the concept of "Pluggable Authentication Module" (PAM) was created, it is a modular system that, by adding the appropriate libraries, usually residing in the `/lib/security` directory, allows programs to use the various authentication systems. The configuration files residing in the `/etc/pam.d/` directory allow you to define for each application which authentication modules to use, such as `pam_ldap`, `pam_winbind`, etc.



Users

→ Admin

→ root

→ Users

→ Server 1 -> N

→ Desktop 1 → N

→ Devices N