Users and groups

Users and groups are used on GNU/Linux for **access control**—that is, to control access to the system's files, directories, and peripherals. Linux offers relatively simple/coarse access control mechanisms by default. For more advanced options, see **ACL**, **Capabilities** and **PAM#Configuration How-Tos**.

Related articles

- **DeveloperWiki:UID / GID Database**
- **Sudo**
- **Polkit**
- **File permissions and attributes**
- **systemd-homed**
- **Reset lost root password**
- **Identity management**

1 Overview

A *user* is anyone who uses a computer. In this case, we are describing the names which represent those users. It may be Mary or Bill, and they may use the names Dragonlady or Pirate in place of their real name. All that matters is that the computer has a name for each account it creates, and it is this name by which a person gains access to use the computer. Some system services also run using restricted or privileged user accounts.

Managing users is done for the purpose of security by limiting access in certain specific ways. The **superuser** (root) has complete access to the operating system and its configuration; it is intended for administrative use only. Unprivileged users can use several programs for controlled **privilege elevation**.

Any individual may have more than one account as long as they use a different name for each account they create. Further, there are some reserved names which may not be used such as "root".

Users may be grouped together into a "group", and users may be added to an existing group to utilize the privileged access it grants.

> **Note**
>
> The beginner should use these tools carefully and stay away from having anything to do with any other *existing* user account, other than their own.

2 Permissions and ownership

From **In UNIX Everything is a File (https://web.archive.org/web/20190816135930/http://ph7spot.com/musings/in-unix-everything-is-a-file)**:

> The UNIX operating system crystallizes a couple of unifying ideas and concepts that shaped its design, user interface, culture and evolution. One of the most important of these is probably the mantra: "everything is a file," widely regarded as one of the defining points of UNIX.
>
> This key design principle consists of providing a unified paradigm for accessing a wide range of input/output resources: documents, directories, hard-drives, CD-ROMs, modems, keyboards, printers, monitors, terminals and even some inter-process and network communications. The trick is to provide a common abstraction for all of these resources, each of which the UNIX

> fathers called a "file." Since every "file" is exposed through the same API, you can use the same set of basic commands to read/write to a disk, keyboard, document or network device.

From **Extending UNIX File Abstraction for General-Purpose Networking (https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=bb95638cdcae4f1a88354734f96a868704bb6684)**:

> A fundamental and very powerful, consistent abstraction provided in UNIX and compatible operating systems is the file abstraction. Many OS services and device interfaces are implemented to provide a file or file system metaphor to applications. This enables new uses for, and greatly increases the power of, existing applications — simple tools designed with specific uses in mind can, with UNIX file abstractions, be used in novel ways. A simple tool, such as cat, designed to read one or more files and output the contents to standard output, can be used to read from I/O devices through special device files, typically found under the `/dev` directory. On many systems, audio recording and playback can be done simply with the commands, "`cat /dev/audio > myfile`" and "`cat myfile > /dev/audio`," respectively.

Every file on a GNU/Linux system is owned by a user and a group. In addition, there are three types of access permissions: read, write, and execute. Different access permissions can be applied to a file's owning user, owning group, and others (those without ownership). One can determine a file's owners and permissions by viewing the long listing format of the **ls** command:

```
$ ls -l /boot/
```

```
total 13740
drwxr-xr-x 2 root root    4096 Jan 12 00:33 grub
-rw-r--r-- 1 root root 8570335 Jan 12 00:33 initramfs-linux-fallback.img
-rw-r--r-- 1 root root 1821573 Jan 12 00:31 initramfs-linux.img
-rw-r--r-- 1 root root 1457315 Jan  8 08:19 System.map26
-rw-r--r-- 1 root root 2209920 Jan  8 08:19 vmlinuz-linux
```

The first column displays the file's permissions (for example, the file `initramfs-linux.img` has permissions `-rw-r--r--`). The third and fourth columns display the file's owning user and group, respectively. In this example, all files are owned by the *root* user and the *root* group.

```
$ ls -l /media/
```

```
total 16
drwxrwx--- 1 root vboxsf 16384 Jan 29 11:02 sf_Shared
```

In this example, the `sf_Shared` directory is owned by the *root* user and the *vboxsf* group. It is also possible to determine a file's owners and permissions using the stat command:

Owning user:

```
$ stat -c %U /media/sf_Shared/
```

```
root
```

Owning group:

```
$ stat -c %G /media/sf_Shared/
```

```
vboxsf
```

Access rights:

```
$ stat -c %A /media/sf_Shared/
```

```
drwxrwx---
```

Access permissions are displayed in three groups of characters, representing the permissions of the owning user, owning group, and others, respectively. For example, the characters `-rw-r--r--` indicate that the file's owner has read and write permission, but not execute (`rw-`), whilst users belonging to the owning group and other users have only read permission (`r--` and `r--`). Meanwhile, the characters `drwxrwx---` indicate that the file's owner and users belonging to the owning group all have read, write, and execute permissions (`rwx` and `rwx`), whilst other users are denied access (`---`). The first character represents the file's type.

List files owned by a user or group with the **find** utility:

```
# find / -group groupname
```

```
# find / -group groupnumber
```

```
# find / -user user
```

A file's owning user and group can be changed with the **chown** (change owner) command. A file's access permissions can be changed with the **chmod** (change mode) command.

See **chown(1) (https://man.archlinux.org/man/chown.1)**, **chmod(1) (https://man.archlinux.org/man/chmod.1)**, and **Linux file permissions (https://www.linux.com/tutorials/understanding-linux-file-permissions)** for additional detail.

3 Shadow

The user, group and password management tools on Arch Linux come from the **shadow (https://archlinux.org/packages/?name=shadow)** package, which is a dependency of the **base (https://archlinux.org/packages/?name=base)** **meta package**.

4 File list

> **Warning**
>
> Do not edit these files by hand. There are utilities that properly handle locking and avoid invalidating the format of the database. See **#User management** and **#Group management** for an overview.

File	Purpose
/etc/shadow	Secure user account information
/etc/passwd	User account information
/etc/gshadow	Contains the shadowed information for group accounts
/etc/group	Defines the groups to which users belong

5 User management

To list users currently logged on the system, the *who* command can be used. To list all existing user accounts including their properties stored in the **user database**, run `passwd -Sa` as root. See **passwd(1) (https://man.archlinux.org/man/passwd.1)** for the description of the output format.

To add a new user, use the *useradd* command:

```
# useradd -m -G additional_groups -s login_shell username
```

-m / --create-home
: the user's home directory is created as `/home/username`. The directory is populated by the files in the skeleton directory. The created files are owned by the new user.

-G / --groups
: a comma separated list of supplementary groups which the user is also a member of. The default is for the user to belong only to the initial group.

-s / --shell
: a path to the user's login shell. Ensure the chosen shell is installed if choosing something other than **Bash**. The default shell for newly created user can be set in `/etc/default/useradd`.

Warning

In order to be able to log in, the login shell must be one of those listed in `/etc/shells`, otherwise the **PAM** module **pam_shells(8) (https://man.archlinux.org/man/pam_shells.8)** will deny the login request.

Note

The password for the newly created user must then be defined, using *passwd* as shown in **#Example adding a user**.

If an initial login group is specified by name or number, it must refer to an already existing group. If not specified, the behaviour of *useradd* will depend on the `USERGROUPS_ENAB` variable contained in `/etc/login.defs`. The default behaviour (`USERGROUPS_ENAB yes`) is to create a group with the same name as the username.

When the login shell is intended to be non-functional, for example when the user account is created for a specific service, `/usr/bin/nologin` may be specified in place of a regular shell to politely refuse a login (see **nologin(8) (https://man.archlinux.org/man/nologin.8)**).

See useradd(8) (https://man.archlinux.org/man/useradd.8) for other supported options.

5.1 Example adding a user

To add a new user named `archie`, creating its home directory and otherwise using all the defaults in terms of groups, directory names, shell used and various other parameters:

```
# useradd -m archie
```

Although it is not required to protect the newly created user `archie` with a password, it is highly recommended to do so:

```
# passwd archie
```

The above *useradd* command will also automatically create a group called `archie` and makes this the default group for the user `archie`. Making each user have their own group (with the group name same as the user name) is the preferred way to add users.

You could also make the default group something else using the `-g` option, but note that, in multi-user systems, using a single default group (e.g. `users`) for every user is not recommended. The reason is that typically, the method for facilitating shared write access for specific groups of users is setting user **umask** value to `002`, which means that the default group will by default always have write access to any file you create. See also **User Private Groups (https://web.archive.org/web/20210724233134/https://security.ias.edu/node/441)**. If a user must be a member of a specific group specify that group as a supplementary group when creating the user.

In the recommended scenario, where the default group has the same name as the user name, all files are by default writeable only for the user who created them. To allow write access to a specific group, shared files/directories can be made writeable by default for everyone in this group and the owning group can be automatically fixed to the group which owns the parent directory by setting the **setgid** bit on this directory:

```
# chmod g+s our_shared_directory
```

Otherwise the file creator's default group (usually the same as the user name) is used.

If a GID change is required temporarily you can also use the *newgrp* command to change the user's default GID to another GID at runtime. For example, after executing `newgrp groupname` files created by the user will be associated with the `groupname` GID, without requiring a re-login. To change back to the default GID, execute *newgrp* without a groupname.

5.1.1 Changing user defaults

The default values used for creating new accounts are set in `/etc/default/useradd` and can be displayed using `useradd --defaults`. For example, to change the default shell globally, set `SHELL=/usr/bin/shell`. A different shell can also be specified individually with the `-s` / `--shell` option. Use `chsh -l` to list valid login shells.

Files can also be specified to be added to newly created user home directories in `/etc/skel`. This is useful for minimalist window managers where config files need manual configuration to reach DE-familiar behavior. For example, to set up default shortcuts for all newly created users:

```
# mkdir /etc/skel/.config
# cp ~archie/.config/sxhkd /etc/skel/.config
```

See also: **Display manager#Run ~/.xinitrc as a session** to add xinitrc as an option to all users on the display manager.

5.2 Example adding a system user

System users can be used to run processes/daemons under a different user, protecting (e.g. with **chown**) files and/or directories and more examples of computer hardening.

With the following command a system user without shell access and without a `home` directory is created (optionally append the `-U` parameter to create a group with the same name as the user, and add the user to this group):

```
# useradd --system -s /usr/bin/nologin username
```

If the system user requires a specific user and group ID, specify them with the `-u` / `--uid` and `-g` / `--gid` options when creating the user:

```
# useradd --system -u 850 -g 850 -s /usr/bin/nologin username
```

5.3 Change a user's login name or home directory

To change a user's home directory:

```
# usermod -d /my/new/home -m username
```

The `-m` option also automatically creates the new directory and moves the content there.

> **Tip**
>
> You can create a link from the user's former home directory to the new one. Doing this will allow programs to find files that have hardcoded paths.
>
> ```
> # ln -s /my/new/home/ /my/old/home
> ```
>
> Make sure there is **no** trailing `/` on `/my/old/home`.

To change a user's login name:

```
# usermod -l newname oldname
```

> **Warning**
>
> Make certain that you are not logged in as the user whose name you are about to change. Open a new tty (e.g. `Ctrl+Alt+F6`) and log in as root or as another user and **elevate to root**. *usermod* should prevent you from doing this by mistake.

Changing a username is safe and easy when done properly, just use the **usermod** command. If the user is associated to a group with the same name, you can rename this with the **groupmod** command.

Alternatively, the `/etc/passwd` file can be edited directly, see **#User database** for an introduction to its format.

Also keep in mind the following notes:

- If you are using **sudo** make sure you update your `/etc/sudoers` to reflect the new username(s) (via the *visudo* command as root).
- Personal **crontabs** need to be adjusted by renaming the user's file in `/var/spool/cron` from the old to the new name, and then opening `crontab -e` to change any relevant paths and have it adjust the file permissions accordingly.
- **Wine's** personal directories/files' contents in `~/.wine/drive_c/users`, `~/.local/share/applications/wine/Programs` and possibly more need to be manually renamed/edited.
- Certain Thunderbird addons, like **Enigmail (https://www.enigmail.net/index.php/en/)**, may need to be reinstalled.
- Anything on your system (desktop shortcuts, shell scripts, etc.) that uses an absolute path to your home dir (i.e. `/home/oldname`) will need to be changed to reflect your new name. To avoid these problems in shell scripts, simply use the `~` or `$HOME` variables for home directories.
- Also do not forget to edit accordingly the configuration files in `/etc/` that relies on your absolute path (e.g. Samba, CUPS, so on). A nice way to learn what files you need to update involves using the grep command this way: `grep -r old_user *`

5.4 Other examples of user management

To enter user information for the **GECOS** comment (e.g. the full user name), type:

```
# chfn username
```

(this way *chfn* runs in interactive mode).

Alternatively the GECOS comment can be set more liberally with:

```
# usermod -c "Comment" username
```

To mark a user's password as expired, requiring them to create a new password the first time they log in, type:

```
# chage -d 0 username
```

User accounts may be deleted with the *userdel* command:

```
# userdel -r username
```

The `-r` option specifies that the user's home directory and mail spool should also be deleted.

To change the user's login shell:

```
# usermod -s /usr/bin/bash username
```

> **Tip**
>
> The **adduser (https://aur.archlinux.org/packages/adduser/)**[AUR] script allows carrying out the jobs of *useradd*, *chfn* and *passwd* interactively. See also **FS#32893 (https://bugs.archlinux.org/task/32893)**.

6 User database

Local user information is stored in the plain-text `/etc/passwd` file: each of its lines represents a user account, and has seven fields delimited by colons.

```
account:password:UID:GID:GECOS:directory:shell
```

Where:

- *`account`* is the user name. This field cannot be blank. Standard *NIX naming rules apply.
- *`password`* is the user password.

> **Warning**
>
> The `passwd` file is world-readable, so storing passwords (hashed or otherwise) in this file is insecure. Instead, Arch Linux uses **shadowed passwords**: the `password` field will contain a placeholder character (`x`) indicating that the hashed password is saved in the access-restricted file `/etc/shadow` . For this reason it is recommended to always change passwords using the **passwd** command.

- *`UID`* is the numerical user ID. In Arch, the first login name (after root) for a so called normal user, as opposed to services, is UID 1000 by default; subsequent UID entries for users should be greater than 1000.
- *`GID`* is the numerical primary group ID for the user. Numeric values for GIDs are listed in **/etc/group**.
- *`GECOS`* is an optional field used for informational purposes; usually it contains the full user name, but it can also be used by services such as *finger* and managed with the **chfn** command. This field is optional and may be left blank.
- *`directory`* is used by the login command to set the `$HOME` environment variable. Several services with their own users use `/` , but normal users usually set a directory under `/home` .

- *shell* is the path to the user's default **command shell**. This field is optional and defaults to `/usr/bin/bash` .

Example:

```
archie:x:1001:1003:Archie,some comment here,,:/home/archie:/usr/bin/bash
```

Broken down, this means: user `archie` , whose password is in `/etc/shadow` , whose UID is 1001 and whose primary group is 1003. Archie is their full name and there is a comment associated to their account; their home directory is `/home/archie` and they are using **Bash**.

The *pwck* command can be used to verify the integrity of the user database. It can sort the user list by UID at the same time, which can be helpful for comparison:

```
# pwck -s
```

Warning

Arch Linux defaults of the files are created as **pacnew files** by new releases of the **filesystem (https://archlinux.org/packages/?name=filesystem)** package. Unless Pacman outputs related messages for action, these *.pacnew* files can, and should, be disregarded/removed. New required default users and groups are added or re-added as needed by **systemd-sysusers(8) (https://man.archlinux.org/man/systemd-sysusers.8)** or the package install script.

7 Automatic integrity checks

Instead of running `pwck` / `grpck` manually, the **systemd timer** `shadow.timer` , which is part of, and is enabled by, installation of the **shadow (https://archlinux.org/packages/?name=shadow)** package, will start `shadow.service` daily. `shadow.service` will run **pwck(8) (https://man.archlinux.org/man/pwck.8)** and **grpck(8) (https://man.archlinux.org/man/grpck.8)** to verify the integrity of both password and group files.

If discrepancies are reported, group can be edited with the **vigr(8) (https://man.archlinux.org/man/vigr.8)** command and users with **vipw(8) (https://man.archlinux.org/man/vipw.8)**. This provides an extra margin of protection in that these commands lock the databases for editing. Note that the default text editor is vi, but an alternative editor will be used if the `EDITOR` **environment variable** is set, then that editor will be used instead.

8 Keeping users on the live system in parity with systemd sysuser.d defaults

Warning

systemd does not provide any official way to migrate existing accounts to fully locked system accounts [1] (https://github.com/systemd/systemd/issues/37179).

As packages adopt the change-sysusers-to-fully-locked-system-accounts (https://archlinux.org/todo/change-sysusers-to-fully-locked-system-accounts/), package created system users generated in the past will not inherit new package defaults for the increased security of locked/expired status. These users need to be modified manually for this change. user-analysis.sh (https://github.com/graysky2/user-analysis) does just that.

In addition, the script also identifies orphaned users (those created by a package no longer on the system) and can automatically delete them.

9 Group management

`/etc/group` is the file that defines the groups on the system (see group(5) (https://man.archlinux.org/man/group.5) for details). There is also its companion `gshadow` which is rarely used. Its details are at gshadow(5) (https://man.archlinux.org/man/gshadow.5).

Display group membership with the *groups* command:

```
$ groups user
```

If `user` is omitted, the current user's group names are displayed.

The *id* command provides additional detail, such as the user's UID and associated GIDs:

```
$ id user
```

To list all groups on the system:

```
$ cat /etc/group
```

Create new groups with the *groupadd* command:

```
# groupadd group
```

> **Note**
>
> If the user is currently logged in, they must log out and in again for changes to take effect.

Add users to a group with the *gpasswd* command (see FS#58262 (https://bugs.archlinux.org/task/58262) regarding errors):

```
# gpasswd -a user group
```

Alternatively, add a user to additional groups with *usermod* (replace `additional_groups` with a comma-separated list):

```
# usermod -aG additional_groups username
```

Warning

If the `-a` option is omitted in the *usermod* command above, the user is removed from all groups not listed in `additional_groups` (i.e. the user will be member only of those groups listed in `additional_groups`).

Modify an existing group with the *groupmod* command, e.g. to rename the `old_group` group to `new_group`:

```
# groupmod -n new_group old_group
```

Note

This will change a group name but not the numerical GID of the group. Hence, all files previously owned by `old_group` will be owned by `new_group`.

To delete existing groups:

```
# groupdel group
```

To remove users from a group:

```
# gpasswd -d user group
```

The *grpck* command can be used to verify the integrity of the system's group files.

Warning

Arch Linux defaults of the files are created as *.pacnew* files by new releases of the **filesystem (https://archlinux.org/packages/?name=filesystem)** package. Unless Pacman outputs related messages for action, these *.pacnew* files can, and should, be disregarded/removed. New required default users and groups are added or re-added as needed by **systemd-sysusers(8) (https://man.archlinux.org/man/systemd-sysusers.8)** or the package install script.

10 Group list

This section explains the purpose of the essential groups from the **filesystem (https://archlinux.org/packages/?name=filesystem)** package. There are many other groups, which will be created with **correct GID** when the relevant package is installed. See the main page for the software for details.

Note

A later removal of a package does not remove the automatically created user/group (UID/GID) again. This is intentional because any files created during its usage would otherwise be left orphaned as a potential security risk.

10.1 User groups

Non-root workstation/desktop users often need to be added to some of following groups to allow access to hardware peripherals and facilitate system administration:

Group	Affected files	Purpose
adm		Administration group, commonly used to give read access to protected logs. It has full read access to **journal** files.
ftp	`/srv/ftp/`	Access to files served by **FTP servers**.
games	`/var/games`	Access to some game software.
http	`/srv/http/`	Access to files served by **HTTP servers**.
log		Access to log files in `/var/log/` created by **syslog-ng**.
rfkill	`/dev/rfkill`	Right to control wireless devices power state (used by **rfkill**).
sys		Right to administer printers in **CUPS**.
systemd-journal	`/var/log/journal/*`	Can be used to provide read-only access to the systemd logs, as an alternative to `adm` and `wheel` **[2] (https://github.com/systemd/systemd/blob/19aadacf92ad86967ffb678e37b2ff9e83cb9480/README#L161)**. Otherwise, only user generated messages are displayed.
uucp	`/dev/ttyS[0-9]+`, `/dev/tts/[0-9]+`, `/dev/ttyUSB[0-9]+`, `/dev/ttyACM[0-9]+`, `/dev/rfcomm[0-9]+`	**RS-232** serial ports and devices connected to them.
wheel		Administration group, commonly used to give privileges to perform administrative actions. It has full read access to **journal** files and the right to administer printers in **CUPS**. Can also be used to give access to the **sudo** and **su** utilities (neither uses it by default).

10.2 System groups

The following groups are used for system purposes, an assignment to users is only required for dedicated purposes:

Group	Affected files	Purpose
dbus		used internally by dbus (https://archlinux.org/packages/?name=dbus)
kmem	/dev/port, /dev/mem, /dev/kmem	
locate	/usr/bin/locate, /var/lib/locate, /var/lib/mlocate, /var/lib/slocate	See Locate.
lp	/dev/lp[0-9]*, /dev/parport[0-9]*	Access to parallel port devices (printers and others).
mail	/usr/bin/mail	
nobody		Unprivileged group.
proc	/proc/*pid*/	A group authorized to learn processes information otherwise prohibited by hidepid= mount option of the proc file system (https://docs.kernel.org/filesystems/proc.html). The group must be explicitly set with the gid= mount option.
root	/*	Complete system administration and control (root, admin).
smmsp		sendmail group.
tty	/dev/tty, /dev/vcc, /dev/vc, /dev/ptmx	
utmp	/run/utmp, /var/log/btmp, /var/log/wtmp	See Wikipedia:utmp#utmp, wtmp and btmp.

10.3 Pre-systemd groups

Before arch migrated to **systemd**, users had to be manually added to these groups in order to be able to access the corresponding devices. This way has been deprecated in favour of **udev** marking the devices with a `uaccess` tag (https://github.com/systemd/systemd/blob/main/rules.d/70-uaccess.rules.in) and *logind* assigning the permissions to users dynamically via **ACLs** according to which session is currently active. Note that the session must not be broken for this to work (see **General troubleshooting#Session permissions** to check it).

There are some notable exceptions which require adding a user to some of these groups: for example if you want to allow users to access the device even when they are not logged in. However, note that adding users to the groups can even cause some functionality to break (for example, the `audio` group will break fast user switching and allows applications to block software mixing).

Group	Affected files	Purpose
audio	`/dev/audio`, `/dev/snd/*`, `/dev/rtc0`	Direct access to sound hardware, for all sessions. It is still required to make ALSA and OSS work in remote sessions, see ALSA#User privileges, otherwise not recommended. Unlike on certain other distros, this group is not used for realtime privileges.
disk	`/dev/sd[a-zA-Z]*[1-9]*`, `/dev/nvme[0-9]*p[1-9]*`, `/dev/mmcblk[0-9]*p[1-9]*`	Access to block devices not affected by other groups such as `optical`, `floppy`, and `storage`.
floppy	`/dev/fd[0-9]*`	Access to floppy drives.
input	`/dev/input/event[0-9]*`, `/dev/input/mouse[0-9]*`	Access to input devices. Introduced in systemd 215 [3] (https://lists.freedesktop.org/archives/systemd-commits/2014-June/006343.html).
kvm	`/dev/kvm`	Access to virtual machines using KVM.
optical	`/dev/sr[0-9]`, `/dev/sg[0-9]`	Access to optical devices such as CD and DVD drives.
scanner	`/var/lock/sane`	Access to scanner hardware.
storage	`/dev/st[0-9]*[lma]*`, `/dev/nst[0-9]*[lma]*`	Used to gain access to removable drives such as USB hard drives, flash/jump drives, MP3 players; enables the user to mount storage devices.[4] (https://lists.archlinux.org/archives/list/arch-dev-public@lists.archlinux.org/thread/7ITKMBKKHUB4SUN5SCS7FLSTEILKGIJH/#IEZCAXRIBMXGOMMW7C4Z3ADED32W24ZI) Now solely for direct access to tapes if no custom udev rules is involved.[5] (https://gitlab.archlinux.org/archlinux/packaging/packages/systemd/-/blob/main/0001-Use-Arch-Linux-device-access-groups.patch)[6] (https://github.com/systemd/systemd/blame/aaaf42cb44d4fcd598fca441a11856f3e8dd06d8/rules.d/50-udev-default.rules.in#L69)[7] (https://elixir.bootlin.com/linux/v5.5.6/source/include/scsi/scsi_proto.h#L251). Also required for manipulating some devices via udisks/udisksctl.
video	`/dev/fb/0`, `/dev/misc/agpgart`	Access to video capture devices, 2D/3D hardware acceleration, framebuffer (X can be used *without* belonging to this group).

10.4 Unused groups

The following groups are currently not used for any purpose:

Group	Affected files	Purpose
bin	none	Historical
daemon		
lock		Used for lockfile access. Required by e.g. gnokii (https://aur.archlinux.org/packages/gnokii/)[AUR].
mem		
network		Unused by default. Can be used e.g. for granting access to NetworkManager (see NetworkManager#Set up PolicyKit permissions).
power		
uuidd		
users		The primary group for users when user private groups are not used (generally not recommended), e.g. when creating users with `USERGROUPS_ENAB no` in `/etc/login.defs` or the `-N` / `--no-user-group` option of *useradd*.

11 Other tools related to these databases

getent(1) (https://man.archlinux.org/man/getent.1) can be used to read a particular record.

```
$ getent group tty
```

As warned in #User database, using specific utilities such as `passwd` and `chfn`, is a better way to change the databases. Nevertheless, there are times when editing them directly is looked after. For those times, `vipw`, `vigr` are provided. It is strongly recommended to use these tailored editors over using a general text editor as they lock the databases against concurrent editing. They also help prevent invalid entries and/or syntax errors. Note that Arch Linux prefers usage of specific tools, such as *chage*, for modifying the shadow database over using `vipw -s` and `vigr -s` from util-linux (https://archlinux.org/packages/?name=util-linux). See also FS#31414 (https://bugs.archlinux.org/task/31414).

lslogins(1) (https://man.archlinux.org/man/lslogins.1) and lastlog2(1) (https://man.archlinux.org/man/lastlog2.1) are some of the tools for viewing utmp related files.