# TP 0 : Installation du poste de travail

## Installation de la distribution Archlinux

### Installation des bases (10 pts)

Pour être en mesure de réaliser vos travaux tout au long de l'année, vous devez disposer d'un poste de travail opérationnel. Nous avons fait le choix d'utiliser la distribution archlinux. L'install d'une telle distribution permettra de mettre en pratique des concepts que vous avez survolé lors de vos années à l'université. *A noter que vous êtes administrateur du poste de travail, vous êtes libre d'installer une autre distribution une fois ce TP complété.*

Archlinux dispose d'un wiki et d'une communauté importante, n'hésitez pas à les consulter :

- https://bbs.archlinux.org/
- https://wiki.archlinux.org/

Prenez également le temps de lire : *Arch compared to other distributions*

Vous pouvez accéder aux hyperliens ci-dessus en utilisant un navigateur en ligne de commande tel que links. Pour installer des packages lors de l'installation, vous pouvez vous référer à la manpage de pacman fournis en annexe de ce TP.

Lors de l'installation vous serez amené à prendre certaines décisions, bien que le choix par défaut est souvent suffisant, vous devez être en mesure de motiver vos choix. Pour ce faire il vous est demandé d'expliquer :

1. Pourquoi et comment faut-il vérifier l'intégrité d'une image télécharger sur internet ?
2. Pourquoi avez choisis ce schéma de partitionnement ? Expliquez l'intérêt de chaque partions.
3. Comment votre machine peut récupérer son adresse IP en utilisant le protocol DHCP, et ce de manière automatique à chaques démarrages ?
4. Comment avez-vous choisis les serveur miroirs ?
5. A quoi sert "*fstab*" ?
6. Pourquoi avez-vous choisis ce "bootloader" ?

Une fois **l'installation minimal réalisée** et avoir **redémarré votre système**, faites valider votre installation.

### Installation des outils de travails (5 pts)

Vous avez à ce stade un environnement de travail plus que minimal. Pour votre confort, il est peut être utile d'installer un environnement graphique, un éditeur de texte, un navigateur web, un émulateur de terminal et un client vpn. Vous êtes libre de choisir les composants logiciel. Pour valider votre installation, vous devez être en mesure de :

1. Lire une vidéo sur une plateforme type youtube, dailymotion ou vimeo avec *le son* dans le navigateur de votre choix ;
2. Ecrire et de compiler un “Hello, world” en C ;
3. De vous rendre sur l’adresse : https://cloud.univ-lille.fr.

## Sécurisation de vos postes de travail (2.5 pts)

Le monde est dangereux, même votre salle de TP. Il vous faut donc installer un “firewall”. Utilisez celui de votre choix, toujours en argumentant, et bloquez toutes connexions sauf sur les ports 22 (SSH), 80 (HTTP) et 443 (HTTPS). Toutes tentatives de connexions UDP sur un port non ouvert doit produire une erreur type : “*Connection reset by peer*”.

Fait valider votre installation.

## Installation de Docker (2.5 pts)

Installer Docker, toutes les machines de la salles doivent être en mesure de consulter une page web qui annonce le hostname de votre poste de travail. Cette page doit être consultable sur le port 80 (HTTP) et 443 (HTTPS) avec un certificat valide.

Fait valider votre installation.

# pacman(8) Manual Page

## NAME

pacman - package manager utility

## Synopsis

*pacman* <operation> [options] [targets]

## Description

Pacman is a package management utility that tracks installed packages on a Linux system. It features dependency support, package groups, install and uninstall scripts, and the ability to sync your local machine with a remote repository to automatically upgrade packages. Pacman packages are a zipped tar format.

Since version 3.0.0, pacman has been the front-end to libalpm(3), the “Arch Linux Package Management” library. This library allows alternative front-ends to be written (for instance, a GUI front-end).

Invoking pacman involves specifying an operation with any potential options and targets to operate on. A *target* is usually a package name, file name, URL, or a search string. Targets can be provided as command line arguments. Additionally, if stdin is not from a terminal and a single hyphen (-) is passed as an argument, targets will be read from stdin.

## Operations

**-D, --database**

Operate on the package database. This operation allows you to modify certain attributes of the installed packages in pacman’s database. It also allows you to check the databases for internal consistency. See Database Options below.

**-Q, --query**

Query the package database. This operation allows you to view installed packages and their files, as well as meta-information about individual packages (dependencies, conflicts, install date, build date, size). This can be run against the local package database or can be used on individual package files. In the first case, if no package names are provided in the

command line, all installed packages will be queried. Additionally, various filters can be applied on the package list. See Query Options below.

### -R, --remove

Remove package(s) from the system. Groups can also be specified to be removed, in which case every package in that group will be removed. Files belonging to the specified package will be deleted, and the database will be updated. Most configuration files will be saved with a *.pacsave* extension unless the *--nosave* option is used. See Remove Options below.

### -S, --sync

Synchronize packages. Packages are installed directly from the remote repositories, including all dependencies required to run the packages. For example, `pacman -S qt` will download and install qt and all the packages it depends on. If a package name exists in more than one repository, the repository can be explicitly specified to clarify the package to install: `pacman -S testing/qt`. You can also specify version requirements: `pacman -S "bash>=3.2"`. Quotes are needed, otherwise the shell interprets ">" as redirection to a file.

In addition to packages, groups can be specified as well. For example, if gnome is a defined package group, then `pacman -S gnome` will provide a prompt allowing you to select which packages to install from a numbered list. The package selection is specified using a space- and/or comma-separated list of package numbers. Sequential packages may be selected by specifying the first and last package numbers separated by a hyphen (`-`). Excluding packages is achieved by prefixing a number or range of numbers with a caret (`^`).

Packages that provide other packages are also handled. For example, `pacman -S foo` will first look for a foo package. If foo is not found, packages that provide the same functionality as foo will be searched for. If any package is found, it will be installed. A selection prompt is provided if multiple packages providing foo are found.

You can also use `pacman -Su` to upgrade all packages that are out-of-date. See Sync Options below. When upgrading, pacman performs version comparison to determine which packages need upgrading. This behavior operates as follows:

```
Alphanumeric:
  1.0a < 1.0b < 1.0beta < 1.0p < 1.0pre < 1.0rc < 1.0
< 1.0.a < 1.0.1
Numeric:
  1 < 1.0 < 1.1 < 1.1.1 < 1.2 < 2.0 < 3.0.0
```

Additionally, version strings can have an *epoch* value defined that will overrule any version comparison, unless the epoch values are equal. This is specified in an `epoch:version-rel` format. For example, `2:1.0-1` is always greater than `1:3.6-1`.

### -T, --deptest

Check dependencies; this is useful in scripts such as makepkg to check installed packages. This operation will check each dependency specified and return a list of dependencies that are not currently satisfied on the system. This operation accepts no other options. Example usage: `pacman -T qt "bash>=3.2"`.

**-U, --upgrade**
Upgrade or add package(s) to the system and install the required dependencies from sync repositories. Either a URL or file path can be specified. This is a “remove-then-add” process. See Upgrade Options below; also see Handling Config Files for an explanation on how pacman takes care of configuration files.

**-F, --files**
Query the files database. This operation allows you to look for packages owning certain files or display files owned by certain packages. Only packages that are part of your sync databases are searched. See File Options below.

**-V, --version**
Display version and exit.

**-h, --help**
Display syntax for the given operation. If no operation was supplied, then the general syntax is shown.

## Options

**-b, --dbpath** <path>
Specify an alternative database location (the default is `/usr/local/var/lib/pacman`). This should not be used unless you know what you are doing. **NOTE**: If specified, this is an absolute path, and the root path is not automatically prepended.

**-r, --root** <path>
Specify an alternative installation root (default is `/`). This should not be used as a way to install software into `/usr/local` instead of `/usr`. **NOTE**: If database path or log file are not specified on either the command line or in pacman.conf(5), their default location will be inside this root path. **NOTE**: This option is not suitable for performing operations on a mounted guest system. See *--sysroot* instead.

**-v, --verbose**
Output paths such as as the Root, Conf File, DB Path, Cache Dirs, etc.

**--arch** <arch>
Specify an alternate architecture.

**--cachedir** <dir>
Specify an alternative package cache location (the default is `/usr/local/var/cache/pacman/pkg`). Multiple cache directories can be specified, and they are tried in the order they are passed to pacman. **NOTE**: This is

an absolute path, and the root path is not automatically prepended.

**--color** <when>
: Specify when to enable coloring. Valid options are *always*, *never*, or *auto*. *always* forces colors on; *never* forces colors off; and *auto* only automatically enables colors when outputting onto a tty.

**--config** <file>
: Specify an alternate configuration file.

**--debug**
: Display debug messages. When reporting bugs, this option is recommended to be used.

**--gpgdir** <dir>
: Specify a directory of files used by GnuPG to verify package signatures (the default is `/usr/local/etc/pacman.d/gnupg`). This directory should contain two files: `pubring.gpg` and `trustdb.gpg`. `pubring.gpg` holds the public keys of all packagers. `trustdb.gpg` contains a so-called trust database, which specifies that the keys are authentic and trusted. **NOTE**: This is an absolute path, and the root path is not automatically prepended.

**--hookdir** <dir>
: Specify a alternative directory containing hook files (the default is `/usr/local/etc/pacman.d/hooks`). Multiple hook directories can be specified with hooks in later directories taking precedence over hooks in earlier directories. **NOTE**: This is an absolute path, and the root path is not automatically prepended.

**--logfile** <file>
: Specify an alternate log file. This is an absolute path, regardless of the installation root setting.

**--noconfirm**
: Bypass any and all “Are you sure?” messages. It’s not a good idea to do this unless you want to run pacman from a script.

**--confirm**
: Cancels the effects of a previous *--noconfirm*.

**--disable-download-timeout**
: Disable defaults for low speed limit and timeout on downloads. Use this if you have issues downloading files with proxy and/or security gateway.

**--sysroot** <dir>
: Specify an alternative system root. Pacman will chroot and chdir into the system root prior to running. This allows mounted guest systems to be properly operated on. Any other paths given will be interpreted as relative to the system root. Requires root privileges.

## Transaction Options (apply to *-S*, *-R* and *-U*)

**-d, --nodeps**

Skips dependency version checks. Package names are still checked. Normally, pacman will always check a package's dependency fields to ensure that all dependencies are installed and there are no package conflicts in the system. Specify this option twice to skip all dependency checks.

**--assume-installed** <package=version>

Add a virtual package "package" with version "version" to the transaction to satisfy dependencies. This allows to disable specific dependency checks without affecting all dependency checks. To disable all dependency checking, see the *--nodeps* option.

**--dbonly**

Adds/removes the database entry only, leaving all files in place.

**--noprogressbar**

Do not show a progress bar when downloading files. This can be useful for scripts that call pacman and capture the output.

**--noscriptlet**

If an install scriptlet exists, do not execute it. Do not use this unless you know what you are doing.

**-p, --print**

Only print the targets instead of performing the actual operation (sync, remove or upgrade). Use *--print-format* to specify how targets are displayed. The default format string is "%l", which displays URLs with *-S*, file names with *-U*, and pkgname-pkgver with *-R*.

**--print-format** <format>

Specify a printf-like format to control the output of the *--print* operation. The possible attributes are: "%n" for pkgname, "%v" for pkgver, "%l" for location, "%r" for repository, and "%s" for size. Implies *--print*.

## Upgrade Options (apply to *-S* and *-U*)

**--asdeps**

Install packages non-explicitly; in other words, fake their install reason to be installed as a dependency. This is useful for makepkg and other build-from-source tools that need to install dependencies before building the package.

**--asexplicit**

Install packages explicitly; in other words, fake their install reason to be explicitly installed. This is useful if you want to mark a dependency as explicitly installed so it will not be removed by the *--recursive* remove operation.

**--ignore** <package>

Directs pacman to ignore upgrades of package even if there is one available. Multiple packages can be specified by separating them with a

comma.

**--ignoregroup** <group>
: Directs pacman to ignore upgrades of all packages in *group*, even if there is one available. Multiple groups can be specified by separating them with a comma.

**--needed**
: Do not reinstall the targets that are already up-to-date.

**--overwrite** <glob>
: Bypass file conflict checks and overwrite conflicting files. If the package that is about to be installed contains files that are already installed and match *glob*, this option will cause all those files to be overwritten. Using *--overwrite* will not allow overwriting a directory with a file or installing packages with conflicting files and directories. Multiple patterns can be specified by separating them with a comma. May be specified multiple times. Patterns can be negated, such that files matching them will not be overwritten, by prefixing them with an exclamation mark. Subsequent matches will override previous ones. A leading literal exclamation mark or backslash needs to be escaped.

## Query Options (apply to *-Q*)

**-c, --changelog**
: View the ChangeLog of a package if it exists.

**-d, --deps**
: Restrict or filter output to packages installed as dependencies. This option can be combined with *-t* for listing real orphans - packages that were installed as dependencies but are no longer required by any installed package.

**-e, --explicit**
: Restrict or filter output to explicitly installed packages. This option can be combined with *-t* to list explicitly installed packages that are not required by any other package.

**-g, --groups**
: Display all packages that are members of a named group. If a name is not specified, list all grouped packages.

**-i, --info**
: Display information on a given package. The *-p* option can be used if querying a package file instead of the local database. Passing two *--info* or *-i* flags will also display the list of backup files and their modification states.

**-k, --check**
: Check that all files owned by the given package(s) are present on the system. If packages are not specified or filter flags are not provided, check all installed packages. Specifying this option twice will perform

more detailed file checking (including permissions, file sizes, and modification times) for packages that contain the needed mtree file.

**-l, --list**
: List all files owned by a given package. Multiple packages can be specified on the command line.

**-m, --foreign**
: Restrict or filter output to packages that were not found in the sync database(s). Typically these are packages that were downloaded manually and installed with *--upgrade*.

**-n, --native**
: Restrict or filter output to packages that are found in the sync database(s). This is the inverse filter of *--foreign*.

**-o, --owns** <file>
: Search for packages that own the specified file(s). The path can be relative or absolute, and one or more files can be specified.

**-p, --file**
: Signifies that the package supplied on the command line is a file and not an entry in the database. The file will be decompressed and queried. This is useful in combination with *--info* and *--list*.

**-q, --quiet**
: Show less information for certain query operations. This is useful when pacman's output is processed in a script. Search will only show package names and not version, group, and description information; owns will only show package names instead of "file is owned by pkg" messages; group will only show package names and omit group names; list will only show files and omit package names; check will only show pairs of package names and missing files; a bare query will only show package names rather than names and versions.

**-s, --search** <regexp>
: Search each locally-installed package for names or descriptions that match `regexp`. When including multiple search terms, only packages with descriptions matching ALL of those terms are returned.

**-t, --unrequired**
: Restrict or filter output to print only packages neither required nor optionally required by any currently installed package. Specify this option twice to include packages which are optionally, but not directly, required by another package.

**-u, --upgrades**
: Restrict or filter output to packages that are out-of-date on the local system. Only package versions are used to find outdated packages; replacements are not checked here. This option works best if the sync database is refreshed using *-Sy*.

## Remove Options (apply to *-R*)

**-c, --cascade**

Remove all target packages, as well as all packages that depend on one or more target packages. This operation is recursive and must be used with care, since it can remove many potentially needed packages.

**-n, --nosave**

Instructs pacman to ignore file backup designations. Normally, when a file is removed from the system, the database is checked to see if the file should be renamed with a *.pacsave* extension.

**-s, --recursive**

Remove each target specified including all of their dependencies, provided that (A) they are not required by other packages; and (B) they were not explicitly installed by the user. This operation is recursive and analogous to a backwards *--sync* operation, and it helps keep a clean system without orphans. If you want to omit condition (B), pass this option twice.

**-u, --unneeded**

Removes targets that are not required by any other packages. This is mostly useful when removing a group without using the *-c* option, to avoid breaking any dependencies.

## Sync Options (apply to *-S*)

**-c, --clean**

Remove packages that are no longer installed from the cache as well as currently unused sync databases to free up disk space. When pacman downloads packages, it saves them in a cache directory. In addition, databases are saved for every sync DB you download from and are not deleted even if they are removed from the configuration file pacman.conf(5). Use one *--clean* switch to only remove packages that are no longer installed; use two to remove all files from the cache. In both cases, you will have a yes or no option to remove packages and/or unused downloaded databases.

If you use a network shared cache, see the *CleanMethod* option in pacman.conf(5).

**-g, --groups**

Display all the members for each package group specified. If no group names are provided, all groups will be listed; pass the flag twice to view all groups and their members.

**-i, --info**

Display information on a given sync database package. Passing two *--info* or *-i* flags will also display those packages in all repositories that depend on this package.

**-l, --list**

List all packages in the specified repositories. Multiple repositories can be specified on the command line.

**-q, --quiet**

Show less information for certain sync operations. This is useful when pacman's output is processed in a script. Search will only show package names and not repository, version, group, and description information; list will only show package names and omit databases and versions; group will only show package names and omit group names.

**-s, --search** <regexp>

This will search each package in the sync databases for names or descriptions that match `regexp`. When you include multiple search terms, only packages with descriptions matching ALL of those terms will be returned.

**-u, --sysupgrade**

Upgrades all packages that are out-of-date. Each currently-installed package will be examined and upgraded if a newer package exists. A report of all packages to upgrade will be presented, and the operation will not proceed without user confirmation. Dependencies are automatically resolved at this level and will be installed/upgraded if necessary.

Pass this option twice to enable package downgrades; in this case, pacman will select sync packages whose versions do not match with the local versions. This can be useful when the user switches from a testing repository to a stable one.

Additional targets can also be specified manually, so that *-Su foo* will do a system upgrade and install/upgrade the "foo" package in the same operation.

**-w, --downloadonly**

Retrieve all packages from the server, but do not install/upgrade anything.

**-y, --refresh**

Download a fresh copy of the master package database from the server(s) defined in pacman.conf(5). This should typically be used each time you use *--sysupgrade* or *-u*. Passing two *--refresh* or *-y* flags will force a refresh of all package databases, even if they appear to be up-to-date.

## Database Options (apply to *-D*)

**--asdeps** <package>

Mark a package as non-explicitly installed; in other words, set their install reason to be installed as a dependency.

**--asexplicit** <package>

Mark a package as explicitly installed; in other words, set their install reason to be explicitly installed. This is useful it you want to keep a package installed even when it was initially installed as a dependency of another package.

**-k, --check**

Check the local package database is internally consistent. This will check all required files are present and that installed packages have the required dependencies, do not conflict and that multiple packages do not own the same file. Specifying this option twice will perform a check on the sync databases to ensure all specified dependencies are available.

**-q, --quiet**
Suppress messages on successful completion of database operations.

## File Options (apply to *-F*)

**-y, --refresh**
Download fresh package databases from the server. Use twice to force a refresh even if databases are up to date.

**-l, --list**
List the files owned by the queried package.

**-x, --regex**
Interpret each query as a regular expression.

**-q, --quiet**
Show less information for certain file operations. This is useful when pacman's output is processed in a script, however, you may want to use *--machinereadable* instead.

**--machinereadable**
Print each match in a machine readable output format. The format is *repository**opkgname**opkgver**opath**n* with *o* being the NULL character and *n* a linefeed.

## Handling Config Files

Pacman uses the same logic as *rpm* to determine action against files that are designated to be backed up. During an upgrade, three MD5 hashes are used for each backup file to determine the required action: one for the original file installed, one for the new file that is about to be installed, and one for the actual file existing on the file system. After comparing these three hashes, the following scenarios can result:

original=X, current=X, new=X
All three files are the same, so overwrites are not an issue. Install the new file.

original=X, current=X, new=Y
The current file is the same as the original, but the new one differs. Since the user did not ever modify the file, and the new one may contain improvements or bug fixes, install the new file.

original=X, current=Y, new=X
Both package versions contain the exact same file, but the one on the file

system has been modified. Leave the current file in place.

original=X, current=Y, new=Y
: The new file is identical to the current file. Install the new file.

original=X, current=Y, new=Z
: All three files are different, so install the new file with a *.pacnew* extension and warn the user. The user must then manually merge any necessary changes into the original file.

original=NULL, current=Y, new=Z
: The package was not previously installed, and the file already exists on the file system. Install the new file with a *.pacnew* extension and warn the user. The user must then manually merge any necessary changes into the original file.

## Examples

pacman -Ss ne.hack
: Search for regexp "ne.hack" in package database.

pacman -S gpm
: Download and install gpm including dependencies.

pacman -U /home/user/ceofhack-0.6-1-x86_64.pkg.tar.gz
: Install ceofhack-0.6-1 package from a local file.

pacman -Syu
: Update package list and upgrade all packages afterwards.

pacman -Syu gpm
: Update package list, upgrade all packages, and then install gpm if it wasn't already installed.

## Configuration

See pacman.conf(5) for more details on configuring pacman using the *pacman.conf* file.

## See Also

alpm-hooks(5), libalpm(3), makepkg(8), pacman.conf(5)

See the pacman website at https://www.archlinux.org/pacman/ for current information on pacman and its related tools.

## Bugs

Bugs? You must be kidding; there are no bugs in this software. But if we

happen to be wrong, submit a bug report with as much detail as possible at the Arch Linux Bug Tracker in the Pacman section.

## Authors

Current maintainers:

- Allan McRae <allan@archlinux.org>
- Andrew Gregory <andrew.gregory.8@gmail.com>
- Dan McGee <dan@archlinux.org>
- Dave Reisner <dreisner@archlinux.org>

Past major contributors:

- Judd Vinet <jvinet@zeroflux.org>
- Aurelien Foret <aurelien@archlinux.org>
- Aaron Griffin <aaron@archlinux.org>
- Xavier Chantry <shiningxc@gmail.com>
- Nagy Gabor <ngaba@bibl.u-szeged.hu>

For additional contributors, use `git shortlog -s` on the pacman.git repository.

---

Last updated 2019-08-12 03:26:42 CEST