# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## Facultad de Ingeniería

# “Pacman: El Motor de Gestión de Software en Arch Linux”

---


**Sistemas Operativos Grupo 6**

**Nombre de integrantes:**

**Saldaña Navarrete Andrea**

**Salgado Cruz Emiliano Roman**

**Fecha de entrega:** **09 de abril de 2025.**

**Semestre: 2025-1**

# "Pacman: El Motor de Gestión de Software en Arch Linux"

## FILOSOFÍA DE ARCH LINUX Y SU SISTEMA DE PAQUETES

*Arch Linux* es una distribución de *Linux* para servidores que ha sido optimizada para AMD64 y se ofrece como una distribución de actualización continua (o *rolling release*, en inglés) desde 2002. La distribución se basa en el principio *KISS*, implantado desde el principio por el entonces director del proyecto Judd Vinet y su equipo. El principio "*KISS*" o "*Keep it simple, stupid*", se centra en darle un enfoque simple a la distribución.

*Arch Linux* prescinde de ayudas visuales para la instalación y la configuración, lo que significa que la distribución está dirigida más bien para desarrolladores experimentados. Desde 2020, Arch Linux ha sido desarrollado por un equipo dirigido por Levente Polyak junto con "usuarios de confianza" y se ha publicado bajo la *GPL* (*General Public License*).

### Sistema administrador de Paquetes

Un paquete es un archivo comprimido que contiene los archivos binarios o ejecutables del *software*, *scripts* para la configuración del mismo y metadatos (nombre, versión, dependencias). El gestor de paquetes es fundamental en cualquier sistema operativo *Linux* ya que sirve para automatizar el proceso de instalación, actualización, configuración y eliminación de *Software*.

***Pacman*** es el gestor de paquetes predeterminado para las distribuciones *Linux* basadas en *Arch*, al igual que una de sus principales características distintivas. Es conocido por su simplicidad, velocidad y enfoque minimalista, pero también por su filosofía diferente a la de otros gestores como *dnf* o *apt*.

## CARACTERÍSTICAS PRINCIPALES

- Diseño minimalista y eficiente. Está escrito en lenguaje C y utiliza *scripts Bash* para operaciones, lo que lo hace rápido y ligero. También utiliza comandos cortos y consistentes, lo cual puede ser una ventaja y desventaja al mismo tiempo ya que al ser menos intuitivos, es necesario memorizarlos.
- Sistema *rolling-release*. Significa que los paquetes en los repositorios siempre están en su última versión estable y no hay versiones fijas del sistema operativo. El sistema se actualiza en su totalidad mediante el comando:

  ```
  pacman -Syu
  ```

  El único riesgo al actualizar de esta forma es que puede romperse algo si hay cambios incompatibles.
- Manejo de dependencias. Sólo se instala lo absolutamente necesario, además no hay "metapaquetes" automáticos, el usuario es quien decide qué instalar. Sin embargo, también se ofrece soporte para dependencias opcionales, si un paquete tiene dependencias opcionales que no se necesitan explícitamente pero que se desean instalar, se puede usar la opción `--asdep`.

  De manera un poco más técnica, *Pacman* no solo resuelve dependencias declaradas por los paquetes, si no que construye internamente un grafo dirigido de dependencias. Internamente, el gestor identifica las relaciones entre paquetes, construye el grafo y detecta posibles ciclos para evitar conflictos durante la instalación.

  En el siguiente fragmento de código obtenido del archivo `deps.c` que podemos encontrar en el repositorio oficial de *Pacman*, podemos observar que se forma el grafo de dependencias entre los paquetes (vértices) y se asignan las relaciones de dependencia (aristas). La detección de ciclos y la resolución de dependencias se manejan en funciones como

`_alpm_warn_dep_cycle` y `_alpm_sortbydeps`, que se ocupan de procesar el grafo de dependencias para determinar el orden correcto de instalación o eliminación.

```
/* Convert a list of alpm_pkg_t * to a graph structure,
 * with an edge for each dependency.
 * Returns a list of vertices (one vertex = one package)
 * (used by alpm_sortbydeps)
 */
static alpm_list_t *dep_graph_init(alpm_handle_t *handle,
    alpm_list_t *targets, alpm_list_t *ignore)
{
alpm_list_t *i, *j;
alpm_list_t *vertices = NULL;
alpm_list_t *localpkgs = alpm_list_diff(
        alpm_db_get_pkgcache(handle->db_local), targets, _alpm_pkg_cmp);

/* We create the vertices */
for(i = targets; i; i = i->next) {
    alpm_graph_t *vertex = _alpm_graph_new();
    vertex->data = (void *)i->data;
    vertices = alpm_list_add(vertices, vertex);
}

/* We compute the edges */
for(i = vertices; i; i = i->next) {
    alpm_graph_t *vertex_i = i->data;
    alpm_pkg_t *p_i = vertex_i->data;
    for(j = vertices; j; j = j->next) {
        alpm_graph_t *vertex_j = j->data;
        alpm_pkg_t *p_j = vertex_j->data;
        if(_alpm_pkg_depends_on(p_i, p_j)) {
            vertex_i->children =
                alpm_list_add(vertex_i->children, vertex_j);
        }
    }
}

return vertices;
}
```

- Repositorios y *AUR* (*Arch User Repository*). Hay algunos repositorios oficiales, tales como *Core*, *Extra*, *Community,* que contienen paquetes oficiales, firmados y mantenidos por desarrolladores, sin embargo es *AUR* el que es considerado como el "superpoder" de *Arch* ya que, al ser comunitario, cualquier persona puede subir sus propios *scripts PKGBUILD*. El único problema con esto es que cada usuario es responsable de verificar la seguridad del *script*.
- Base de datos simple y transparente. Toda la información se guarda en archivos de texto en `var/lib/pacman`, además no hay *hidden locks,* si algo falla, el usuario puede editar o limpiar manualmente la base.
- Instalación desde código fuente. Como mencionamos antes, no se compilan paquetes por defecto, pero se soportan *PKGBUILD*, los cuales son *scripts* de *Bash* que contienen instrucciones para descargar, compilar y empaquetar *software* desde su código fuente. Normalmente contienen:
    - Metadatos del paquete: nombre, versión del *software,* revisión, descripción, arquitecturas soportadas, página oficial del proyecto, licencia.
    - Dependencias: aquellas necesarias para compilar y ejecutar el *software* y en algunas ocasiones se incluyen otras opcionales.
    - Orígenes del código fuente.
    - Funciones de construcción: `build()` y `package()`.
    - También pueden contener parches personalizados, variables útiles y funciones adicionales.

  Estos archivos generan paquetes con extensión `.pkg.tar.zst`, que son una combinación de *tape archives* (archivos `tar`) comprimidos con el algoritmo *Zstandard* (`zstd`) y son los que utilizamos para instalar *software.* Se crean al ejecutar `makepkg` en el directorio correspondiente al *PKGBUILD* y su estructura normalmente se compone de:
    - Archivos del *software* instalado: binarios, bibliotecas, documentación y configuraciones.
    - Metadatos del paquete: `.PKGINFO`, `.MTREE`

- *Scripts* opcionales: archivos de texto en *Bash* que pueden ejecutarse antes o después de instalar o eliminar el paquete.

**COMPARACIÓN CON OTROS GESTORES DE PAQUETES**

Lo primero que haremos será dar una breve descripción de los gestores que utilizaremos en esta sección, *WinGet* y *DNF*.

***WinGet.***

Esta herramienta es la interfaz cliente para el servicio del administrador de paquetes de *Windows*, disponible para las versiones 10, 11 y *Server* 2025.

Entre sus características principales están las siguientes:

- Maneja un comando principal *winget*
- Lista y busca aplicaciones: `winget search <nombre>` para buscar aplicaciones disponibles.
- Ofrece una serie de comandos que le permiten mostrar los detalles de las aplicaciones, cambiar de origen y validar los paquetes. Para obtener una lista completa de comandos, usamos el comando: `winget --help`.

En cuanto a las ventajas que ofrece están la integración directa con *Windows*, que funciona bien con *software* popular y que está en constante actualización en la *Microsoft Store*.

Entre sus desventajas está el hecho de que es relativamente nuevo, puesto que salió recientemente en las actualizaciones de mayo de 2020, aún está en fases de expansión, por lo que no todo el *software* está disponible, además puede resultar menos flexible comparado con otros gestores de paquetes.

***DNF***

Es el sistema de gestión de paquetes por defecto en la mayoría de las distribuciones *Linux* basadas en *RPM*.

Entre sus características principales están las siguientes:

- Gestiona automáticamente las dependencias entre paquetes, asegurando que todas las bibliotecas necesarias se instalen.
- Facilidad de uso: *DNF* ha mejorado la experiencia del usuario respecto a *YUM* con un rendimiento más rápido y una mejor gestión de dependencias.

Mientras *Pacman* se enfoca principalmente en *Arch Linux* y su comunidad, que es más de "hazlo tú mismo", *DNF*, por otro lado, está más integrado en ecosistemas empresariales *(Red Hat, Fedora*), y su uso tiende a ser más orientado a usuarios que prefieren estabilidad y soporte a largo plazo, como en servidores.

Los criterios que usaremos para comparar los gestores son:

- Reproductibilidad.
- Velocidad de instalación de paquetes.
- Tamaño promedio de los repositorios que suele utilizar cada gestor.

**VENTAJAS Y DESVENTAJAS DE PACMAN EN ENTORNOS DE PRODUCCIÓN**

Dado que Pacman está diseñado para ser rápido y eficiente en la gestión de paquetes, esto representa una clara ventaja en entornos de producción, en donde el tiempo es esencial y muy valioso. Otra ventaja que puede otorgar Pacman en estos entornos es la flexibilidad que otorga para la personalización y configuración del sistema de los usuarios.

Una de las claras desventajas que tenemos con este gestor es la curva de aprendizaje, que podría ser mayor y más pronunciada en comparación con otros gestores de paquetes más simples.
La cantidad de paquetes podría ser más limitada en comparación con otros sistemas, lo que podría limitar las capacidades y alcances en un entorno de producción específico.

**LINK DEL VIDEO.**

"Pacman: El Motor de Gestión de Software en Arch Linux".mp4

**REFERENCIAS**


- *pacman - ArchWiki*. (s. f.). https://wiki.archlinux.org/title/Pacman
- De Linux, C. (2024, 30 mayo). *pacman: El Corazón de Arch Linux para la Gestión de Paquetes - Cosas de Linux*. Cosas de Linux. https://www.cosasdelinux.com/pacman-el-corazon-de-arch-linux-para-la-gestion-de-paquetes/
- Mattwojo. (2025, 13 febrero). *Administrador de paquetes de Windows*. Microsoft Learn. https://learn.microsoft.com/es-es/windows/package-manager/
- https://gitlab.archlinux.org/pacman/pacman/-/blob/master/lib/libalpm/deps.c?ref_type=heads
- *Using the DNF software package manager*. (s. f.). Fedora Docs. https://docs.fedoraproject.org/en-US/quick-docs/dnf/