Cuando instalamos una aplicación desde el código fuente usando **make install**, podemos encontrarnos con un problema bastante incómodo: desinstalarla después no siempre es directo.

Esto ocurre porque **make install no sigue un estándar universal de desinstalación**. Cada proyecto decide qué hace durante la instalación y si proporciona o no una forma limpia de revertirla. Por eso, instalar software desde código fuente puede ser práctico, pero también puede ensuciar el sistema si no sabemos exactamente qué archivos se han copiado.

Lo primero que debemos comprobar es si el propio proyecto incluye una regla de desinstalación. Para ello, debemos volver al directorio del código fuente desde el que compilamos e instalamos la aplicación y ejecutar:

```
make uninstall
```

Si el Makefile incluye el objetivo **uninstall**, este comando debería eliminar los archivos instalados previamente. El problema es que esto depende completamente de los autores del programa. Algunos proyectos lo incluyen, otros no.

Además, debemos ejecutar este comando desde el **mismo árbol de código fuente**, usando la **misma versión** y, preferiblemente, la **misma configuración** con la que hicimos la instalación original. Si compilamos el programa con opciones concretas, rutas personalizadas o prefijos diferentes, conviene repetir exactamente ese contexto antes de intentar desinstalarlo.

Un error habitual es pensar que **make clean** desinstala el programa. No es así.

```
make clean
```

El comando **make clean** normalmente solo limpia el directorio de compilación. Es decir, elimina archivos intermedios, binarios generados, objetos compilados y otros restos del proceso de construcción dentro del propio árbol de código fuente.

Pero **make clean no toca el resto del sistema de archivos**. Si durante la instalación se copiaron binarios a /usr/local/bin, librerías a /usr/local/lib o páginas de manual a /usr/local/share/man, make clean no eliminará nada de eso.

Si el proyecto no incluye una regla **uninstall**, tendremos que averiguar qué hizo exactamente la instalación para poder revertirla manualmente. Una forma útil de verlo es ejecutar una simulación de instalación:

```
make -n install
```

La opción **-n** indica a make que muestre los comandos que ejecutaría, pero sin ejecutarlos realmente. Esto nos permite ver qué archivos copiaría, dónde los instalaría y qué directorios modificaría.

Con esa información podemos deshacer manualmente los pasos. Por ejemplo, si vemos que la instalación copia un binario a /usr/local/bin, una librería a /usr/local/lib y documentación a /usr/local/share, podremos eliminar esos archivos concretos.

```
sudo rm /usr/local/bin/nombre-del-binario
sudo rm /usr/local/lib/libnombre.so
sudo rm -r /usr/local/share/nombre-del-programa
```

Debemos tener mucho cuidado con este método. No conviene borrar rutas completas a ciegas, especialmente dentro de /usr, /usr/local, /lib o /etc. Lo correcto es eliminar únicamente los archivos que sepamos que pertenecen a la aplicación instalada.

Otra técnica útil, si el Makefile soporta la variable **DESTDIR**, consiste en repetir la instalación dentro de un directorio temporal. Así podemos ver qué archivos habría instalado el programa sin tocar directamente el sistema real.

```
mkdir /tmp/instalacion-prueba
make install DESTDIR=/tmp/instalacion-prueba
find /tmp/instalacion-prueba -type f
```

Esto nos permite obtener una lista bastante aproximada de los archivos que el programa instala. Después podemos comparar

esas rutas con las rutas reales del sistema y eliminarlas manualmente si corresponde.

Por ejemplo, si dentro de /tmp/instalacion-prueba aparece esta ruta:

```
/tmp/instalacion-prueba/usr/local/bin/programa
```

La ruta real instalada probablemente será:

```
/usr/local/bin/programa
```

En ese caso podríamos eliminarla con:

```
sudo rm /usr/local/bin/programa
```

En proyectos generados con CMake, a veces existe un archivo llamado **install_manifest.txt**. Este archivo contiene una lista de los archivos instalados durante el proceso de instalación.

```
cat install_manifest.txt
```

Si existe, podemos usarlo como referencia para saber qué archivos debemos eliminar. Aun así, conviene revisar el contenido antes de borrar nada.

```
sudo rm $(cat install_manifest.txt)
```

Este comando solo deberíamos usarlo si hemos revisado antes el archivo y estamos seguros de que todas las rutas son correctas. En sistemas en producción, es más prudente borrar los archivos uno por uno o hacer primero una simulación visual.

La mejor forma de evitar este problema es no instalar directamente con **make install** sobre el sistema principal sin control. Cuando sea posible, es preferible instalar software usando paquetes del sistema, crear un paquete .deb, usar un directorio prefijado o emplear herramientas que permitan rastrear qué archivos se instalan.

Por ejemplo, podemos instalar en /usr/local usando un prefijo claro:

```
./configure --prefix=/usr/local
make
sudo make install
```

O incluso instalar en una ruta aislada:

```
./configure --prefix=/opt/nombre-del-programa
make
sudo make install
```

Instalar en /opt/nombre-del-programa facilita mucho la limpieza posterior, porque todos los archivos quedan agrupados en una ubicación concreta. En ese caso, desinstalar puede ser tan simple como borrar ese directorio y retirar enlaces simbólicos si los hemos creado manualmente.

```
sudo rm -r /opt/nombre-del-programa
```

En resumen práctico: si tenemos suerte, **make uninstall** resolverá el problema. Si no existe esa opción, tendremos que reconstruir qué hizo **make install**, revisar las rutas instaladas y eliminar manualmente los archivos correspondientes.