Compilación e instalación de programas a partir de las fuentes

Las distribuciones Linux disponen de mecanismos que se ocupan de la instalación de paquetes de programas a partir de los repositorios (*repositories*) disponibles en la red. Esto permite una gestión completa de los elementos de software implementados en un sistema: listado, actualización y desinstalación.

Sin embargo, a veces es necesario instalar un programa a partir de sus archivos de código fuente, copiando estos últimos en el sistema local y compilándolos para poder generar los programas ejecutables correspondientes.

Este procedimiento solo debe ser utilizado como último recurso, cuando no exista ningún paquete para la distribución que estamos utilizando que contenga el programa que se quiere instalar o la versión deseada. Este tipo de instalación no será asumida por el gestor de aplicaciones estándar, que no conservará ninguna huella del elemento instalado y no podrá ni actualizarlo ni desinstalarlo.

También puede ser necesario aplicar parches (*patches*) a algunos programas instalados, en particular si son relativos a problemas de seguridad.

1. Instalación a partir de los archivos de código fuente

Para facilitar la instalación de una aplicación a partir de sus archivos de código fuente, un procedimiento estándar de compilación ha sido creado en el marco del proyecto GNU (*Gnu is Not Unix*). Este procedimiento uniformiza la operación para los diferentes programas de código libre y permite a usuarios que no tengan el perfil de programador poder hacerlo.

a. Búsqueda y descarga del código fuente

Generalmente, los archivos de código fuente de una aplicación pueden descargarse, bajo la forma de un archivo comprimido (*tarball file*).

La primera etapa consiste en descargar este archivo en el directorio donde se vaya a efectuar la compilación.

Después hay que descomprimir y restaurar el archivo, con el o los comandos estándares que soportan los formatos utilizados en la creación y compresión del archivo comprimido.

Después de esta operación encontraremos, generalmente en el directorio de instalación, un archivo de texto llamado `README`, el cual tiene que ser leído imperativamente antes de seguir el procedimiento. Puede contener información sobre el programa que se va a instalar e instrucciones para su configuración y su compilación.

Si se trata de una aplicación de código libre, deberíamos encontrar también en la raíz del directorio de instalación un archivo llamado `LICENSE`, que contendrá la información acerca de licencia del programa.

b. Descompresión de los archivos del código fuente

La mayoría de las veces, el archivo que contiene los archivos de código fuente está comprimido en alguno de los formatos diferentes que pueden identificarse gracias a la extensión del archivo:

Archivo.tar.Z

Archivo TAR, comprimido usando `compress`. Se descomprime usando el comando `uncompress Archivo.tar.Z` o con la opción `-Z` del comando `tar`.

Archivo.tar.gz

Archivo TAR, comprimido usando

`gzip`. Se descomprime usando el comando `gunzip Archivo.tar.gz` o con la opción `-z` del comando `tar`.

Archivo.tar.bz2

Archivo TAR, comprimido usando `bzip2`. Se descomprime usando el comando `bunzip2 Archivo.tar.bz` o con la opción `-j` del comando `tar`.

Archivo.tar.xz

Archivo TAR, comprimido usando `xz`. Se descomprime usando el comando `unxzArchivo.tar.xz` o con la opción `-J` el comando `tar`.

El archivo descomprimido ya no presenta ninguna extensión de archivo, el archivo original se pierde.

c. Restauración del archivo comprimido

El formato estándar para archivos comprimidos en Linux es el utilizado por el comando `tar`. Para restaurar el contenido de un archivo comprimido, una vez que haya sido descomprimido, se procede de la manera siguiente:

- Nos trasladamos al directorio de trabajo: la restauración de un archivo comprimido TAR crear los archivos y los directorios comprimidos en el directorio actual, por lo tanto hay que situarse en el directorio donde queramos ejecutar el procedimiento de generación de los ejecutables, y no en el directorio de instalación definitiva.

Es necesario tener derecho de escritura en ese directorio, sin embargo no se aconseja realizar el procedimiento siendo el superusuario del sistema (UID 0), ya que no es necesario y además es peligroso.

- Ejecutar el comando `tar`:

La línea de comando habitual es la siguiente:

```
tar [-]xvf archivo.tar
```

Donde:

`X`	Extracción del contenido del archivo comprimido.
`v`	Visualización detallada.
`f archivo.tar`	Archivo comprimido que se va a restaurar.

Si el archivo no ha sido descomprimido y el comando `tar` soporta el formato de compresión, habrá que añadir la opción correspondiente (`Z` para un archivo comprimido por `compress`, `z` para un archivo comprimido por `gzip`, `j` para un archivo comprimido por `bzip2` y `J` para un archivo comprimido por `xz`).

Ejemplo

Queremos instalar la última versión del servidor HTTP Nginx. Decidimos instalarla a partir del código fuente, ya que el paquete de esta versión todavía no está disponible en nuestra distribución.

El archivo comprimido se encuentra en la URL http://nginx.org/download.

Descargamos el archivo comprimido `nginx-1.17.10.tar.gz` :

```
wget http://nginx.org/download/nginx-1.17.10.tar.gz
--2021-10-05 08:50:16--  http://nginx.org/download/nginx-1.17.10.tar.gz
Resolviendo nginx.org (nginx.org)... 3.125.197.172, 52.58.199.22,
2a05:d014:edb:5702::6, ...
Conectando con nginx.org (nginx.org)[3.125.197.172]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 1039541 (1015K) [application/octet-stream]
Grabando a: “nginx-1.17.10.tar.gz”

nginx-1.17.10.tar.g 100%[===================>]   1015K  3,14MB/s    en 0,3s

2021-10-05 08:50:16 (3,14 MB/s) - “nginx-1.17.10.tar.gz” guardado
[1039541/1039541]

ls -l nginx*
-rw-r--r-- 1 pba  pba  1039541 abr 14  2020 nginx-1.17.10.tar.gz
```

Antes de restaurar el archivo comprimido, es prudente hacer un listado de su contenido. El archivo comprimido tiene un una extensión `tar.gz`*, se trata probablemente de un archivo TAR comprimido usando el comando* `gzip`*. Podemos usar la opción* `z` *del comando* `tar` *para gestionar la descompresión, y la opción* `t` *(table of contents) que mostrará el contenido del archivo sin restaurarlo:*

```
tar tvzf nginx-1.17.10.tar.gz
drwxr-xr-x mdounin/mdounin    0 2020-04-14 16:19 nginx-1.17.10/
drwxr-xr-x mdounin/mdounin    0 2020-04-14 16:19 nginx-1.17.10/auto/
drwxr-xr-x mdounin/mdounin    0 2020-04-14 16:19 nginx-1.17.10/conf/
drwxr-xr-x mdounin/mdounin    0 2020-04-14 16:19 nginx-1.17.10/contrib/
drwxr-xr-x mdounin/mdounin    0 2020-04-14 16:19 nginx-1.17.10/src/
-rwxr-xr-x mdounin/mdounin 2502 2020-04-14 16:19 nginx-1.17.10/configure
-rw-r--r-- mdounin/mdounin 1397 2020-04-14 16:19 nginx-1.17.10/LICENSE
-rw-r--r-- mdounin/mdounin   49 2020-04-14 16:19 nginx-1.17.10/README
[...]
```

Todos los archivos incluidos en el archivo comprimido se encuentran en el directorio `nginx-1.17.10` *, que será restaurado en el directorio actual. A priori, no hay ningún*

riesgo a la hora de restaurar el contenido del archivo, especialmente porque no se efectúa con la cuenta del usuario `root`*:*

```
tar xvzf nginx-1.17.10.tar.gz
[...]
nginx-1.17.10/auto/cc/gcc
nginx-1.17.10/auto/cc/icc
nginx-1.17.10/auto/cc/msvc
nginx-1.17.10/auto/cc/name
nginx-1.17.10/auto/cc/owc
nginx-1.17.10/auto/cc/sunc
```

El archivo comprimido ha sido restaurado en el subdirectorio `nginx-1.17.10` *del directorio actual:*

```
ls -l nginx-1.17.10
total 784
drwxr-xr-x 6 pba pba   4096 abril 15 14:38 auto
-rw-r--r-- 1 pba pba 302754 abril 14 16:19 CHANGES
-rw-r--r-- 1 pba pba 462076 abril 14 16:19 CHANGES.ru
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 conf
-rwxr-xr-x 1 pba pba   2502 abril 14 16:19 configure
drwxr-xr-x 4 pba pba   4096 abril 15 14:38 contrib
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 html
-rw-r--r-- 1 pba pba   1397 abril 14 16:19 LICENSE
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 man
-rw-r--r-- 1 pba pba     49 abril 14 16:19 README
drwxr-xr-x 9 pba pba   4096 abril 15 14:38 src
```

d. Configuración de la compilación

Antes de proceder a la compilación propiamente dicha, hay que efectuar una configuración en función del sistema donde se hará la compilación y de las diferentes opciones deseadas.

De esta configuración se encarga un script llamado `configure`, situado en el directorio donde vamos a extraer el archivo comprimido. Este script comprobará que el entorno

necesario para la compilación es correcto (versión del sistema, presencia de bibliotecas necesarias, etc.), y si fuera necesario, solicitará los valores de distintos parámetros, y después creará un archivo de texto conteniendo el conjunto de las instrucciones necesarias para la generación de los programas ejecutables que constituyen el programa: el archivo `Makefile`. Este archivo, que tiene que respetar una sintaxis específica, será leído e interpretado por el comando `make`.

La operación de configuración previa a la compilación se efectúa, por lo tanto, solicitando la ejecución del archivo `./configure`.

El script puede necesitar, opcionalmente, algunos argumentos de configuración cuando se ejecute. Hay que asegurarse de ello leyendo el archivo `README`, o lanzándolo con la opción `--help`.

Esta parte de la instalación no requiere derechos de administración del sistema. Por lo tanto, no se aconseja ejecutar el script de configuración con la cuenta del superusuario (UID 0).

El procedimiento de compilado depende del tipo de sistema objetivo. Frecuentemente hay que especificar el tipo de distribución y la versión del sistema local. El comando `uname -a` (*all*) permite mostrar las características del sistema.

Ejemplo

Uso del comando en una distribución Debian 10:

```
uname -a
Linux debian10 4.19.0-8-amd64 #1 SMP Debian 4.19.98-1 (2020-01-26) x86_64
GNU/Linux
```

Ejemplo

Continuamos con la instalación de una versión de Nginx a partir de los archivos de código fuente, empezada en el ejemplo anterior.

Entramos en el directorio raíz de la instalación y hacemos un listado de su contenido:

```
cd nginx-1.17.10/
ls -l
total 784
drwxr-xr-x 6 pba pba   4096 abril 15 14:38 auto
-rw-r--r-- 1 pba pba 302754 abril 14 16:19 CHANGES
-rw-r--r-- 1 pba pba 462076 abril 14 16:19 CHANGES.ru
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 conf
-rwxr-xr-x 1 pba pba   2502 abril 14 16:19 configure
drwxr-xr-x 4 pba pba   4096 abril 15 14:38 contrib
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 html
-rw-r--r-- 1 pba pba   1397 abril 14 16:19 LICENSE
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 man
-rw-r--r-- 1 pba pba     49 abril 14 16:19 README
drwxr-xr-x 9 pba pba   4096 abril 15 14:38 src
```

El archivo `configure` está configurado con el derecho de ejecución para cualquier usuario, por eso se podrá solicitar su ejecución directamente.

Leemos el contenido del archivo `README`:

```
vi README
Documentation is available at http://nginx.org
```

Se puede comprobar que el script comprueba el tipo de sistema en el que se hará la instalación con el comando `uname` y diferentes opciones:

```
grep uname configure
  NGX_SYSTEM=`uname -s 2>/dev/null`
  NGX_RELEASE=`uname -r 2>/dev/null`
  NGX_MACHINE=`uname -m 2>/dev/null`
```

Ejecutamos el script de configuración de la instalación:

```
./configure
checking for OS
 + Linux 4.19.0-8-amd64 x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version:8.3.0 (Debian 8.3.0-6)
checking for gcc -pipe switch ... found
checking for -Wl,-E switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for gcc builtin 64 bit byteswap ... found
checking for unistd.h ... found
checking for inttypes.h ... found
checking for limits.h ... found
checking for sys/filio.h ... not found
checking for sys/param.h ... found
checking for sys/mount.h ... found
checking for sys/statvfs.h ... found
checking for crypt.h ... found
checking for Linux specific features
[...]
checking for getaddrinfo() ... found
checking for PCRE library ... not found
checking for PCRE library in /usr/local/ ... not found
checking for PCRE library in /usr/include/pcre/ ... not found
checking for PCRE library in /usr/pkg/ ... not found
checking for PCRE library in /opt/local/ ... not found

./configure:error:the HTTP rewrite module requires the PCRE library.
You can either disable the module by using --without-http_rewrite_module
option, or install the PCRE library into the system, or build the PCRE
library statically from the source with nginx by using
--with-pcre=<path> option.
```

Como suele producirse a menudo, el script detecta la ausencia de un elemento requerido para la instalación del servidor y termina en error. Hay que proceder a la instalación del elemento necesario y volver a ejecutar el procedimiento. En este caso, consideramos que el módulo ausente no será necesario y desactivamos su integración en Nginx:

```
./configure --without-http_rewrite_module
[...]
./configure:error:the HTTP gzip module requires the zlib library.
You can either disable the module by using --without-http_gzip_module
option, or install the zlib library into the system, or build the zlib library
statically from the source with nginx by using --with-zlib=<path> option.
```

Esta vez falta el módulo HTTP `gzip`. Consideramos que tampoco será necesario para nuestro servidor Ngnix y relanzamos el procedimiento.

```
./configure --without-http_rewrite_module  --without-http_gzip_module
[...]
creating objs/Makefile

Configuration summary
 + PCRE library is not used
 + OpenSSL library is not used
 + zlib library is not used

 nginx path prefix:"/usr/local/nginx"
 nginx binary file:"/usr/local/nginx/sbin/nginx"
 nginx modules path:"/usr/local/nginx/modules"
 nginx configuration prefix:"/usr/local/nginx/conf"
 nginx configuration file:"/usr/local/nginx/conf/nginx.conf"
 nginx pid file:"/usr/local/nginx/logs/nginx.pid"
 nginx error log file:"/usr/local/nginx/logs/error.log"
 nginx http access log file:"/usr/local/nginx/logs/access.log"
 nginx http client request body temporary files:"client_body_temp"
 nginx http proxy temporary files:"proxy_temp"
 nginx http fastcgi temporary files:"fastcgi_temp"
 nginx http uwsgi temporary files:"uwsgi_temp"
 nginx http scgi temporary files:"scgi_temp"
```

La configuración de la instalación ha terminado.

```
ls -l
```

```
total 792
drwxr-xr-x 6 pba pba   4096 abril 15 14:38 auto
-rw-r--r-- 1 pba pba 302754 abril 14 16:19 CHANGES
-rw-r--r-- 1 pba pba 462076 abril 14 16:19 CHANGES.ru
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 conf
-rwxr-xr-x 1 pba pba   2502 abril 14 16:19 configure
drwxr-xr-x 4 pba pba   4096 abril 15 14:38 contrib
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 html
-rw-r--r-- 1 pba pba   1397 abril 14 16:19 LICENSE
-rw-r--r-- 1 pba pba    376 abril 15 15:08 Makefile
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 man
drwxr-xr-x 3 pba pba   4096 abril 15 15:08 objs
-rw-r--r-- 1 pba pba     49 abril 14 16:19 README
drwxr-xr-x 9 pba pba   4096 abril 15 14:38 src
```

El script ha creado un archivo `Makefile`, que será usado para la fase de compilación e instalación.

e. Compilación

Una vez que el archivo `Makefile` haya sido generado, la compilación se hace lanzando el comando `make`. Tenemos que estar seguros de estar en el directorio correcto, porque el comando leerá el archivo `Makefile` del directorio actual.

Dependiendo de la aplicación que haya que generar, el comando puede durar más o menos tiempo, ocupar más o menos espacio y solicitar más o menos tiempo al procesador.

Los archivos ejecutables generados por el comando se encuentran en la arborescencia dentro del directorio actual y no en los directorios definitivos de instalación.

Esta parte de la instalación no precisa ningún derecho de administración del sistema. Por lo tanto se aconseja ejecutar el script con una cuenta normal y no con la cuenta del superusuario (UID 0).

Ejemplo

Siguiendo los ejemplos precedentes, procedemos a la compilación de la aplicación servidor HTTP Nginx.

Examinamos el contenido del archivo `Makefile` *:*

vi Makefile

```
default:    build
clean:
	rm -rf Makefile objs
build:
	$(MAKE) -f objs/Makefile
install:
	$(MAKE) -f objs/Makefile install
modules:
	$(MAKE) -f objs/Makefile modules
upgrade:
	/usr/local/nginx/sbin/nginx -t
	kill -USR2 `cat /usr/local/nginx/logs/nginx.pid`
	sleep 1
	test -f /usr/local/nginx/logs/nginx.pid.oldbin
	kill -QUIT `cat /usr/local/nginx/logs/nginx.pid.oldbin`
```

La sintaxis del archivo es específica para el comando `make` *. Los objetivos (una palabra seguida del carácter ":") describen diferentes operaciones y líneas de comandos que se tienen que ejecutar para generarlos. Sin argumento, el comando* `make` *ejecutará el objetivo especificado en* `default` *, aquí* `build` *, es decir, la generación de los ejecutables.*

Ejecutamos el comando `make` *:*

make

```
make -f objs/Makefile
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g
-I src/core -I src/event -I src/event/modules -I src/os/unix -I objs -I src/http
-I src/http/modules \
	-o objs/src/http/ngx_http_postpone_filter_module.o \
	src/http/ngx_http_postpone_filter_module.c
```

```
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g
-I src/core -I src/event -I src/event/modules -I src/os/unix -I objs -I src/http
-I src/http/modules \
    -o objs/src/http/modules/ngx_http_ssi_filter_module.o \
    src/http/modules/ngx_http_ssi_filter_module.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g
-I src/core -I src/event -I src/event/modules -I src/os/unix -I objs -I src/http
-I src/http/modules \
    -o objs/src/http/modules/ngx_http_charset_filter_module.o \
    src/http/modules/ngx_http_charset_filter_module.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g
-I src/core -I src/event -I src/event/modules -I src/os/unix -I objs -I src/http
-I src/http/modules \
    -o objs/src/http/modules/ngx_http_userid_filter_module.o \
    src/http/modules/ngx_http_userid_filter_module.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g
-I src/core -I src/event -I src/event/modules -I src/os/unix -I objs -I src/http
-I src/http/modules \
    -o objs/src/http/modules/ngx_http_headers_filter_module.o \
    src/http/modules/ngx_http_headers_filter_module.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g
-I src/core -I src/event -I src/event/modules -I src/os/unix -I objs -I src/http
-I src/http/modules \
    -o objs/src/http/ngx_http_copy_filter_module.o \
    src/http/ngx_http_copy_filter_module.c
[...]
objs/src/http/modules/ngx_http_upstream_keepalive_module.o \
objs/src/http/modules/ngx_http_upstream_zone_module.o \
objs/ngx_modules.o \
-ldl -lpthread -lcrypt \
-Wl,-E
sed -e "s|%%PREFIX%%|/usr/local/nginx|" \
    -e "s|%%PID_PATH%%|/usr/local/nginx/logs/nginx.pid|" \
    -e "s|%%CONF_PATH%%|/usr/local/nginx/conf/nginx.conf|" \
    -e "s|%%ERROR_LOG_PATH%%|/usr/local/nginx/logs/error.log|" \
    < man/nginx.8 > objs/nginx.8
make[1]: se sale del directorio '/home/pba/nginx-1.17.10'
```

La compilación se ha terminado. Los módulos objetos y los ejecutables se encuentran en el

subdirectorio `objs`*:*

```
ls -ltr objs
total 4860
-rw-r--r-- 1 pba pba    657 abril 15 15:07 ngx_auto_headers.h
-rw-r--r-- 1 pba pba  17339 abril 15 15:08 autoconf.err
-rw-r--r-- 1 pba pba   5544 abril 15 15:08 ngx_modules.c
drwxr-xr-x 9 pba pba   4096 abril 15 15:08 src
-rw-r--r-- 1 pba pba  39124 abril 15 15:08 Makefile
-rw-r--r-- 1 pba pba   6841 abril 15 15:08 ngx_auto_config.h
-rw-r--r-- 1 pba pba  41888 abril 15 15:56 ngx_modules.o
-rwxr-xr-x 1 pba pba 4833592 abril 15 15:56 nginx
-rw-r--r-- 1 pba pba   5375 abril 15 15:56 nginx.8
file objs/nginx
objs/nginx:ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/
Linux 3.2.0, BuildID[sha1]=0617031ac5d1d66b4cca640e52838cebcf154f8b, with
debug_info, not stripped
```

El ejecutable es `objs/nginx` . Se puede observar que esta fase se ha ejecutado con el usuario `pba`, y no con el usuario `root`.

f. Instalación de los ejecutables

La etapa siguiente consiste en instalar los diferentes elementos de la aplicación (ejecutables, archivos de configuración y documentación) en sus directorios definitivos. Para ello, lanzamos el comando `make install`.

En esta parte de la instalación se necesitan, en general, los derechos de administración del sistema.

Ejemplo

Procedemos a la instalación del servidor Nginx compilado en el ejemplo anterior.

```
make install
make -f objs/Makefile install
make[1]: se entra en el directorio '/home/pba/nginx-1.17.10'
test -d '/usr/local/nginx' || mkdir -p '/usr/local/nginx'
test -d '/usr/local/nginx/sbin' \
    || mkdir -p '/usr/local/nginx/sbin'
test ! -f '/usr/local/nginx/sbin/nginx' \
    || mv '/usr/local/nginx/sbin/nginx' \
        '/usr/local/nginx/sbin/nginx.old'
mv: no se puede mover '/usr/local/nginx/sbin/nginx' a '/usr/local/nginx/
sbin/nginx.old': Permiso denegado
make[1]: *** [objs/Makefile:1180: install] Error 1
make[1]: se sale del directorio '/home/pba/nginx-1.17.10'
make: *** [Makefile:11: install] Error 2
```

La instalación ha fallado, porque para crear un directorio en el directorio `/usr/local`, hay que ser superusuario (UID=0).

```
su -c "make install"
Contraseña: XXX
make[1]: se entra en el directorio '/home/pba/nginx-1.17.10'
test -d '/usr/local/nginx' || mkdir -p '/usr/local/nginx'
test -d '/usr/local/nginx/sbin' \
    || mkdir -p '/usr/local/nginx/sbin'
test ! -f '/usr/local/nginx/sbin/nginx' \
    || mv '/usr/local/nginx/sbin/nginx' \
        '/usr/local/nginx/sbin/nginx.old'
cp objs/nginx '/usr/local/nginx/sbin/nginx'
test -d '/usr/local/nginx/conf' \
    || mkdir -p '/usr/local/nginx/conf'
cp conf/koi-win '/usr/local/nginx/conf'
cp conf/koi-utf '/usr/local/nginx/conf'
cp conf/win-utf '/usr/local/nginx/conf'
test -f '/usr/local/nginx/conf/mime.types' \
    || cp conf/mime.types '/usr/local/nginx/conf'
cp conf/mime.types '/usr/local/nginx/conf/mime.types.default'
test -f '/usr/local/nginx/conf/fastcgi_params' \
    || cp conf/fastcgi_params '/usr/local/nginx/conf'
cp conf/fastcgi_params \
```

```
    '/usr/local/nginx/conf/fastcgi_params.default'
test -f '/usr/local/nginx/conf/fastcgi.conf' \
    || cp conf/fastcgi.conf '/usr/local/nginx/conf'
cp conf/fastcgi.conf '/usr/local/nginx/conf/fastcgi.conf.default'
test -f '/usr/local/nginx/conf/uwsgi_params' \
    || cp conf/uwsgi_params '/usr/local/nginx/conf'
cp conf/uwsgi_params \
    '/usr/local/nginx/conf/uwsgi_params.default'
test -f '/usr/local/nginx/conf/scgi_params' \
    || cp conf/scgi_params '/usr/local/nginx/conf'
cp conf/scgi_params \
    '/usr/local/nginx/conf/scgi_params.default'
test -f '/usr/local/nginx/conf/nginx.conf' \
    || cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf'
cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf.default'
test -d '/usr/local/nginx/logs' \
    || mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/logs' \
    || mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/html' \
    || cp -R html '/usr/local/nginx'
test -d '/usr/local/nginx/logs' \
    || mkdir -p '/usr/local/nginx/logs'
make[1]: se sale del directorio '/home/pba/nginx-1.17.10'
```

La aplicación se ha instalado en el directorio `/usr/local/nginx` :

```
ls -l /usr/local/nginx
total 16
drwxr-xr-x 2 root root 4096 abril 15 16:06 conf
drwxr-xr-x 2 root root 4096 abril 15 16:06 html
drwxr-xr-x 2 root root 4096 abril 15 16:06 logs
drwxr-xr-x 2 root root 4096 abril 15 16:06 sbin
```

El ejecutable es: `/usr/local/nginx/sbin/nginx`

```
ls -l /usr/local/nginx/sbin/nginx
```

```
-rwxr-xr-x 1 root root 4833592 abril 15 16:06 /usr/local/nginx/sbin/nginx
```

Podemos comprobar la versión del ejecutable lanzándolo con la opción `-V`:

```
/usr/local/nginx/sbin/nginx -V
nginx version:nginx/1.17.10
built by gcc 8.3.0 (Debian 8.3.0-6)
configure arguments:--without-http_rewrite_module --without-http_gzip_module
```

g. Limpieza de los archivos de código fuente

En caso de problema durante la instalación, a veces es necesario volver a ejecutar el procedimiento desde el principio. Pero antes habrá que "limpiar" el directorio de instalación y su arborescencia suprimiendo los elementos que han sido generados anteriormente.

Para ello, según el grado de reinicialización deseado del procedimiento de instalación, se pueden ejecutar los comandos `make clean` o `make mrproper`.

```
make clean
```

Este comando suprime los archivos de tipo objeto ejecutables generados por el comando `make`.

```
make mrproper
```

Este comando suprime todos los archivos generados por el uso del comando `make` (archivos de tipo objeto, ejecutables, archivos de configuración y de dependencia, etc.).

Ejemplo

Queremos suprimir los archivos creados durante una compilación anterior de los archivos fuente de `nginx`.

Examinamos el archivo `Makefile` *:*

```
cat Makefile
default:    build
clean:
    rm -rf Makefile objs
build:
    $(MAKE) -f objs/Makefile
install:
    $(MAKE) -f objs/Makefile install
modules:
    $(MAKE) -f objs/Makefile modules
upgrade:
    /usr/local/nginx/sbin/nginx -t
    kill -USR2 `cat /usr/local/nginx/logs/nginx.pid`
    sleep 1
    test -f /usr/local/nginx/logs/nginx.pid.oldbin
    kill -QUIT `cat /usr/local/nginx/logs/nginx.pid.oldbin`
```

Hay un objetivo `clean`*, pero no existe el objetivo* `mrproper`*. Solo podemos "limpiar" con el comando* `make clean` *:*

```
ls -l
total 792
drwxr-xr-x 6 pba pba   4096 abril 15 14:38 auto
-rw-r--r-- 1 pba pba 302754 abril 14 16:19 CHANGES
-rw-r--r-- 1 pba pba 462076 abril 14 16:19 CHANGES.ru
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 conf
-rwxr-xr-x 1 pba pba   2502 abril 14 16:19 configure
drwxr-xr-x 4 pba pba   4096 abril 15 14:38 contrib
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 html
-rw-r--r-- 1 pba pba   1397 abril 14 16:19 LICENSE
-rw-r--r-- 1 pba pba    376 abril 15 15:08 Makefile
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 man
drwxr-xr-x 3 pba pba   4096 abril 15 15:56 objs
-rw-r--r-- 1 pba pba     49 abril 14 16:19 README
drwxr-xr-x 9 pba pba   4096 abril 15 14:38 src
```

Ejecutamos el comando:

```
make clean
rm -rf Makefile objs
ls -l
total 784
drwxr-xr-x 6 pba pba   4096 abril 15 14:38 auto
-rw-r--r-- 1 pba pba 302754 abril 14 16:19 CHANGES
-rw-r--r-- 1 pba pba 462076 abril 14 16:19 CHANGES.ru
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 conf
-rwxr-xr-x 1 pba pba   2502 abril 14 16:19 configure
drwxr-xr-x 4 pba pba   4096 abril 15 14:38 contrib
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 html
-rw-r--r-- 1 pba pba   1397 abril 14 16:19 LICENSE
drwxr-xr-x 2 pba pba   4096 abril 15 14:38 man
-rw-r--r-- 1 pba pba     49 abril 14 16:19 README
drwxr-xr-x 9 pba pba   4096 abril 15 14:38 src
```

El comando ha suprimido el directorio `objs` y su contenido, así como el archivo `Makefile`.

h. Desinstalación de un programa

El comando `make uninstall` permite desinstalar los elementos instalados por el comando `make install`. Se tiene que ejecutar desde el directorio desde el que se lanzó la instalación.

El objetivo del comando `make` es posible que no exista. En ese caso, hay que consultar la documentación de la aplicación para obtener el procedimiento de desinstalación.

2. Instalación de correctivos de software (patches)

Algunas aplicaciones a veces se actualizan utilizando archivos de corrección, llamados patches. Estos parches, a menudo relativos a problemas de seguridad que hay que corregir rápidamente, modifican directamente los archivos de código fuente para generar una nueva versión del ejecutable. Se aplican usando el comando `patch`.

Sintaxis

```
patch [Opciones] < ArchivoPatch
```

Parámetros principales

`Opciones`	Opciones.
`ArchivoPatch`	Archivo con el parche que se aplicará.

Descripción

El comando aplica los parches contenidos en el archivo `ArchivoPatch`.

Las opciones más corrientes son:

```
-p[de 0 a n]
```

Suprime de `0` a n niveles en los caminos de acceso contenidos en el parche. Por ejemplo, si el parche se refiere a `/usr/dev/src/main.c`, la opción `-p3` usará `src/main.c`.

```
-b
```

Hace una copia de respaldo (*backup*) de los archivos originales.

```
--dry-run
```

Muestra los resultados del parche, pero sin aplicarlo realmente.

-f

Fuerza (*force*) la ejecución en modo silencioso del parche, respondiendo "sí" a todas las preguntas.

--verbose

Visualización detallada.

-R

Supresión del parche. El comando anula el parche aplicado.

Un archivo de parche es el resultado del comando `diff`, aplicado a diferentes versiones de los archivos. Aplicando el parche, ejecutamos las operaciones necesarias para que los archivos originales sean idénticos a los archivos de destino.

Ejemplo

En la rama de desarrollo `nginx`, en el sitio de desarrollo GitHub, recuperamos un archivo de parche para `nginx`: https://github.com/kn007/patch/blob/master/nginx.patch

Leemos el contenido del archivo, que indica en sus primeras líneas cómo aplicar el parche:

```
vi nginx.patch
Add HTTP2 HPACK Encoding Support.
Add Dynamic TLS Record support.

Using:patch -p1 < nginx.patch

diff -uNr a/auto/modules b/auto/modules
[...]
```

Comprobamos el parche sin aplicarlo realmente:

```
patch --dry-run -p1 < nginx.patch
checking file auto/modules
checking file auto/options
checking file src/core/ngx_murmurhash.c
checking file src/core/ngx_murmurhash.h
checking file src/event/ngx_event_openssl.c
checking file src/event/ngx_event_openssl.h
checking file src/http/modules/ngx_http_ssl_module.c
checking file src/http/modules/ngx_http_ssl_module.h
checking file src/http/v2/ngx_http_v2.c
checking file src/http/v2/ngx_http_v2_encode.c
checking file src/http/v2/ngx_http_v2_filter_module.c
checking file src/http/v2/ngx_http_v2.h
checking file src/http/v2/ngx_http_v2_table.c
```

Ejecutamos el parche, como se especifica en el archivo, y con la opción de respaldo `-b`:

```
patch -b -p1 < nginx.patch
patching file auto/modules
patching file auto/options
patching file src/core/ngx_murmurhash.c
patching file src/core/ngx_murmurhash.h
patching file src/event/ngx_event_openssl.c
patching file src/event/ngx_event_openssl.h
patching file src/http/modules/ngx_http_ssl_module.c
patching file src/http/modules/ngx_http_ssl_module.h
patching file src/http/v2/ngx_http_v2.c
patching file src/http/v2/ngx_http_v2_encode.c
patching file src/http/v2/ngx_http_v2_filter_module.c
patching file src/http/v2/ngx_http_v2.h
patching file src/http/v2/ngx_http_v2_table.c
```

Podemos ver los archivos modificados y las copias de respaldo con la extensión `.orig`:

```
ls -l src/http/v2/
```

```
total 632
-rw-r--r-- 1 pba pba 132369 abril 15 16:58 ngx_http_v2.c
-rw-r--r-- 1 pba pba 132034 abril 14 16:19 ngx_http_v2.c.orig
-rw-r--r-- 1 pba pba   1177 abril 15 16:58 ngx_http_v2_encode.c
-rw-r--r-- 1 pba pba   1191 abril 14 16:19 ngx_http_v2_encode.c.orig
-rw-r--r-- 1 pba pba  56559 abril 15 16:58 ngx_http_v2_filter_module.c
-rw-r--r-- 1 pba pba  58000 abril 14 16:19 ngx_http_v2_filter_module.c.orig
-rw-r--r-- 1 pba pba  17146 abril 15 16:58 ngx_http_v2.h
-rw-r--r-- 1 pba pba  13553 abril 14 16:19 ngx_http_v2.h.orig
-rw-r--r-- 1 pba pba 128803 abril 14 16:19 ngx_http_v2_huff_decode.c
-rw-r--r-- 1 pba pba  12829 abril 14 16:19 ngx_http_v2_huff_encode.c
-rw-r--r-- 1 pba pba  16454 abril 14 16:19 ngx_http_v2_module.c
-rw-r--r-- 1 pba pba   1261 abril 14 16:19 ngx_http_v2_module.h
-rw-r--r-- 1 pba pba  25450 abril 15 16:58 ngx_http_v2_table.c
-rw-r--r-- 1 pba pba  11697 abril 14 16:19 ngx_http_v2_table.c.orig
```

Respaldamos la versión actual del ejecutable (con la cuenta de `root`):

```
cp /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx-nopatch
```

Generamos el nuevo ejecutable (si hemos ejecutado anteriormente el comando `make clean`, habrá que ejecutar primero el script `configure` para volver a crear el archivo `Makefile`):

```
make
[...]
-Wl,-E
sed -e "s|%%PREFIX%%|/usr/local/nginx|" \
    -e "s|%%PID_PATH%%|/usr/local/nginx/logs/nginx.pid|" \
    -e "s|%%CONF_PATH%%|/usr/local/nginx/conf/nginx.conf|" \
    -e "s|%%ERROR_LOG_PATH%%|/usr/local/nginx/logs/error.log|" \
    < man/nginx.8 > objs/nginx.8
make[1]: se sale del directorio '/home/pba/nginx-1.17.10'
```

Instalamos el nuevo ejecutable:

```
make install
[...]
test -d '/usr/local/nginx/logs' \
     || mkdir -p '/usr/local/nginx/logs'
make[1]: se sale del directorio '/home/pba/nginx-1.17.10'
```

Se ha creado una nueva versión:

```
ls -l /usr/local/nginx/sbin
total 14172
-rwxr-xr-x 1 root root 4835696 abril 15 17:03 nginx
-rwxr-xr-x 1 root root 4833592 abril 15 17:00 nginx-nopatch
-rwxr-xr-x 1 root root 4833592 abril 15 16:06 nginx.old
```