How To: Solve Installation Problems
PAINSTAKINGLY RECREATED

HANS-GEORG EßER

The installation of Linux programs from source code generally follows the ”classic” three step path of ./configure, make and make install. But what happens if, for instance, packages required during the ”make” process are missing? We examine some typical problems.

When installing a program from a source code archive the aim is to create executable binary programs that can be run by the computer. To do this, a compiler must be installed. As Linux programs are normally written in C or C++ you will need a C compiler. In most distributions this is part of the *egcs* package, which must be installed.

But the compiler alone is not sufficient. During compilation what is known as a *makefile* is used. This file contains the instructions as to how the program will be built. The program *make* reads this file and carries out these instructions: this must also be installed.

To check whether these packages are present run the commands:

```
rpm -q egcs
rpm -q make
```

rpm should respond in each case with the package name and a version number (e.g. *egcs-1.1.2-30* and *make-3.78.1-4*). If it tells you that the package is missing, look on the installation CD of your Linux distribution for the corresponding packages (e.g. *egcs-1.1.2-30.i386.rpm* and *make-3.78.14.i386.rpm*) and install these while logged in as *root* using the command:

```
rpm -Uvh package-name.rpm
```

The first step after unpacking the source code archive (and changing to the directory containing the code) is the configuration of the sources. This is the step during which the *makefile* is created. The command for this is:

```
./configure
```

The dot and slash at the start are normally needed so that your shell searches in the current folder for the script *configure*. The shell runs this script which searches through your system checking for the presence of the programs and libraries which are required to successfully compile the new software. If you want to install, say, a KDE program, X Window and KDE **libraries** and the respective **include files** at the very least are required. The same applies to GNOME programs.

If files needed to compile the program are missing, *./configure* will produce error messages like this,

```
checking for X... configure: error: Can't fi➘
nd X includes.
Please check your installation and add the c➘
orrect paths!
```

The packages you’ll most often need are:

```
XFree86-devel-3.3.6-20.i386.rpm
kdelibs-1.1.2-15.i386.rpm
kdelibs-devel-1.1.2-15.i386.rpm
qt-2.1.0-4.beta1.i386.rpm
qt-devel-2.1.0-4.beta1.i386.rpm
gnome-libs-1.2.0-0mdk_helix_2.i586.rpm
gnome-libs-devel-1.2.0-0mdk_helix_2.i586.rpm
```

The version numbers shown above are of course only examples. If queries *rpm -q XFree86-devel*, *rpm -q kdelibs-devel*, *rpm -q gnome-libs-devel* etc. give you the reply ”package … is not installed”, look on your distribution CD for similarly named packages and install these as explained above). Note that GNOME libraries are not required for KDE programs and vice versa. ■

libraries: *Libraries contain program code that is designed to be called by other programs. This avoids the need for each program to have its own version of this code, saving programmertime and memory usage. Libraries usually reside in /usr/lib/, /usr/local/lib/ or their subfolders. Only a few important system libraries are located directly in /lib.*

include files: *Include files provide information to the compiler as to how the program gains access to the libraries. They have a file extension of .h and are usually found in /usr/include, /usr/local/include and their subfolders* ■

KDEDIR and QTDIR

When compiling KDE programs a common problem is that the environment variables KDEDIR and QTDIR are not correctly set. This occurs if both Qt library versions (1.x and 2.x) are installed. If you encounter problems compiling, set QTDIR to the correct folder (for example with export QTDIR=/usr/lib/qt-2.1.0). KDEDIR should also be correctly set. With most distributions the correct value is /opt/kde or /usr.