



## RPM-devel package dependencies

In this document I will try to describe a problem with the dependencies of so called "-devel" rpm packages and describe a number of ways in which this problem may be solved.

Many rpm packages have been split up into a number of sub packages:

- package containing binaries
- package containing libraries ( % { \_libdir } / \* .so . \* )
- package containing -devel files ( % { \_includedir } / \* .h % { \_libdir } / \* .la % { \_libdir } / \* .so )
- package containing –static-devel files ( % { \_libdir } / \* .a )

(for more details on which files belong where, see Mandrake's RPM Howto, [libpolicy](#) section)

With some packages this happened earlier than with others. Some packages have always been split up into sub packages. In this document I don't want to get into the exact history of this.

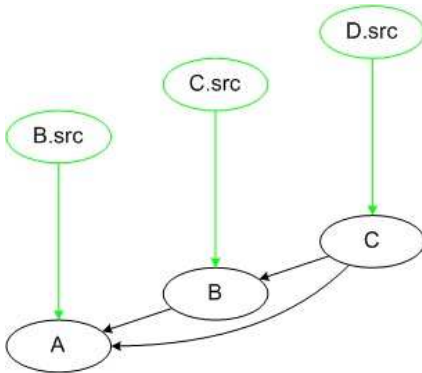
Let's make the assumption that once upon a time there were 4 packages. Package A, B, C and D. These packages have the following dependency structure:

- C Requires B and A
- B Requires A

The following dependencies are in place for the source packages (C.src and D.src):

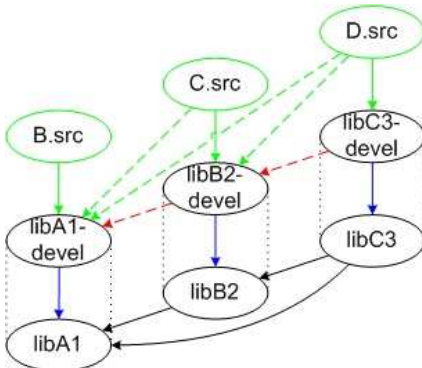
- C.src [BuildRequires](#) B
- D.src [BuildRequires](#) C

This picture reflects this situation:



The black lines in this picture reflect dependencies (Requires) which have been found by find-provides and find-requires.

Then the distribution decided to split the packages up and apply some fancy naming to the packages. This is very similar to the current state of many distributions. The following picture reflects this situation:



The blue lines (between lib\*-devel and lib\*) in the picture reflect the manual dependencies that have been added to the spec file of the src.rpm of A, B and C. The dotted line shows that those two packages originate from the same src.rpm. The red line between the -devel packages represent the optional dependencies that a developer may put into the .spec file. Since they are optional they are dashed. The [BuildRequires](#) (the green lines) have to be placed

according to the (manual) dependencies that have or haven't been set between the -devel packages, these result in being dashed. The [BuildRequires](#) for the "top level" -devel package is always needed and therefore it's a solid line. These optional dependencies on the -devel file make it difficult to apply the correct [BuildRequires](#) to the package.

What I'm looking for is to find a way to correctly and reliably provide dependency information to the -devel packages. There might be several ways to do this:

### Option 1: Find dependency information from .h (header) files

A -devel package consist of header files and symbolic links to the shared library. The header files contain information that can be used to find dependencies. However, it's not as simple as just grepping through the header files for include statements. The mechanism will need to be able to cope with the define statements in the header files. Perhaps using tools like:

Perl:

- C-Include <http://search.cpan.org/author/AMICHAUER/C-Include-1.40/Include.pm>
- C-Scan <http://search.cpan.org/author/HVDS/C-Scan-0.74/Scan.pm>

C:

- Cflow <http://www.unet.univie.ac.at/aix/cmds/aixcmds1/cflow.htm> <http://packages.debian.org/stable/devel/cflow.html>

Pro's

- ?

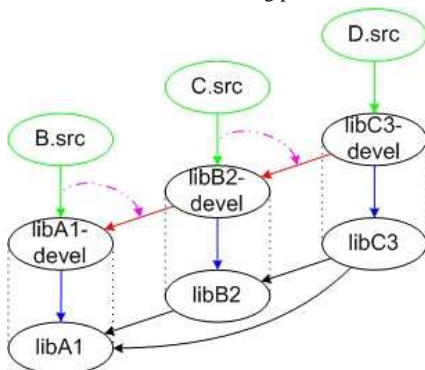
Con's

- It's difficult to extract information from header files;
- It will require adding "Provides" and "Requires" for header files;
- The dependency information distilled from this may not be correct;

### Option 2: Re-use [BuildRequires](#) as dependency information

It might be possible to use the [BuildRequires](#) of the src.rpm for the Requires of the -devel package resulting from the src.rpm. Perhaps grep on "-devel" (to filter out other [BuildRequires](#) like bison & flex). This can be done by an automatic process (find-provides & find-requires) or manually (entries in the .spec file). The manual process can be backed up by a rpmlint check.

This will result in the following picture:



The [BuildRequires](#) are now clear, only the "top level" [BuildRequires](#) are needed. The purple dashed dotted lines show where the dependency information came from. In this case the dependency information between the -devel packages was obtained from the [BuildRequires](#).

Pro's

- Does not require new Provides to be set;

Con's

- The -devel dependencies will rely on [BuildRequires](#). [BuildRequires](#) are often not correct, also due to the -devel dependencies not being correct;
- There is no reliable way to find [BuildRequires](#) (at the moment);
- Keeping the [BuildRequires](#) and -devel dependencies correct will be a constant evolving process, since they influence each other. Multiple iterations of rebuilding (all?) packages will be required to get it stable.

### Option 3: Re-use Requires from binary package as dependency information

Find a valid .so\$ file. In general a valid .so\$ file must comply to these rules:

- filename ends on ".so";
- objdump -p contains "SONAME";
- be a symlink.

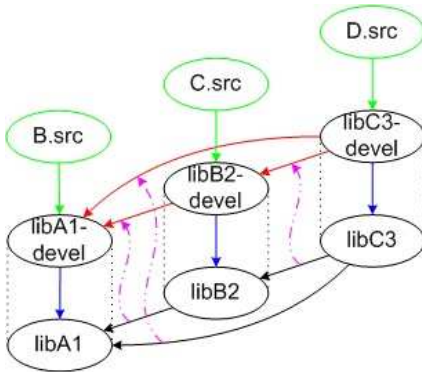
The Provides is obtained from the "SONAME" filed of objdump -p, with the version information behind the .so stripped off. The Requires based on the Requires that is found on the (binary) lib package, but strip off the versioning (Requires libasound.so.2 for the lib package becomes Requires devel(libasound) for the -devel package):

libao2-0.8.3-2mdk	libao2-devel-0.8.3-2mdk
libasound.so.2	devel(libasound)
libc.so.6	devel(libc)
libc.so.6(GLIBC_2.0)	devel(libc)
libc.so.6(GLIBC_2.1)	devel(libc)
libc.so.6(GLIBC_2.1.3)	devel(libc)
libdl.so.2	devel(libdl)
libdl.so.2(GLIBC_2.0)	devel(libdl)
libdl.so.2(GLIBC_2.1)	devel(libdl)
libm.so.6	devel(libdm)
libpthread.so.0	devel(libpthread)
libpthread.so.0(GLIBC_2.0)	devel(libpthread)

In the glibc-devel package won't Provide a number of these dependencies, so they are grepped out of the Requires of other packages. These include: "libc.so", "libpthread.so" and "librt.so". When building on hosts with a 2.6 kernel "linux-gate.so" also needs to be grepped out.

"librt.so" used to be a regular file but was recently (with 2.3.3-3mdk) turned into a symlink again, which makes it Provide devel(librt) for the glibc-devel package. An alternative is to add these (libc.so, libpthread.so and linux-gate.so) Provides to the glibc-devel package manually so they don't need to be grepped out on the Requires side.

This will result in the following picture:



The [BuildRequires](#) are now clear, only the "top level" [BuildRequires](#) are needed. The purple dashed dotted lines show where the dependency information came from. In this case the dependency information between the -devel packages was obtained from the Requires of the lib package.

#### Pro's

- Only need to touch each package once, no need for an iterating process;
- Dependency information for the -devel package covers all dependencies (all the way down to libc), just like with it binary (lib) counterpart;

#### Con's

- Some packages / software need some bugs to be ironed out.

#### Usecases

This is a short list of packages that have some weirdness, so I can use them as a "usecases":

package name	use case	example

glibc-devel	filter out some (non symlink lib*.so files)	/usr/lib/libcrypt.so -> ../lib/libcrypt.so.1
glibc-devel	filter out some (non symlink lib*.so files) --> don't include	/usr/lib/libpthread.so
libSDL_ttf2.0-devel	version name in symlink destination	/usr/lib/libSDL_ttf.so -> libSDL_ttf-2.0.so.0.0.5
libstdc++5-devel	"++" in the name	/usr/lib/gcc-lib/i586-mandrake-linux-gnu/3.2.3/libstdc++.so -> ../libstdc++.so.5.0.3
pam	aren't real .so's --> don't include	/lib/security/pam_unix_acct.so -> pam_unix.so
rpm-devel	(version name in symlink destination, symlink destination not a .so.*	/usr/lib/librpm.so -> librpm-4.2.so
zlib1-devel	normal package?	/usr/lib/libz.so -> libz.so.1.1.4

### Exceptions:

- gnome-games:

These are plugins for a game and cannot be moved to a -devel package. According to the script the package will Requires glibc-devel. These files have to be converted to "plugin" style .so files (not a symlink).

```
gnome-games Provides: devel(libgnome-stones)
gnome-games Provides: devel(libgnomekoban)
gnome-games Requires: devel(libm)
```

### Implementation examples

Current implementation (as implemented in rpm-build-4.2-25mdk):

find-provides:

```
#
# --- .so files.
for i in `echo $filelist | tr '[:blank:]' "\n" | egrep '/usr(/X11R6)?/lib(|64)(/gcc-lib/.+)?/[^/]+\so$`; do
  objd=`objdump -p ${i} | grep SONAME`
  [ -h ${i} -a -n "${objd}" ] && \
  lib64=`if file -L $i 2>/dev/null | grep "ELF 64-bit" >/dev/null; then echo "(64bit)"; fi` && \
  echo ${objd} | perl -p -e "s/.*SONAME\s+(\S+)\so.*/devel(\1$lib64)/g"
done | sort -u
```

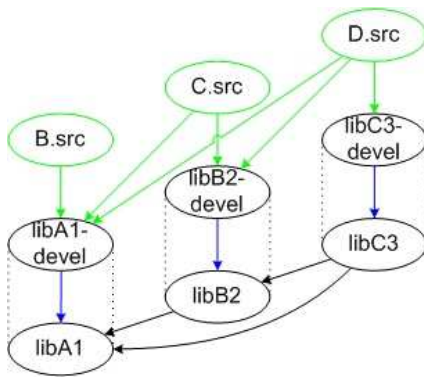
find-requires:

```
#
# --- .so files.
for i in `echo $filelist | tr '[:blank:]' "\n" | egrep "/usr(/X11R6)?/lib(|64)/[^\s]+\so$`; do
  objd=`objdump -p ${i} | grep SONAME`
  lib64=`if file -L $i 2>/dev/null | grep "ELF 64-bit" >/dev/null; then echo "(64bit)"; fi` && \
  [ -h ${i} -a -n "${objd}" ] && \
  ldd ${i} \
  | grep -v "/\(\lib\|lib64\)\/ld-linux.*\so" \
  | perl -p -e "s/\s+(\S+)\so.*/devel(\1$lib64)/g"
done | egrep -v 'devel\(\linux-gate|lib(c|pthread|rt)(\(\64bit\))?\)' | sort -u
```

### Option 4: No inter -devel package dependencies

What would also work is to remove all inter -devel package dependencies. This will result in package libA-devel "Requires: libA", and nothing more.

This will result in the following picture:

**Pro's**

- Only need to touch each package one, no need for an iterating process;
- Less

**Con's**

- A package's [BuildRequires](#) will have a large list of -devel package dependencies. This will be very difficult to maintain;

<a href="#">Attachment:</a>	<a href="#">Action:</a>	<a href="#">Size:</a>	<a href="#">Date:</a>	<a href="#">Who:</a>	<a href="#">Comment:</a>
<a href="#">rpm-devel-1.jpg</a>	<a href="#">action</a>	16940	25 Jun 2003 - 19:13	Main.stefan	rpm-devel-1
<a href="#">rpm-devel-2.jpg</a>	<a href="#">action</a>	31656	25 Jun 2003 - 19:13	Main.stefan	rpm-devel-2
<a href="#">rpm-devel-3.jpg</a>	<a href="#">action</a>	29282	25 Jun 2003 - 19:14	Main.stefan	rpm-devel-3
<a href="#">rpm-devel-4.jpg</a>	<a href="#">action</a>	32175	25 Jun 2003 - 19:14	Main.stefan	rpm-devel-4
<a href="#">rpm-devel-5.jpg</a>	<a href="#">action</a>	30511	25 Jun 2003 - 19:15	Main.stefan	rpm-devel-5

Topic **RpmDevelDependencies** . { [Edit](#) | [Attach](#) | [Ref-Bv](#) | [Printable](#) | [Diffs](#) | [r1.7](#) | [≥](#) | [r1.6](#) | [≥](#) | [r1.5](#) | [More](#) }

Revision r1.7 - 27 Jan 2004 - 16:30 GMT - [JohnKeller](#)

Parents: [TechnicalNotes](#)

Mandrake Linux Development Community webs: [Main](#) | [TWiki](#) | [Know](#) | [Sandbox](#)

Copyright © 1999-2004 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.

The Mandrake Linux Development Community is maintained by volunteers and does not necessarily reflect the official views or opinions of [MandrakeSoft](#).

Ideas, requests, problems regarding the Mandrake Linux Development Community? [Send feedback](#).