RPM Creation

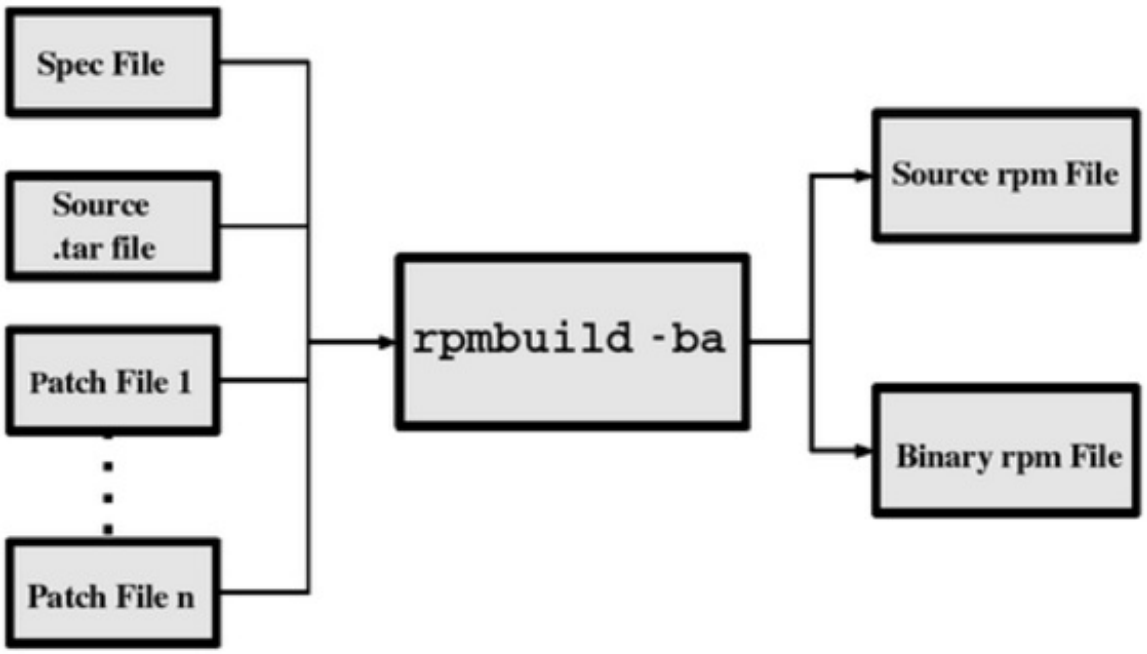
The RPM packaging system separates its main functionality methods using two main programs:

- **rpm**: Handles querying, installing, upgrading and erasing.
- **rpmbuild**: Handles creating and manipulating source and binary packages.

Each of these programs comes packaged as part of its own **rpm** and comes together with other related utilities we will not discuss in this section:

```
$ rpm -qil rpm-build | grep bin
/usr/bin/gendiff
/usr/bin/rpmbuild
/usr/bin/rpmspec
$ rpm -qil rpm | grep bin
/bin/rpm
/usr/bin/rpm2cpio
/usr/bin/rpmdb
/usr/bin/rpmkeys
/usr/bin/rpmquery
/usr/bin/rpmverify
```

It is a rather obvious point, but you should never remove the **rpm** program itself!



There are three basic ingredients required to assemble the source and binary packages:

- A **tarball** file containing all source code, makefiles, default configuration files, documentation, etc.
- Any **patch** files needed to be applied to the upstream source encapsulated in the tarball.
- A **spec** file that must be written that describes the package and how to patch and build it, dependency information, etc.

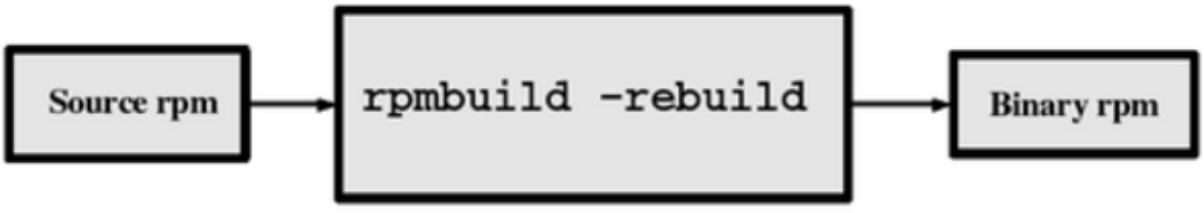
Once you have all these files together, the command:

```
$ rpmbuild -ba specFile
```

will perform the following steps:

- Coalesce the input files into the source RPM.
- Unpack the source tar file.
- Apply all patch files.
- Build binaries for the current architecture.
- Package binaries and config files into a binary RPM file.

Each of these steps is described by the **spec** file that the packager wrote.



RPM can be picky about where these files are placed. We will return to this question in the laboratory exercise.

If you want binaries for multiple platforms, then once you have the source RPM you can just copy it to a machine of another architecture and run **rpmbuild --rebuild SourceRPM** (or you can do a cross-compile if you have the cross-compile toolchain installed). This command can also be used to re-create the binary RPM from the source RPM in case you ever need to do so.

Also note that some "binary" rpms contain no binary files, i.e. they contain only scripts, configuration files, etc., that work independently of architecture. They will have the string **noarch** in their name rather than **i686 or x86_64**, etc., in their binary package name.