Cabal FAQ

Contents

Select the name of a problem to jump to a full explanation and solution.

I just used cabal to install parsec. Why did it give me version 2 rather than 3?

`runghc Setup` complains of missing packages

```
$ runghc Setup.hs configure
Configuring blah-1.0...
Setup.hs: At least the following dependencies are missing:
time -any
```

But I just did `cabal install time`! What is going on?

Cabal goes into an infinite loop / runs out of memory

I just upgraded to ghc-6.10 and now cabal runs out of memory when I try to install something

```
Resolving dependencies...
cabal: memory allocation failed (requested 2097152 bytes)
```

internal error: could not construct a valid install plan

```
$ cabal install leksah
Resolving dependencies...
cabal: internal error: could not construct a valid install plan.
The proposed (invalid) plan contained the following problems:
The following packages are involved in a dependency cycle leksah-0.10.0.4
```

What's going on? How do I fix it?

Dependencies conflict

> ```
> Resolving dependencies...
> cabal: dependencies conflict: ghc-6.10.1 requires process ==1.0.1.1 however
> process-1.0.1.1 was excluded because ghc-6.10.1 requires process ==1.0.1.0
> ```
>
> What does this message mean? It makes little sense to me — ghc-6.10.1 requires both `process ==1.0.1.1` and `process ==1.0.1.0`?

Yes, that's exactly what it looks like to Cabal which is why it is horribly confused.

The reason is shadowing between the global and user package dbs. The way ghc package databases work is that the user one is just slapped on top of the global one. Like `Data.Map.union`.

Suppose you've got these packages registered in the global package db:

```
    A-1
   / |
  /  |
 B-1 |
  \  |
   \ |
    C-1.0.1.0
```

Now, suppose you register B-1 in the user package db, then it masks the existing B-1 from the global db. But suppose that you build this instance against C-1.0.1.1...

```
    A-1
   / \
  /   \
 B-1   \
  \     C-1.0.1.0
   \
    C-1.0.1.1
```

Oh no! Now A-1 appears to depend on two versions of C! The cabal-install dependency resolver is designed to look for solutions where this does not happen, but here it's already happened in the installed packages, so it cannot ever pick the A-1 package as part of an install plan.

It's not just cabal-install that will get confused here. ghc --make will too. If you're lucky you'll get linker errors. If you're unlucky you'll get segfaults.

The solution we've been discussing for the next major ghc release is to track package ABIs and possibly even to allow slotting packages in their ABI. That would make this problem disappear and generally allow safer and more flexible management of installed packages.

In the mean time I'm thinking of making the representation of installed package databases record broken packages. Then when we overlay package databases we could mark clashes like the above as breaking A-1. We should also try and get the solver to help us find solutions that avoid getting into this situation in the first place.

> And, more importantly, how do I solve this?

Run `ghc-pkg list` and look for packages where you have the same version registered in the user and global package db. Unreigister the user one. For example you might have `Cabal-1.6.0.1` registered per-user and globally.

In this specific example you should also unregister `process-1.0.1.1`:

```
ghc-pkg unregister --user process-1.0.1.1
```

If it says it'll break other things, then do it anyway using `--force`. But you will need to rebuild all those other packages that get broken. To check what packages are broken use:

```
ghc-pkg check
```

To avoid this problem in the future, avoid upgrading core packages. The latest version of cabal-install has disabled the `upgrade` command to make it a bit harder for people to break their systems in this way.

Hidden packages (a)

You build a package that you are writing yourself and get a message like:

```
Could not find module `Data.Map': it is a member of package
containers-0.1.0.0, which is hidden.
```

What you need to do is to add `containers` to the `build-depends` in your `.cabal` file.

The reason you get this message is because when cabal asks ghc to build your package, it tells ghc to ignore all available packages except for the ones explicitly listed in the `.cabal` file. The terminology ghc uses for this is "hidden".

Do not be confused by the `ghc-pkg` tool commands to `hide` and `expose` packages. That makes no difference to this issue.

Hidden packages (b)

You build a package (not one you're writing yourself) and get a message like:

```
Could not find module `Data.Map': it is a member of package
containers-0.1.0.0, which is hidden.
```

This is because the package has not been updated for ghc-6.8 which has split the base package into lots of smaller packages. The package needs to be updated to say that it depends on these new split base packages, like containers, process and several others.

If you just want to get the package to build, add the missing package names to the build-depends: line in the .cabal file. For example given the above error message we would add the `containers` package to the build-depends.

Developers of packages who want to know how to update their package properly so that it will continue to work with old and new compilers should see the article on **upgrading packages**.

runProcess: does not exist

You try to install a package on Windows and it fails with the message

```
sh: runProcess: does not exist (No such file or directory)
```

What it means is that it cannot find the program `sh.exe` which is needed to run the `./configure` script that this package uses.

Packages that use `./configure` scripts are not very good citizens on Windows. They have to be run from within an MSYS shell because `./configure` scripts are actually Unix shell scripts. The MSYS shell provides all the programs that the script needs to run.

BTW, if you want to make the error message better see Cabal ticket **#403**.

ExitFailure 1

You use cabal to build a bunch of packages and it fails with a message like:

```
cabal: Error: some packages failed to install: foo-1.0 failed during the
configure step. The exception was: exit: ExitFailure 1
```

Hooray for unhelpful error messages! Although this final message is unhelpful, there is almost always an actual error message further up in the build log. Try scrolling up to the bit where it was trying to build that package.

runghc Setup complains of missing packages

> I get
>
> ```
> $ runghc Setup.hs configure
> Configuring PER-0.0.20...
> Setup.hs: At least the following dependencies are missing:
> time -any && -any
> ```
>
> But I already have that package installed.
>
> ```
> $ ghc-pkg list time
> /usr/lib/ghc-6.10.1/./package.conf:
> /home/me/.ghc/x86-linux-6.10.1/package.conf:
>     time-1.1.2.4
> ```

The default for `runghc Setup.hs configure` is `--global`, but the default for `cabal configure` is `--user`. Global packages cannot depend on user packages. So if you're using the `cabal` program to install packages, then you can also us it to configure other packages. There is usually no need to use `runghc Setup.hs` at all.

If you need to use the `runghc Setup.hs` interface (e.g. in some system build scripts) and you want it to pick up packages from the user package db then use the `--user` flag. If you're constantly having to use the `runghc Setup.hs` interface and doing per-user installs is a pain then you can set the default for the cabal program to be global installs in the cabal config file (`~/.cabal/config`).

Parsec 2 vs 3

> I just used cabal to install parsec. Why did it give me version 2 rather than 3?

The default version of the parsec package is version 2.x. You can explicitly request version 3 using

```
$ cabal install 'parsec >= 3'
```

Cabal goes into an infinite loop / runs out of memory

> I just upgraded to ghc-6.10 and now cabal runs out of memory when I try to install something
>
> ```
> Resolving dependencies...
> cabal: memory allocation failed (requested 2097152 bytes)
> ```

This happens when you use cabal-install version 0.5.x with ghc-6.10. The older cabal-install version goes into an infinite loop when resolving dependencies because it cannot cope with the fact that base 3 depends on base 4.

If you have still got ghc-6.8.x installed then you can upgrade to the latest cabal version using:

```
$ cabal install cabal-install --with-compiler=ghc-6.8.3
```

otherwise you will need to bootstrap cabal-install freshly with ghc-6.10. When you have upgraded, double-check you have the right version on your `$PATH` using

```
$ cabal --version
cabal-install version 0.6.2
using version 1.6.0.2 of the Cabal library
```

Internal error: invalid install plan

> ```
> $ cabal install leksah
> Resolving dependencies...
> cabal: internal error: could not construct a valid install plan.
> The proposed (invalid) plan contained the following problems:
> The following packages are involved in a dependency cycle leksah-0.10.0.4
> ```

The solution is to upgrade cabal-install:

```
$ cabal install 'cabal-install >= 0.10'
```

This happens when you use an older version of cabal-install to try to install a package that uses a new Cabal feature that allows intra-package dependencies (that is, components within a package that depend on each other).

The older version of cabal-install does not understand these new intra-package dependencies and it thinks that the package depends on itself (which would be a dependency cycle) so it gets upset.