Creating Packages with setup.rb

Creating Single Package Archive

"Single Package Archive" means the archive which includes only one PACKAGE. "package" means one set of ruby scripts, ruby extentions, commands, and data files.

setup.rb requires that the archive is structured like this:

```
PackageTop/
    setup.rb
    lib/
        (ruby scripts)
    ext/
        (ruby extentions)
    bin/
        (commands)
    data/
        (data files)
    conf/
        (configuration files)
    man/
        (manual pages)
    test/
        (tests)
```

Each file/directories acts as below:

setup.rb

> The installer. This file is included in this archive. Just copy it to your package.

lib/, bin/, data/, conf/, man/

> These directories includes files which are to be installed. This directory tree is mirrored to the target directory, from 'lib/' to 'RUBYLIB/', from 'bin/' to 'BINDIR/', from 'data/' to 'DATADIR/' ....
>
> Use 'lib/' for ruby scripts, 'bin/' for commands, 'data/' for any other data files, 'conf/' for configuration files, 'man/' for manual pages.

ext/

> 'ext/' directory includes source code of ruby extentions. If you want to install 'RUBYLIB/ARCH/someext.so', create a directory 'ext/someext/' and put source files into it.
>
> [WARNING] All extention source directories MUST include extconf.rb or MANIFEST.

test/

'test/' directory contains test scripts. You must write test scripts which run on test/unit library.

Creating Multi-Package Archive

setup.rb can handle an archive which includes multiple PACKAGEs.

setup.rb requires the archive is structured as below:

```
PackageTop/
    setup.rb
    packages/         <--- fixed name
        tmail/        <--- tmail package
            bin/
            lib/
            ext/
            data/
            conf/
            man/
            test/
        raccrt/       <--- raccrt package
            bin/
            lib/
            ext/
            data/
            conf/
            man/
            test/
        strscan/      <--- strscan package
            bin/
            lib/
            ext/
            data/
            conf/
            man/
            test/
        amstd/        <--- amstd package
            bin/
            lib/
            ext/
            data/
            conf/
            man/
            test/
```

Hooking Tasks

You can hook any tasks, such as "config" "setup". For example, you want to make some files in 'lib/tmail/' when setup. Then create file 'lib/tmail/pre-setup.rb' and put this:

```
# pre-setup.rb

# process grammer file
system "racc #{srcdir_root + '/src/mp.y'} -o mailp.rb"

# require all ruby scripts in this directory from  loadlib.rb.
```

```
list = Dir.glob(curr_srcdir + '/*.rb').collect {|n| File.basename(n) }
File.open( '_loadlib.rb', 'w' ) {|f|
  f.puts list.collect {|n| "require 'tmail/" + n + "'" }
}
File.open( '../tmail.rb', 'w' ) {|f|
  f.puts "require 'tmail/_loadlib'"
}
```

This file is evaluated on task "setup" in the directory, before processing any other thing. Acceptable hook file names are:

```
{pre,post}-{config,setup,install,test,clean,distclean}.rb
```

[NOTE] You can also put hook files in the top directory of archive and/or the type-root directory ('bin/', 'lib/',...).

srcdir/objdir support

setup.rb supports srcdir/objdir separation. In other words, you can compile everything out of the source directory.

If you write hooks, you should supports srcdir/objdir system. When you read source code, read it from srcdir. When you write anything, write it to the current directory. There's also some APIs to help your work. see ,Hook Script APIs Reference Manual

metaconfig

You can add new config options by writing file "metaconfig". metaconfig must be placed in the package-root directory.

Here is a simple example of metaconfig.

```
add_path_config 'libc', '/lib/libc.so', 'path to the C standard library'
add_bool_config 'win32', false, 'compile with Win32 support'
```

This script defined new config option --libc and --win32.

In 'metaconfig', you can use some APIs described in ,metaconfig API Reference Manual

Backward Compatibility

I do not assure any backward compatibility for the setup.rb. If you'd like old behavior, just use old version.

License

GNU LGPL, Lesser General Public License version 2.1. For details, see file "COPYING".

NOTE: You CAN distribute your program under the any licenses you like. LGPL does not force you to make your programs LGPL while the installer is LGPL'ed one.

Installation Manual

You can freely copy/edit and/or distribute Usage_*.txt files which are included in this archive. I do not claim any rights on them. Removing my copyright is also OK.