http://xahlee.org/emacs/elisp\_library\_system.html

Emacs Lisp's Library System: What's require, load, load-file, autoload, feature?

Emacs Lisp Library系统：什么是require，load，load-file，autoload，feature？

This page explains emacs library system. For example, what's the difference between library, package, features? And what's the difference between {load-file, load, require, autoload}?

这篇文章解释emacs库系统。例如，库，包，特性的差异是什么？load-file， load， require， autoload之间的区别又是什么？

What's Library, Package, Feature?

什么是库，包，特性？

No Namespace

Emacs lisp the language does not have namespaces. Everything is global, with dynamic scope, with some shadowing mechanism. So, don't expect library or module to be language defined namespace constructs like Perl or Python's module system or Java's Package system.

无命名空间

Emacs lisp是没有命名空间的语言。一切是全局的，伴随动态作用域和一些跟踪机制。所以，不要指望库或模块是语言定义的命名空间结构，就像Perl或Python的模块系统或Java的包系统。

What's the difference between a Package and Library?

The terms “package” and “library”, are used losely in emacs/elisp manual to refer to any useful elisp file. They are not technical definitions in elisp.

包和库差异何在？

术语“包”和“库”,在emacs/elisp手册来用来指任何elisp文件。在emacs/elisp中它们并没有从技术来定义。

A “library” usually refers to elisp file containing a collection of lisp functions, to be called by other lisp source code. For example, the command comment-dwim is defined in 〔newcomment.el〕, which is a library of functions.

”库“通常是一个包含众多lisp函数的elisp文件，它被其他的lisp源码调用。例如，命令comment-dwin在定义一个叫做newcomment.el的函数库中。

A “package” usually refers to something useful for emacs users. e.g. major mode or minor mode.

“包”通常是指对Emacs用户有用的东西，例如主模式或辅模式。

The term “module” is not used by emacs.

术语“模块”没有用被emacs使用。

Emacs's Concept of “Feature” and the Functions {provide, require}

The term “feature” has some meaning in elisp, but is not mechanical. A “feature” is elisp symbol that is intended to represent the functionality provided by a emacs package.

emacs中“特性”的概念和函数{provide，require}

术语“特性”在elisp中有特殊的意义，但并不是机械的。“特性”是一个elisp符号，它旨在代表一个emacs包提供的功能。

For example, type 【Alt+x describe-variable Enter features Enter】, and here's its value:

例如，键入Alt+x describe-variable Enter features Enter】后，下面是它的值：

ibuffer etags ring cc-mode cc-fonts cc-menus cc-cmds cc-styles

cc-align cc-engine cc-vars cc-defs xlsl-mode encoded-kb speck

sgml-mode dired info newcomment desktop recentf tree-widget wid-edit

advice help-fns …

A elisp file can call (provide '<symbol name>) near the end. Then emacs will add that symbol to the “features” list.

elisp文件在文件末尾调用(provide '<symbol name>)之后，emacs就会将这个symbol添加到“features”列表中。

The purpose of features is for emacs to know if a package is already loaded. The “features” variable and the functions {provide, require}, is the mechanism. The require function checks the “features” to see if a symbol is already there before loading the file.

features的目的是让Emacs知道一个包是否已经加载。变量“features”和函数{provide, require}构成这种机制。require函数通过检查“features”列表查看符号在加载文件之前是否已经存在。

In summary:

总结：

“features” is a global variable, holding a list of elisp symbols that are features.

“features”是一个全局变量，维护了一个elisp符号列表，每个符号都是一种特性。

A “feature” is a elisp symbol that represent a lisp package.

“特性”是一个代表lisp包的符号。

A package can call (provide '<symbol name>). When this code is evaluated, emacs will add the symbol name to the “features” list.

一个包能够调用(provide '<symbol name>)。当此代码被计算后，emacs会将这个symbol添加到“features”列表中。

When the code (require '<symbol name>) is called, emacs check if that symbol name is already in the “features” list. If not, load it. (emacs will guess a file name based on the symbol name. Or, the “require” function may specify a file name in the 2nd argument.)

当代码(require '<symbol name>)被调用时，emacs检查该符号是否已经是在“features”列表中。如果没有就加载它。（ Emacs基于符号名猜测文件名。或者， “require”函数可以在第二个参数指定一个文件名。 ）

Package/Library/Feature are not Managed

没有管理包/库/特性

There is no absolute relation between any concept of package/library/feature/autoload facilities and the file name.

包/库/特性/自动转载/文件名之间并无绝对的联系。

By convention, if a elisp file name is 〔xyz-mode.el〕, it OFTEN provides a lisp symbol “xyz-mode” as its feature name (if it does at all), and the command to invoke the mode is OFTEN named “xyz-mode”. Sometimes the “mode” part is omitted in any of {file name, feature symbol name, command name}.

按照惯例，如果一个elisp文件的名称是〔xyz-mode.el〕，它往往提供一个Lisp符号“xyz-mode”作为其特性的名称（如果完全这样做的话） ，并且调用的该模式命令往往是“xyz-mode” 。有时{文件名，特性符号名，命令名} 中的“mode”部分会被忽略不写。

This is only a lose convention. There are a lot exceptions in many bundled emacs packages. For example:

这只是一个模糊的公约。在不少捆绑的emacs包中可以看到很多例外。例如：

The file 〔lisp-mode.el〕 provides the symbol lisp-mode as feature, and is invoked by a command named emacs-lisp-mode.

The 〔cua-base.el〕 file provides symbols “cua-base” and “cua” as features, and is invoked by a command named cua-mode.

The 〔text-mode.el〕 file does not provide any symbol for feature, but is a useful mode anyway. It is invoked by a command named text-mode.

The file 〔desktop.el〕 provides the symbol “desktop” as feature, and the command name to invoke it is desktop-save-mode.

All the above means, you could have a file named 〔Joe-xyz-mode\_v2.1.el〕, which provides a feature named “abc”, while the command name to activate it may be “opq”, and it might be displayed in mode line as “OPQ helper”. And, this file can be considered as a package as well as library.

通过上面所述，你可以定义一个名为〔Joe-xyz-mode\_v2.1.el〕它提供了一个名为“abc”的特性，但是激活它的命令名称可能是“ opq ” ，它可能会模式行显示为“OPQ helper”。而且，这个文件既可以被看做一个包也可以被看做一个库。

File/Package Loading Mechanisms

文件/包加载机制

Emacs's module system is a primitive system, centered on loading file, with some slightly high level things such as its “features”, autoload, require. However, nothing is strict or enforced by elisp.

Emacs的模块系统是一种原始的系统，围绕加载文件，有一些稍微高层次的东西，如它的“features” ，autoload, require 。然而，没有任何东西对于elisp是严格的的或强制的。

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Purpose** | **Tech Detail** | **Comment** |
| load-file | Load a file. | Load one specific file. | Use this if you have one very SPECIFIC file at one particular file path. |
| load | Load a file. | Load a file by searching thru var “load-path”. Also, tries to load a compiled version (.elc) if exists. | Use this if the path for the file is not known in advance, and you are using a file name without full path, such as “undo” or “undo.el”, and you want to load the compiled version if it exists (undo.elc). |
| require | Load a package if it has not already been loaded. | Checks the var “features”, if symbol is not there, then call load to load it. | Best used in elisp source code, similar to other lang's “require” or “import”. |
| autoload | Load a file only when a function is called. | Associate a function name with a file path. When the function is called, load the file, and execute the function. | If you are writing a major mode, it's good to have your package installation go by autoload (if possible). It saves startup time. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Purpose** | **Tech Detail** | **Comment** |
| load-file | 加载一个文件 | 加载一个指定文件 | 如果你指定的文件在一个特定的文件路径，使用此 |
| load | 加载一个文件 | 通过搜索通过变量 “load-path ” 加载一个文件。此外，会尝试载入存在的编译版本（ .elc ）。 | 若事先不知道文件路径，或你使用不包含完整路径的文件名，如“undo”或“ undo.el ” ，或要加载已存在的编译版本，使用此。 |
| require | 加载一个未载入的包 | 检查变量 “features”, 如果符号不存在，则通过load加载它。. | 最好用在elisp的源码或其他语言的 “require” or “import”. |
| autoload | 只在函数调用时加载包 | 通过文件路径关联一个函数名称。当函数被调用，加载该文件，并执行该函数 | If you are writing a major mode, it's good to have your package installation go by autoload (if possible). It saves startup time. 如果你正在编写一个主要模式，最好让你的包通过autoloader进行安装（如果可以的话）。它节省了启动时间。 |