## Package Management

	10		
<u>Description</u>	Keystroke The list-nackage comman	Function ad opens a buffer listing all packages avai	Note  lable from the sites identified by the URLs listed in the 'package-archives'.
Management	<ul> <li>Once the information has</li> <li>Then you can use one of</li> </ul>	s been retrieved from the sites, the packa f the single keys listed below to operate of	ages are listed one per line with name, version, state and description.
	• C-h p • <f1> p</f1>	(finder-by-keyword)	Find packages matching a given keyword. Useful to search for packages supporting a specific concept.
	• C-h P	(describe-package PACKAGE)	Display the full documentation of PACKAGE (a symbol).  • Prompts for the package name.
See also: <u>▼ Help/Info</u>	• <f1> P</f1>		Shows whether it is installed or not, its version, the features it implements and some extra notes.
Install Package	M-x package-install	(package-install PKG &optional DONT-SELECT)	Install the package PKG.  • Prompts for the package name.
	M-x package- refresh-contents	(package-refresh-contents &optional ASYNC)	Download descriptions of all configured ELPA packages.  For each archive configured in the variable 'package-archives', inform Emacs about the latest versions of all packages it offers, and make them available for download.  Optional argument ASYNC specifies whether to perform the downloads in the background.
Show PEL user option and package info  See also:  See Help/Info Customize	<f11> ? e ?</f11>	(pel-package-info &optional FULL-REPORT ON-STDOUT	Display the following information inside a *pel-user-options* buffer:  • name of custom file, package-user-dir, the number of PEL user-options, and the number of them that are active, number of loaded files, and features.  • The number of Elpa packages active: the count of the ones directly installed because of active PEL user-options and the count of them installed as dependencies of the first group.  • The number of Emacs Lisp files stored in the ~/.emacs.d/utils (or equivalent directory) as a result of PEL user options.  • The number of elpa-compliant packages that have a newer version and could be updated.  • With optional argument, like C-u, generates a full report with more details.
_	• <f11> ? e P • <f11> SPC 1 1 p</f11></f11>	(list-packages &optional NO-FETCH)	List packages available in from used package managers in a *Packages* buffer, showing the package name, its package manager site name, status and description. Clicking (or pressing enter) on the name opens a buffer with the description of the package. See minor mode commands below.
	• <f12> 1 p</f12>		▼ The <f12> 1 p binding is available for buffers in Emacs-Lisp mode.</f12>
Operations	-		the buffer that lists the packages, in the *Package* buffer.
·	h	(package-menu-quick-help)	Show short key binding help for 'package-menu-mode'.  • The full list of keys can be viewed with C-h m (or its equivalent <f1> m)</f1>
	n	(next-line &optional ARG TRY- VSCROLL)	Move to next line
Move point to previous line	P	(previous-line &optional ARG TRY- VSCROLL)	Move to previous line
Filter list of packages	f	(package-menu-filter KEYWORD)	Filter the *Packages* buffer.  • Show only those items that relate to the specified KEYWORD.  • KEYWORD can be a string or a list of strings. If it is a list, a package will be displayed if it matches any of the keywords. Interactively, it is a list of strings separated by commas.  • To restore the full package list, type 'q'.
Toggle visibility of obsolete packages	(	(package-menu-toggle-hiding)	Toggle visibility of obsolete available packages.
Hide package/describe associate package	н	(package-menu-hide-package)	hide-package: Hide a package under point.  If optional arg BUTTON is non-nil, describe its associated package.
Sort columns s	s	(tabulated-list-sort &optional N)	Sort Tabulated List entries by the column at point.  • With a numeric prefix argument N, sort the Nth column.
	• ? • RET	(package-menu-describe-package &optional BUTTON)	Describe the current package.  If optional arg BUTTON is non-nil, describe its associated package.  The description buffer also contains buttons to install the package.
Refresh buffer	g	(revert-buffer &optional IGNORE- AUTO NOCONFIRM PRESERVE- MODES)	Redisplay/refresh information.
Refresh content (download package list again)	r	(package-menu-refresh)	Download the Emacs Lisp package archive.  • This fetches the contents of each archive specified in 'package-archives', and then refreshes the package menu.
Mark package for deletion	d	(package-menu-mark-delete &optional NUM)	Mark an (installed) package for deletion and move to the next line.  NUM argument is unused.
Mark package for installation	i	(package-menu-mark-install &optional NUM)	Mark a package for installation and move to the next line.  NUM argument is unused.
	• u • <del></del>	(package-menu-mark-unmark &optional NUM)	Clear any marks on a package and move to the next line.  NUM argument is unused.
Mark all upgradable packages:  'D' for old versions to delete 'I' for new versions to install	U	(package-menu-mark-upgrades)	Mark all upgradable packages in the Package Menu.  For each installed package with a newer version available, place an (I)nstall flag on the available version and a (D)elete flag on the installed version.  A subsequent M-x package-menu-execute call will upgrade the package. It can also be done with the x key in the 'Packages' buffer.  If there's an async refresh operation in progress, the flags will be placed as part of 'package-menupost-refresh' instead of immediately.
Execute action specified by marks	x	(package-menu-execute &optional NOQUERY)	execute: Perform marked Package Menu actions.     Packages marked for installation are downloaded and installed;     packages marked for deletion are removed.     Optional argument NOQUERY non-nil means do not ask the user to confirm.
Quit Window	q	(quit-window &optional KILL WINDOW)	Quit WINDOW and bury its buffer.

## Package Managers — References

cases, Elpa has packages that are also in MELPA, like crisp.  This can also be installed, see: <a href="http://melpa.org/#/getting-started">http://melpa.org/#/getting-started</a> An older site. Deprecated. Seems no longer active. There seems to be a security issue with this site.  Emacs Package Managers  Emacs provides the built-in package.el package manager. This is also what PEL currently uses. package.el provides an easy-to-use package management facility where each package is maintained inside its own directory and where the directory identifies the package name and version. Each package directory is placed in Emacs load-path variable. Several other package managers (if not all) use the same mechanism.  Unfortunately as the number of packages increase over time as you use more packages, the longer load-path causes Emacs startup time increase.  I have been investigating the possibility of putting all Emacs Lisp code from all installed packages inside a single directory to speed up	Topic & Link	Note
Februaries   The original ONU package distribution site.	Emacs Package Repositories	
The service to include more packages than flips; for example SLIME and rust-mode are available at MELPA but not at Elpa. In some other services, Elba but not at Elpa. In some other services, Elba but not at Elpa. In some other services, Elba but not at Elpa. In some other services, Elba but not at Elpa. In some other services and the services of the SLIME and rust-mode are available at MELPA but not at Elpa. In some other services. An other state of the services are as the services and the services and the services are as the services. The services are as the services are as the services are as the services.  **Packago.et**  **Packago.et**  **Imacs provides the bull-in package at package manager. This is also what PEL currently uses, package and diversor, Early bedage increase package informs to standard the services are as the services.  **Unfortunately as the package areas and versor. Early bedage inducting state path variation. Several other package increases and package increases are package increases and package increases are package.  **Cask ** Cask **Cask ** Cask *		The original GNU package distribution site.
cases. (Sibh his procleages that are also in MELPA, like or risp The can all so the relatified, see https://mea.oru/Meg/ett	Emacs Wiki - ELPA	
Emacs Package Managers  • package.el  • package.el provides the bulli-in package analyto-isse package managerent tacility where each package in maintained inside its own directory and where it directory identifies the package manager and vertion. Each package directory is placed in Emacs load-path variable. Several other package managers are and vertion. Each package directory is placed in Emacs load-path variable. Several other package managers are and vertion. Each package directory is placed in Emacs load-path variable. Several other package increase over time as you use more packages, the longer load-path cause Emacs startup time increase.  • It have been investigating the posibility of putting all Emacs Listo and from all installed packages inside a single directory to speed up the re-organization are resolved.  • Cask  • Cask  • Cask  • Cask  • Cask Decementation  • Cask Offitub  • Cask Offitub  • Cask Offitub  • Same author than Cask's.  • el-get  • Borra  • Straight  • Another package manager. External package meant to replace package.el  • Borra  • Straight  • Borry straight and package manager. External package meant to replace package.el  • Borra  • Straight  • Borry straight and package manager. External package meant to replace package.el  • Borra  • Brance's Key Syntax Explained  • How can I simulate an arbitrary key event from Eliap?  • Modifier Keys  • Emacs is print key binding and cannot append the directory in the pinding and cannot append the directory in the pinding for a command or list all key binding for a command or list all key bindings.	MELPA	
package.el      package.e	Marmalade repo	An older site. Deprecated. Seems no longer active. There seems to be a security issue with this site.
package of provides an aesy-to-use package imanagement fallity where each package is maintained inside its own directory and where it directory identifies the package of management control is placed in Emass laced-path variable. Several other package imanagers (if not all) use the same mechanism.  Unfortunately as the number of packages increase over time as you use more packages, the longer load-path causes Emacs startup time increase.  I have been investigating the packages increase over time as you use more packages, the longer load-path causes Emacs startup time reduction if all problems remaining from the recognization of the packages in the package in the packages in the package in the packages in the package i	Emacs Package Managers	
Preparing Lisp code for distribution  • Quelpa  • Cask  Modern Emacs package management with Cask and Pallet glambdaCat Cask Decembration Cask Github EPL © Github Same author than Cask's.  • el-get  • Borg  • Straight Another package manager. External package meant to replace package.el Articles on Straight Articles on Straight Emacs's Key Syntax, Explained  Keymaps Rebinding keys in your init file Emacs's Key Syntax, Explained Hove an Listmadate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the celebete character  It <del> Fails to Delete Modifier Keys Emacs: print key binding for a sommand or list all key bindings</del>	• package.el	package el provides an easy-to-use package management facility where each package is maintained inside its own directory and where the directory identifies the package name and version. Each package directory is placed in Emacs load-path variable. Several other package managers (if not all) use the same mechanism.  Unfortunately as the number of packages increase over time as you use more packages, the longer load-path causes Emacs startup time to increase.  I have been investigating the possibility of putting all Emacs Lisp code from all installed packages inside a single directory to speed up Emacs startup. This investigation is not yet complete but could lead to significant startup time reduction if all problems remaining from this
Ouelpa      Cask      Modern Emacs package management with Cask and Pallet elambdaCat      Cask Documentation Cask Documentation Read the complete info there as the second step.      Same author than Cask's.      el-get      Straight Another package manager. External package meant to replace package.el  Articles on Straight Another package manager. External package meant to replace package.el  Articles on Straight Articles on Straight Binding keys and loading code Auto loading Keymaps Rebinding keys in your init file Emacs's Key Syntax Explained How can L simulate an arbitary key event from Elisp? writing lisp emacs key binding and annot specify the sidelete-character II < DEL> Fails to Delete Modifier Keys Emacs: print key binding for a sommand or list all key binding and sommand or list all key bindings	Emacs Lisp Packages	
• Cask  Modern Emacs package management with Cask and Pallet		
Modern Emacs package management with Cask and Pallet & LambdaCat  Cask Documentation Cask © Github  EPL © Github  **Borg  **Straight**  Another package manager. External package meant to replace package.el  Articles on Straight  **Borg vs Straight.el?*  Binding keys and loading code Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  **Writing lisp emass key binding and cannot specify the <delete> character  If <delete cash="" of="" replace="" spint="" td="" the="" the<=""><td>• Quelpa</td><td></td></delete></delete>	• Quelpa	
Modern Emacs package management with Cask and Pallet & LambdaCat &	. OI-	
Cask @ Github  EPL @ Github  Same author than Cask's.  el-get  Borg  Straight  Another package manager. External package meant to replace package.el  Articles on Straight  Bindling keys and loading code  Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete- a="" a<="" all="" binding="" bindliff="" character="" command="" emacs:="" for="" key="" keys="" list="" or="" print="" td=""><td>management with Cask and Pallet</td><td>General Intro and experience about Cask &amp; Pallet. Read First!</td></delete->	management with Cask and Pallet	General Intro and experience about Cask & Pallet. Read First!
EPL@ Github Same author than Cask's.  • el-get  • Borg  • Straight Another package manager. External package meant to replace package.el  Articles on Straight • Borg vs Straight.el?  Binding keys and loading code Auto loading Keymaps Rebinding keys in your init file Emacs's Key Syntax Explained How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character If <del> Fails to Delete  Modifier Keys Emacs: print key binding for a command or list all key bindings</del></delete>	Cask Documentation	Read the complete info there as the second step.
Borg  Borg  Another package manager. External package meant to replace package.el  Articles on Straight  Binding keys and loading code Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character If <del another="" bindings="" delete="" emacs:="" external="" fails="" key="" keys="" manager.="" manager.<="" meant="" modifier="" package="" package.el="" print="" replace="" td="" to=""><td>Cask @ Github</td><td></td></del></delete>	Cask @ Github	
Borg  Straight Another package manager. External package meant to replace package.el  Articles on Straight  Binding keys and loading code Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>	EPL @ Github	Same author than Cask's.
Straight Another package manager. External package meant to replace package.el  Articles on Straight  • Borg vs Straight.el?  Binding keys and loading code Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the cdeletes character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del>	• el-get	
Articles on Straight  Binding keys and loading code  Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>	• Borg	
Articles on Straight  Binding keys and loading code  Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>	• Straight	Another package manager. External package meant to replace package el
Binding keys and loading code  Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>		
Auto loading  Keymaps  Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>		Doig to Changilladi:
Keymaps Rebinding keys in your init file Emacs's Key Syntax Explained How can I simulate an arbitary key event from Elisp? writing lisp emacs key binding and cannot specify the <delete> character If <del> Fails to Delete Modifier Keys Emacs: print key binding for a command or list all key bindings</del></delete>		
Rebinding keys in your init file  Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>		
Emacs's Key Syntax Explained  How can I simulate an arbitary key event from Elisp?  writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>		
How can I simulate an arbitary key event from Elisp? writing lisp emacs key binding and cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>		
cannot specify the <delete> character  If <del> Fails to Delete  Modifier Keys  Emacs: print key binding for a command or list all key bindings</del></delete>	How can I simulate an arbitary key	
Modifier Keys  Emacs: print key binding for a command or list all key bindings		
Emacs: print key binding for a command or list all key bindings	If <del> Fails to Delete</del>	
command or list all key bindings	Modifier Keys	
Mastering Key Bindings in Emacs		
	Mastering Key Bindings in Emacs	