Grep

Description	Keystroke	Function	Note

Grep under Emacs		Emacs support several find and grep commands that can be executed from within Emacs. Doing so has several advantages:
		• The command output is collected in the *grep* Emacs buffer while the command runs asynchronously.
		  • The buffer is read-only, you can search within it right away.
		  • If you type RET on a found line, Emacs visit the file of the match at the appropriate line.
		• Without even going inside the search result buffer you can type one of the 'goto match' commands to move to the next or previous match:
		  • (next-error) : `C-x` `` ` ``, `M-g n` or `M-g M-n`
		  • (previous_error): `M-g p` or `M-g M-p`
		• Stop the grep asynchronous operation with `C-c C-k` or `<f11> g k`
		• The search commands keep a history for the searches. You can browse the history at the prompt.
		📦 PEL supports installation and use of several grep-enhancing external packages. 🔲 All PEL user-options for grep are in the **pel-pkg-for-grep** group.
		∑ **Customize** grep support to get new features and enhance its speed using the following commands.

Open this PDF file. See also: ∑ **Help/Info**	`<f11> g <f1>`	**(pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the local copy of the ∑ **Grep** PDF file unless a command prefix (like `C-u`) was used. In that case it opens the Github-hosted file instead.
Customize PEL Grep Support See also: ∑ **Customize**	`<f11> g <f2>`	**(pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL grep support. Groups: grep, ag, rg, ripgrep, wgrep. • If OTHER-WINDOW is non-nil (use `C-u`) , display in other window.
Customize Emacs Grep Support See also: ∑ **Customize**	`<f11> g <f3>`	**(pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs grep support. Groups: grep, ag, rg, ripgrep, wgrep.
Grep Search in Emacs		Use one of the following commands to perform grep operation on files. All of these commands share the following: • They collect output in the *grep* buffer. • Grep runs asynchronously, you can use any of the grep finding navigation commands while it is still running. See the list below. • To kill the grep job before it finishes, type `C-c C-k`. • This commands use a special history list for its arguments, so you can easily repeat a grep and find command.
Run grep via find See also: ∑ **File mngt**	• `<f11> f g` • `<f11> g f`	**(grep-find** COMMAND-ARGS)	Grep in files found by the find utility: run grep via find with user-specified args COMMAND-ARGS. • Collect output in the *grep* buffer. • While find runs asynchronously, you can use the `C-x` `` ` `` to find the text that grep hits refer to. • This command uses a special history list for its arguments, so you can easily repeat a find command. ☝ Faster alternatives are available if you use ripgrep or ag. See below.

Run grep	`<f11> g g`	**(grep** COMMAND-ARGS)	Run Grep with user-specified COMMAND-ARGS, collect output in a buffer.
			• Without prefix, prompts to complete a grep command.
			  • The default grep command is: `grep -nH —null -e`
			  • For example to search for main in all .c files complete it like this: `grep -nH —null -e main *.c`
			• If you prefix the command with `C-u`, Emacs creates a command to grep for the string at point in the same types of files as the current file.

Local grep	`<f11> g l`	**(lgrep** REGEXP &optional FILES DIR CONFIRM)	Run grep, searching for REGEXP in FILES in directory DIR. • The search is limited to file names matching shell pattern FILES. • FILES may use abbreviations defined in 'grep-files-aliases', e.g. entering 'ch' is equivalent to '*.[ch]'. As whitespace triggers completion when entering a pattern, including it requires quoting, e.g. '`C-q<space>`'. • With `C-u` prefix, you can edit the constructed shell command line before it is executed. • With `C-u C-u` prefix, directly edit and run 'grep-command'. • This command shares argument histories with `<f11> g r` and `<f11> g g`.
Recursive grep over files in directory tree using grep	`<f11> g r`	**(rgrep** REGEXP &optional FILES DIR CONFIRM)	Recursively grep for REGEXP in FILES in directory tree rooted at DIR. • The search is limited to file names matching shell pattern FILES. • FILES may use abbreviations defined in 'grep-files-aliases', e.g. entering 'ch' is equivalent to '*.[ch]'. As whitespace triggers completion when entering a pattern, including it requires quoting, e.g. '`C-q<space>`'. • With `C-u` prefix, you can edit the constructed shell command line before it is executed. With `C-u C-u` prefix, directly edit and run 'grep-find-command'. • This command shares argument histories with M-x lgrep and M-x grep-find. • When called programmatically and FILES is nil, REGEXP is expected to specify a command to run. As seen above, typing the `C-u` prefix before the command keystroke, we can modify the command line found, e.g. for adding a **| sort -V** so that output list the files in sorted order with lines number also sorted.
Recursive Gzip grep	`<f11> g z`	**(zrgrep** REGEXP &optional FILES DIR CONFIRM TEMPLATE)	Recursively grep for REGEXP in gzipped FILES in tree rooted at DIR. • Like 'rgrep' but uses 'zgrep' for 'grep-program', sets the default file name to '*.gz', and sets 'grep-highlight-matches' to 'always'.
Kill grep process	• `C-c C-k` • `<f11> g k`	**(kill-grep)**	Kill the grep process that currently runs asynchronously. • Useful if you want to interrupt a long-running grep command.
Grep findings navigation commands		Once a grep operation has been launched, the following commands can be used to navigate through the result, moving to the location of a match, inside the file at the matched line number. Fo async grep operations the commands can be used while the grep is still processing. The commands can be issued while point is inside the *grep* buffer but also outside.
Move to first match	`<f11> g 1`	**(first-error** &optional N)	Restart at the first grep found. • Visit corresponding source code. With prefix arg N, visit the source code of the Nth error.
Move to next match	• `C-` `` ` `` • `C-x` `` ` `` • `M-g n` • `M-g M-n`	**(next-error** &optional ARG RESET)	A prefix ARG specifies how many error messages to move; negative means move back to previous error messages. Just C-u as a prefix means reparse the error message buffer and start at the first error.
Move to previous match	• `M-g p` • `M-g M-p`	**(previous-error** &optional N)	Prefix arg N says how many error messages to move backwards (or forwards, if negative).
Search in project with projectile See ∑ **Projectile**		If projectile is available, you can also search in project-related files with Projectile. With PEL, activate the projectile mode with `<f11> <f8> <f8>` Note that both ripgrep and ag also provide project-based searching but projectile provides more control over the concept of project. 📦 These commands require **projectile external package** 🔲 PEL activates projectile when the **pel-use-projectile** user option is non-nil. More on ripgrep and ag below.
Search in project files with recursive grep	`<f8> s g`	**(projectile-grep** &optional REGEXP ARG)	Perform rgrep in the project. • With a prefix ARG asks for files (globbing-aware) which to grep in. • With prefix ARG of '-' (such as 'M--'), default the files (without prompt), to 'projectile-grep-default-files'. • With REGEXP given, don't query the user for a regexp.
Search in project files with ripgrep	`<f8> s r`	**(projectile-ripgrep** SEARCH-TERM &optional ARG)	Run a Ripgrep search with 'SEARCH-TERM' at current project root. • With an optional prefix argument ARG SEARCH-TERM is interpreted as a regular expression. 📦 Requires the projectile, ripgrep.el external packages as well as the ripgrep command line utility. 🔲 PEL activates this command when **pel-use-projectile** is non-nil. But to make it work you must also set **pel-use-ripgrep** to **t**. Also note that the ripgrep command line utility must be installed manually.
Search in project files with ag	`<f8> s s`	**(projectile-ag** SEARCH-TERM &optional ARG)	Run an ag search with SEARCH-TERM in the project. • With an optional prefix argument ARG SEARCH-TERM is interpreted as a regular expression. 📦 Requires the projectile, ag.el external packages as well as the ag command line utility. 🔲 PEL activates this command when **pel-use-projectile** is non-nil. But to make it work you must also set **pel-use-ag** to **t**. Also note that the ag command line utility must be installed manually.

Description	Keystroke	Function	Note
Speeding up Emacs grep searching	By default Emacs uses the standard find, grep, rgrep and zgrep command line utilities to perform the grep-based search operations. • The actual utility used depends on what is available on your Operating System. 📦 As fast as grep might be, tools like **ripgrep** and **ag** are much faster. See the section below on ripgrep and ag. Also,the grep-find-command user option can be modified to use ripgrep for the normal grep-find command. See notes in the reference table below.		

Searching with RipGrep explicitly using rg.el	The following commands use ripgrep explicitly to perform various types of searches. These search are <u>much faster than grep on large file set</u>.
	• Using ripgrep through the rg.el package inside Emacs makes it **very** flexible with lots of features available on the command line that can be easily activated via rg-menu.
	  • To get the list of all **rg** command lines you can use man (or woman) inside Emacs. With PEL you can type: **<f11> ? m rg RET**
	    • See ∑ **Help/Info** for info on man and woman.
	
	The following commands use ripgrep explicitly to perform **fast** grep-type searches. Using ripgrep is highly recommended.
	📦 The commands require the **rg.el** package. 🗹 PEL activates the use of **ripgrep** with **rg.el** inside Emacs when **pel-use-ripgrep** user option is set to t. You must **install ripgrep command line utility** separately. When pel-use-projectile is set to **t**, PEL also installs **ripgrep.el** that projectile uses.		

Recursive regexp grep over files in directory tree using RipGrep	• **<f11> g i** • *C-c s r*	**(rg** QUERY FILES DIR)	Run ripgrep, searching for REGEXP in FILES in directory DIR. • The search is limited to file names matching shell pattern FILES. • FILES may use abbreviations defined in 'rg-custom-type-aliases' or ripgrep builtin type aliases, e.g. entering 'elisp' is equivalent to '*.el'. REGEXP is a regexp as defined by the ripgrep executable. • With **C-u** prefix (CONFIRM), you can edit the constructed shell command line before it is executed. ☛ Useful to add the -z option to search into compressed files (.zip, .el.gz, etc…). • Collect output in a buffer. While ripgrep runs asynchronously, you can use **C-x ` ** (M-x 'next-error'), or RET in the rg output buffer, to go to the lines where rg found matches. 👍 ⚠️ With PEL, only **<f11> g i** forces loading of the rg package. At first the **C-s s r** key binding is not set. It will be as soon as rg is loaded (forced by the other bindings).
Recursive literal grep over files in directory tree using RipGrep	• **<f11> g t** • *C-c s t*	**(rg-literal** QUERY FILES DIR)	Run ripgrep, searching for literal PATTERN in FILES in directory DIR. • With C-u prefix (CONFIRM), you can edit the constructed shell command line before it is executed. ☛ Useful to add the -z option to search into compressed files (.zip, .el.gz, etc…). 👍 ⚠️ With PEL, only **<f11> g I** forces loading of the rg package. At first the **C-s s t** key binding is not set. It will be as soon as rg is loaded (forced by the other bindings).

Open RipGrep transient menu	• **<f11> g m**
	• **C-c s**
		(rg-menu)	Shows the transient menu for rg.
			• While the menu is active all keys and mouse actions are controlled by the rg menu, even though point remains where it was previously.
			  • The menu list options and commands shown below.
			  • Type the options first and include the dash in what you type. Then select the command.
			• The current directory affects what is proposed in the directory and file type prompts of several commands.
			• The results are show inside a *rg* buffer. There are <u>several commands available inside the *rg* buffer</u> which provide other functionality. Type ? in the *rg* buffer for help.
			• ⚠️ The default binding **C-c s** is also used for several cross-reference tools such as CScope which hijacks the binding when enabled if the **cscope-keymap-prefix** user option is not changed.
			  • With PEL, the PEL binding is always available in any case.

	`Search          Manage`
	`  d Dwim          l List`
	`  r Regex         s Save`
	`  t Literal       S Save as name`
	`  p Project`
	`  c Current directory`
	`  f Current file`
			`Switches`
			`  -h Search hidden files (--hidden)`
			`  -z Search zipped files (--search-zip)`
			`  -v Invert match (--invert-match)`
			`  -U Multi line (--multiline-dotall --multiline)`
			`  -w Search words (--word-regexp)`
			`  -l Don't cross file system (--one-file-system)`
			`  -n Override ignore files (--no-ignore)`
			
			`Options`
			`  -C Show context (--context=)`
			`  -M Omit long lines (--max-columns=)`
			`  -m Max matches per file (--max-count=)`
			`  -g Filter files glob (--glob=)`
			`  -T Exclude files types (--type-not=)`

Searching with ripgrep using deadgrep	📦 The **deadgrep** external package uses the **ripgrep** command line tool to provide a buffer-central control of ripgrep searches. 🗹 PEL installs and activates it when the **pel-use-deadgrep** user-option is turned on (set to **t**). Use **<f11> g <f2>** to open the customization buffer for it. When activated the following commands are available.		
Start a deadgrep search	**<f11> g d**	**(deadgrep** SEARCH-TERM)	Start a ripgrep search for SEARCH-TERM. Prompt for the search string. • If called with a prefix argument, create the results buffer but don't actually start the search. • Opens a *deadgrep* search result and control buffer where you can see the findings, navigate through them and perform other ripgrep searches with the ability to modify various search parameters. • See the deadgrep-mode commands available in this buffer in the following rows below.
Visit result at point	**RET**	**(deadgrep-visit-result)**	Goto the search result at point.
Visit result at point, in other window	**o**	**(deadgrep-visit-result-other-window)**	Goto the search result at point, opening in another window.
Move point to Next result/button	**n**	**(deadgrep-forward)**	Move forward to the next item. • This will either be a button, a filename, or a search result. See also 'deadgrep-forward-match'.
Move point to Previous result/button	**p**	**(deadgrep-backward)**	Move backward to the previous item. • This will either be a button, a filename, or a search result. See also 'deadgrep-backward-match'.
Move point to Next search hit	**N**	**(deadgrep-forward-match)**	Move point forward to the beginning of next match. • Note that a result line may contain more than one match, or zero matches (if the result line has been truncated).
Move point to Previous search hit	**P**	**(deadgrep-backward-match)**	Move point backward to the beginning of previous match.
Move point to next file header	**M-n**	**(deadgrep-forward-filename)**	Move forward to the next filename.
Move point to previous file header	**M-p**	**(deadgrep-backward-filename)**	Move backward to the previous filename.
Re-run the search	**g**	**(deadgrep-restart)**	Re-run ripgrep with the current search settings.
Collapse/Expand result for a file	**<tab>**	**(deadgrep-toggle-file-results)**	Show/hide the results of the file at point. • Point must be located on a file header line.
Stop a running search	**C-c C-k**	**(deadgrep-kill-process)**	Kill the deadgrep process associated with the current buffer.

Description	Keystroke	Function	Note

Searching with Ag explicitly	The following commands use ag explicitly to perform various types of searches.
	• Some of the commands restrict the search to files of specified types. They search in field with extensions corresponding to the specified type.
	  • For a list of the file types , execute the command **ag --list-file-types**.
	• It is also possible to provide more command lines to the ag command executed by a command.
	  • For that, type a command prefix (like **C-u**) before typing the command key sequence.
	  • For instance, you can use the -z command line switch to *search into zip files*.
	  • To get the list of all **ag** command lines you can use man (or woman) inside Emacs. With PEL you can type: **<f11> ? m ag RET**
	    • See ∑ **Help/Info** for info on man and woman.
	📦 These command require the **ag.el** package and the **ag** command line utility. 🔲 PEL activates the use of **ag** inside Emacs via **ag.el** when **pel-use-ag** user option is set to **t**. You must **install ag command line utility** separately.		

Literal search in files of specified directory tree	**<f11> g a a**	**(ag** STRING DIRECTORY)	Search using ag in a given DIRECTORY for a given literal search STRING, with STRING defaulting to the symbol under point. • If called with a prefix, prompts for flags to pass to ag.
Regexp search in files of specified directory tree	**<f11> g a x**	**(ag-regexp** STRING DIRECTORY)	Search using ag in a given directory for a given regexp. • The regexp should be in **PCRE syntax**, not Emacs regexp syntax. • If called with a prefix, prompts for flags to pass to ag.
Literal search in files of specific types in specified directory tree	**<f11> g a f**	**(ag-files** STRING FILE-TYPE DIRECTORY)	Search using ag in a given DIRECTORY for a given literal search STRING, limited to files that match FILE-TYPE. • Prompt for file type. Support tab completion. Type tab to get the list of file types. • STRING defaults to the symbol under point. • If called with a prefix, prompts for flags to pass to ag.
Ag search in project files	The following commands perform an ag search in the field of the current project directory tree. • The project directory tree is identified by the presence of a (D)VCS depot file (.git, .hg, etc..) in a directory.		
Literal search in project files of specific type	**<f11> g a p f**	**(ag-project-files** STRING FILE-TYPE)	Search using ag for a given literal search STRING, limited to files that match FILE-TYPE. • Prompt for file type. Support tab completion. Type tab to get the list of file types. • STRING defaults to the symbol under point. • If called with a prefix, prompts for flags to pass to ag.
Literal search in all project files	**<f11> g a p p**	**(ag-project** STRING)	Guess the root of the current project and search it with ag for the given literal search STRING. • If called with a prefix, prompts for flags to pass to ag.
Regexp Grep in all project files	**<f11> g a p x**	**(ag-project-regexp** REGEXP)	Guess the root of the current project and search it with ag for the given regexp. • The regexp should be in **PCRE syntax**, not Emacs regexp syntax. • If called with a prefix, prompts for flags to pass to ag.
Find files with matching name in directory tree	**<f11> g a d d**	**(ag-dired** DIR STRING)	Recursively find files in DIR matching literal search STRING. • The PATTERN is matched against the full path to the file, not only against the file name. • The results are presented as a 'dired-mode' buffer with 'default-directory' being DIR.
Find files with name matching regex in directory tree	**<f11> g a d x**	**(ag-dired-regexp** DIR REGEXP)	Recursively find files in DIR matching REGEXP. • The regexp should be in **PCRE syntax**, not Emacs regexp syntax. • The REGEXP is matched against the full path to the file, not only against the file name. • Results are presented as a 'dired-mode' buffer with 'default-directory' being DIR.
Find project files with a name matching regex	**<f11> g a d f**	**(ag-project-dired-regexp** REGEXP)	Recursively find files in current project matching REGEXP.
Kill all ag buffers	**<f11> g a k a**	**(ag-kill-buffers)**	Kill all 'ag-mode' buffers.
Kill other ag buffers	**<f11> g a k o**	**(ag-kill-other-buffers)**	Kill all 'ag-mode' buffers other than the current buffer.
Kill ag process	**<f11> g a k p**	**(ag/kill-process)**	Kill the 'ag' process running in the current buffer.

Grep under Emacs - Reference

Grep	The standard grep command is available on all POSIX systems.
	• Emacs default uses grep for the grep, zgrep and zrgrep Emacs commands.
	  • For rgrep and zrgrep, Emacs uses a command line that invokes grep (or zgrep) via find.
	    • The actual used command line is specified by the **grep-find-template** user option.

GNU EMacs Manual — Searching with Grep under Emacs	Describes Emacs basic grep support - with the use of standard GNU grep
GNU Grep 3.5 manual	GNU grep is available by default on Linux distributions. The actual version depends on the distribution.
freeBSD Grep	freeBSD is available on freeBSD and macOS
OpenBSD Grep	
ripgrep : rg	You can use ripgrep instead of GNU|BSD grep to significantly speed grep searches. It's possible to use ripgrep by itself, with its own command. It is also possible to replace grep in the standard Emacs commands using grep.
Ripgreg utility: rg @ Github	Github home for the ripgrep command line utility: rg
Emacs rg.el package @ Github	Github home for rg.el
rg.el manual	Latest version of the rg.el manual (previous versions available on the same site)
ripgrep.el @ Github	Github home of ripgrep.el
Blazing-fast jump-to-grep in Emacs using ripgrep	Describes how the **grep-find-command** user option can be modified to use ripgrep for the normal grep-find command. Note that you do not need to write Emacs code as described in that article, all you have to do is customize gre-find-command user option and change the value. By default, grep-find-command is nil, but is set by Emacs to: '("find . -type f -exec grep -nH --null -e \\{\\} +" . 42) Also note that changing grep-find-command will not impact the behaviour of the zgrep command, so it will still be slow. A better solution this problem is to leave grep-find-command alone and use the **rg.el** package which provides fast ripgrep searches, including gripping into zip files.
deadgrep @ GitHub	
Ag: the silver searcher	The ag command line utility is another fast search tool with features similar to ripgrep.
Ag @ GitHub	Github home of the ag command line utility.
ag.el @ Github	Github home for the Emacs package that provides access to the ag command line utility.