GNU Global Source Code Tag System

Edition 6.6.12, for GNU Global version 6.6.12
5 March 2024

by Tama Communications Corporation

Copyright © 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2008, 2010, 2011, 2012, 2013 Tama Communications Corporation

This manual is for GNU Global (version 6.6.12, 5 March 2024), a source code tagging system that works the same way across diverse environments.

Published by Tama Communications Corporation
Tama-city, Tokyo, Japan.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

1 Overview of this tool

1.1 What is GNU Global?

GNU Global is a source code tagging system that works the same way across diverse environments, such as Emacs editor, Vi editor, Less viewer, Bash shell, various web browsers, etc. You can find the locations of symbols such as functions, macros, structs and classes in your source code and move there easily. It is useful for hacking a large project which contains many sub-directories, many `#ifdef` and many `main()` functions. It is similar to ctags or etags, but is different from them at the point of independence of any editor.

1.2 Concept of project

GNU Global can treat a source tree containing sub-directories as a project. Anywhere in the project, you can utilize a high performance tag database. You need not specify where the database is, as global(1) locates it by itself. Because of this feature, you can move freely in a project, and in and out of many projects.

1.3 Features

GNU Global has the following features:

- supports C, Yacc, Java, PHP4 and assembly.
- works the same way across diverse environments like follows:
  - Shell command line
  - Bash shell
  - Vi editor (Nvi, Elvis, vim)
  - Less viewer
  - Emacs editor (Emacs, Mule, Xemacs)
  - Web browser
  - Doxygen documentation system
- finds locations of specified symbol quickly.
- locates not only definitions but also references.
- allows duplicate tags.
- locates paths which matches to the specified pattern.
- hierarchical searches by default.
- searches not only in a source project but also in library projects.
- generates completion list for completing input method.
- supports various output formats.
- allows customizing of a set of candidate files to be tagged.
- understands POSIX 1003.2 regular expression.
- supports idutils as an external search engine.
- tag files are independent of machine architecture.

- supports incremental updating of tag files.
- plug-in parser is available to treat new language.
- supports customizing using `gtags.conf`.
- generates a hypertext of source code.
- compact format to save disk space.
- supports client/server environment (TRAMP ready).
- ignores binary files, dot files and specified files.
- includes cscope-compatible program (gtags-cscope).
- includes grep-like command (-g command).
- supports grep-like symbol highlighting.

2 Command line

You can use tag facilities from the shell command line. It is a big merit of Global compared with any other tagging system.

2.1 Preparation

Before beginning, please read the FAQ (Frequently Asked Questions) file.

```
$ more /usr/local/share/gtags/FAQ
```

First of all, you must execute gtags(1) (see Section 5.2 [gtags], page 32) at the root of the source tree. For example, if you want to browse the source code of Vi editor in FreeBSD, please move to the source directory and invoke gtags(1).

```
$ cd /usr/src/usr.bin/vi
$ gtags
```

Gtags traverses sub-directories, picks up symbols from the source files and makes three tag files at the current directory. After this, all files and directories under the current directory are treated as a project.

```
$ ls G*
GPATH   GRTAGS  GTAGS
```

- `GTAGS` definition database
- `GRTAGS` reference database
- `GPATH` path name database

You should prepare for considerable disk space for tag files. For example, Linux-2.6.32 source code requires the following disk space.

```
source code(Linux-2.6.32)       390MB

GPATH                             6MB
GTAGS                            81MB
GRTAGS                          202MB
-------------------------------------
total of tag files              289MB
```

2.2 Basic usage

Consider the following source tree:

```
/home/user/
 |
 |-ROOT/      <- the root of source tree (GTAGS,GRTAGS,...)
    |
    |- README       .....   +---------------+
    |                       |The function of|
    |                       +---------------+
    |- DIR1/
    |  |
    |  |- fileA.c   .....   +---------------+
```

```
|  |                    |main(){        |
|  |                    |       func1();|
|  |                    |       func2();|
|  |                    |}              |
|  |                    +---------------+
|  |
|  |- fileB.c   .....   +---------------+
|                       |func1(){ ... } |
|                       +---------------+
|- DIR2/
   |
   |- fileC.c   .....   +---------------+
                       |#ifdef X       |
                       |func2(){ i++; }|
                       |#else          |
                       |func2(){ i--; }|
                       |#endif         |
                       |func3(){       |
                       |       func1();|
                       |}              |
                       +---------------+
```

- Once you make tag files at the root directory of a project, you can invoke global(1) from anywhere in the project. By default, you get relative paths of the located files.

```
$ cd /home/user/ROOT
$ global func1
DIR1/fileB.c            # func1() is defined in fileB.c
$ cd DIR1
$ global func1
fileB.c                 # relative path from DIR1
$ cd ../DIR2
$ global func1
../DIR1/fileB.c         # relative path from DIR2
```

  You can use global(1) only when you are in a project. If you are out of any project, it shows an error message like follows:

```
$ cd /home/user
$ global func1
global: GTAGS not found.
```

- The `-r` option locates references.

```
$ global -r func2
../DIR1/fileA.c         # func2() is referred from fileA.c
```

- You can use POSIX regular expressions.

```
$ cd /home/user/ROOT
$ global 'func[1-3]'
DIR1/fileB.c            # func1, func2 and func3 are matched
DIR2/fileC.c
```

- The `-x` option shows details. It is similar to the `-x` option of ctags(1).

```
$ global func2
DIR2/fileC.c
$ global -x func2
func2              2 DIR2/fileC.c       func2(){ i++; }
func2              4 DIR2/fileC.c       func2(){ i--; }
```

- The `-a` option produces absolute path names.

```
$ global -a func1
/home/user/ROOT/DIR1/fileB.c
```

- The `-s` command locates symbols which are not defined in `GTAGS`.

```
$ global -xs X
X                  1 DIR2/fileC.c #ifdef X
```

- The `-g` command locates lines which have the specified pattern.

```
$ global -xg '#ifdef'
#ifdef             1 DIR2/fileC.c #ifdef X
```

  It is similar to egrep(1) but is far more convenient for source code reading, because it allows you to search through a project, and only in source files.

  Additionally, you can use various options:

  `-O` search only in text files.

  `-o` search in both source files and text files.

  `-l` search only under the current directory.

  The `-e`, `-G` and `-i` options are available too. The usage is almost same as egrep(1).

  You can even change the output format of global(1) to grep style using `--result=grep` option.

  Of course, these options can be used even by other commands.

- The `-P` command locates path names which include the specified pattern.

```
$ global -P fileB
DIR1/fileB.c
$ global -P '1/'
DIR1/fileA.c
DIR1/fileB.c
$ global -P '\.c$'
DIR1/fileA.c
DIR1/fileB.c
DIR2/fileC.c
```

- The `-f` command prints a list of tags in the specified file(s).

```
$ global -f DIR2/fileC.c
func2              2 DIR2/fileC.c   func2(){ i++; }
func2              4 DIR2/fileC.c   func2(){ i--; }
func3              6 DIR2/fileC.c   func3(){
```

- The `-l` option limits the range of retrieval under the current directory.

```
$ cd DIR1
```

```
$ global -xl func[1-3]
func1        1 fileB.c      func1(){...}
```

2.3 Applied usage

- You can customize a set of candidate files to be tagged.

```
$ find . -type f -print >/tmp/list     # make a file set
$ vi /tmp/list                         # customize the file set
$ gtags -f /tmp/list
```

- If your source files are on a read-only device, such as CDROM, then you cannot make tag files at the root of the source tree. In such case, you can make tag files in another place using `GTAGSROOT` environment variable.

```
$ mkdir /var/dbpath
$ cd /cdrom/src                  # the root of source tree
$ gtags /var/dbpath              # make tag files in /var/dbpath
$ export GTAGSROOT=`pwd`
$ export GTAGSDBPATH=/var/dbpath
$ global func
```

  There is another method for it. Since global(1) locates tag files also in `/usr/obj` + <current directory>, you can setup like follows: It is 'obj' directory.

```
$ cd /cdrom/src                  # the root of source tree
$ mkdir -p /usr/obj/cdrom/src
$ gtags /usr/obj/cdrom/src       # make tag files in /usr/obj/cdrom/src
$ global func
```

  The `-O, --objdir` option does it automatically for you.

  The path of obj directory can be changed by environment variable `MAKEOBJDIRPREFIX`.

- If you want to locate symbols that are not defined in the source tree, then you can specify library directories with `GTAGSLIBPATH` environment variable.

  You should execute gtags(1) at each directory in the `GTAGSLIBPATH`. If `GTAGS` is not found there, global ignores such directories.

```
$ pwd
/develop/src/mh                  # this is a source project
$ gtags
$ ls G*TAGS
GRTAGS  GTAGS
$ global mhl
uip/mhlsbr.c                     # mhl() is found
$ global strlen                  # strlen() is not found
$ (cd /usr/src/lib; gtags)       # library source
$ (cd /usr/src/sys; gtags)       # kernel source
$ export GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
$ global strlen
../../../usr/src/lib/libc/string/strlen.c  # found in library
$ global access
../../../usr/src/sys/kern/vfs_syscalls.c   # found in kernel
```

Or, you can take a more straightforward way to do the same thing. In the following example, we treat as if the system library and the kernel are part of our project.

```
$ ln -s /usr/src/lib .
$ ln -s /usr/src/sys .
$ gtags
$ global strlen
lib/libc/string/strlen.c
$ global access
sys/kern/vfs_syscalls.c
```

- If you forget symbol names, you can use the `-c` (complete) command.

```
$ global -c kmem                # maybe k..k.. kmem..
kmem_alloc
kmem_alloc_pageable
kmem_alloc_wait
kmem_free
kmem_free_wakeup
kmem_init
kmem_malloc
kmem_suballoc                   # This is what I need!
$ global kmem_suballoc
../vm/vm_kern.c
```

- You can use the `-c` command with the complete command in the shell.

  In Bash:

```
$ funcs()
> {
>         local cur
>         cur=${COMP_WORDS[COMP_CWORD]}
>         COMPREPLY=(`global -c $cur`)
> }
$ complete -F funcs global
$ global kmem_TABTAB
kmem_alloc           kmem_alloc_wait      kmem_init
kmem_alloc_nofault   kmem_free            kmem_malloc
kmem_alloc_pageable  kmem_free_wakeup     kmem_suballoc
$ global kmem_sTAB
$ global kmem_suballoc
../vm/vm_kern.c
```

  If you like input completion, you should try globash (see Section 3.1 [GloBash], page 9). It supports you in a suitable way without any preparation.

- You can edit all files which have specified tags by typing one command. For example:

```
$ vi `global func1`     # edit fileB.c
```

- If you want to browse many files in order, do the following:

```
$ global -xr fork | awk '{printf "view +%s %s\n",$2,$3}'
```

```
view +650 ../dev/aic7xxx/aic7xxx_asm.c
view +250 ibcs2/ibcs2_misc.c
view +401 linux/linux_misc.c
view +310 ../kern/init_main.c
view +318 ../kern/init_main.c
view +336 ../kern/init_main.c
view +351 ../kern/init_main.c
$ !! | sh            # from now on, go to next tag with 'ZZ'.
```

3 Various applications

3.1 Global facility for Bash

Special support for Bash is available.

3.1.1 Features

- Vi-like tag stack is available.
- Emacs-like tag name completion is available.
- Automatic invoking of editor.
- Tag mark facility is available.
- You can manage a directory list by cookie facility.

3.1.2 Preparation

First, do the preparation of global. See Section 2.1 [Preparation], page 3. Then you can invoke globash(1) command.

```
$ globash
```

Only first time, you will see the following message.

```
GloBash needs a working directory. Do you create '/home/you/.globash'? ([y]/n)
```

Pressing the `ENTER` key, you will see a prompt as follows:

```
[/usr/src/sys]/kern _
```

This prompt means that the current directory is '/usr/src/sys/kern' and the root directory of the project is '/usr/src/sys'. Tag and marker are valid only in a project.

When you try to go out of the project, globash warns like:

```
[/usr/src/sys] cd ..
You are going to get out of the current project.
Tag stack and marker will be removed. Sure? ([y]/n)_
```

If you answer *y* `RET` or just `RET` then the tag stack and marker (described later) will be removed.

If you need help then please type *`ghelp`*.

3.1.3 Usage

- Most of global(1)'s (see Section 5.1 [global], page 26) command characters are available as commands.

```
[/usr/src/sys] x fork           <- (global -x fork)
>    1  fork              94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] r                <- (global -xr fork)
>    1  fork              85 alpha/linux/linux_machdep.c
     2  fork             184 i386/linux/linux_machdep.c
[/usr/src/sys] s lbolt          <- (global -xs lbolt)
>    1  lbolt            1210 i386/isa/wd_cd.c     tsleep((cad
     2  lbolt            1211 i386/isa/wd_cd.c     tsleep((cad
     3  lbolt             709 i386/isa/wfd.c     tsleep ((caddr
```

```
...
[/usr/src/sys] g                <- (global -xg lbolt)
>    1  lbolt            1210 i386/isa/wd_cd.c     tsleep((cad
...
[/usr/src/sys] P init           <- (global -xP init)
>    1  path    1 dev/hea/eni_init.c
     2  path    1 dev/hfa/fore_init.c
     3  path    1 i386/i386/initcpu.c
     4  path    1 kern/init_main.c
     5  path    1 kern/init_sysent.c
     6  path    1 kern/vfs_init.c
     7  path    1 vm/vm_init.c
[/usr/src/sys] _
```

If no argument is specified then the latest argument is used.

- Input completion facility is available. For each command, suitable completion is applied.

```
[/usr/src/sys] x kmem_TABTAB
kmem_alloc          kmem_free           kmem_malloc
kmem_alloc_nofault  kmem_free_wakeup    kmem_object
kmem_alloc_wait     kmem_init           kmem_suballoc
[/usr/src/sys] x kmem_sTAB
[/usr/src/sys] x kmem_suballoc
```

- You can select a tag by `show` command.

```
[/usr/src/sys] x main
>    1  main             70 alpha/alpha/gensetdefs.c main(in
     2  main            1500 alpha/alpha/ieee_float.c main(i
     3  main            227 boot/alpha/boot1/boot1.c main()
....
[/usr/src/sys] show 3
(Load editor and show boot/alpha/boot1/boot1.c at line 227.)
```

The default editor is vi(1). You can specify it statically by `EDITOR` environment variable or temporarily by options.

```
[/usr/src/sys] show -e 3
(Preloaded emacs show boot/alpha/boot1/boot1.c at line 227.)
[/usr/src/sys] show -l 3
(Load less and show boot/alpha/boot1/boot1.c at line 227.)
[/usr/src/sys] show -g 3
(Preloaded firefox show boot/alpha/boot1/boot1.c at line 227.)
```

Otherwise, you can use the following commands (with abbreviated form):

`list (l)` print tag list.

`first` go to the first tag.

`last` go to the last tag.

`next (n)` go to the next tag.

`prev (p)` go to the previous tag.

`show N (1,2,3,..,999)`
go to Nth tag

- You can use vi-like tag stack. You can return to the previous tag list by the `pop` or *`CTRL-T`* command.

```
[/usr/src/sys] x main
>    1  main             70 alpha/alpha/gensetdefs.c main(in
     2  main           1500 alpha/alpha/ieee_float.c main(i
     3  main            227 boot/alpha/boot1/boot1.c main()
....
[/usr/src/sys] show 3
(Load editor and show boot/alpha/boot1/boot1.c at line 227.)
[/usr/src/sys] x fork           <- push new tag on the tag stack.
>    1  fork             94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] pop              <- pop tag stack.
[/usr/src/sys] show
(Load editor and show boot/alpha/boot1/boot1.c at line 227.)
```

  You can print the current tag stack with `tags` command.

- You can remember tags using the `mark` command.

```
[/usr/src/sys] x fork
>    1  fork             94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] mark
[/usr/src/sys] x main
>    1  main             70 alpha/alpha/gensetdefs.c main(in
     2  main           1500 alpha/alpha/ieee_float.c main(i
     3  main            227 boot/alpha/boot1/boot1.c main()
....
[/usr/src/sys] mark -l          <- show marker list.
     1  fork             94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] mark 1           <- select a marker.
(Load editor and show kern/kern_fork.c at line 227.)
[/usr/src/sys] list
>    1  main             70 alpha/alpha/gensetdefs.c main(in
     2  main           1500 alpha/alpha/ieee_float.c main(i
     3  main            227 boot/alpha/boot1/boot1.c main()
....
```

  Marked tags are valid until you go out of the current project or quit the current bash(1) session.

- You can remember directories using the `cookie` command, and return there using the `warp` command.

```
[/usr/src/sys] cookie           <- drop a cookie.
[/usr/src/sys] cd kern
[/usr/src/sys]/kern cookie      <- drop a cookie again.
[/usr/src/sys]/kern cd ../i386
[/usr/src/sys]/i386 cookie -l   <- show cookie list.
```

```
    1  /usr/src/sys/kern
    2  /usr/src/sys
[/usr/src/sys]/i386 warp 2      <- warp to the selected cookie.
[/usr/src/sys] _
```

Cookie directories are valid until you delete them.

3.2 Less using Global

You can use Global as a tagging system of less(1) viewer instead of ctags.

3.2.1 Features

- You can use most of Global's facilities from less(1) viewer.
- Less viewer supports duplicated tag.

3.2.2 Preparation

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from less(1), you need to set environment variable `LESSGLOBALTAGS` to '`global`'.

```
$ export LESSGLOBALTAGS=global
```

3.2.3 Usage

- To go to `func1`, you can say

  ```
  $ less -t func1
  ```

  Please note that if `tags` exists in the current directory then less(1) uses it. If you want to use `GTAGS` even if `tags` exists then please specify the tag file explicitly like this:

  ```
  $ less -TGTAGS -t func1
  ```

- To go to the referenced point of `func1`, please specify `GRTAGS`.

  ```
  $ less -TGRTAGS -t func1
  ```

  In the same way, you can use `GTAGS`, `GRTAGS` or `GPATH` as tag files.

- If a number of tags are located, less(1) goes to the first tag. You can go to next tag by typing `t` and back by typing `T`.

  `t` go to next tag.

  `T` go to previous tag.

- In a less(1) session, you can use the `:t` command to locate a new symbol. But in this case, you cannot change the tag file from the one specified by the `-T` option.
- With the `-T-` option, less(1) reads standard input as a tag file. You can connect global and less(1) with a pipe. It is very convenient.

  ```
  $ global -x func | less -T-
  ```

  In the same way, you can use the following command lines:

  ```
  # pattern match with grep(1).
  $ global -xg 'lseek(.*)' | less -T-
  ```

```
# pattern match with idutils(1).
$ global -xI func | less -T-

# all definitions in *.c.
$ global -f *.c | less -T-

# all files including 'init' in their path.
$ global -Px init | less -T-
```

- If your editor doesn't support Global directly then you can use less(1) as a footstool.

```
# invoke less
$ less -t main
main(int argc, char **argv)
{
int i;
.....
[xxx/main.c (tag 1 of 55)]

# type 'v'(vi) command in less session.
v

# load vi and show the same position.
.....
main((int argc, char **argv)
{
int i;
.....
[xxx/main.c 313 lines, 7783 char]

# type 'ZZ' command in vi session.
ZZ

# exit vi and back to less session.
main(int argc, char **argv)
{
int i;
.....
[xxx/main.c (tag 1 of 55)]
```

3.3 Nvi-1.81.5 using Global

You can use Global as a tagging system of Nvi editor instead of ctags.

3.3.1 Features

- You can use most of Global's facilities from Nvi.
- Intelligent recognition of the current token and its type.

3.3.2 Preparation

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from Nvi, you need to write to `.nexrc` like this. It is assumed that `gtags.pl` is put in `$HOME/perl`.

```
$HOME/.nexrc
+----------------------------
|perl use lib "$ENV{'HOME'}/perl"
|perl require 'gtags.pl'
|map ^P :tagprev^M
|map ^N :tagnext^M
|map ^] :perl tag^M
|ab gtag perl tag qw(
|ab gta perl tag qw(
|ab gt perl tag qw(
```

You must start Nvi in a project as described in Section 2.1 [Preparation], page 3.

3.3.3 Usage

- To go to `func1`, you can say

  ```
  :perl tag qw(func1)
  ```

  Suggested .nexrc:

  ```
  ab gtag perl tag qw(
  ab gta perl tag qw(
  ab gt perl tag qw(
  ```

- To go to the referenced point of `func1`, add the option `-r`

  ```
  :perl tag qw(-r func1)
  ```

- If a number of tags are located, Nvi goes to the first tag. You can go to next tag by typing *`:tagnext`* and back by typing *`:tagprev`*.

  Suggested .nexrc:

  ```
  map ^N :tagnext^M
  map ^P :tagprev^M
  ```

- If you don't specify any argument, ':perl tag' command does the following:

  If the context of the current token is a definition then it is equivalent to *`:perl tag qw(-r current-token)`*. Otherwise, if it is a reference to some definitions then it is equivalent to *`:perl tag qw(current-token)`* else it is equivalent to *`:perl tag qw(-s current-token)`*.

  Suggested .nexrc:

  ```
  map ^] :perl tag^M
  ```

  It is similar to *`CTRL-]`* command.

- You can use the `-s` option; it locates symbols which are not defined in `GTAGS`.

  ```
  :perl tag qw(-s pat)
  ```

- The `-g`, `-f` and `-P` options are also available. It works like the command line.

  ```
  :perl tag qw(-g pat)
  ```

- When you want to locate tags the name of which start with '`set`' or '`get`', use:

```
:perl tag qw(^[sg]et)
```

- Other tag commands are also available:

CTRL-T return to the most recent tag location.

:tagpop return to the most recent tag location.

:tagtop return to the top of the tag stack.

:display tags
display the tags stack.

3.4 Elvis using Global

Elvis 2.1 or later has two variables, `tagprg` and `tagprgonce`, for running an external tag search program. You can use them for Global.

3.4.1 Features

- You can use most of Global's facilities from Elvis.
- Mouse is supported.

3.4.2 Preparation

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, start Elvis and execute `set tagprg="global -t $1"` like this:

```
$ elvis
~
~
~
~
~
~
:set tagprg="global -t $1"
```

3.4.3 Usage

- To go to `func1`, you can say

```
:tag func1
```

It seems the same as original Elvis, but Elvis executes `global -t func1` internally and read the output instead of tags file.

- To go to the referenced point of `func1`, add `-r` option.

```
:tag -r func1
```

- To locate symbols which are not defined in `GTAGS`, try this:

```
:tag -s lbolt
```

- To locate strings, try this:

```
:tag -g Copyright
```

- When a lot of results are expected, it's better to use the browse command.

```
:browse -r fork
```

  It brings a following selection list. You can select a tag line and go to the point.

```
Browse -r fork (2 matches)
+----------------+----------------+--------------------
| TAG NAME       | SOURCE FILE    | SOURCE LINE
+----------------+----------------+--------------------
|fork            |ux/linux_misc.c | (line 565)
|fork            |ern/init_main.c | (line 191)
+----------------+----------------+--------------------
```

- To get a list of tags in specified files, use the `-f` command.

```
:browse -f main.c                <- locate definitions in main.c
```

- Other tag commands are also available:

  CTRL-] go to the definition of the current token.

  CTRL-T return to the most recent tag context.

  :tag without argument, go to the next tag.

  :pop return to the most recent tag context.

  :stack display the tags stack.

  :stag create a new window and move its cursor to the tag's definition point.

  :sbrowse same as *browse* but show in a new window.

- You can use POSIX regular expressions.

```
:tag ^put_                    <- locate tags start with 'put_'

:browse -g 'fseek(.*L_SET)' <- locate fseek() using L_SET argument
```

- You can browse tag's list of many files.

```
:browse -f *.c                <- locate tags in *.c
```

- You can browse the files whose path includes specified pattern.

```
:browse -P /vm/               <- under vm/ directory
:browse -P \.h$               <- all include files
:browse -P init               <- path including 'init'
```

- You can use mouse for tag operations.

  If you have a mouse, then you can use the left button to double-click on a word in the text, to have Elvis perform a *:tag* search on that word. Double-clicking the right button anywhere in the text will perform a *:pop* command.

  In the selection list of the *browse* command, you can use the left button to double-click on a tag name, to have Elvis select the tag. To come back, double-click the right button.

3.5 Vim using Global

In Vim 6.2 or later, you can use the `gtags.vim` script.

3.5.1 Features

- You can use most of Global's facilities from Vim.
- Intelligent recognition of the current token and its type.
- Special characters '`%`', '`#`' and input completion are available.

To our regret, the tag stack facility is not available. If you want to use the facility, please try gtags-cscope.vim. See Section 3.7 [Gtags-cscope], page 21.

3.5.2 Preparation

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, copy `gtags.vim` to your plug-in directory or source it from your vimrc.

```
$ cp /usr/local/share/gtags/gtags.vim $HOME/.vim/plugin
```

3.5.3 Usage

- To go to `main`, you can say

  ```
  :Gtags main
  ```

  Vim executes global(1), parses the output, lists located tags in quickfix window and loads the first entry. The quickfix window is like this:

  ```
  gozilla/gozilla.c|200| main(int argc, char **argv)
  gtags-cscope/gtags-cscope.c|124| main(int argc, char **argv)
  gtags-parser/asm_scan.c|2056| int main()
  gtags-parser/gctags.c|157| main(int argc, char **argv)
  gtags-parser/php.c|2116| int main()
  gtags/gtags.c|152| main(int argc, char **argv)
  [Quickfix List]
  ```

  You can go to any entry using quickfix command.

  :cn go to the next entry.

  :cp go to the previous entry.

  :ccN go to the N'th entry.

  :cl list all entries.

  You can see the help of quickfix like this:

  ```
  :h quickfix
  ```

  Suggested map:

  ```
  map <C-n> :cn<CR>
  map <C-p> :cp<CR>
  ```

- To go to the referenced point of `func1`, add the `-r` option.

  ```
  :Gtags -r func1
  ```

- To locate symbols which are not defined in `GTAGS`, try this:

```
:Gtags -s lbolt
```

- To locate strings, try this:

```
:Gtags -g int argc

:Gtags -g "root"

:Gtags -ge -C                <- locate '-C'
```

- To get a list of tags in specified files, use the `-f` command.

```
:Gtags -f main.c             <- locate tags in main.c
```

  If you are editing `main.c` itself, you can use '`%`' instead.

```
:Gtags -f %                  <- locate tags in main.c
```

- You can use POSIX regular expressions.

```
:Gtags ^put_               <- locate tags starting with 'put_'

:Gtags -g fseek(.*SEEK_SET) <- locate fseek() using SEEK_SET
```

- Input completion is available.

  In the command line, press *`CTRL-D`* after some typing and Vim will show a list of tag names that start with the string. Press `TAB` and Vim will complete the tag name.

```
:Gtags fuTAB

:Gtags func1                 <- 'nc1' is appended by vim
```

- You can browse files whose path includes specified pattern.

```
:Gtags -P /vm/               <- under vm/ directory
:Gtags -P \.h$               <- all include files
:Gtags -P init               <- path including 'init'
```

- You can use all options of global(1) except for `-c -n -p -q -u -v` and all long name options. They are sent to global(1) as is. For example:

```
:Gtags -gi paTtern         <- matches both 'PATTERN' and 'pattern'

:Gtags -POi make           <- matches Makefile but not makeit.c
```

  About the other options, please see Section 5.1 [global], page 26.

- The GtagsCursor command brings you to the definition or reference of the current token.

  If the context of the current token is a definition then it is equivalent to *`:Gtags -r current-token`*; if it is a reference to some definitions then it is equivalent to *`:Gtags current-token`*; else it is equivalent to *`:Gtags -s current-token`*.

```
:GtagsCursor
```

  Suggested map:

```
map <C-\>^] :GtagsCursor<CR>
```

  Though the mapping of *`:GtagsCursor`* to *`^]`* seems suitable, it will bring an inconvenience in the help screen.

- If you have a hypertext generated by htags(1) then you can display the same part of the source code in your browser. Let's load firefox and try this:

  ```
  :Gozilla
  ```

  Suggested map:

  ```
  map <C-g> :Gozilla<CR>
  ```

- If you want to load Vim with all main()s then following command line is useful.

  ```
  $ vim '+Gtags main'
  ```

3.6 Extended Emacs using Global

You can use Global as a tagging system of Emacs editor instead of etags.

3.6.1 Features

- You can use most of Global's facilities from the editor.
- More intelligent recognition of the current token and its type.
- Tag completion is available for input tag name.
- Mouse is supported.

3.6.2 Preparation

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from Emacs, you need to load the `gtags.el` and execute `gtags-mode` function in it.

Write the call to autoload function to your `$HOME/.emacs`, start Emacs and execute `gtags-mode` function. If you put `gtags.el` in a directory other than the standard macro directory, you need to add it to `load-path`.

```
$HOME/.emacs
+-------------------------------------------------------
|(setq load-path (cons "/home/owner/global" load-path))
|(autoload 'gtags-mode "gtags" "" t)

$ emacs

|
|J_:-----Mule: *scratch*       (Lisp Interaction)--L16--All----
|M-x gtags-mode[RET]
+-------------------------------------------------------
```

If you want to get into gtags-mode whenever you get into c-mode then you can append the following code to your `$HOME/.emacs`.

```
(setq c-mode-hook
    '(lambda ()
        (gtags-mode 1)
))
```

About key mappings, please see the comment of gtags.el.

3.6.3 Usage

- To go to `func1`, invoke `gtags-find-tag` and you can see a prompt in the mini-buffer. Then input the tag name.

  ```
  Find tag: func1 <- 'Find tag: ' is a prompt
  ```

- To go to the referenced point of `func1`, invoke `gtags-find-rtag`.

  ```
  Find tag (reference): func1
  ```

- Tag name completion is available.

  ```
  Find tag: fuTAB

  Find tag: func1                          <- 'nc1' is appended by emacs
  ```

- If a number of tags are located, Emacs goes into *GTAGS SELECT MODE* like this:

  ```
  +-------------------------------------------------------------
  |main             347 i386/isa/ultra14f.c main()
  |main             128 kern/init_main.c   main(framep)
  |main             104 netiso/clnp_debug.c main()
  |main             164 netiso/xebec/main.c main(argc, argv)
  |
  |
  |
  |
  |
  |J_:--%*-Mule: *scratch*       (Gtags Select)--L1--All----
  |[GTAGS SELECT MODE] 4 lines
  +-------------------------------------------------------------
  ```

  Please select a tag line by any Emacs command and press `RET`, and you can go to the tag's point. When you want to go to the next or previous tag, please return to the above mode with `gtags-pop-stack` and reselect.

  You can customize the path style in this mode by setting `gtags-path-style` variable.

  `root` relative from the root of the project (Default)

  `relative` relative from the current directory

  `absolute` absolute (relative from the system root directory)

  There are two methods to set this variable:

  - You can change it dynamically using the `customize` command of Emacs. You will find the entry in the Programming/Tools/Gtags group.
  - You can change it when Emacs is loaded using `.emacs` file like this:

    ```
    (setq gtags-mode-hook
      '(lambda ()
        (setq gtags-path-style 'relative)))
    ```

- `gtags-find-tag-from-here` command is available.

  If current token is a definition, it is equivalent to *`Find tag (reference): current-token`*`RET`, otherwise it is equivalent to *`Find tag: current-token`*`RET`.

- To locate symbols which are not defined in `GTAGS`, try `gtags-find-symbol`.

```
Find symbol: lbolt          <- 'Find symbol:' is a prompt
```

- To locate strings, try `gtags-find-with-grep`.

```
Find pattern: Copyright
```

- You can use POSIX regular expressions.

```
Find tag: ^put_             <- locate tags starting with 'put_'
```

- Mouse command is available.

  If you use X version Emacs, try the following:

  Move the mouse cursor to a symbol name and click the middle button, and you can go to the point of the definitions, or to its references, depending on the context. In 'GTAGS SELECT MODE', move the mouse cursor to a line and click the center button.

  To return to the previous position, click the right button.

3.7 Gtags-cscope

You can also use cscope as a client of GNU Global. Global package includes a command named 'gtags-cscope' which is a port of cscope, that is, it is cscope itself except that it use Global as a search engine instead of cscope's one.

```
$ gtags-cscope
```

Its function is available from vim editor. Please input like follows in the command mode.

```
:set csprg=gtags-cscope
:cs add GTAGS
```

After this, you can use built-in `cs find` commands in the Vim editor. Though the deceit is not perfect (`cs find d` is not implemented), this method might be more convenient than `gtags.vim` in that you can use the tag stack facility of Vim.

3.8 Hypertext generator

You can use Global's facilities from web browsers.

3.8.1 Features

- Htags makes a hypertext from C, Yacc and Java source files.
- Once the hypertext is generated, you need nothing other than a web browser.
- You can move the hypertext to anywhere; it is independent of the source code.
- You can use all of your browser's functions, such as search, history, bookmark, save, frames, windows, etc.

3.8.2 Preparation

At first, you must ensure that you have a lot of disk space for hypertext. For example, Linux-2.6.32 source code (390MB) requires 4–6 GB of disk space.

```
source code(Linux-2.6.32)           390MB
```

```
GPATH,GTAGS,GRTAGS                  289MB

hypertext (with no option)          3.8GB
hypertext (with --suggest option)   5.7GB
```

Please invoke gtags(1) (see Section 5.2 [gtags], page 32) and htags(1) (see Section 5.3 [htags], page 38) in order like this:

```
(at the root directory of your source project)
$ gtags                 # make tag files(GPATH,GTAGS,GRTAGS)
$ htags                 # make hypertext(HTML/)
```

Then you will find a directory named `HTML` in the current directory.

Htags has rich options. If you are new on htags then you are recommended to use the `--suggest` option. This option makes some popular options effective, and invokes gtags(1) if there is no tag files.

```
$ htags --suggest
```

If HTTP server is available then the `-D` and `-f` options are also useful.

3.8.3 Usage

Please start a web browser like this:

```
$ lynx HTML/index.html
```

You will understand the usage by looking at the examples.

You can move the HTML directory to anywhere. It is independent of the source code as long as CGI facility is not used.

Using firefox, you can also utilize the hypertext from your command line like this:

```
$ global -x main
main        10 main.c main(int argc, char *argv[]) {
$ gozilla +10 main.c                # usage is similar to vi editor.
(show main.c at 10 on your browser.)
```

But in this case, you must not move the HTML directory from the source directory.

3.9 Doxygen using Global

You can use Global as a source browser of Doxygen.

Doxygen Release 1.4.3 or later has config option `USE_HTAGS`. When enabled in combination with `SOURCE_BROWSER=YES`, htags(1) is used as a source browser instead of Doxygen's own.

Here is an example.

```
(in source directory)
$ doxygen -g
$ vi Doxyfile
+---------------------------------
|...
|INPUT                  = .
|RECURSIVE              = YES
|SOURCE_BROWSER         = YES
```

```
|USE_HTAGS               = YES
|...

$ doxygen
$ lynx html/index.html
```

4 Other topics

4.1 How to configure Global

You can customize Global using configuration file.

```
# cp gtags.conf /etc/gtags.conf         # system wide config file.
# vi /etc/gtags.conf

$ cp gtags.conf $HOME/.globalrc         # personal config file.
$ vi $HOME/.globalrc
```

If `$HOME/.globalrc` exists then Global uses it; else if `/etc/gtags.conf` exists then Global uses it; otherwise default value is used. The format of `gtags.conf` resembles termcap(5). By default, 'default' target is used. About the capabilities, please see each command manual. See Chapter 5 [Reference], page 26.

4.2 Plug-in parser

You can write a new parser for gtags(1).

Command layer plug-in parser was abolished. Please write function layer plug-in parser instead. See `plugin-factory/` to discover the function layer plug-in parser.

You can use Universal Ctags as a plug-in parser too.

```
# Installation of Global
# It assumed that ctags command is installed in '/usr/local/bin'.

$ ./configure --with-universal-ctags=/usr/local/bin/ctags
$ make
$ sudo make install

# Executing of gtags
# It assumed that Global is installed in '/usr/local'.

$ export GTAGSCONF=/usr/local/share/gtags/gtags.conf
$ export GTAGSLABEL=new-ctags
$ gtags                          # gtags invokes Universal Ctags internally
```

4.3 Incremental updating

Modifying some source files, you need not remake the entire tag files. Instead, you can use incremental updating facility (`-u` option).

```
$ gtags
$ cd kernel
$ vi user.c                              # modify user.c
...
:wq
$ global -vu                             # -v means verbose
[Sat May 29 00:31:41 JST 2010] Gtags started.
```

```
 Tag found in '/usr/local/src/linux-2.6.32'.
 Incremental updating.
[Sat May 29 00:31:43 JST 2010] Updating 'GTAGS' and 'GRTAGS'.
 [1/1] deleting tags of kernel/user.c
 [1/1] extracting tags of kernel/user.c
 Global databases have been modified.
[Sat May 29 00:31:51 JST 2010] Done.

$ global -vu                            # try again
[Sat May 29 00:33:16 JST 2010] Gtags started.
 Tag found in '/usr/local/src/linux-2.6.32'.
 Incremental updating.
 Global databases are up to date.       # do nothing
[Sat May 29 00:33:19 JST 2010] Done.
```

5 Command References

5.1 global - print locations of given symbols

NAME

global - print locations of given symbols

SYNOPSIS

global [-adEFGilMnNqrstTvx][-S dir][-e] pattern
global -c[dFiIMoOPrsT] prefix
global -f[adlnqrstvx][-L file-list][-S dir] files
global -g[aEGilMnoOqtvVx][-L file-list][-S dir][-e] pattern [files]
global -I[ailMnqtvx][-S dir][-e] pattern
global -P[aEGilMnoOqtvVx][-S dir][-e] pattern
global -p[qrv]
global -u[qv]

DESCRIPTION

Global finds locations of given symbols in C, Yacc, Java, PHP and assembly source files, and prints the path name, line number and line image of the locations. Global can locate not only definitions but also references and other symbols.

Global can treat a source tree, that is, a directory that has sub-directories and source files, as a project. In advance of using this command, you must execute gtags(1) at the root directory of the project which you want to investigate to make tag files. Then you can use global command anywhere in the project. You need not specify where the tag file is. Instead, global locates it by itself.

You can specify a regular expression for pattern. Global understands two different versions of regular expression syntax: basic and extended (default).

COMMANDS

The following commands are available:

<no command> pattern
: No command means tag search command. Print tags which match to pattern. By default, print definition tags.

'`-c`', '`--completion[=limit]`' [prefix]
: Print symbols which start with prefix. If prefix is not given, print all symbols. If limit is given, print up to limit lines.

'`-f`', '`--file`' files
: Print all tags in the files. This command implies the '`-x`' option.

'`-g`', '`--grep`' pattern [files]
: Print all lines which match to the pattern. If files are given, this command searches in those files.

‘`--help`’ Print a usage message.

‘`-I`’, ‘`--idutils`’ pattern
: Print all lines which match to pattern. This function uses idutils(1) as a search engine. To use this command, you need to install idutils(1) in your system and execute gtags(1) with the ‘`-I`’ option.

‘`-P`’, ‘`--path`’ [pattern]
: Print path names which match to pattern. If no pattern is given, print all paths in the project.

‘`-p`’, ‘`--print-dbpath`’
: Print location of `GTAGS`.

‘`--print`’ name
: Print location of name, which may be one of: ‘`root`’, ‘`dbpath`’ or ‘`conf`’. ‘`root`’ means project's root directory. ‘`dbpath`’ means a directory where tag databases exist. ‘`conf`’ means configuration file.

‘`-u`’, ‘`--update`’
: Update tag files incrementally. This command internally invokes gtags(1). You can execute this command anywhere in the project, differing from gtags(1).

‘`--version`’
: Show version number.

OPTIONS

The following options are available:

‘`-a`’, ‘`--absolute`’
: Print absolute path names. By default, print relative path names.

‘`--color`’ when
: Use color to highlight the pattern within the line; when may be one of: never, always or auto (default). The default color is bold red text on current background; the environment variable *GREP_COLORS* (only *mt* and *ms* are effective at present) or *GREP_COLOR* defines it. This option is effective to the following commands: <no command>, -f, -g, -I, -P.

‘`-C`’, ‘`--directory`’ dir
: Change the directory before doing all the work including parameter analysis.

‘`-d`’, ‘`--definition`’
: Print locations of definitions.

‘`-e`’, ‘`--regexp`’ pattern
: Use pattern as the pattern; useful to protect patterns starting with ‘`-`’.

‘`-E`’, ‘`--extended-regexp`’
: Interpret pattern as a extended regular expression. This is the default.

‘`--encode-path`’ chars
: Convert path characters in chars into a ‘`%`’ symbol, followed by the two-digit hexadecimal representation of the character. A blank will be converted to ‘`%20`’.

‘`-F`’, ‘`--first-match`’
: End the search without going through all the tag files listed in *GTAGSLIBPATH* when tags are found in a tag file. This is the default.

‘`--from-here`’ context
: Decide tag type by context. Its syntax should be ‘`lineno:path`’. If the context is a definition of the pattern then use ‘`-r`’, else if there is at least one definition of the pattern then use ‘`-d`’, else use ‘`-s`’. If this option is specified then ‘`-d`’, ‘`-r`’ and ‘`-s`’ on the command line are ignored. Regular expression is not allowed for pattern. This option assumes use in conversational environments such as editors and IDEs.

‘`-G`’, ‘`--basic-regexp`’
: Interpret pattern as a basic regular expression. The default is an extended regular expression.

‘`--gtagsconf`’ file
: Set environment variable *GTAGSCONF* to file.

‘`--gtagslabel`’ label
: Set environment variable *GTAGSLABEL* to label.

‘`-i`’, ‘`--ignore-case`’
: Ignore case distinctions in the pattern.

‘`-L`’, ‘`--file-list`’ file
: Obtain files from file in addition to the arguments. The argument file can be set to ’-’ to accept a list of files from the standard input. File names must be separated by newline.

‘`-l`’, ‘`--local`’
: Print only tags which exist under the current directory.

‘`--literal`’
: Execute literal search instead of regular expression search. This option works with the tag search command, ‘`-g`’ command, ‘`-P`’ command and ‘`-I`’ command.

‘`-M`’, ‘`--match-case`’
: Search is case-sensitive. This is the default.

‘`--match-part part`’
: Specify how path name completion should match, where part is one of: ‘`first`’, ‘`last`’ or ‘`all`’ (default). This option is valid only with the ‘`-c`’ command in conjunction with ‘`-P`’.

‘`-n`’, ‘`--nofilter`’
: Suppress sort filter and path conversion filter.

‘`-N`’, ‘`--nearness`’[=start]
: Use Nearness sort method (sorting by closest from start) for the output. By default, alphabetical sort method is used. This option is effective for the tag search command, ‘`-P`’ command and ‘`-g`’ command. As an exception, ‘`-g`’ command ignores this option when files are specified by arguments. The nearness is defined by how many parent directories to go up to reach the target. The

result of nearness sort is concatenation of the following ([0]-[n]) in this order. The default of start is the current directory.

```
[0] If the start is a file, output of local search in the file.
[1] Output of local search in the start directory except for [0].
[2] Output of local search in the parent directory except for [0]-[1].
[3] Output of local search in the grandparent directory except for [0]-[2].
... (repeat until the project root directory)
[n] Output of local search in the project root directory except for [0]-[n-1
```

In each directory, they are sorted by alphabetical order.

'`-O`', '`--only-other`'
: Treat only text files other than source code, like `README`. This option is valid only with the '`-g`' or '`-P`' command. This option overrides the '`-o`' option.

'`-o`', '`--other`'
: Treat not only source files but also text files other than source code, like `README`. This option is valid only with the '`-g`' or '`-P`' command.

'`--path-style`' format
: Print path names using format, which may be one of: '`relative`', '`absolute`', '`shorter`', '`abslib`' or '`through`'. '`relative`' means relative path. '`absolute`' means absolute path. '`shorter`' means the shorter one of relative and absolute path. '`abslib`' means absolute path for libraries (GTAGSLIBPATH) and relative path for the rest. '`through`' means the relative path from the project root directory (internal format of GPATH). The default is '`relative`'. The '`--path-style`' option is given more priority than the '`-a`' option.

'`--print0`'
: Print each record followed by a null character instead of a newline.

'`-q`', '`--quiet`'
: Quiet mode.

'`-r`', '`--reference`', '`--rootdir`'
: Print reference tags. Reference means the reference to a symbol which has definitions. With the '`-p`' option, print the root directory of the project.

'`--result`' format
: Print out using format, which may be one of: '`path`' (default), '`ctags`', '`ctags-x`', '`grep`' or '`cscope`'. The '`--result=ctags`' and '`--result=ctags-x`' options are equivalent to the '`-t`' and '`-x`' options respectively. The '`--result`' option is given more priority than the '`-t`' and '`-x`' options.

'`--single-update`' file
: Update tag files using gtags(1) with the '`--single-update`' option. It is considered that file was added, updated or deleted, and there is no change in other files. This option implies the '`-u`' option.

'`-s`', '`--symbol`'
: Print other symbol tags. Other symbol means the reference to a symbol which has no definition.

'`-S`', '`--scope`' dir
: Print only tags which exist under dir directory. It is similar to the '`-l`' option, but you need not change directory.

'`-T`', '`--through`'
: Go through all the tag files listed in *GTAGSLIBPATH*. By default, stop searching when tag is found. This option is ignored when either '`-s`', '`-r`' or '`-l`' option is specified.

'`-t`', '`--tags`'
: Use standard ctags format.

'`-V`', '`--invert-match`'
: Invert the sense of matching, to select non-matching lines. This option is valid only with the '`-g`' or '`-P`' commands.

'`-v`', '`--verbose`'
: Verbose mode.

'`-x`', '`--cxref`'
: Use standard ctags cxref (with '`-x`') format.

EXAMPLES

```
$ ls -F
Makefile      src/    lib/
$ gtags
$ ls G*
GPATH   GRTAGS  GTAGS
$ global main
src/main.c
$ (cd src; global main)
main.c
$ global -x main
main              10 src/main.c  main (argc, argv) {
$ global -f src/main.c
main              10 src/main.c  main (argc, argv) {
func1             55 src/main.c  func1() {
func2             72 src/main.c  func2() {
func3            120 src/main.c  func3() {
$ global -x '^[sg]et'
set_num           20 lib/util.c  set_num(values) {
get_num           30 lib/util.c  get_num() {
$ global -rx set_num
set_num          113 src/op.c            set_num(32);
set_num          225 src/opop.c               if (set_num(0) > 0) {
$ global strlen
$ (cd /usr/src/sys; gtags)
$ export GTAGSLIBPATH=/usr/src/sys
$ global -a strlen
/usr/src/sys/libkern/strlen.c
```

```
$ (cd /usr/src/lib; gtags)
$ GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
$ global -a strlen
/usr/src/lib/libc/string/strlen.c
```

FILES

`GTAGS` Tag file for definitions.

`GRTAGS` Tag file for references.

`GPATH` Tag file for source files.

`GTAGSROOT`
: If environment variable *GTAGSROOT* is not set and file `GTAGSROOT` exists in the same directory as `GTAGS` then global sets *GTAGSROOT* to the contents of the file.

`gtags.conf`, `$HOME/.globalrc`
: Configuration data for GNU Global. See gtags.conf(5).

ENVIRONMENT

The following environment variables affect the execution of global:

GREP_COLOR
: The color to use for '`--color`'; *GREP_COLORS* has precedence.

GREP_COLORS
: The color (mt or ms) to use for '`--color`'; see grep(1).

GTAGSBLANKENCODE
: If this variable is set, the '`--encode-path=" <TAB>"`' option is specified.

GTAGSCACHE
: The size of the B-tree cache. The default is 50000000 (bytes).

GTAGSCONF
: Configuration file.

GTAGSDBPATH
: The directory in which the tag files exist. This value is ignored when *GTAGSROOT* is not defined. Use of this variable is not recommended.

GTAGSFORCECPP
: If this variable is set, each file whose suffix is `.h` is treated as a C++ source file.

GTAGSLABEL
: Configuration label. The default is default.

GTAGSLIBPATH
: If this variable is set, it is used as the path to search for library functions. If the given symbol is not found in the current project, global also searches in these paths. Since only `GTAGS` is targeted in the retrieval, this variable is ignored when '`-r`' or '`-s`' is specified.

GTAGSLOGGING
: If this variable is set, `$GTAGSLOGGING` is used as the path name of a log file. There is no default value.

GTAGSROOT
: The root directory of the project. Usually, it is recognized by existence of `GTAGS`. Use of this variable is not recommended.

GTAGSTHROUGH
: If this variable is set, the '`-T`' option is specified.

GTAGSOBJDIR, *MAKEOBJDIR*
: If eigher of the two variable is set, it is used as the name of BSD-style objdir. The former is given priority. The default is `obj`.

GTAGSOBJDIRPREFIX, *MAKEOBJDIRPREFIX*
: If eigher of the two variable is set, it is used as the prefix of BSD-style objdir. The former is given priority. The default is `/usr/obj`.

CONFIGURATION

The following configuration variables affect the execution of global:

`icase_path` (boolean)
: Ignore case distinctions in pattern.

DIAGNOSTICS

Global exits with a non-0 value if an error occurred, 0 otherwise.

SEE ALSO

gtags(1), htags(1), less(1), gtags.conf(5).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

AUTHOR

Shigio YAMAGUCHI, Hideki IWAMOTO and others.

HISTORY

The global command appeared in FreeBSD 2.2.2.

5.2 gtags - create tag files for global

NAME

gtags - create tag files for global

SYNOPSIS

gtags [-ciIOqvw][-C dir][-d tag-file][-f file][dbpath]

DESCRIPTION

Gtags is used to create tag files for global(1).

Gtags recursively collects source files under the current directory, picks up symbols and writes the cross-reference data into the tag files (`GTAGS`, `GRTAGS` and `GPATH`).

By default, gtags picks up C, Yacc, assembly language, Java, C++ and PHP source files. Files whose names end in `.c`, `.h` are assumed to be C source files. Files whose names end in `.y` are assumed to be Yacc source files. Files whose names end in `.s`, `.S` are assumed to be assembly language source files. Files whose names end in `.java` are assumed to be Java source files. Files whose names end in `.c++`, `.cc`, `.hh`, `.cpp`, `.cxx`, `.hxx`, `.hpp`, `.C`, `.H` are assumed to be C++ source files. Files whose names end in `.php`, `.php3`, `.phtml` are assumed to be PHP source files. Other files are assumed to be text files. Gtags does not treat binary files.

If `gtags.files` exists in the current directory or a file is specified by the '`-f`' option, target files are limited by it. Lines starting with '`. `' are comments.

OPTIONS

The following options are available:

'`--accept-dotfiles`'
: Accept files and directories whose names begin with a dot. By default, gtags ignores them.

'`-c`', '`--compact`'
: Make `GTAGS` in compact format. This option does not influence `GRTAGS`, because that is always made in compact format.

'`-C`', '`--directory`' dir
: Change the directory before doing all the work including parameter analysis. This option is ignored in *GTAGS_OPTIONS*. Please specify it on the command line directly.

'`--config`'[=name]
: Print the value of config variable name. If name is not specified then print all names and values. In addition to the variables listed in the ENVIRONMENT section, you can refer to install directories by read only variables: *bindir*, *libdir*, *datadir*, *localstatedir* and *sysconfdir*.

'`-d`', '`--dump`' tag-file
: Dump a tag file as text to the standard output. Output format is 'key<tab>data'. This is for debugging.

'`--explain`'
: Explain handling files.

'`-f`', '`--file`' file
: Give a list of candidates of target files. Files which are not on the list are ignored. The argument file can be set to '`-`' to accept a list of files from the standard input. File names must be separated by newline. To make the list you may use find(1), which has rich options for selecting files.

‘`--gtagsconf`’ file
: Set environment variable *GTAGSCONF* to file.

‘`--gtagslabel`’ label
: Set environment variable *GTAGSLABEL* to label.

‘`--help`’ Print a usage message.

‘`-I`’, ‘`--idutils`’
: In addition to tag files, make ID database for idutils(1).

‘`-i`’, ‘`--incremental`’
: Update tag files incrementally. It's better to use global(1) with the ‘`-u`’ command.

‘`-O`’, ‘`--objdir`’
: Use BSD-style obj directory as the location of tag files. If *GTAGSOBJDIRPREFIX* is set and `$GTAGSOBJDIRPREFIX` directory exists, gtags creates `$GTAGSOBJDIRPREFIX/<current directory>` directory and makes tag files in it. Though you can use *MAKEOBJDIRPREFIX* instead of *GTAGSOBJDIRPREFIX*, it is deprecated. If dbpath is specified, this option is ignored.

‘`--single-update`’ file
: Update tag files for a single file. It is considered that file was added, updated or deleted, and there is no change in other files. This option implies the ‘`-i`’ option.

‘`--skip-unreadable`’
: Skip unreadable files.

‘`--skip-symlink`’ [=type]
: Skip symbolic links. If type is 'f' then skip only symbolic links for file, else if 'd' then skip only symbolic links for directory. The default value of type is 'a' (all symbolic links).

‘`--sqlite3`’
: Use Sqlite 3 API to make tag files. By default, BSD/DB 1.85 API is used. To use this option, you need to invoke configure script with ‘`--with-sqlite3`’ in the build phase.

‘`--statistics`’
: Print statistics information.

‘`-q`’, ‘`--quiet`’
: Quiet mode.

‘`-v`’, ‘`--verbose`’
: Verbose mode.

‘`--version`’
: Show version number.

‘`-w`’, ‘`--warning`’
: Print warning messages.

dbpath The directory in which tag files are generated. The default is the current directory.

EXAMPLES

```
$ ls -F
Makefile      src/    lib/
$ gtags -v
$ global -x main
main              10 src/main.c  main (argc, argv) {
```

FILES

`GTAGS` Tag file for definitions.

`GRTAGS` Tag file for references.

`GPATH` Tag file for source files.

`gtags.conf`, `$HOME/.globalrc`
Configuration data for GNU Global. See gtags.conf(5).

`gtags.files`
The list of candidates of target files.

`.notfunction`
The list of symbols which is not a definition. If this file exists in the project root directory, gtags does not regard the symbols listed in this file as definitions. Lines starting with ';' are comment lines.

ENVIRONMENT

The following environment variables affect the execution of gtags:

GTAGSCACHE
The size of the B-tree cache. The default is 50000000 (bytes).

GTAGSCONF
Configuration file.

GTAGSFORCECPP
If this variable is set, each file whose suffix is `.h` is forcibly parsed by the built-in C++ parser. Do not set this variable if you are using plug-in parsers. This variable and the built-in C++ parser are deprecated.

GTAGSFORCEENDBLOCK
If this variable is set, each } at the first column brings end of block of 0 level.

GTAGSLABEL
Configuration label. The default is '`default`'.

GTAGSLOGGING
If this variable is set, `$GTAGSLOGGING` is used as the path name of a log file. There is no default value.

GTAGS_COMMANDLINE
: This variable can only be referenced from the hook (see gtags_hook). Gtags sets its own effective command line to this variable before calling the hook. Each argument is separated by whitespace, and real whitespace is represented as '%20'. This is read only.

GTAGS_OPTIONS
: The value of this variable is inserted in the head of arguments.

GTAGSOBJDIR
: If this variable is set, it is used as the name of BSD-style objdir. The default is `obj`. Though you can use *MAKEOBJDIR* instead of *GTAGSOBJDIR*, it is deprecated.

GTAGSOBJDIRPREFIX
: If this variable is set, it is used as the prefix of BSD-style objdir. The default is `/usr/obj`. Though you can use *MAKEOBJDIRPREFIX* instead of *GTAGSOBJDIRPREFIX*, it is deprecated.

TMPDIR
: The location used to stored temporary files. The default is `/tmp`.

CONFIGURATION

The following configuration variables affect the execution of gtags. You can see the default value for each variable with the '`--config`' option.

`gtags_parser` (comma separated list)
: Specify the mapping of language names and plug-in parsers. Each part delimited by the comma consists of a language name, a colon, the shared object path, an optional colon followed by a function name. If the function name is not specified, 'parser' is assumed. As a special exception, gtags collects values from multiple `gtags_parser` variables. For these mappings, the first match is adopted.

`gtags_hook` (command line)
: Specify a command line which should be executed at the beginning of gtags after loading configuration file. You can use this hook to update `gtags.files` dynamically. `"./"` in it always means the project root directory, since gtags is always invoked there.
This hook is ignored when the following options are specified: –version, –help, –config, –dump.

`icase_path` (boolean)
: Ignore case distinctions in the path. Suffixes check is affected by this capability.

`langmap` (comma separated list)
: Language mapping. Each comma-separated map consists of a language name, a colon, and a list of file extensions. You can specify a glob pattern surrounded by parentheses instead of an extension for the files without extensions (e.g. Make:([Mm]akefile).mak.mk). As a special exception, gtags collects values from multiple `langmap` variables. For these mappings, the first match is adopted. Default mapping is:
'`c:.c.h,yacc:.y,asm:.s.S,java:.java,cpp:.c++.cc.hh.cpp.cxx.hxx.hpp.C.H,php:.php.ph`

`skip` (comma separated list)

Gtags skips files and directories which are given in this list. As a special exception, gtags collects values from multiple `skip` variables. If the value ends with '`/`', it is assumed as a directory and gtags skips all files under it. The value may include glob patterns (*, ?, [...], [!...], [^...]).
If the value starts with '`/`', it is assumed a relative path name from the root directory of the project. You cannot use glob patterns for a path name. However, this direction is out-of-date, and is not recommended. Instead, you can use '`-f`' option which can be combined with find(1). Since find(1) has rich options to select files, you can do everything. Additionally, this technique can also be applied to any other tagging systems like ctags(1), cscope(1), etc.
Skip list is also effective when you use the '`-f`' or `gtags.files`.

EXAMPLES

```
#
# There are several ways to use Universal Ctags plug-in parser.
#
$ gtags --gtagsconf=/usr/local/share/gtags/gtags.conf --gtagslabel=universal-ctags
$ gtags --explain
or
$ export GTAGSCONF=/usr/local/share/gtags/gtags.conf
$ export GTAGSLABEL=universal-ctags
$ gtags --explain
or
# In your project
$ cp /usr/local/share/gtags/gtags.conf .
$ vi gtags.conf # default:tc=native => default:tc=universal-ctags
$ gtags --explain
```

DIAGNOSTICS

Gtags exits with a non-0 value if an error occurred, 0 otherwise.

Note that files created by gtags with a non-zero exit code should be considered corrupted.

SEE ALSO

global(1), htags(1), gtags.conf(5).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

BUG

`GTAGS` and `GRTAGS` are very large. In advance of using this command, check the space of your disk.

Assembly language support is far from complete. It extracts only `ENTRY()` and `ALTENTRY()` from source file. Probably valid only for FreeBSD and Linux kernel source.

C++ support is deprecated.

There is no concurrency control about tag files.

AUTHOR

Shigio YAMAGUCHI, Hideki IWAMOTO and others.

HISTORY

The gtags command appeared in FreeBSD 2.2.2.

5.3 htags - generate a hypertext from a set of source files

NAME

htags - generate a hypertext from a set of source files

SYNOPSIS

htags [-aDfFghInosTvw][-d dbpath][-m name][-t title][dir]

DESCRIPTION

Htags generates a hypertext from a set of source files of C, Yacc, Java, PHP and assembly.

In advance of using this command, you should execute gtags(1) in the root directory of a source project. Then you can execute htags in the same place. Htags makes a directory named `HTML`, and puts a hypertext in it. You can start browsing at `HTML/index.html`.

Since htags generates a static hypertext as long as neither '`-D`' nor '`-f`' option is not specified, you can move it anywhere and browse it by any browser without any HTTP server. If you use '`-D`' or '`-f`' option, you need to use HTTP server. Though any HTTP server is available, htags-server(1) is recommended.

This command has so many options. If you are new to htags, it is recommended to use '`--suggest`' option. With that option, htags chooses popular options on your behalf.

OPTIONS

The following options are available:

'`-a`', '`--alphabet`'
: Make an alphabetical index suitable for a large project.

'`--auto-completion`'[=limit]
: Enable auto-completion facility for the input form. If limit is specified, number of candidates is limited to the value. Please note this function requires javascript language in your browser.

'`-C`', '`--directory`' dir
: Change the directory before doing all the work including parameter analysis.

'`--caution`'
: Display a caution message on the top page.

'`--cflow`' cflowfile
: Add a call tree by cflow(1). cflowfile must be posix format. If you use GNU cflow, invoke the command at the project root directory with the

'`--format=posix`' option. This option is deprecated; please use '`--call-tree`' or '`--callee-tree`' instead.

'`--call-tree`' callfile
: Add a call tree by cflow(1); callfile must be posix format. If you use GNU cflow, invoke the command at the project root directory with the '`--format=posix`' option.

'`--callee-tree`' calleefile
: Add a callee tree by cflow(1); calleefile must be posix format. If you use GNU cflow, invoke the command at the project root directory with the '`--format=posix`' and '`--reverse`' options.

'`--colorize-warned-line`'
: Use color to highlight warned lines.

'`--cvsweb`' url
: Add a link to cvsweb; url is used as the base of URL. When directory `CVS` exists in the root directory of the project, the content of `CVS/Repository` is used as the relative path from the base.

'`--cvsweb-cvsroot`' cvsroot
: Specify cvsroot in cvsweb URL.

'`-D`', '`--dynamic`'
: Generate a tag list dynamically using CGI program. Though this option decreases both the size and generation time of hypertext, you need to start up HTTP server.

'`-d`', '`--dbpath`' dbpath
: Specify the directory in which `GTAGS` exists. The default is the current directory.

'`--disable-grep`'
: Disable grep in the search form ('`-f`', '`--form`').

'`--disable-idutils`'
: Disable idutils in the search form ('`-f`', '`--form`').

'`-F`', '`--frame`'
: Use frames for the top page.

'`-f`', '`--form`'
: Add a search form using CGI program. You need to start a HTTP server for it.

'`--fixed-guide`'
: Put a fixed guide at the bottom of the screen of the browser.

'`--full-path`'
: Use full path name in the file index. By default, use just the last component of a path.

'`-g`', '`--gtags`'
: Execute gtags(1) before starting job. The '`-v`', '`-w`' and dbpath options are passed to gtags.

‘`--gtagsconf`’ file
: Set environment variable *GTAGSCONF* to file.

‘`--gtagslabel`’ label
: Set environment variable *GTAGSLABEL* to label.

‘`-h`’, ‘`--func-header`’[=position]
: Insert a function header for each function. By default, htags doesn't generate them. You can choose the position using position, which allows one of ‘`before`’, ‘`right`’ or ‘`after`’ (default).

‘`--help`’ Print a usage message.

‘`--html-header`’ file
: Insert a header record derived from file into the HTML header of each file.

‘`-I`’, ‘`--icon`’
: Use icons instead of text for some links.

‘`--insert-footer`’ file
: Insert custom footer derived from file before `</body>` tag.

‘`--insert-header`’ file
: Insert custom header derived from file after `<body>` tag.

‘`--item-order`’ spec
: Specify the order of the items in the top page. The spec is a string consisting of item signs in order. Each sign means as follows: ‘`c`’: caution; ‘`s`’: search form; ‘`m`’: mains; ‘`d`’: definitions; ‘`f`’: files; ‘`t`’: call tree. The default is ‘`csmdf`’.

‘`-m`’, ‘`--main-func`’ name
: Specify startup function name; the default is ‘`main`’.

‘`--map-file`’
: Generate file `MAP`.

‘`-n`’, ‘`--line-number`’[=columns]
: Print line numbers. By default, don't print line numbers. The default value of columns is 4.

‘`--no-order-list`’
: Numbers are not given in list form.

‘`-o`’, ‘`--other`’
: Pick up not only source files but also other files for the file index.

‘`-s`’, ‘`--symbol`’
: Make anchors not only for definitions and references but also other symbols.

‘`--show-position`’
: Print the position string per function definition. The string can be interpreted by general editors in UNIX. The default is false.

‘`--statistics`’
: Print statistics information.

'`--suggest`'
: Htags chooses popular options on behalf of beginners. It is equivalent to '`-aghInosTxv --show-position --fixed-guide`' now.

'`--suggest2`'
: Htags chooses popular options on behalf of beginners. This option enables frame, AJAX and CGI facility in addition to the facilities by the '`--suggest`' option. It is equivalent to '`--suggest -DfF --auto-completion --tree-view=filetree`' now.

'`-T`', '`--table-flist`'[=rows]
: Use `<table>` tag to display the file index. You can optionally specify the number of rows; the default is 5.

'`-t`', '`--title`' title
: Title of the hypertext. The default is the last component of the path of the current directory.

'`--tabs`' cols
: Tab stops. The default is 8.

'`--table-list`'
: Use `<table>` tag to display the tag list.

'`--tree-view`'[=type]
: Use treeview for the file index. Please note this function requires javascript language in your browser. Possible values of type are as follows: treeview, filetree, treeview-red, treeview-black, treeview-gray, treeview-famfamfam. The default is treeview.

'`-v`', '`--verbose`'
: Verbose mode.

'`--version`'
: Show version number.

'`-w`', '`--warning`'
: Print warning messages.

dir
: The directory in which the result of this command is stored. The default is the current directory.

EXAMPLES

```
$ gtags -v
$ htags -sanohITvt 'Welcome to XXX source tour!'
$ firefox HTML/index.html
$ htags --suggest2
$ htags-server >& log &
$ firefox http://127.0.0.1:8000
```

FILES

`GTAGS`
: Tag file for definitions.

`GRTAGS` Tag file for references.

`GPATH` Tag file for source files.

`gtags.conf`, `$HOME/.globalrc`
Configuration data for GNU Global. See gtags.conf(5).

`HTML/FILEMAP`
Mapping file for converting file name into the path of the file.

`HTML/GTAGSROOT`
If this file exists, CGI program `global.cgi` sets environment variable *GTAGSROOT* to the contents of it. If you move directory `HTML` from the original place, please make this file.

`HTML/.htaccess`
Local configuration file for Apache. This file is generated when the '`-f`' or '`-D`' options are specified.

`HTML/index.html`
Start-up file.

`HTML/MAP` Mapping file for converting tag name into the path of tag list.

`HTML/style.css`
Style sheet file.

`/usr/local/share/gtags/style.css.tmpl`
The template of the style sheet file (`HTML/style.css`).

ENVIRONMENT

The following environment variables affect the execution of htags:

GTAGSCACHE
The size of the B-tree cache. The default is 50000000 (bytes).

GTAGSCONF
Configuration file.

GTAGSLABEL
Configuration label. The default is '`default`'.

HTAGS_OPTIONS
The value of this variable is inserted in the head of arguments.

TMPDIR The location used to stored temporary files. The default is `/tmp`.

GTAGSFORCECPP
If this variable is set, each file whose suffix is `.h` is treated as a C++ source file.

CONFIGURATION

The following configuration variables affect the execution of htags:

`datadir`(string)
Shared data directory. The default is `/usr/local/share` but you can change the value using configure script. Htags looks up template files in the `gtags` directory in this data directory.

`include_file_suffixes`(comma separated list)
: Suffixes of include files. The default is:
'`h,hh,hxx,hpp,H,inc.php`'.

`langmap`(comma separated list)
: Language mapping. Each comma-separated map consists of the language name, a colon, and a list of file extensions. Default mapping is:
'`c:.c.h,yacc:.y,asm:.s.S,java:.java,cpp:.c++.cc.hh.cpp.cxx.hxx.hpp.C.H,php:.php.ph`

DIAGNOSTICS

Htags exits with a non-0 value if an error occurred, 0 otherwise.

SEE ALSO

htags-server(1), global(1), gtags(1), gtags.conf(5).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

BUG

Generated hypertext is VERY LARGE. In advance, check the space of your disk.

PHP support is far from complete.

The '`-f`' and '`-D`' options generate CGI programs. If you open the hypertext to the public, please recognize security dangers.

Htags does not support plug-in parser.

AUTHOR

Shigio YAMAGUCHI, Hideki IWAMOTO and others.

HISTORY

The htags command appeared in FreeBSD 2.2.2.

5.4 htags-server - a private Web/CGI server for htags

NAME

htags-server - a private Web/CGI server for htags

SYNOPSIS

htags-server [-b ip-address][-u language][–retry[=n]][port]

DESCRIPTION

Htags-server is a private Web/CGI server for the hyper-text generated by htags(1).

In advance of using this command, you should execute htags(1). Then you can execute htags-server at the same place. You can start browsing at 'http://localhost:8000/' by default. To stop the server, just press on 'CTRL-C'.

Python 2/3 or Ruby equipped with WEBrick is required. By default, htags-server looks for python first, if not found then looks for ruby.

OPTIONS

The following options are available:

'`-b`', '`--bind`' ip-address
: Specifies the IP address on which htags-server listen. The default value is 127.0.0.1.

'`-u`', '`--use`' language
: Specifies language to use, which may be one of: '`python`' and '`ruby`'.

'`--retry`'[=n]
: If the port is already in use, retry n times with incrementing the port number. The default of n is 20.

port
: Specifies the port on which htags-server listen. If you want to use multiple sessions, you must use unique number for each htags-server. The default value is 8000.

EXAMPLES

```
$ gtags
$ htags --suggest2
$ htags-server
Python2 http/cgi server
Serving HTTP on 127.0.0.1 port 8000 ...
(another terminal)
$ firefox http://localhost:8000/
$ htags-server
Error: port 8000 already in use.
$ htags-server --retry
Warning: port 8000 already in use (skipped)
Please access at http://127.0.0.1:8001
Python2 http/cgi server
Serving HTTP on 127.0.0.1 port 8001 ...
```

DIAGNOSTICS

Htags-server exits with a non-0 value if an error occurred, 0 otherwise.

SEE ALSO

global(1), gtags(1), htags(1), python(1), ruby(1).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

AUTHOR

Shigio YAMAGUCHI.

HISTORY

The htags-server command appeared in Global-6.3 (2014).

5.5 gozilla - force firefox to display specified part of a source file

NAME

gozilla - force firefox to display specified part of a source file

SYNOPSIS

gozilla [-b browser][-p][+no] file
gozilla [-b browser][-p] -d name

DESCRIPTION

Gozilla forces firefox to display specified part of a source file. Gozilla can be used with other browsers like chrome.

In advance of using this command, you must execute gtags(1) and htags(1) at the root directory of the project to make tag files. Then you can execute this command anywhere in the project.

First form:
You can specify a source file and optional line number. This syntax is similar to vi(1) and emacs(1).

Second form:
You can specify a definition name directly. The definition name should exist in `GTAGS`. This option requires `HTML/MAP` generated by htags(1).

Some browsers require you to load it before executing gozilla.

OPTIONS

The following options are available:

‘`+no`’ Line number.

‘`-b`’ browser
Browser to use. By default, it is assumed firefox.

‘`-d`’ name Print definitions.

‘`--help`’ Show help.

‘`-p`’ Just print a generated URL instead of displaying it.

file File name or alias name.

‘`-q`’, ‘`--quiet`’
Quiet mode.

‘`-v`’, ‘`--verbose`’
Verbose mode.

‘`--version`’
Show version number.

FILES

`GTAGS` Tag file for definitions.

`HTML/` Hypertext of source code.

`HTML/MAP` Mapping file for converting tag name into the path of tag list.

`gtags.conf`, `$HOME/.globalrc`
Configuration data for GNU Global. See gtags.conf(5).

ENVIRONMENT

BROWSER
Browser to use. By default, it is assumed firefox. If you want to load the default browser in OSX, you may set this variable to osx-default.

GTAGSDBPATH
The directory in which the tag files exist. This value is ignored when *GTAGSROOT* is not defined.

GTAGSROOT
The root directory of the project.

EXAMPLES

```
$ gtags
$ htags
$ global -x main
main              82 ctags.c          main(argc, argv)
$ gozilla +82 ctags.c
$ gozilla -d main
$ gozilla -b lynx +82 ctags.c
```

DIAGNOSTICS

Gozilla exits with a non-0 value if an error occurred, 0 otherwise.

SEE ALSO

global(1), gtags(1), htags(1), epiphany(1), firefox(1), gtags.conf(5).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

AUTHORS

Shigio YAMAGUCHI.

HISTORY

The gozilla command appeared in FreeBSD 2.2.2 but was not installed by default.

5.6 gtags-cscope - interactively examine a C program

NAME

gtags-cscope - interactively examine a C program

SYNOPSIS

gtags-cscope [-bCdehLlVv][-F file][-012345678 pattern][-p n]

DESCRIPTION

gtags-cscope is an interactive, screen-oriented tool that allows the user to browse through source files for specified elements of code.

gtags-cscope builds the symbol cross-reference the first time it is used on the source files for the program being browsed. On a subsequent invocation, gtags-cscope rebuilds the cross-reference only if a source file has changed or the list of source files is different. When the cross-reference is rebuilt, it is updated incrementally, which makes rebuilding faster than the initial build.

gtags-cscope is a tool which just borrows user interface of cscope; it is Global itself for the substance.

OPTIONS

Some command line arguments can only occur as the only argument in the execution of gtags-cscope. They cause the program to just print out some output and exit immediately:

‘`-h`’ View the long usage help display.

‘`-V`’ Print the version number of gtags-cscope.

‘`--help`’ Same as ‘`-h`’

‘`--version`’
Same as ‘`-V`’

The following options can appear in any combination:

‘`-a`’ Print absolute path names.

‘`-b`’ Build the cross-reference only.

‘`-C`’ Ignore letter case when searching.

‘`-d`’ Do not update the cross-reference.

‘`-e`’ Suppress the `^e` command prompt between files.

‘`-F`’ file Read symbol reference lines from file. (A symbol reference file is created by `>` and `>>`, and can also be read using the `<` command, described under “Issuing Subsequent Requests”, below.)

‘`-i`’ Ignore SIGINT signal in line-oriented mode.

‘`-L`’ Do a single search with line-oriented output when used with the -num pattern option.

‘`-l`’ Line-oriented interface. This option implies the ‘`-d`’ option.

‘`-[0-9]`’ pattern
Go to input field num (counting from 0) and find pattern.

‘`-p`’ n Display the last n file path components instead of the default (1). Use ‘`0`’ to not display the file name at all.

‘`-v`’ Be more verbose in line-oriented mode.

Requesting the initial search

After the cross-reference is ready, gtags-cscope will display this menu:

```
Find this symbol:
Find this global definition:
Find functions called by this function (N/A):
Find references of this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

Press the <Up> or <Down> keys repeatedly to move to the desired input field, type the text to search for, and then press the <Return> key.

Issuing subsequent requests

If the search is successful, any of these single-character commands can be used:

0-9a-zA-Z Edit the file referenced by the given line number.

<Space> Display next set of matching lines.

<Tab> Alternate between the menu and the list of matching lines

<Up> Move to the previous menu item (if the cursor is in the menu) or move to the previous matching line (if the cursor is in the matching line list).

<Down> Move to the next menu item (if the cursor is in the menu) or move to the next matching line (if the cursor is in the matching line list).

`+` Display next set of matching lines.

- Display previous set of matching lines.

`^e` Edit displayed files in order.

`>` Write the displayed list of lines to a file.

`>>` Append the displayed list of lines to a file.

`<` Read lines from a file that is in symbol reference format (created by `>` or `>>`), just like the ‘`-F`’ option.

`^` Filter all lines through a shell command and display the resulting lines, replacing the lines that were already there.

|	Pipe all lines to a shell command and display them without changing them.
^g	Read lines from the result of the execution of global(1).

At any time these single-character commands can also be used:

<Return>	Move to next input field.
^n	Move to next input field.
^p	Move to previous input field.
^y	Search with the last text typed.
^b	Move to previous input field and search pattern.
^f	Move to next input field and search pattern.
^c	Toggle ignore/use letter case when searching. (When ignoring letter case, a search for '`FILE`' will match '`File`' and '`file`'.)
^r	Rebuild the cross-reference.
!	Start an interactive shell (type `^d` to return to gtags-cscope).
^l	Redraw the screen.
?	Give help information about gtags-cscope commands.
^d	Exit gtags-cscope.

NOTE: If the first character of the text to be searched for matches one of the above commands, escape it by typing a \ (backslash) first.

Substituting new text for old text

After the text to be changed has been typed, gtags-cscope will prompt for the new text, and then it will display the lines containing the old text. Select the lines to be changed with these single-character commands:

0-9a-zA-Z	Mark or unmark the line to be changed.
*	Mark or unmark all displayed lines to be changed.
<Space>	Display next set of lines.
+	Display next set of lines.
-	Display previous set of lines.
^a	Mark or unmark all lines to be changed.
^d	Change the marked lines and exit.
<Esc>	Exit without changing the marked lines.
!	Start an interactive shell (type `^d` to return to gtags-cscope).
^l	Redraw the screen.
?	Give help information about gtags-cscope commands.

Special keys

If your terminal has arrow keys that work in vi, you can use them to move around the input fields. The up-arrow key is useful to move to the previous input field instead of using the <Tab> key repeatedly. If you have <CLEAR>, <NEXT>, or <PREV> keys they will act as the `^l`, `+`, and `-` commands, respectively.

Line-Oriented interface

The '`-l`' option lets you use gtags-cscope where a screen-oriented interface would not be useful, for example, from another screen-oriented program.

gtags-cscope will prompt with '`>>`' when it is ready for an input line, which starts with the field number (counting from 0), immediately followed by the search pattern. For example, '`1main`' finds the definition of the '`main`' function.

If you just want a single search, instead of the '`-l`' option use the '`-L`' and '`-num pattern`' options, and you won't get the '`>>`' prompt.

For '`-l`', gtags-cscope outputs the number of reference lines:
cscope: 2 lines

For each reference found, gtags-cscope outputs a line consisting of the file name, function name, line number, and line text, separated by spaces. For example:
main.c main 161 main(argc, argv)

Note that the editor is not called to display a single reference, unlike the screen-oriented interface.

You can use the *c* command to toggle ignore/use letter case when searching. (When ignoring letter case, a search for '`FILE`' will match '`File`' and '`file`'.)

You can use the *r* command to rebuild the database.

gtags-cscope will quit when it detects end-of-file, or when the first character of an input line is `^d` or *q*.

ENVIRONMENT VARIABLES

The following environment variables are of cscope origin.

CSCOPE_EDITOR

Overrides the *EDITOR* and *VIEWER* variables. Use this if you wish to use a different editor with cscope than that specified by your *EDITOR*/*VIEWER* variables.

CSCOPE_LINEFLAG

Format of the line number flag for your editor. By default, cscope invokes your editor via the equivalent of '`editor +N file`', where N is the line number that the editor should jump to. This format is used by both emacs and vi. If your editor needs something different, specify it in this variable, with '`%s`' as a placeholder for the line number. Eg: if your editor needs to be invoked as '`editor -#103 file`' to go to line 103, set this variable to '`-#%s`'.

CSCOPE_LINEFLAG_AFTER_FILE

Set this variable to '`yes`' if your editor needs to be invoked with the line number option after the filename to be edited. To continue the example

from *CSCOPE_LINEFLAG*, above: if your editor needs to see '`editor file -#number`', set this environment variable. Users of most standard editors (vi, emacs) do not need to set this variable.

EDITOR Preferred editor, which defaults to vi.

HOME Home directory, which is automatically set at login.

SHELL Preferred shell, which defaults to sh.

TERM Terminal type, which must be a screen terminal.

TERMINFO
Terminal information directory full path name. If your terminal is not in the standard terminfo directory, see curses and terminfo for how to make your own terminal description.

TMPDIR Temporary file directory, which defaults to `/tmp`.

VIEWER Preferred file display program (such as less), which overrides *EDITOR* (see above).

The following environment variables are of Global origin.

GTAGSCONF
Configuration file.

GTAGSGLOBAL
If this variable is set, `$GTAGSGLOBAL` is used as the name of global(1). The default is global.

GTAGSGTAGS
If this variable is set, `$GTAGSGTAGS` is used as the name of gtags(1). The default is gtags.

GTAGSDBPATH
The directory in which the tag files exist. This value is ignored when *GTAGSROOT* is not defined.

GTAGSLABEL
Configuration label. The default is '`default`'.

GTAGSLIBPATH
If this variable is set, it is used as the path to search for library functions. If the specified tags is not found in the project, global also searches in these paths. Since only `GTAGS` is targeted in the retrieval, this variable is ignored when '`-r`' or '`-s`' is specified.

GTAGSROOT
The root directory of the project.

MAKEOBJDIR
If this variable is set, `$MAKEOBJDIR` is used as the name of BSD-style objdir. The default is `obj`.

MAKEOBJDIRPREFIX
If this variable is set, `$MAKEOBJDIRPREFIX` is used as the prefix of BSD-style objdir. The default is `/usr/obj`.

FILES

`GTAGS` Tag file for definitions.

`GRTAGS` Tag file for references.

`GPATH` Tag file for source files.

`GTAGSROOT`
If environment variable *GTAGSROOT* is not set and file `GTAGSROOT` exists in the same directory as `GTAGS` then global sets *GTAGSROOT* to the contents of the file.

`gtags.conf`, `$HOME/.globalrc`
Configuration data for GNU Global. See gtags.conf(5).

SEE ALSO

gtags(1), global(1), htags(1).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

BUG

The function field of the display is almost <unknown> since Global doesn't recognize it.

'`Find functions called by this function`' is not implemented.

AUTHOR

Joe Steffen (original author) and others.

HISTORY

Cscope was originally developed at Bell Labs in the early 1980s, and was released as free software under the BSD license in April 2000. Gtags-cscope is a derivative of cscope to use Global as the back-end. Its line-oriented interface was originally written in 2006, and was re-implemented in 2011 using cscope itself.

5.7 globash - a special shell for Global using GNU bash

NAME

globash - a special shell for Global using GNU bash

SYNOPSIS

globash

DESCRIPTION

Globash is a special shell for Global using GNU bash. You can use a lot of functions to ease reading source code, like tag stack, tag mark and cookie. At first, you should make

tag files using gtags and invoke this command in the project. Please refer to the help (type *ghelp<RET>*) for detailed usage.

FILES

`GTAGS` Tag file for definitions.

`GRTAGS` Tag file for references.

`GPATH` Tag file for source files.

`~/.globashrc`
Start-up file.

`~/.globash/`
Session directory.

ENVIRONMENT

The following environment variables affect the execution of globash:

EDITOR The editor used by the show command.

SEE ALSO

gtags(1), htags(1), less(1).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

AUTHOR

Shigio YAMAGUCHI.

HISTORY

The globash command appeared in Global-4.1 (2001).

5.8 gtags.conf - configuration data for GNU Global

NAME

gtags.conf - configuration data for GNU Global

SYNOPSIS

```
gtags.conf
~/.globalrc
```

DESCRIPTION

`gtags.conf` has configuration data for global(1), gozilla(1), gtags(1) and htags(1). They look for the following files in this order, and read from the first one that exists and is readable.

(1) –gtagsconf file
(2) $GTAGSCONF
(3) [project root]/gtags.conf
(4) [project root]/[objdir]/gtags.conf
(5) $HOME/.globalrc
(6) /etc/gtags.conf
(7) [sysconfdir]/gtags.conf

[sysconfdir] is `/usr/local/etc` by default; you can change it by the –sysconfdir option of the configure script. [objdir] is `obj` by default; you can change it by *GTAGSOBJDIR* or *MAKEOBJDIR*.

Its syntax is similar to termcap(5) except for some extensions described later. A file includes one or more records. You should select a record using a label. Default label is '`default`'. If *GTAGSLABEL* is set, its value is used. Please be aware the meaning of '`default`' is different from that of termcap(5).

Lines starting with '`#`' are ignored. A record consists of one or more fields separated by '`:`' and ends with a newline. The first field must have a label name. Each field of the rest has a variable definition. There are three types in those variables.

Boolean: This type has no value; existence of variable means true else false.
Number: This type has a number value after '`#`'.
String: This type has a string value after '`=`'.

Most variables are relevant to string type.

If data includes '`:`' or newline, you need to quote it by '\'. An empty field is permitted, but it doesn't make sense.

When there is more than one definition, the previous one is adopted. Exceptionally, all values of skip, langmap and gtags_parser are adopted and connected respectively.

Variable substitution

Variable substitution similar to sh(1) is available. You can refer to any string variable using '`$`' prefix. For example, the following b has a value '`XYZ`'.

```
:a=Y:b=X${a}Z:
```

Including another record

Tc is a special variable; it reads in definitions in another record. It is similar to the tc of termcap(5). Its syntax is as follows:
tc=label[@file]
If only a label is given, tc is read from a record specified by the label in the same file. If @file is also given, tc is read from a record specified by the label in the specified file. File should be an absolute path, and should not include variables. Different from termcap(5), you can use tc anytime anywhere.

CONFIGURATION

About variables for each command, please see each manual.

You can use the following environment variables as a string variable. If these variables are set then environment variables of the same name are set before command execution.
GREP_COLOR
GREP_COLORS
GTAGSBLANKENCODE
GTAGSCACHE
GTAGSFORCECPP
GTAGSGLOBAL
GTAGSGTAGS
GTAGSLIBPATH
GTAGSLOGGING
GTAGSTHROUGH
GTAGS_OPTIONS
HTAGS_OPTIONS
MAKEOBJDIR
MAKEOBJDIRPREFIX
TMPDIR

The following variables are also available as a string variable. By default, each of them has a directory path given by the configure script.
bindir
datadir
libdir
localstatedir
sysconfdir

ENVIRONMENT

The following environment variables affect the execution of the commands.

GTAGSCONF
: If this variable is set, the value is used as a configuration file.

GTAGSLABEL
: If this variable is set, the value is used as a label of the configuration file. The default is default.

SEE ALSO

global(1), gozilla(1), gtags(1), htags(1).

GNU Global source code tag system
(http://www.gnu.org/software/global/).

HISTORY

The gtags.conf file appeared in Global-3.0 (1998).

Appendix A Copying This Manual

A.1 GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.1.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Option Index

—

`--result` 5
`-a` 5
`-c` 7
`-f` 5, 6, 16, 18
`-g` 5, 15, 16, 18
`-l` 5
`-o` 5
`-O` 5, 6
`-P` 5, 16, 18
`-r` 4, 15, 17
`-s` 5, 15, 18
`-t` 15
`-u` 24
`-x` 5

F

FDL, GNU Free Documentation License 56

Table of Contents

1 Overview of this tool 1
- 1.1 What is GNU Global? 1
- 1.2 Concept of project 1
- 1.3 Features 1

2 Command line 3
- 2.1 Preparation 3
- 2.2 Basic usage 3
- 2.3 Applied usage 6

3 Various applications 9
- 3.1 Global facility for Bash 9
  - 3.1.1 Features 9
  - 3.1.2 Preparation 9
  - 3.1.3 Usage 9
- 3.2 Less using Global 12
  - 3.2.1 Features 12
  - 3.2.2 Preparation 12
  - 3.2.3 Usage 12
- 3.3 Nvi-1.81.5 using Global 13
  - 3.3.1 Features 13
  - 3.3.2 Preparation 14
  - 3.3.3 Usage 14
- 3.4 Elvis using Global 15
  - 3.4.1 Features 15
  - 3.4.2 Preparation 15
  - 3.4.3 Usage 15
- 3.5 Vim using Global 17
  - 3.5.1 Features 17
  - 3.5.2 Preparation 17
  - 3.5.3 Usage 17
- 3.6 Extended Emacs using Global 19
  - 3.6.1 Features 19
  - 3.6.2 Preparation 19
  - 3.6.3 Usage 20
- 3.7 Gtags-cscope 21
- 3.8 Hypertext generator 21
  - 3.8.1 Features 21
  - 3.8.2 Preparation 21
  - 3.8.3 Usage 22
- 3.9 Doxygen using Global 22

4 Other topics 24

4.1 How to configure Global 24
4.2 Plug-in parser 24
4.3 Incremental updating 24

5 Command References 26

5.1 global - print locations of given symbols 26
NAME 26
SYNOPSIS 26
DESCRIPTION 26
COMMANDS 26
OPTIONS 27
EXAMPLES 30
FILES 31
ENVIRONMENT 31
CONFIGURATION 32
DIAGNOSTICS 32
SEE ALSO 32
AUTHOR 32
HISTORY 32
5.2 gtags - create tag files for global 32
NAME 32
SYNOPSIS 32
DESCRIPTION 33
OPTIONS 33
EXAMPLES 35
FILES 35
ENVIRONMENT 35
CONFIGURATION 36
EXAMPLES 37
DIAGNOSTICS 37
SEE ALSO 37
BUG 37
AUTHOR 38
HISTORY 38
5.3 htags - generate a hypertext from a set of source files 38
NAME 38
SYNOPSIS 38
DESCRIPTION 38
OPTIONS 38
EXAMPLES 41
FILES 41
ENVIRONMENT 42
CONFIGURATION 42
DIAGNOSTICS 43
SEE ALSO 43

BUG . . . 43
AUTHOR . . . 43
HISTORY . . . 43
5.4 htags-server - a private Web/CGI server for htags . . . 43
NAME . . . 43
SYNOPSIS . . . 43
DESCRIPTION . . . 43
OPTIONS . . . 44
EXAMPLES . . . 44
DIAGNOSTICS . . . 44
SEE ALSO . . . 44
AUTHOR . . . 44
HISTORY . . . 45
5.5 gozilla - force firefox to display specified part of a source file . . . 45
NAME . . . 45
SYNOPSIS . . . 45
DESCRIPTION . . . 45
OPTIONS . . . 45
FILES . . . 46
ENVIRONMENT . . . 46
EXAMPLES . . . 46
DIAGNOSTICS . . . 46
SEE ALSO . . . 46
AUTHORS . . . 46
HISTORY . . . 46
5.6 gtags-cscope - interactively examine a C program . . . 47
NAME . . . 47
SYNOPSIS . . . 47
DESCRIPTION . . . 47
OPTIONS . . . 47
Requesting the initial search . . . 48
Issuing subsequent requests . . . 48
Line-Oriented interface . . . 50
ENVIRONMENT VARIABLES . . . 50
FILES . . . 52
SEE ALSO . . . 52
BUG . . . 52
AUTHOR . . . 52
HISTORY . . . 52
5.7 globash - a special shell for Global using GNU bash . . . 52
NAME . . . 52
SYNOPSIS . . . 52
DESCRIPTION . . . 52
FILES . . . 53
ENVIRONMENT . . . 53
SEE ALSO . . . 53
AUTHOR . . . 53

HISTORY 53
5.8 gtags.conf - configuration data for GNU Global 53
NAME 53
SYNOPSIS 53
DESCRIPTION 53
Variable substitution 54
Including another record 54
CONFIGURATION 54
ENVIRONMENT 55
SEE ALSO 55
HISTORY 55

Appendix A Copying This Manual 56
A.1 GNU Free Documentation License 56
A.1.1 ADDENDUM: How to use this License for your documents .. 62

Option Index 63