GNU GLOBAL Source Code Tag System

Edition 5.9.6, for GNU GLOBAL version 5.9.6
7 June 2011

by Tama Communications Corporation

This manual is for GNU GLOBAL (version 5.9.6, 7 June 2011), a source code tag system that works the same way across diverse environments.

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

1 Overview of this tool

1.1 What is GNU GLOBAL?

GNU GLOBAL is a source code tag 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 locate specified objects such as functions, macros, structs, classes in your source files and move there easily. It is useful for hacking large projects which contain 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 high performance tag database. You need not specify where the database is. Instead, 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 following features:

- support C, C++, Yacc, Java, PHP4 and assembly.
- work 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
- find the locations of specified object quickly.
- locate not only object definitions but also object references.
- allows duplicate objects.
- locate path names which include specified pattern.
- hierarchical search is available.
- search not only in a source project but also in library projects.
- generate completion list for completing input method.
- support various output format.
- allows customizing of the set of candidate files to be tagged.
- understand POSIX 1003.2 regular expression.
- support idutils as an external search engine.
- tag files are independent of machine architecture.

- support incremental updating of tag files.
- plug-in parser is available to treat new language.
- support customizing with '`gtags.conf`'.
- generate a hypertext of source code.

2 Command line GLOBAL

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

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 33) at the root of 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 source files and makes four tag files at the current directory. After this, the whole files under this directory is 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 the tag files. For example, Linux-2.6.32 source code requires the following disk space.

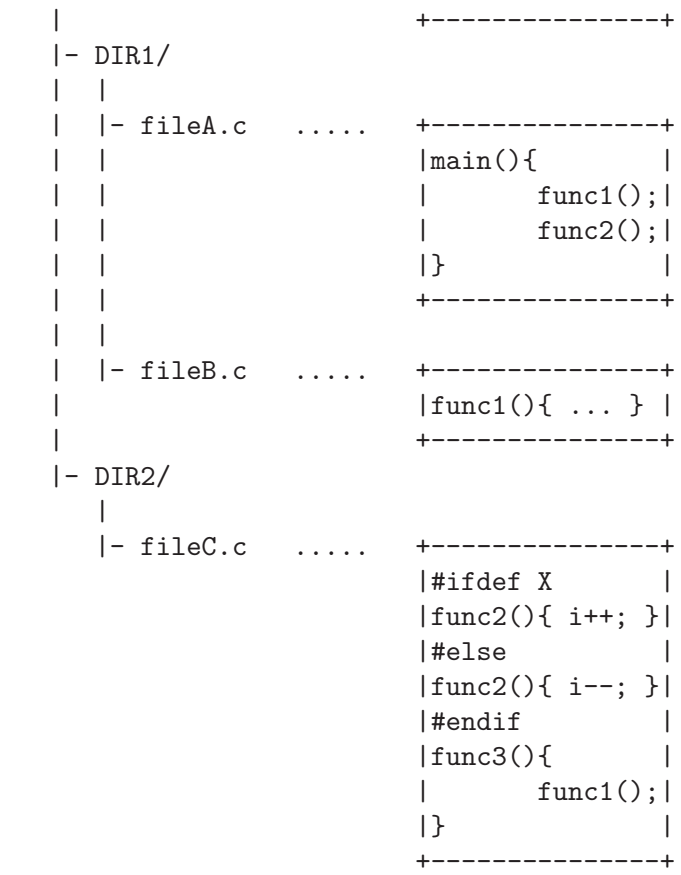
```
source code(Linux-2.6.32)       390MB

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|
```



- Once you make tag files at the root directory of the source tree, you can execute global(1) from anywhere in the tree. By default, you get the relative path of the located files. You need not specify where the tag file is. Instead, global(1) locates it by itself.

```
$ 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
```

  Global command is possible to use only when you are in a project. If you are out of any project, it brings an error message like follows:

```
$ cd /home/user
$ global func1
```

```
global: GTAGS not found.
```

- The '`-r`' option locates object 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 the details. It is similar to the '`-x`' option in 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 the absolute path name.

```
$ 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 the lines which have 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 the source files.

Additionally, you can use various options:

-O search only in the text files.

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

-l search only under the current directory.

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

You can even change the output format of global(1) to the grep style using the '`--result=grep`' option. Of course, these options can be used even by other commands.

- The '`-P`' command locates path names which include 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 print a list of objects in specified file.

```
$ 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 the 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 the `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:

```
$ 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 value ‘`/usr/obj`’ can be changed by environment variable `MAKEOBJDIRPREFIX`. The ‘`-O, --objdir`’ option do it automatically instead of you.

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

  You should execute gtags at each directory of the path. If ‘`GTAGS`’ is not found there, global ignores it.

```
$ 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 object 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 had better try globash(see Section 3.1 [GloBash], page 10). It support you in a suitable way without any preparation.
- You can edit all files which have specified objects 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.
- Yoo 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. And 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)
```

Inputing the ENTER key, you will see a prompt like this:

```
[/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* and *RET* or just *RET* in the above prompt then the tag stack and marker (described later) will be removed.

If you need help then please type *ghelp*.

3.1.3 Usage

- Almost global(1)(see Section 5.1 [global], page 29)'s command characters are available as a command.

```
[/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 the *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) but 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 mozilla show boot/alpha/boot1/boot1.c at line 227.)
```

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

`list (l)` print tag list.

`first` go to the first tag.

`last` go to the last tag.

`next (n)` go to next tag.

`prev (p)` go to previous tag.

`show n (1,2,3,..,999)`
go to nth tag

- You can use vi-like tag stack. You can return the previous tag list by the *pop* or *CTL-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 tag stack by *tags* command.

- You can memory 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 session.

- You can memory 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 the tag system of Less(1) viewer instead of ctags.

3.2.1 Features

- You can use most of GLOBAL's facilities from Less viewer.
- Less viewer support 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, 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 use 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`', '`GPATH`' as tag files.
- If a number of objects are located, Less 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 session, you can use `:t` command to locate new symbol. But in this case, you cannot change tag file from one specified by '`-T`' option.
- With the '`-T-`' option, Less read standard input as a tag file. You can connect global and Less 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 objects definitions in *.c.
$ global -f *.c | less -T-

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

- If your editor doesn't support GLOBAL directly then you can use Less 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 the tag 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 write to '`.nexrc`' like this: It assumed that '`gtags.pl`' is put on '`$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 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 objects 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 do the followings:

  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 *`CTL-]`* 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`' option are also available. It works like command line.

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

- When you want to locate objects the name of which start with "set" or "get", use:

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

- Other tag commands are also available:

CTL-T returns to the most recent tag location.

:tagpop returns to the most recent tag location.

:tagtop returns to the top of the tag stack.

:display tags
display the tags stack.

3.4 Elvis using GLOBAL

Elvis 2.1 or the 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 same as original Elvis, but Elvis execute `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, you had better 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 objects in specified files, use '`-f`' command.

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

- Other tag commands are also available:

  CTL-] go to the definition of current token.

  CTL-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 creates a new window and moves its cursor to the tag's definition point.

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

- You can use POSIX regular expressions.

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

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

- You can browse an object list of many files.

```
:browse -f *.c              <- locate objects 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 by 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 on the right button.

3.5 Vim using GLOBAL

In Vim 6.2 or later, you can use '`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 character '%', '#' and input completion are available.

To our regret, tag stack facility is not available. If you want to use the facility, please try gtags-cscope See Section 3.7 [Gtags-cscope (fake cscope)], page 25.

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 execute global(1), parse the output, list located objects in quickfix window and load the first entry. The quickfix windows 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 '`-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 objects in specified files, use -f command.

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

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

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

- You can use POSIX regular expressions.

```
:Gtags ^put_                  <- locate objects start with 'put_'

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

- Input completion is available.

  In the command line, press *CTL-D* after some typings 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 the -c, -n, -p, -q, -u, -v and all long name options. They are sent to global(1) as is. For example,

```
:Gtags -gi paTtern      <- match to both 'PATTERN' and 'pattern'.

:Gtags -POi make        <- match to Makefile but doesn't match to makeit.c.
```

  About the other options, please see See Section 5.1 [global], page 29.

- 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*. Otherwise, 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 ':GtagsCursor' to '^]' seems suitable, it will bring an inconvenience in the help screen.

- If you have the hypertext generated by htags(1) then you can display the same part of the source code on the mozilla browser. Let's load mozilla 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 the tag 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)
))
```

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. You need to execute `gtags-make-complete-list` command before it.

  `Find tag: fu`*`TAB`*

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

- If a number of objects 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 the 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 method 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 start with 'put_'
```

- Mouse command is available.

  If you use X version Emacs, try the following:

  Move the mouse cursor to an object 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 (fake cscope)

You can use gtags-cscope(1) instead of cscope(1). For example, you can deceive Vim editor using the following commands:

```
: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 the point 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, 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 G byte 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 33) and htags(1)(see Section 5.3 [htags], page 37) 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 option 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 mozilla, you can also utilize the hypertext from your command line like this:

```
$ mozilla # load mozilla
$ 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 mozilla's screen.)
```

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 the 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 the 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 config 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 use it, else if '`/etc/gtags.conf`' exists then GLOBAL use it. Otherwise default value is used. The format of '`gtags.conf`' is resemble to termcap(5). By default, 'default' target is used. About the capabilities, please see each command manual. See Chapter 5 [Reference], page 29.

4.2 Plug-in parser

You can write new parser for gtags(1).

Command layer plug-in parser was abolished. Please write function layer plug-in parser instead. See '`plugin-example/`' to know function layer plug-in parser.

4.3 Incremental updating

Modifying some source files, you need not remake the whole 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 the specified object.

NAME

global - print locations of the specified object.

SYNOPSIS

global [-adGilnqrstTvx][-e] pattern
global -c[iIqsv] prefix
global -f[anqrstvx][-L file-list] files
global -g[aGilnoOqtvVx][-L file-list][-e] pattern [files]
global -I[ailnqtvx][-e] pattern
global -P[aGilnoOqtvVx][-e] pattern
global -p[qrv]
global -u[qv]

DESCRIPTION

Global finds locations of the specified object in C, C++, Yacc, Java, PHP and Assembly source files, and print the path name, line number and line image of the locations. Global can locate not only object definitions but also object 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 a project to make tag files. Then you can execute this 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 the pattern. Global understands two different versions of regular expression syntax: basic and extended. The default is extended.

COMMANDS

The following commands are available:

<no command> pattern
: Print objects which match to the pattern. By default, print object definitions.

‘-c’, ‘--completion’ [prefix]
: Print object names which start with the specified prefix. If prefix is not specified, print all object names.

‘-f’, ‘--file’ files
: Print all objects in the files. This option implies the ‘-x’ option.

‘-g’, ‘--grep’ pattern [files]
: Print all lines which match to the pattern. If files is specified, this command searches in the files.

‘--help’ Show help.

‘`-I`’, ‘`--idutils`’ pattern
: Print all lines which match to the pattern. This function use 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 the pattern. If no pattern specified, print all path names in the project.

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

‘`-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 name. By default, print relative path name.

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

‘`--from-here`’ context
: Decide tag type by the context. The context must be ’lineno:path’. If this option is specified then the ‘`-s`’ and ‘`-r`’ are ignored. Regular expression is not allowed for the pattern. This option assumes use in conversational environments such as editors and IDEs.

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

‘`--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’.

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

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

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

‘`-L`’, ‘`--file-list`’ file-list
: Obtain files from file-list in addition to the arguments.

‘`--literal`’
: Execute literal search instead of regular expression search. This option is only valid when the ‘`-g`’ command is specified.

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

‘`-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.

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

‘`-q`’, ‘`--quiet`’
: Quiet mode.

‘`-r`’, ‘`--reference`’, ‘`--rootdir`’
: Print locations of object references. With the ‘`-p`’ option, print the root directory of the project.

‘`--result`’ format
: Print out using the specified format. format may be path, ctags, ctags-x, grep or cscope. The default is path. The ‘`--result=ctags`’ and ‘`--result=ctags-x`’ are equivalent to the ‘`-t`’ and ‘`-x`’ respectively. The ‘`--result`’ option is given more priority than the -t and -x options.

‘`-s`’, ‘`--symbol`’
: Print locations of the specified symbol other than definitions.

‘`-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`’, ‘`--verbose`’
: Verbose mode.

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

‘`-x`’, ‘`--cxref`’
: Use standard ctags cxref (with the -x) format.

EXAMPLES

```
$ ls -F
Makefile      src/    lib/
$ gtags
$ global main
src/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 '^[sg]et'
get_num          205 src/op.c            while (get_num() > 0) {
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 strlen
../../../usr/src/sys/libkern/strlen.c
$ (cd /usr/src/lib; gtags)
$ GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
$ global strlen
../../../usr/src/lib/libc/string/strlen.c
```

FILES

'`GTAGS`' Tag file for object definitions.

'`GRTAGS`' Tag file for object references.

'`GPATH`' Tag file for path of source files.

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

'`$HOME/.globalrc`', '`/etc/gtags.conf`', '`[sysconfdir]/gtags.conf`'
: Configuration files.

ENVIRONMENT

The following environment variables affect the execution of global:

GTAGSROOT
: The root directory of the project.

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

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

GTAGSCONF
: Configuration file. The default is '`$HOME/.globalrc`'.

GTAGSLABEL
: Configuration label. The default is default.

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

CONFIGURATION

The following configuration variables affect the execution of global:

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

DIAGNOSTICS

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

SEE ALSO

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

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][-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, pickup symbols and write the cross-reference data into the tag files (‘`GTAGS`’, ‘`GRTAGS`’ and ‘`GPATH`’).

If ‘`gtags.files`’ exists or the ‘`-f`’ option is specified, target files are limited by it. Lines starting with `". "` are comments.

C, C++, yacc, java, PHP and Assembly source files are supported. Files whose names end in ‘`.c`’ or ‘`.h`’ are assumed to be C source files. Files whose names end in ‘`.c++`’ ‘`.cc`’ ‘`.cpp`’ ‘`.cxx`’ ‘`.hxx`’ ‘`.hpp`’ ‘`.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 ‘`.java`’ are assumed to be Java source files. Files whose names end in ‘`.php`’ ‘`.php3`’ ‘`.phtml`’ are assumed to be PHP source files. Files whose names end in ‘`.s`’ or ‘`.S`’ are assumed to be Assembly source files. Other files are assumed to be text files.

OPTIONS

The following options are available:

‘`-c`’, ‘`--compact`’
: Make GTAGS in compact format. This option does not influence GRTAGS, because they are always made in compact format.

‘`--config`’[=name]
: Print the value of config variable name. If name is not specified then print all names and values.

‘`-d`’, ‘`--dump`’ tag-file
: Dump a tag file. The output format is ’key<tab>data’. This is for debugging.

‘`-f`’, ‘`--file`’ file
: Browse through all source files whose names are listed in file. The argument file can be set to ‘`-`’ to accept a list of files from the standard input. File names must be separated by newline.

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

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

‘`-I`’, ‘`--idutils`’
: Also make the ID database file for idutils(1).

‘`-i`’, ‘`--incremental`’
: Update tag files incrementally. You had better use global(1) with the -u option.

‘-O’, ‘--objdir’
: Use BSD-style objdir as the location of tag files. If ‘$MAKEOBJDIRPREFIX’ directory exists, gtags creates ‘$MAKEOBJDIRPREFIX/<current directory>’ directory and makes tag files in it. If dbpath is specified, this option is ignored.

‘--single-update’ file
: Update tag files for single file. It is considered that file was updated, and other files were not updated. The file must be relative path name from the current directory. This option implies the -i option. If the file is new then ‘--single-update’ is ignored, and the processing is automatically switched to normal incremental updating.

‘--statistics’
: Print statistics information. This option is valid only for normal creation of tag files.

‘-q’, ‘--quiet’
: Quiet mode.

‘-v’, ‘--verbose’
: Verbose mode.

‘-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 object definitions.

‘GRTAGS’
: Tag file for object references.

‘GPATH’
: Tag file for path of source files.

‘$HOME/.globalrc’, ‘/etc/gtags.conf’, ‘[sysconfdir]/gtags.conf’
: Configuration files.

‘gtags.files’
: The list of candidates of target files.

ENVIRONMENT

The following environment variables affect the execution of gtags:

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

GTAGSCONF
: Configuration file. The default is '`$HOME/.globalrc`'.

GTAGSLABEL
: Configuration label. The default is default.

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

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

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

CONFIGURATION

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

`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 the language name, a colon, and a list of file extensions. As a special exception, gtags collects values from multiple `langmap` variables. Default mapping is 'c:.c.h,yacc:.y,asm:.s.S,java:.java,cpp:.c++.cc.cpp.cxx.hxx.hpp.C.H,php:.php.php3.phtml'.

`gtags_parser`(comma separated list)
: Specify the mapping of funtion layer plugin parser. Each part delimited by the comma consists of the 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.

`skip`(comma separated list)
: Gtags skips files which are listed 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. If the value starts with '/', it is assumed a relative path from the root of source directory.

DIAGNOSTICS

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

SEE ALSO

global(1), htags(1).

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

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 [-acDfFghInosTvwx][-d dbpath][-m name][-t title][dir]

DESCRIPTION

Htags generates a hypertext from a set of source files of C, 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 the '`-D`' or '`-f`' option is not specified, you can move it anywhere and browse it by any browser without any HTTP server.

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

OPTIONS

The following options are available:

'`-a`', '`--alphabet`'
: Make an alphabetical object index which is suitable for large projects.

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

'`--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 root directory of the project with the ‘`--format=posix`’ option. See EXAMPLES.

‘`-c`’, ‘`--compact`’
: Compress html files by gzip(1). You need to configure HTTP server so that gzip(1) is invoked for each compressed file. See ‘`HTML/.htaccess`’ that is generated by htags.

‘`--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 source 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 object lists dynamically using CGI program. Though this option decrease both the size and generation time of hypertext, you need to start up HTTP server.

‘`-d`’, ‘`--dbpath`’ dbpath
: Specify a directory in which ‘`GTAGS`’ exist. 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 up HTTP server for it.

‘`--fixed-guide`’
: Put a fixed guide at the bottom of the source code.

‘`--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 the *GTAGSCONF* environment variable to file.

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

‘`-h`’, ‘`--func-header`’[=position]
: Insert function header for each function. By default, htags doesn't generate it. You can specify the position using position argument, which allows one of before, right and after. The default position is after.

‘`--html`’ Generate HTML hypertext instead of XHTML.

‘`--html-header`’ file
: Insert header records derived from file into the HTML header.

‘`-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: definition; f: files; t: call tree. The default is csmdf.

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

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

‘`--map-file`’
: Generate files ‘`MAP`’.

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

‘`--overwrite-key`’
: Allow the same key as the parameter of the ‘`--system-cgi`’ option.

‘`--system-cgi`’ key
: Use the system CGI script. The key must be a unique key in your site. At the first time, you should (1) copy the CGI script written by this command into the system CGI directory, and (2) execute bless.sh script at the HTML directory as a root user.

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

‘`--show-position`’
: Show position per function definition. 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.

‘`-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.

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

‘`--tree-view`’[=type]
: Use treeview for the file index. Please note that this function requires javascript language in your browser.

‘`-v`’, ‘`--verbose`’
: Verbose mode.

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

‘`-x`’, ‘`--xhtml`’[=version]
: Generate XHTML hypertext. This is the default. If the ‘`--frame`’ option is specified then generate XHTML-1.0 Frameset for index.html and generate XHTML-1.0 Transitional for other files, else if version is `1.1` or config variable `xhtml_version` is set to `1.1` then generate XHTML-1.1 else XHTML 1.0 Transitional.

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 --suggest

$ cflow --tree --format=posix *.[ch] >cflow.out
$ htags --cflow=cflow.out
```

FILES

‘`GTAGS`’
: Tag file for object definitions.

'`GRTAGS`' Tag file for object references.

'`GPATH`' Tag file for files.

'`$HOME/.globalrc`', '`/etc/gtags.conf`', '`[sysconfdir]/gtags.conf`'
Configuration files.

'`HTML/index.html`'
Startup file.

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

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

'`HTML/style.css`'
Style sheet file. This file is generated when the '`--xhtml`' option is specified.

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

'`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.

ENVIRONMENT

The following environment variables affect the execution of htags:

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

GTAGSCONF
Configuration file. The default is '`$HOME/.globalrc`'.

GTAGSLABEL
Configuration label. The default is default.

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

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: If the '`--xhtml`' option is specified then all definitions of HTML tag are ignored. Instead, you can customize the appearance using style sheet file ('`style.css`').

`body_begin`(string)
Begin tag for body. The default is '<body text=#191970 bgcolor=#f5f5dc vlink=gray>'.

`body_end`(string)
End tag for body. The default is '</body>'.

`brace_begin`(string)
: Begin tag for brace. The default is ''.

`brace_end`(string)
: End tag for brace. The default is ''.

`colorize_warned_line`(boolean)
: Colorize warned line using `warned_line_begin` and `warned_line_end`. The default is false.

`comment_begin`(string)
: Begin tag for comments. The default is ''.

`comment_end`(string)
: End tag for comments. The default is ''.

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

`gzipped_suffix`(string)
: Suffix for compressed html file. The default is 'ghtml'.

`hr`(string) Horizontal rules. The default is '<hr>'.

`htags_options`(string)
: Default options for htags. This value is inserted into the head of arguments.

`include_file_suffixes`(comma separated list)
: Suffixes of include files. The default is 'h,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.cpp.cxx.hxx.hpp.C.H,php:.php.php3.phtml'.

`ncol`(number)
: Columns of line number. The default is 4.

`normal_suffix`(string)
: Suffix for normal html file. The default is 'html'.

`position_begin`(string)
: Begin tag for position mark. The default is ''.

`position_end`(string)
: End tag for position mark. The default is ''.

`reserved_begin`(string)
: Begin tag for reserved word. The default is ''.

`reserved_end`(string)
: End tag for reserved word. The default is ''.

`script_alias`(string)
: Script alias for system cgi script ('`--system-cgi`').

`sharp_begin`(string)
Begin tag for 'define'. The default is ''.

`sharp_end`(string)
End tag for 'define'. The default is ''.

`table_begin`(string)
Begin tag for table. The default is '<table>'.

`table_end`(string)
End tag for table. The default is '</table>'.

`tabs`(number)
Tab stop. The default is 8.

`title_begin`(string)
Begin tag for Title. The default is '<h1>'.

`title_end`(string)
End tag for Title. The default is '</h1>'.

`warned_line_begin`(string)
Begin tag for line which htags warned. The default is ''.

`warned_line_end`(string)
End tag for line which htags warned. The default is ''.

`xhtml_version`(1.0|1.1)
XHTML version. 1.0 and 1.1 are acceptable. The default is 1.0.

DIAGNOSTICS

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

SEE ALSO

global(1), gtags(1).

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, -D or -c option generates CGI programs. If you open the result to the public, please recognize the security dangers.

AUTHOR

Shigio YAMAGUCHI, Hideki IWAMOTO and others.

HISTORY

The htags command appeared in FreeBSD 2.2.2.

5.4 gozilla - force mozilla to display specified part of a source file.

NAME

gozilla - force mozilla to display specified part of a source file.

SYNOPSIS

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

DESCRIPTION

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

In advance of using this command, you must execute gtags(1) and htags(1) at the root directory of a 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`'.

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 mozilla.

'`-d`' name Print object definitions.

'`--help`' Show help.

'`-p`' Print just 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

'`HTML/`' Hypertext of source code.

'`GTAGS/`' Tag file for object definitions.

'`$HOME/.gozillarc`'
Alias file. Please read source code for the detail.

ENVIRONMENT

GTAGSROOT
The root directory of the project.

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

BROWSER
Browser to use. By default, it is assumed mozilla.

EXAMPLES

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

$ firefox &
$ gozilla -b firefox +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), firefox(1), epiphany(1), mozilla(1).

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

NOTES

Gozilla means 'Global for mozilla'.

BUGS

Gozilla can accept not only source files but also text files, directories, HTML files and even URLs, because it is omnivorous.

AUTHORS

Shigio YAMAGUCHI.

HISTORY

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

5.5 gtags-cscope - interactively examine a C program

NAME

gtags-cscope - interactively examine a C program

SYNOPSIS

gtags-cscope [-bCdehLlVv][-F symfile][-0123456789 pattern][-p n]

DESCRIPTION

gtags-cscope is an interactive, screen-oriented tool that allows the user to browse through C 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, the data for the unchanged files are copied from the old cross-reference, which makes rebuilding faster than the initial build.

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 on the first line of screen the version number of gtags-cscope.

‘`--help`’ Same as ‘`-h`’

‘`--version`’
Same as ‘`-V`’

The following options can appear in any combination:

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

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

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

‘`-e`’ Suppress the <Ctrl>-e command prompt between files.

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

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

‘-l’ Line-oriented interface (see “Line-Oriented Interface” below).

‘-[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 not to display the file name at all.

‘-v’ Be more verbose in line-oriented mode. Output progress updates during database building and searches.

Requesting the initial search

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

```
Find this C symbol:
Find this function definition:
Find functions called by this function (N/A):
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
```

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.

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

<Return>	Move to next input field.
^g	Read lines from the result of the execution of specified command.
^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, 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 starting with the field number (counting from 0) immediately followed by the search pattern, for example, "lmain" 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, 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 the 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. Ex: 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 the GLOBAL origin.

GTAGSROOT
The root directory of the project.

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

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

GTAGSCONF
Configuration file. The default is '`$HOME/.globalrc`'.

GTAGSLABEL
Configuration label. The default is default.

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 object definitions.

'`GRTAGS`' Tag file for object references.

'`GPATH`' Tag file for path of source files.

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

‘`$HOME/.globalrc`’, ‘`/etc/gtags.conf`’, ‘`[sysconfdir]/gtags.conf`’
Configuration files.

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 1980's, 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.

5.6 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 function 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'<ENTER>) about a detailed usage.

FILES

‘`GTAGS`’ Tag file for object definitions.

‘`GRTAGS`’ Tag file for object references.

‘`GPATH`’ Tag file for path of source files.

‘`~/.globashrc`’
The personal initialization file, executed for globash.

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).

Appendix A Copying This Manual

A.1 GNU Free Documentation License

Version 1.2, November 2002

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.

Appendix B Business Model

B.1 The BOKIN Model Definition

Introduction

BOKIN Model is a business model to obtain proceeds by widely collecting donations while developing and distributing free software. This model is constructed not to take away consumer's freedom of software.

The business which comply with the following criteria can be called *a business based on BOKIN Model*.

Criteria

1. CORPORATION
   The person who start a business based on BOKIN Model must be a business corporation registered in the home country. (Herein after called *the corporation*)
2. FREE SOFTWARE
   The corporation develops free software. (Herein after called *the BOKINware*)
3. LICENSE
   The corporation distributes the BOKINware under GNU GPL (GNU General Public License) and GNU FDL (GNU Free Documentation License). Exceptionally, external packages which the BOKINware uses, small supporting files, short manuals and rough documentation can use simple all-permissive license, compatible with GNU GPL.
4. COPYRIGHT MANAGEMENT
   The corporation manages copyright on the BOKINware for consumers to keep on using it at ease.
   - Every file in the BOKINware should have a legally valid copyright notice and a license notice.
   - To include program which is assigned from another developer, the corporation receives a disclaimer paper or assignment paper signed by the author.
   - To include program which is not assigned, the corporation confirms its license is GNU GPL or compatible with GNU GPL, lists the files and authors in a file named '`AUTHORS`', and lists the license in a file named '`LICENSE`'. The BOKINware should contain these two files.
5. MAILING LIST
   The corporation maintains mailing lists for consumers to cooperate one another.

   The list members, including the corporation, don't owe any duty.

   The mailing lists should include the following two at least.

- Bug mailing list
  This list distributes, to the active maintainers of the BOKINware, bug reports and fixes for, and suggestions for improvements in the BOKINware. This list is also for user discussion.
- Help mailing list
  This list is the place for authors, users and installers of the BOKINware to ask for help.

The mailing lists can be replaced with a similar communication tool.

The corporation can decide the operation policy of the list, but must not obstruct the list members to cooperate one another.

6. COLLECTING DONATIONS
   The corporation collects donations widely as its proceeds.

   The corporation must not offer the donor an individual supply of profit.
7. DONOR LIST
   The corporation open the donor list to the public.

   The donor list includes the following information.

   - Date of donation (The date when the corporation received the donation)
   - Amount of donation (Amount which the corporation received)
   - Donor's name
   - Donor's nationality

   When donor's name and nationality are unknown or the donor prefers to remain anonymous, they are treated as *anonymous*.

   The BOKINware should contain the donor list as a file named '`DONORS`'. It is preferable that the list is open to the public even on the Internet.
8. BOKIN MODEL DEFINITION
   The BOKINware should contain the present definition as a file named '`BOKIN_MODEL`'.

Renewal

The author may publish revised and/or new versions of the BOKIN Model Definition 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.

B.2 Frequently Asked Questions

Version 1.0, December 17, 2005

BOKIN Model Frequently Asked Questions

1. What does *BOKIN* mean?
   BOKIN means collecting donations in Japanese. (*BO*=collect, *KIN*=money)
2. What is the purpose to require the person who start a BOKIN model business being a registered corporation?
   The purpose is to prevent people from donating to the person who does not exist actually.
3. Is annoying copyright management necessary?
   Yes, it is. Copyright management is absolutely necessary for consumers to keep on using the BOKINware at ease.

   It is dangerous to use the software whose copyright is not neatly managed. If you use such software, you might suddenly be prohibited to use it, or be claimed a license fee of high priced. These are not imaginary fears but troubles of reality.
4. Why is program license limited to GNU GPL?
   Because GNU GPL defends consumers in two points.
   - Copyleft License
     Since GNU GPL is copyleft license, it makes a program free, and requiring all modified and extended versions of the program to be free as well. As a result, consumer can keep on using the BOKINware at ease in the future.
   - Widely Known
     Since GNU GPL is widely known, and is explained frequently, it does not become the load to consumer. It is troublesome for consumer to understand new licenses.
5. What is the purpose of the donor list?
   There are two purposes.
   - To defend freedom of donation.
     The consumer can decide whether to donate after understanding the situation of the donation. If nothing being informed, freedom does not exist there. In BOKIN model, consumers are not isolated existence.
   - To praise donation.
     To praise donation brings new donors. Since BOKIN model owes all to people's free wills, we cannot praise the donation too much.
6. Is donation spent on the BOKINware?
   It depends on the management of the corporation. Since donations become the proceeds of the corporation, the corporation itself decides the usage under its freedom.
7. Is the donor list kept true?
   It is very difficult to mix lies in the public information, because it is checked by various methods.

- Donors can confirm whether they are listed.
- People can ask whether to have donated to the donors in the list.
- The tax office can examine the contradiction between the content of the list and the content of the declaration of the corporation's taxation business.

8. Why is the corporation prohibited from doing an individual supply of profit for the donors?
   When individual supply of profit becomes ordinary, donation fall into the payment for the profit. We cannot call it donation. BOKIN Model business should be supported only by people's free will.

Option Index

Table of Contents