開発のお供に——タグ・リファレンスツール

*Debian Projectオフィシャルメンバ、Debian JP Projectメンバ、YLUG発起人。コンピュータ全般、とりわけフリーソフトウェアとUn*xに関心を持って活動している。*

後藤 正徳

探している関数が定義されている場所にエディタでジャンプしたい、どこで使われているのか調べたい、そんなときはないだろうか？プログラムを書いているときはしょっちゅうそういう場面にぶつかるはずである。自分が作ったプログラムではないソースを見るときは、特にそうだろう。ひょっとして、findやgrepを使って探してはないだろうか？

```
grep -r main *
```

この方法だと、関係ない部分まで探してきてしまう。特定のあまり使われないような文字列を検索するときには便利だが、“main”のようなありきたりの名前を探し出すのは大変だろう。

そういうときに、真価を発揮する便利なツールが、タグ・リファレンスツールだ。タグ・リファレンスツールは、ソース中のシンボルをタグとして別のファイルに記憶させておき、ユーザが検索をかけるときは、そのファイルから参照情報を引っ張ってきてエディタなどで表示・ジャンプするものだ。特に大規模なプログラムをハックするときは非常に便利である。

そこで今回は、私が使ってみたタグ・リファレンスツールのうち、ctags,etags,cscopeのみに絞って、独断と偏見を交えて紹介したい。どれも見た目や使い方は原始的・単純かもしれないが、Un*xな開発環境とルック&フィールには非常にマッチしていて、Un*x使いにはご機嫌なはずだ。一見すると、Windowsの統合開発環境のように、見た目が綺麗で、GUIで操作できるようなものがよいと思う人もいるかもしれない。しかし、MDIウィンドウの中にすべてのウィンドウが入っていないと使えないような不便さも、起動・処理の遅さもない。一度慣れれば大変快適であること間違いなしである。

ちなみに、私はまだまだこれらのツールを最近普通に使い始めたというだらしない状態なので、まったく使いこなしていない :-D。もっと活用できるはずなので、興味を持たれたらぜひさまざまな機能を試して頂きたい。

● ctags、 etags

最もよく知られているタグ・リファレンスツールがこの ctags、 etags である。 ctags は vi 用のタグを、 etags は emacsen[*1] 用のタグを、 それぞれ生成するツールである。 一般的な ctags,etags は、 emacsen の中に一緒にバンドルされて配布されている。 対応している言語は、 新しいｅｍａｃｓｅｎ 付属の ctags,etags では、 C、 C++ を始め 10 以上に対応している。

○vi で使う ctags

vi でタグ・リファレンスツールを使うためには、 ctags によって最初にタグファイルを生成する。

```
ctags *.[ch]
```

これで、 カレントディレクトリにタグファイル “tags” が生成される。 なお ctags のオプションは色々とあるので、

```
ctags —help
```

として見てみるとよいだろう。

さて、 生成したタグは、 次のようにして使う。

```
vi -t もげ
```

これで、 「もげ」 に指定された関数 「もげ()」 を検索して、 vi を立ち上げ、該当関数のところにジャンプしてくれる。

また、 コマンドラインからタグ一覧を参照することもできる。

```
ctags -x *.[ch]
```

とすると、 関数名とそれが定義されている行数、 そしてファイル名とどのように定義されているかがそれぞれ表示されるはずだ。

○ctags 実戦!

ということで、 早速使ってみよう。 今回ターゲットにするのは、 後程紹介する cscope の最新版 cscope 15.3 のソースである。 ソースの tarball を展開すると、 src の下にコードが入っているので、 そのディレクトリに移動しよう。 まず、 ctags でタグファイルを生成する。

```
ctags *.[ch]
```

続いて mymalloc()という関数を探してみよう。

```
vi -t mymalloc
```

どうだろうか？ alloc.c の 64 行目にカーソルが飛んだはずである。

○emacsen で使う etags

emacsen でタグ・リファレンスツールを使うためには、 ctags と同様、 最初に etags を使ってタグファイルを生成する。

```
etags *.[ch]
```

これでTAGSというファイルができる。

続いてemacsenを立ち上げよう。 そして

```
M-x visit-tags-table
```

とする。

```
Visit tags table: (default TAGS) ~/
```

と出てくるので、 参照したいTAGSファイルが存在するディレクトリを指定しよう。 TAGSがemacsenに読み込まれる。

実際にタグを参照したいときは、

```
M-x find-tag
```

または

```
M-.
```

とする。

```
Find tag:
```

と出てくるので、 探したい関数名を入力しよう。

存在すれば、 該当ファイルの関数が定義されている行にジャンプするはずである。

また、 ソースファイル中で定義された名称を検索したいときがあるだろう。通常文字列検索に使う

```
M-x search-forward
```

または

```
C-s
```

では、 単一ファイル中しか検索してくれない。

そこで、

```
M-x tags-search
```

として、 検索文字列を入力しよう。 emacsenが該当文字列を各ファイルごとに検索していき、 該当ファイルを自動的に開いてその文字列のところにカーソルを持っていってくれる。

また、 該当文字列を更に検索していきたいときは、

```
M-x tags-loop-continue
```

または

```
M-,
```

とすれば、 次々と各ファイルの中を探し当てて表示してくれる。

ほかにもfind-tag-regexpやtags-query-replace、 tags-apropos があるので、

いろいろと試してほしい。

○etags 実戦！

それではetagsを使ってみよう。 ctags同様cscopeのソースを使用してmymalloc()という関数を探してみる。 まずはetagsを使ってタグファイルを生成する。

```
etags *.[ch]
```

次にemacsenを起動し、

```
M-x visit-tags-table
```

として、 TAGSファイルの在処を入力しよう。 次に

```
M-.
```

と入力して

```
mymalloc
```

を検索対象とする。 無事alloc.cにカーソルが飛んだだろうか？

○おまけ

あまり知られていないことだが、 Linux kernelのソースディレクトリで

```
make TAGS
```

とすると、 etagsで生成したタグファイルが作成される。

```
make tags
```

で、 タグファイルがctagsで生成されたりする。

このほかに、 ctags には

```
http://ctags.sourceforge.net/ ctags-exuberant
```

“A multilanguage implementation of CTAGS” というのもある。 こちらは、 さらに多くの言語や機能をサポートしているようだ。

● cscope

私が最近愛用しているのが、 このcscopeである。 歴史的にはSCO （Santa Cruz Operation） によって開発されて、 主に商用方面で使用されてきたらしい。 現在では、 cscopeは次のURLから入手できる。 最新版は15.3である。

```
http://cscope.sourceforge.net/
```

対象言語は、 主にCが想定されている。 また、 編集エディタとしてはviを想定しているが、 xemacsやemacs用の.elを使えば、 emacsenからでもcscopeが使えるらしい。

○使い方

cscopeは、 先ほどのctags, etagsと異なり、 端末対話型インターフェースになっている。 まず、 端末でcscopeと入力して立ち上げよう。 すると、 起動

時にタグ （cscope ではcross-reference と呼ぶ） ファイルを自動的に生成してくれて、 cscope のメイン画面に移るだろう。

なお、 詳しいオプションは

```
cscope -h
```

で表示されるものや man ページを参照してもらうことにして、 ここでは 「すぐ実戦」 となるオプションを紹介しよう。

```
Usage: cscope [-bcCdehklLqRTuUV] [-f file] [-F file] [-i file] [-I dir] [-s dir]
              [-p number] [-P path] [-[0-8] pattern] [source files]
```

-b　　cross-reference を生成するだけですぐ終了する。

-C　　大文字・小文字を無視して検索する。

-d　　cross-reference を更新しない。-b オプションで生成した cross-reference があるディレクトリでは、すぐcscope が起動するようになる。

-I incdir　　incdir 中の#include ファイルを読むようになる。

-k　　Kernel モードで起動。/usr/include の#include ファイルを読まなくなる。

-p n　　cscope で表示されるファイル名でn ディレクトリまでたどる。

-q　　逆参照インデックスファイルを作成する。検索が高速になる。

-R　　再帰的にディレクトリを参照する。

普段私は、 カーネルソースなど/usr/include 以下を参照させたくないとき（cscope はデフォルトで /usr/include 下のヘッダも自動的に読込んでくれる）は、

```
cscope -bkqR
```

通常のプログラムを見るときは、

```
cscope -bqR
```

としてcross-reference を生成し、 ソースに手を入れてない状態で cscope を立ち上げるときは

```
cscope -dqR -p 2
```

などとして高速に起動させている。

さて、 cscope が立ち上がると次の画面になるだろう。

```
Find this C symbol:     ※ここで指定したシンボルが使用されている個所を検索
Find this global definition:  ※定義されている個所を検索
Find functions called by this function:       ※検索関数が別の関数を呼び出す個
```

所を検索

Find functions calling this function: ※この関数を呼び出している個所を検索

Find this text string: ※文字列を単純検索

Change this text string: ※検索文字列を置換する

Find this egrep pattern: ※正規表現で検索する

Find this file: ※入力文字列をファイル名としてエディタを起動

Find files #including this file: ※文字列を#include している個所を検索

ここで、 検索させたい機能の行にカーソルを移動させ、 検索文字列を入力する。 たとえばFind this C symbolでは、 入力検索文字列が使用されている個所を探し出してくれて、 一覧表示してくれる。 そして、 最初の行に表示された数字 （または文字） を叩けば、 勝手に該当行にジャンプしてくれる。 もし該当する個所が1つしかないときは、 自動的にviが起動される。 viで編集した後cscopeの画面に戻るには、 viを終了させればよい。 また、 cscope自体を終了させたいときは

```
C-d
```

とすれば抜けられる。

なおcscopeについては、 さらに詳しい使い方の解説が

```
http://docs.sun.com/htmlcoll/coll.674.1/euc-jp/CUG/cscope.html
```

Sunのページより見られる。 なお、 このページには、 cscopeを使ったデバッグの分かりやすい例が解説されている （mymalloc()を例題にしているので、 私の文章を読んでからだと入り込みやすいはずである :-)。

○cscope 実戦！

それでは、 cscope 15.3のソースを使用して、 mymallocの定義を検索してみよう。

```
cscope -dqR -p 2
```

など、 お好みのオプションを付けてリファレンスファイルを生成する。 次に、cscopeを起動する。

```
cscope
```

ここで

```
Find this global definition:
```

にカーソルを移動しよう。 そしてmymallocと入力する。 これでalloc.cに画面が切り替わったはずである。

次にmymalloc関数が使われている個所を検索してみよう。 ctags、 etagsでは、 実質出来ない変数名・関数名の使われている場所の検索も、 cscopeなら楽々である。

```
Find this C symbol:
```

にカーソルを移動して mymalloc と入力する。 すると、 画面上部に使われている個所がずらりと表示されたはずだ。 カーソルで移動して、 mymalloc が使用されている関数へ、 簡単に飛ぶことができることを確認できるだろう。

●おわりに

ある程度ソースコードが大きくなってくると、 こういったタグ・リファレンスツールは必須になってくる。 むしろ、 タグ・リファレンスツールは基本的な開発の道具といっても過言ではない。 今回は取り上げられなかったが、 Un*x には次のようなツールもあり、 大変有用である。

```
http://lxr.linux.no/ lxr - Linux Cross-Reference
http://www.stack.nl/~dimitri/doxygen/index.html Doxygen
http://www.tamacom.com/global/index.html global
```

便利に使いこなして、 是非ハックの速度を向上してほしい。

Happy Hacking！

*1) emacsen とは emacs 一族のこと。 emacs、 xemacs、 mule などを指す。