

## CTAGS

Ctags est un outil d'indexation (ou de référence croisée) pour les langages de programmation (C, C++, Lisp, Python, Perl, ...). C'est un outils très important pour les programmeurs.

### Où le télécharger ?

<http://ctags.sourceforge.net/>

- Prendre le binaire RPM (ctags-5.5.4-1.i386.rpm) pour Redhat ou Mandrake
- Prendre les sources (ctags-5.5.4.tar.gz) pour les autres distributions.
- Il existe également une version pour Windows (ec554w32.zip)

### Comment l'installer ?

- Pour Redhat ou mandrake :

```
$ rpm -iv ctags-5.5.4-1.i386.rpm
```

- Pour les autres distributions :

```
$ tar xzvf ctags-5.5.4.tar.gz  
$ cd ctags-5.5.4/  
$ ./configure  
$ make && make install // il faut être root
```

- Pour Windows :

Dé-zipper ec554w32.zip et double-cliquer sur l'exécutable.

### A quoi cela sert-il ?

Ctags génère un fichier d'index des objets (variables, fonctions, maccros) des langages connus.

Cet index, qui répertorie les informations sur divers objets trouvés dans une série de fichiers écrits dans les langages connus par ctags :

- Permettra aux éditeurs de textes (GNU/Emacs, Vi, Nedit, ...) de localiser plus aisément les éléments indexés.
- Peut être affiché sur la sortie standard, sous un format compréhensible pour l'Homme.

### Comment l'utiliser ?

```
$ ctags [options] [fichier(s)] cela va créer un fichier nommé « tags » .
```

Pour plus d'informations :

- Le manuel : `$ man ctags`
- Le built-in du shell : `$ help ctags` ou `$ ctags --help`

## **Les options**

Pour les options suivantes, ctags ne crée pas de fichier « tags », il utilise la sortie standard comme fichier.

`--list-languages` Affiche la liste des langages de programmation supportés par ctags.

**Exemple :**

```
$ ctags --list-languages
```

```
Asm
```

```
Asp
```

```
Awk
```

```
C
```

```
C++
```

```
Cobol
```

```
...
```

`--list-maps=x` Affiche la liste des extensions possibles pour un langage *x*.  
*x* peut prendre les valeurs suivantes : C, C++, Lisp, Python, ... ou all pour afficher les extensions de tous les langages supportés.

`-x` Affiche une référence croisée sur la sortie standard dans un format agréable pour un être humain (formaté comme l'option `-l` de la commande `ls`).

L'affichage se fait avec une ligne de 5 colonnes par objet indexé.

Colonne 1 : nom de l'objet indexé.

Colonne 2 : type de l'objet (variable, fonctions, macros, ...).

Colonne 3 : ligne ou cet objet est défini.

Colonne 4 : fichier ou est défini l'objet.

Colonne 5 : déclarations ou prototypes de l'objet.

**Exemple :**

```
$ ctags -x *.c *.h Cf. Capture 5 pour le résultat de cette commande
```

`-x --c-kinds = t` Affiche sur la sortie standard les objets indexés de type *t* uniquement.  
*t* peut prendre les valeurs suivantes :

- *v* pour afficher seulement les variables indexés.
- *f* pour les fonctions.
- *d* pour les macro.
- *m* pour les structures / classes.
- *t* pour les typedefs.
- ...

Pour connaître les valeurs possibles de *t* (et leurs significations) pour les langages connus par ctags, utiliser la commande `ctags --list-kinds`

**Exemple :**

```
$ ctags -x -c-kinds=f *.c
```

Cf. Capture 2 pour le résultat de cette commande

Pour les options suivantes, `ctags` va créer un fichier nommé « tags » par défaut. C'est ce fichier qui sera utilisé par les éditeurs de texte.

-a Ajoute les index dans un fichier t « tags » existant (sans écrasement).

-e Le fichier d'index se nommera « TAGS » et non « tags ».

Pour qu'Emacs puisse l'utiliser.

Équivalent à l'outil *etags* (`ctags` spécialement conçu pour Emacs).

**Exemple :**

```
$ ctags -e *.c *.h
```

Cette commande va créer un fichier d'index de tous les fichiers se terminant par `.c` ou par `.h`, dans le répertoire où elle a été lancée ;

Cela implique que l'utilisateur qui lance cette commande doit avoir les droits d'écriture dans le répertoire d'où il se trouve.

-f *filename* Le fichier d'index s'appellera *filename* au lieu de « tags ».

Équivalent à l'option `-o filename`.

**Exemple :**

```
$ ctags -f mestags *.c *.h
```

 va créer un fichier d'index nommé « mestags ».

Cf. Capture 3.

-R Pour lancé `ctags` récursivement au niveau des répertoires.

Le fichier de référence sera créé dans le répertoire où la commande a été tapé.

--totals=yes affiche le nombre de fichier indexé, le nombre d'objet indexé, les temps mis pour construire l'index et pour le trier par ordre alphabétique.

Cf. Capture 3

...

Pour les autres options consulter le man ou le built-in du shell.

### **Comment l'utiliser avec l'éditeur gvim et Vi?**

```
$ ctags *.c *.h
```

```
$ gvim -t plus
```

Ceci éditera le fichier programme C qui contient la fonction *plus()* et placera directement le curseur sur la première ligne de la fonction *plus()*.

```
$ gvim -t main
```

De même, cette commande vous placera sur la ligne contenant la définition de la fonction *main()*.

Dans l'éditeur Vi, vous pouvez sauter à une fonction en tapant : (double point) `tag nom_de_la_fonction` comme ci dessous :

DUPUY Daddimy  
169 634

Sassi M. Amine  
177 138

: *tag nom\_fonction*

: *tag div ()*

Ceci placera le curseur sur la première ligne de fonction ().

Si vous voulez sauter dans la fonction à partir de la ligne du fichier contenant le nom de la fonction, placez le curseur juste avant le nom de la fonction et tapez CTRL+] (tapez la touche de contrôle et le crochet gauche simultanément).

```
int main ()  
{ printf(“%d\n”, div(UN,DEUX))
```

^  
|  
|

Placez le curseur ici (juste avant div()) et tapez CTRL+]

Ceci vous emmènera à la définition de la fonction "div()" (même s'il est dans un autre fichier).

Pour revenir à cette ligne tapez CTRL+t (la touche CTRL et la lettre 't' simultanément).

Continuez à appuyer sur CTRL+t pour inverser et revenir à la première ligne où vous avez commencé la navigation. C'est-à-dire que vous pouvez conserver pressées CTRL+] et ensuite taper CTRL+t pour revenir. Vous pouvez refaire ceci aussi souvent que vous le désirez pour avoir une navigation complète au travers de toutes les fonctions C ou C++.

### **Comment l'utiliser avec GNU/Emacs ?**

Par défaut, Emacs va essayer de charger un fichier d'index nommé « TAGS » dans le répertoire courant ; c'est pour cette raison que l'on utilise l'option -e qui crée ce fichier. Une fois, le fichier d'indexation créé, on peut exécuter les commandes suivantes :

*M-x visit-tags-table <RET> filename <RET>* (par défaut, *filename* = TAGS).

Cette commande permet de sélectionner et de charger le fichier d'index à utiliser par Emacs.

*M-. [TAG] <RET>*

Trouve le premier objet indexé. Par défaut, c'est l'objet (fonction ou variables) sous le curseur.

*M-\**

C'est l'équivalent du CTRL+t dans Vi. (cf. le paragraphe précédent).

On peut également utiliser l'interface graphique de Emacs :

*Edit / Go to / Set Tags File Name* pour charger le fichier d'index.

*Edit / Go to / Find Tag* équivalent à *M-. [TAG] <RET>*

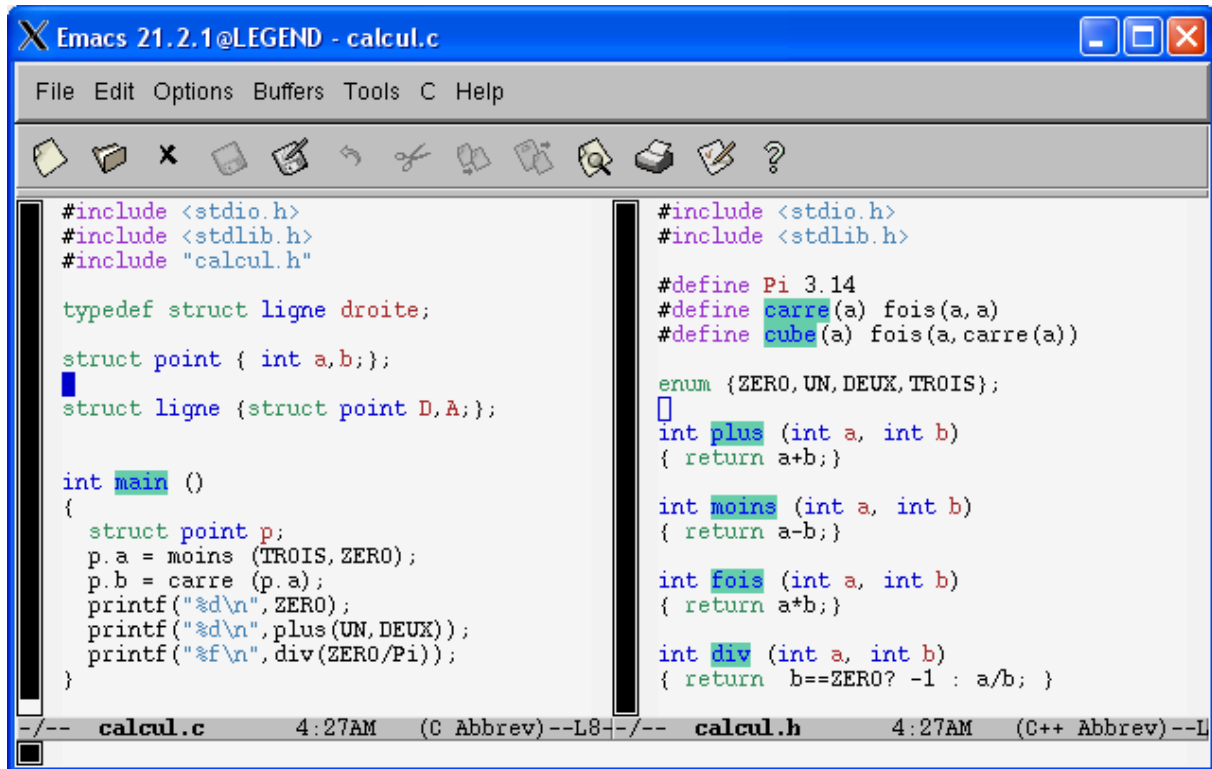
### **Autres**

Le fichier d'index ainsi créé peut aussi servir à d'autres éditeurs ou autres programmes, notamment dans des scripts shell.

Cf. Capture 4 pour voir le contenu d'un fichier d'index créé par ctags.

## Les sources et captures d'écran pour les exemples:

Les fichiers sont placés dans un répertoire nommé Exemple :



The screenshot shows the Emacs editor window titled "Emacs 21.2.1@LEGEND - calcul.c". The window contains two source files side-by-side. The left pane shows the content of `calcul.c` and the right pane shows the content of `calcul.h`. The status bar at the bottom indicates the current file and line number for both panes.

```
#include <stdio.h>
#include <stdlib.h>
#include "calcul.h"

typedef struct ligne droite;

struct point { int a,b;};
struct ligne {struct point D,A;};

int main ()
{
  struct point p;
  p.a = moins (TROIS, ZERO);
  p.b = carre (p.a);
  printf("%d\n", ZERO);
  printf("%d\n", plus(UN, DEUX));
  printf("%f\n", div(ZERO/Pi));
}

#include <stdio.h>
#include <stdlib.h>

#define Pi 3.14
#define carre(a) fois(a, a)
#define cube(a) fois(a, carre(a))

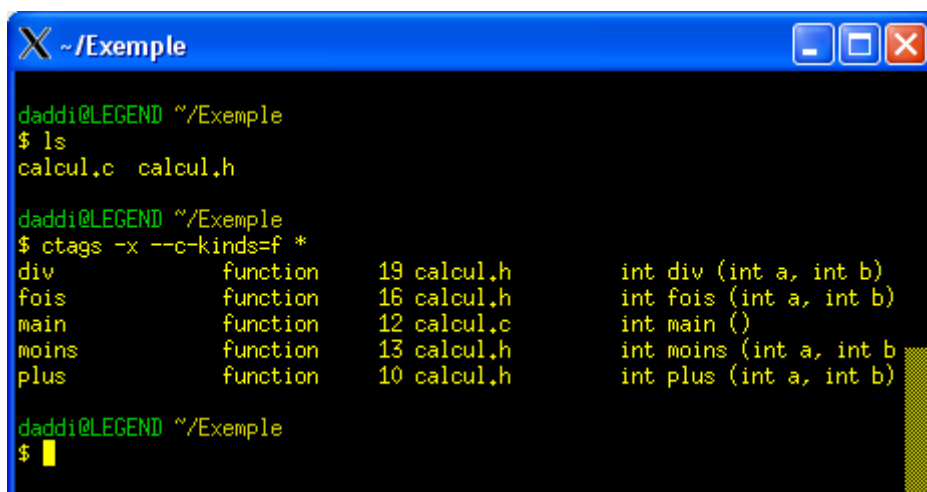
enum {ZERO, UN, DEUX, TROIS};
int plus (int a, int b)
{ return a+b;}

int moins (int a, int b)
{ return a-b;}

int fois (int a, int b)
{ return a*b;}

int div (int a, int b)
{ return b==ZERO? -1 : a/b; }
```

Capture 1 : les fichiers du répertoire Exemple/



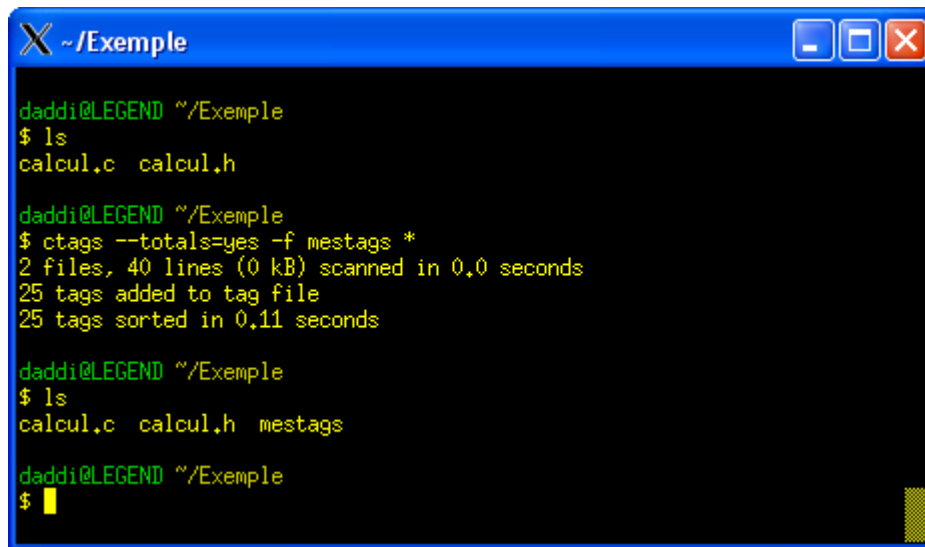
The screenshot shows a terminal window titled "~ / Exemple". The user has run the command `ctags -x --c-kinds=f *` to generate a tag file. The output shows a list of functions and their locations in the source files.

```
daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h

daddi@LEGEND ~/Exemple
$ ctags -x --c-kinds=f *
div          function    19 calcul.h    int div (int a, int b)
fois        function    16 calcul.h    int fois (int a, int b)
main        function    12 calcul.c    int main ()
moins      function    13 calcul.h    int moins (int a, int b)
plus       function    10 calcul.h    int plus (int a, int b)

daddi@LEGEND ~/Exemple
$
```

Capture 2 : l'option `-x --c-kinds=f`



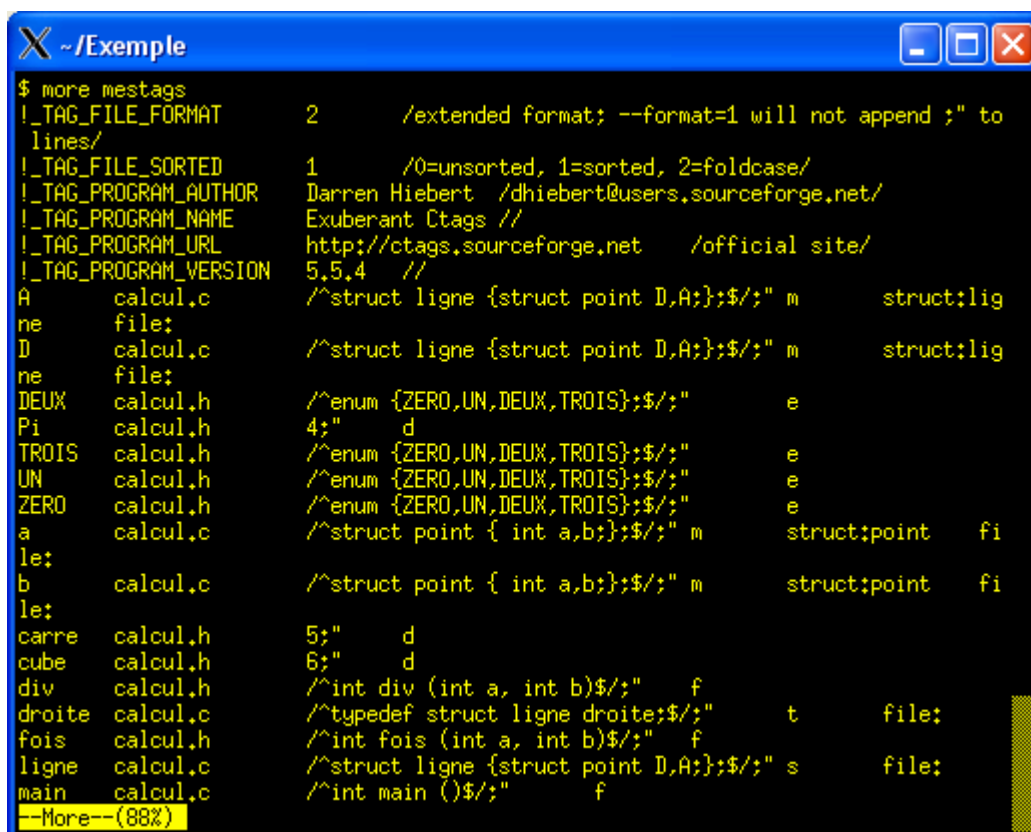
```
~/Exemple
daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h

daddi@LEGEND ~/Exemple
$ ctags --totals=yes -f mestags *
2 files, 40 lines (0 kB) scanned in 0,0 seconds
25 tags added to tag file
25 tags sorted in 0,11 seconds

daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h mestags

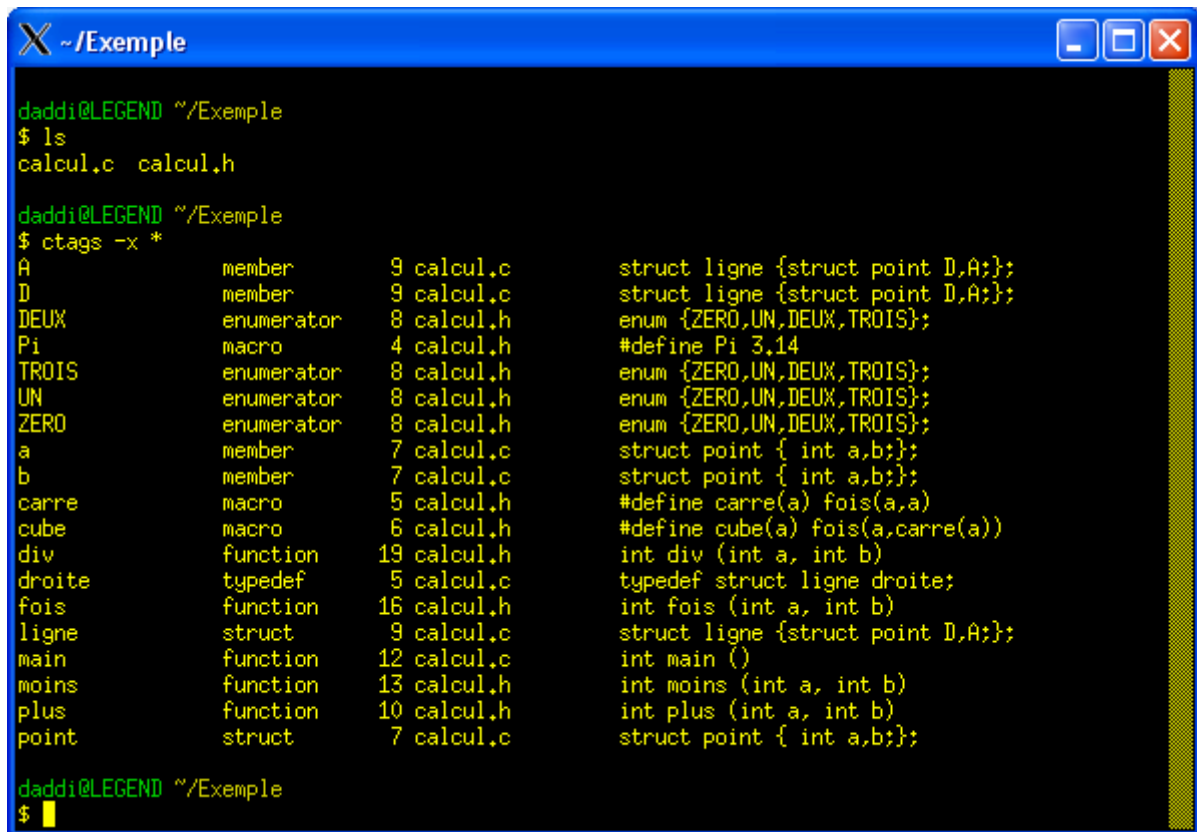
daddi@LEGEND ~/Exemple
$
```

Capture 3 : l'option -f et --totals=yes



```
~/Exemple
$ more mestags
!_TAG_FILE_FORMAT      2          /extended format; --format=1 will not append ;" to
lines/
!_TAG_FILE_SORTED      1          /0=unsorted, 1=sorted, 2=foldcase/
!_TAG_PROGRAM_AUTHOR   Darren Hiebert /dhiebert@users.sourceforge.net/
!_TAG_PROGRAM_NAME     Exuberant Ctags //
!_TAG_PROGRAM_URL      http://ctags.sourceforge.net /official site/
!_TAG_PROGRAM_VERSION  5.5.4 //
A calcul.c             /^struct ligne {struct point D,A};$/" m      struct:lig
ne file:
D calcul.c             /^struct ligne {struct point D,A};$/" m      struct:lig
ne file:
DEUX calcul.h          /^enum {ZERO,UN,DEUX,TROIS};$/" e
Pi calcul.h            4;" d
TROIS calcul.h         /^enum {ZERO,UN,DEUX,TROIS};$/" e
UN calcul.h            /^enum {ZERO,UN,DEUX,TROIS};$/" e
ZERO calcul.h          /^enum {ZERO,UN,DEUX,TROIS};$/" e
a calcul.c             /^struct point { int a,b};$/" m      struct:point fi
le:
b calcul.c             /^struct point { int a,b};$/" m      struct:point fi
le:
carre calcul.h         5;" d
cube calcul.h          6;" d
div calcul.c           /^int div (int a, int b)$/" f
droite calcul.c        /^typedef struct ligne droite;$/" t      file:
fois calcul.h          /^int fois (int a, int b)$/" f
ligne calcul.c         /^struct ligne {struct point D,A};$/" s      file:
main calcul.c          /^int main ()$/" f
--More--(88%)
```

Capture 4 : le contenu du fichier d'index (mestags).



```
X ~/Exemple
daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h

daddi@LEGEND ~/Exemple
$ ctags -x *
A          member      9 calcul.c      struct ligne {struct point D,A;};
D          member      9 calcul.c      struct ligne {struct point D,A;};
DEUX      enumerator  8 calcul.h      enum {ZERO,UN,DEUX,TROIS};
Pi        macro       4 calcul.h      #define Pi 3,14
TROIS     enumerator  8 calcul.h      enum {ZERO,UN,DEUX,TROIS};
UN        enumerator  8 calcul.h      enum {ZERO,UN,DEUX,TROIS};
ZERO      enumerator  8 calcul.h      enum {ZERO,UN,DEUX,TROIS};
a         member      7 calcul.c      struct point { int a,b;};
b         member      7 calcul.c      struct point { int a,b;};
carre     macro       5 calcul.h      #define carre(a) fois(a,a)
cube      macro       6 calcul.h      #define cube(a) fois(a,carre(a))
div       function  19 calcul.h     int div (int a, int b)
droite    typedef    5 calcul.c     typedef struct ligne droite;
fois      function  16 calcul.h     int fois (int a, int b)
ligne     struct     9 calcul.c     struct ligne {struct point D,A;};
main      function  12 calcul.c     int main ()
moins     function  13 calcul.h     int moins (int a, int b)
plus      function  10 calcul.h     int plus (int a, int b)
point     struct     7 calcul.c     struct point { int a,b;};

daddi@LEGEND ~/Exemple
$
```

Capture 5 : l'option -x

## Remarques

Les options peuvent être mis par défaut dans le fichier /etc/ctags.conf ou ~/ctags.conf1  
En général dans les dernières versions, il faut créer soit même ce fichier.

Voici à quoi cela ressemble :

```
/etc/ctags.conf
--verbose=yes
--totals=yes
--recurse
--langmap=asm:
--langdef=Splus
--langmap=Splus:.s.S.R.r.q
--regex-Splus=/^[ \t]+"?([.A-Za-z][.A-Za-z0-9]*)"?[ \t]*(<-|_)[ \t]*function^I/

--regex-Splus=/^"?([.A-Za-z][.A-Za-z0-9]*)"?[ \t]*(<-|_)^I/
```

## Ressources

Internet ([ctags.sourceforge.net](http://ctags.sourceforge.net))  
Le manpage et le help du shel.

## CTAGS

Ctags est un outil d'indexation (ou de référence croisée) pour les langages de programmation (C, C++, Lisp, Python, Perl, ...). C'est un outils très important pour les programmeurs.

### Où le télécharger ?

<http://ctags.sourceforge.net/>

- Prendre le binaire RPM (ctags-5.5.4-1.i386.rpm) pour Redhat ou Mandrake
- Prendre les sources (ctags-5.5.4.tar.gz) pour les autres distributions.
- Il existe également une version pour Windows (ec554w32.zip)

### Comment l'installer ?

- Pour Redhat ou mandrake :

```
$ rpm -iv ctags-5.5.4-1.i386.rpm
```

- Pour les autres distributions :

```
$ tar zxvf ctags-5.5.4.tar.gz  
$ cd ctags-5.5.4/  
$ ./configure  
$ make && make install // il faut être root
```

- Pour Windows :

Dé-zipper ec554w32.zip et double-cliquer sur l'exécutable.

### A quoi cela sert-il ?

Ctags génère un fichier d'index des objets (variables, fonctions, maccros) des langages connus.

Cet index, qui répertorie les informations sur divers objets trouvés dans une série de fichiers écrits dans les langages connus par ctags :

- Permettra aux éditeurs de textes (GNU/Emacs, Vi, Nedit, ...) de localiser plus aisément les éléments indexés.
- Peut être affiché sur la sortie standard, sous un format compréhensible pour l'Homme.

### Comment l'utiliser ?

```
$ ctags [options] [fichier(s)] cela va créer un fichier nommé « tags » .
```

Pour plus d'informations :

- Le manuel : `$ man ctags`
- Le built-in du shell : `$ help ctags` ou `$ ctags --help`

## **Les options**

Pour les options suivantes, ctags ne crée pas de fichier « tags », il utilise la sortie standard comme fichier.

`--list-languages` Affiche la liste des langages de programmation supportés par ctags.

**Exemple :**

```
$ ctags --list-languages
```

```
Asm
```

```
Asp
```

```
Awk
```

```
C
```

```
C++
```

```
Cobol
```

```
...
```

`--list-maps=x` Affiche la liste des extensions possibles pour un langage `x`.  
`x` peut prendre les valeurs suivantes : C, C++, Lisp, Python, ... ou all pour afficher les extensions de tous les langages supportés.

`-x` Affiche une référence croisée sur la sortie standard dans un format agréable pour un être humain (formaté comme l'option `-l` de la commande `ls`).

L'affichage se fait avec une ligne de 5 colonnes par objet indexé.

Colonne 1 : nom de l'objet indexé.

Colonne 2 : type de l'objet (variable, fonctions, macros, ...).

Colonne 3 : ligne ou cet objet est défini.

Colonne 4 : fichier ou est défini l'objet.

Colonne 5 : déclarations ou prototypes de l'objet.

**Exemple :**

```
$ ctags -x *.c *.h Cf. Capture 5 pour le résultat de cette commande
```

`-x --c-kinds = t` Affiche sur la sortie standard les objets indexés de type `t` uniquement.  
`t` peut prendre les valeurs suivantes :

- `v` pour afficher seulement les variables indexés.
- `f` pour les fonctions.
- `d` pour les macro.
- `m` pour les structures / classes.
- `t` pour les typedefs.
- ...

Pour connaître les valeurs possibles de `t` (et leurs significations) pour les langages connus par ctags, utiliser la commande `ctags --list-kinds`

**Exemple :**

```
$ ctags -x -c-kinds=f *.c
```

Cf. Capture 2 pour le résultat de cette commande

Pour les options suivantes, `ctags` va créer un fichier nommé « tags » par défaut. C'est ce fichier qui sera utilisé par les éditeurs de texte.

-a Ajoute les index dans un fichier t « tags » existant (sans écrasement).

-e Le fichier d'index se nommera « TAGS » et non « tags ».

Pour qu'Emacs puisse l'utiliser.

Équivalent à l'outil *etags* (`ctags` spécialement conçu pour Emacs).

**Exemple :**

```
$ ctags -e *.c *.h
```

Cette commande va créer un fichier d'index de tous les fichiers se terminant par `.c` ou par `.h`, dans le répertoire où elle a été lancée ;

Cela implique que l'utilisateur qui lance cette commande doit avoir les droits d'écriture dans le répertoire d'où il se trouve.

-f *filename* Le fichier d'index s'appellera *filename* au lieu de « tags ».

Équivalent à l'option `-o filename`.

**Exemple :**

```
$ ctags -f mestags *.c *.h
```

 va créer un fichier d'index nommé « mestags ».

Cf. Capture 3.

-R Pour lancé `ctags` récursivement au niveau des répertoires.

Le fichier de référence sera créé dans le répertoire où la commande a été tapé.

--totals=yes affiche le nombre de fichier indexé, le nombre d'objet indexé, les temps mis pour construire l'index et pour le trier par ordre alphabétique.

Cf. Capture 3

...

Pour les autres options consulter le man ou le built-in du shell.

### **Comment l'utiliser avec l'éditeur gvim et Vi?**

```
$ ctags *.c *.h
```

```
$ gvim -t plus
```

Ceci éditera le fichier programme C qui contient la fonction *plus()* et placera directement le curseur sur la première ligne de la fonction *plus()*.

```
$ gvim -t main
```

De même, cette commande vous placera sur la ligne contenant la définition de la fonction *main()*.

Dans l'éditeur Vi, vous pouvez sauter à une fonction en tapant : (double point) `tag nom_de_la_fonction` comme ci dessous :

DUPUY Daddimy  
169 634

Sassi M. Amine  
177 138

: *tag nom\_fonction*

: *tag div ()*

Ceci placera le curseur sur la première ligne de fonction ().

Si vous voulez sauter dans la fonction à partir de la ligne du fichier contenant le nom de la fonction, placez le curseur juste avant le nom de la fonction et tapez CTRL+] (tapez la touche de contrôle et le crochet gauche simultanément).

```
int main ()  
{ printf(“%d\n”, div(UN,DEUX))
```

^  
|  
|

Placez le curseur ici (juste avant div()) et tapez CTRL+]

Ceci vous emmènera à la définition de la fonction "div()" (même s'il est dans un autre fichier).

Pour revenir à cette ligne tapez CTRL+t (la touche CTRL et la lettre 't' simultanément).

Continuez à appuyer sur CTRL+t pour inverser et revenir à la première ligne où vous avez commencé la navigation. C'est-à-dire que vous pouvez conserver pressées CTRL+] et ensuite taper CTRL+t pour revenir. Vous pouvez refaire ceci aussi souvent que vous le désirez pour avoir une navigation complète au travers de toutes les fonctions C ou C++.

### **Comment l'utiliser avec GNU/Emacs ?**

Par défaut, Emacs va essayer de charger un fichier d'index nommé « TAGS » dans le répertoire courant ; c'est pour cette raison que l'on utilise l'option -e qui crée ce fichier. Une fois, le fichier d'indexation créé, on peut exécuter les commandes suivantes :

*M-x visit-tags-table <RET> filename <RET>* (par défaut, *filename* = TAGS).

Cette commande permet de sélectionner et de charger le fichier d'index à utiliser par Emacs.

*M-. [TAG] <RET>*

Trouve le premier objet indexé. Par défaut, c'est l'objet (fonction ou variables) sous le curseur.

*M-\**

C'est l'équivalent du CTRL+t dans Vi. (cf. le paragraphe précédent).

On peut également utiliser l'interface graphique de Emacs :

*Edit / Go to / Set Tags File Name* pour charger le fichier d'index.

*Edit / Go to / Find Tag* équivalent à *M-. [TAG] <RET>*

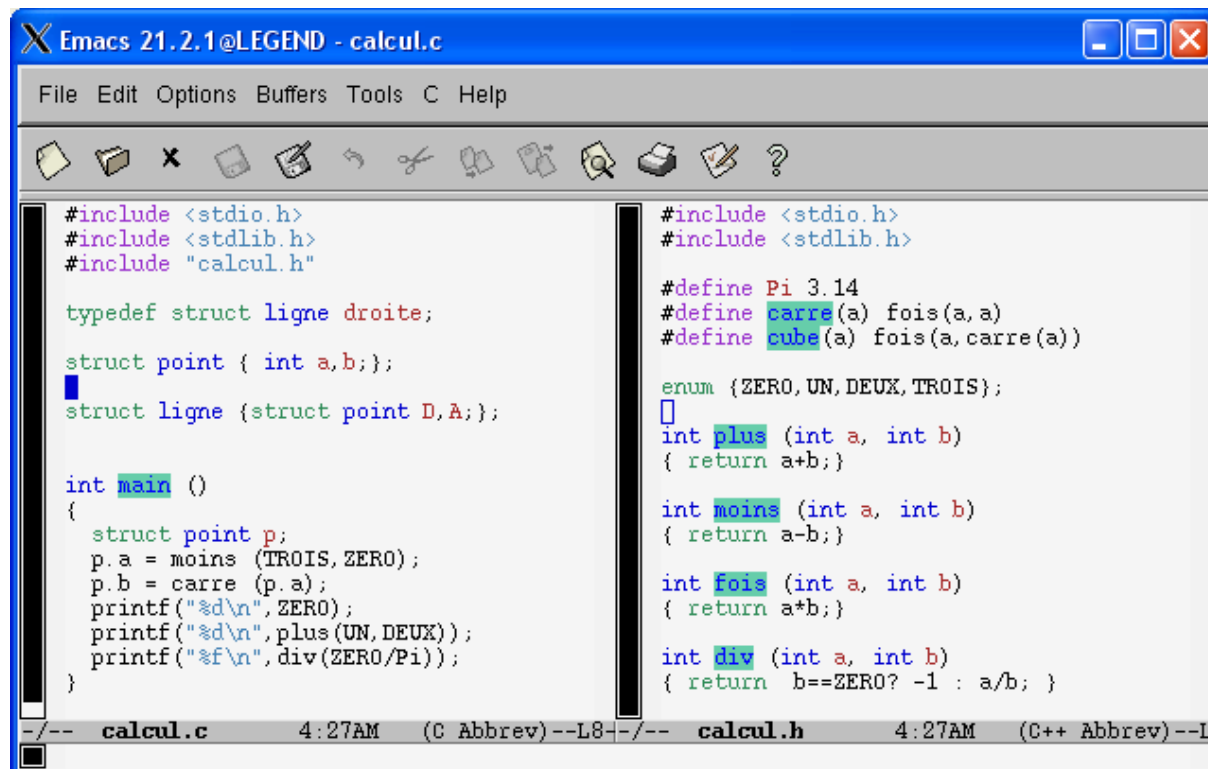
### **Autres**

Le fichier d'index ainsi créé peut aussi servir à d'autres éditeurs ou autres programmes, notamment dans des scripts shell.

Cf. Capture 4 pour voir le contenu d'un fichier d'index créé par ctags.

## Les sources et captures d'écran pour les exemples:

Les fichiers sont placés dans un répertoire nommé Exemple :



The screenshot shows the Emacs editor window titled "Emacs 21.2.1@LEGEND - calcul.c". The window contains two source files side-by-side. The left pane shows the content of `calcul.c` and the right pane shows the content of `calcul.h`. The status bar at the bottom indicates the current file and line number for both panes.

```
#include <stdio.h>
#include <stdlib.h>
#include "calcul.h"

typedef struct ligne droite;

struct point { int a,b;};
struct ligne {struct point D,A;};

int main ()
{
  struct point p;
  p.a = moins (TROIS, ZERO);
  p.b = carre (p.a);
  printf("%d\n", ZERO);
  printf("%d\n", plus(UN, DEUX));
  printf("%f\n", div(ZERO/Pi));
}
```

```
#include <stdio.h>
#include <stdlib.h>

#define Pi 3.14
#define carre(a) fois(a, a)
#define cube(a) fois(a, carre(a))

enum {ZERO, UN, DEUX, TROIS};

int plus (int a, int b)
{ return a+b;}

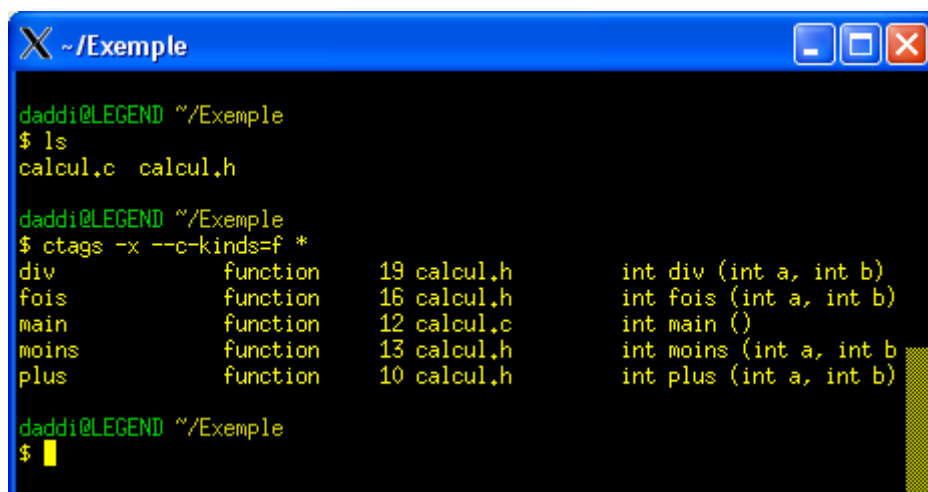
int moins (int a, int b)
{ return a-b;}

int fois (int a, int b)
{ return a*b;}

int div (int a, int b)
{ return b==ZERO? -1 : a/b; }
```

-- calcul.c 4:27AM (C Abbrev)--L8-- calcul.h 4:27AM (C++ Abbrev)--L

Capture 1 : les fichiers du répertoire Exemple/



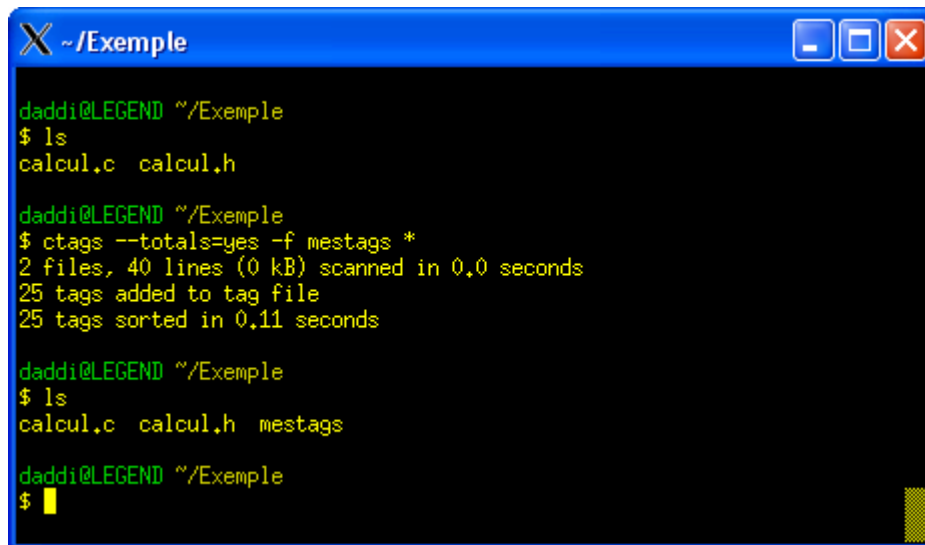
The screenshot shows a terminal window titled "~ / Exemple". The user has run the command `ctags -x --c-kinds=f *` to generate a tags file. The output shows a list of functions and their locations in the source files.

```
daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h

daddi@LEGEND ~/Exemple
$ ctags -x --c-kinds=f *
div          function    19 calcul.h    int div (int a, int b)
fois        function    16 calcul.h    int fois (int a, int b)
main        function    12 calcul.c    int main ()
moins      function    13 calcul.h    int moins (int a, int b)
plus       function    10 calcul.h    int plus (int a, int b)

daddi@LEGEND ~/Exemple
$
```

Capture 2 : l'option `-x --c-kinds=f`



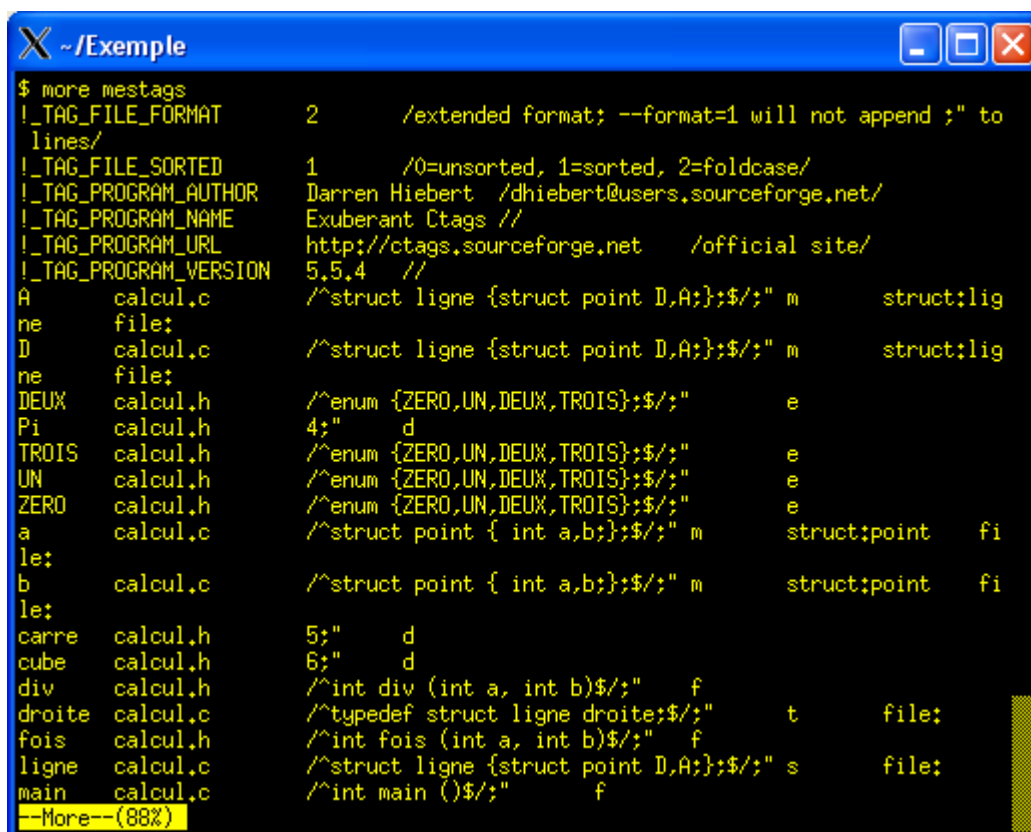
```
~/Exemple
daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h

daddi@LEGEND ~/Exemple
$ ctags --totals=yes -f mestags *
2 files, 40 lines (0 kB) scanned in 0,0 seconds
25 tags added to tag file
25 tags sorted in 0,11 seconds

daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h mestags

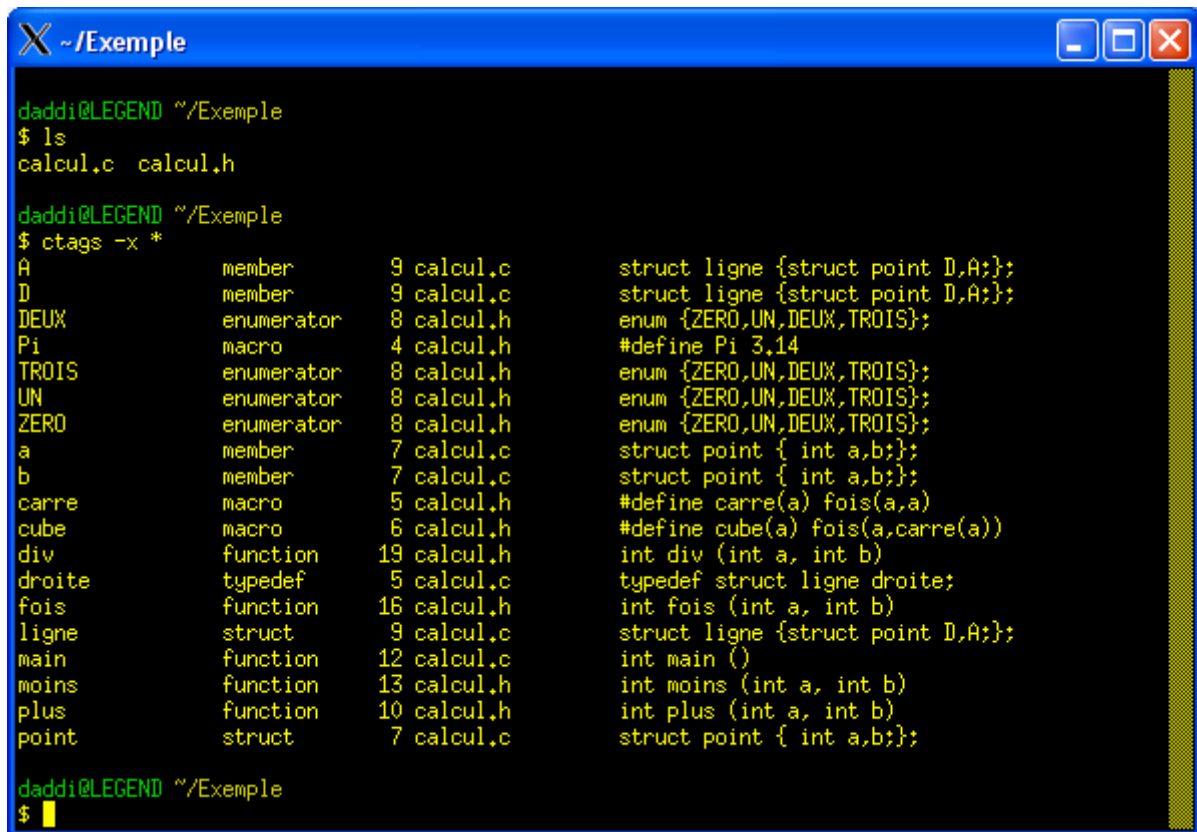
daddi@LEGEND ~/Exemple
$
```

Capture 3 : l'option -f et --totals=yes



```
~/Exemple
$ more mestags
!_TAG_FILE_FORMAT      2       /extended format; --format=1 will not append ;" to
lines/
!_TAG_FILE_SORTED      1       /0=unsorted, 1=sorted, 2=foldcase/
!_TAG_PROGRAM_AUTHOR    Darren Hiebert /dhiebert@users.sourceforge.net/
!_TAG_PROGRAM_NAME      Exuberant Ctags //
!_TAG_PROGRAM_URL       http://ctags.sourceforge.net /official site/
!_TAG_PROGRAM_VERSION   5.5.4 //
A calcul.c              /^struct ligne {struct point D,A};$/" m      struct:lig
ne file:
D calcul.c              /^struct ligne {struct point D,A};$/" m      struct:lig
ne file:
DEUX calcul.h           /^enum {ZERO,UN,DEUX,TROIS};$/" e
Pi calcul.h             4;" d
TROIS calcul.h          /^enum {ZERO,UN,DEUX,TROIS};$/" e
UN calcul.h             /^enum {ZERO,UN,DEUX,TROIS};$/" e
ZERO calcul.h           /^enum {ZERO,UN,DEUX,TROIS};$/" e
a calcul.c              /^struct point { int a,b};$/" m      struct:point fi
le:
b calcul.c              /^struct point { int a,b};$/" m      struct:point fi
le:
carre calcul.h          5;" d
cube calcul.h           6;" d
div calcul.c            /^int div (int a, int b)$/" f
droite calcul.c         /^typedef struct ligne droite;$/" t      file:
fois calcul.h           /^int fois (int a, int b)$/" f
ligne calcul.c          /^struct ligne {struct point D,A};$/" s      file:
main calcul.c           /^int main ()$/" f
--More--(88%)
```

Capture 4 : le contenu du fichier d'index (mestags).



```
~/Exemple
daddi@LEGEND ~/Exemple
$ ls
calcul.c calcul.h

daddi@LEGEND ~/Exemple
$ ctags -x *
A          member      9 calcul.c          struct ligne {struct point D,A;};
D          member      9 calcul.c          struct ligne {struct point D,A;};
DEUX      enumerator  8 calcul.h          enum {ZERO,UN,DEUX,TROIS};
Pi        macro       4 calcul.h          #define Pi 3,14
TROIS     enumerator  8 calcul.h          enum {ZERO,UN,DEUX,TROIS};
UN        enumerator  8 calcul.h          enum {ZERO,UN,DEUX,TROIS};
ZERO      enumerator  8 calcul.h          enum {ZERO,UN,DEUX,TROIS};
a         member      7 calcul.c          struct point { int a,b;};
b         member      7 calcul.c          struct point { int a,b;};
carre     macro       5 calcul.h          #define carre(a) fois(a,a)
cube      macro       6 calcul.h          #define cube(a) fois(a,carre(a))
div       function  19 calcul.h         int div (int a, int b)
droite    typedef    5 calcul.c          typedef struct ligne droite;
fois      function  16 calcul.h         int fois (int a, int b)
ligne     struct     9 calcul.c          struct ligne {struct point D,A;};
main      function  12 calcul.c          int main ()
moins     function  13 calcul.h         int moins (int a, int b)
plus      function  10 calcul.h         int plus (int a, int b)
point     struct     7 calcul.c          struct point { int a,b;};

daddi@LEGEND ~/Exemple
$
```

Capture 5 : l'option -x

## Remarques

Les options peuvent être mis par défaut dans le fichier /etc/ctags.conf ou ~/ctags.conf1  
En général dans les dernières versions, il faut créer soit même ce fichier.

Voici à quoi cela ressemble :

```
/etc/ctags.conf
--verbose=yes
--totals=yes
--recurse
--langmap=asm:
--langdef=Splus
--langmap=Splus:.s.S.R.r.q
--regex-Splus=/^[ \t]+"?([.A-Za-z][.A-Za-z0-9]*)"?[ \t]*(<-|_)[ \t]*function^I/

--regex-Splus=/^"?([.A-Za-z][.A-Za-z0-9]*)"?[ \t]*(<-|_)^I/
```

## Ressources

Internet ([ctags.sourceforge.net](http://ctags.sourceforge.net))  
Le manpage et le help du shell.