“libgsc.a”

Guide de l’utilisateur

Manuel de Référence

Luc Weber

Observatoire de Genève
24 octobre 2012

Table des matières

Chapitre 1

Tutorial

1.1 Utilisation

La librarie `libgsc.a` permet le contrôle du générateur de cartes stellaires basé sur le Guide Star Catalogue (`xgsc` sous OpenWindows) depuis un programme client. Le client peut être un programme écrit soit en Fortran soit en C. Les appels sont similaires, à la différence qu'en C, le status est retourné par la fonction alors qu'en Fortran le status est retourné dans le dernier argument supplémentaire.

Attention, les chaînes de caractères écrites en Fortran doivent impérativement être terminées par un caractère nul (compatibilité avec le C). Exemple :

```
name = 'Wolfgang' // char(0)
```

Cette librarie est construite sur le protocole de communication GOP `Geneva Observatory Protocol`.

Les programmes clients se compilent en C avec la commande :

```
cc prog.c -o prog -I/share/include [options] -lgsc -lgop -ltpudummy
```

ou en Fortran avec

```
f77 prog.f -o prog [options] -lgsc -lgop -ltpudummy
```

1.2 Connection – Déconnection

La connection à `xgsc` se fait avec la fonction `gsc_client()`. Cette fonction alloue une structure pour le canal de communication et initialise la connection. La déconnection est automatique, et `xgsc` survit au client. Si le client qui désire tuer `xgsc` le fait avec la fonction `gsc_close_connection()`. Les exemples suivants montrent une connection suivit d'une déconnection en C et en Fortran.

```
#include <gop.h>
int    actual_status=0;

main()
{
   struct gop_connect *connect_gsc;
   char   option[20];
   int    delay=1;
   int    repeat=10;
   ...
   if (gsc_client(connect_gsc, actual_status, option, delay, repeat, verbose)
                != GOP_OK){...}
   actual_status = 1;
   ...
   if (gsc_quit(connect_gsc) != GOP_OK){...}
}
```

```
      program xxxx

      integer    connect_gsc
      integer    actual_status     /0/
      common     / status_common / actual_status
      character*20   option[20]
      integer    delay             /1/
      integer    repeat            /10/
      integer    status
      ...
      call gsc_client(connect_gsc, actual_status, option, delay,
     .                repeat, verbose, status)
      if(status.ne.0) ...
      actual_status = 1
      ...
      call gsc_quit(connect_gsc, status)
      if(status.ne.0) ...
      ...
      end
```

La fonction `gsc_client` exécute la commande système `gsc` suivie des options donnée par la chaîne de caractère `option`.
Comme le démarage de `xgsc` peut prendre du temps, le client attend que la socket soit créée. La présence de la socket est testée au maximum `repeat` fois avec un interval de temps donné en secondes entières de durée = `delay` [secondes]. Le niveau de verbosité du protocole de communication est initialisé avec la variable verbose : 0, 1, 2 et 3 à 9 (voir libgop.a).
Le flag `actual_status` indique par `1` que le client se croit connecté et par `0` qu'il n'est pas connecté. Ce flag global de déterminer la tactique de connection si la commande `gsc_client` est lancée plusieurs fois dans le même programme.

1.3 Affichage d'un champ

La selection d'un champ se fait en deux étapes. La première étape consiste à charger le panneau de contrôle selon les spécifications du champ avec les commandes `gsc_set_alpha(), gsc_set_delta(), gsc_set_identi` `gsc_set_magnitudes(), gsc_set_scale()` et `gsc_set_year()`. La seconde étape consiste à donner l'ordre à `xgsc` de lire le panneau et à charger le champ avec la fonction `gsc_plot()`. Exemple :

```
program xxxx

integer    connect_gsc
...

call gsc_client(connect_gsc, actual_status, option, delay, repeat, status)
if(status.ne.0) ...


call gsc_set_alpha(connect_gsc, '12h36'//char(0), status)
call gsc_set_delta(connect_gsc, '3d 24 55'//char(0), status)
call gsc_set_magnitudes(connect_gsc, 9., 12., status)
call gsc_set_scale(connect_gsc, '15min'//char(0), status)
call gsc_set_year(connect_gsc, '1900'//char(0), status)

call gsc_plot(connect_gsc, status)

....
end
```

Si l'affichage du champ n'est pas désiré, la fonction `gsc_read_gsc()` peut être substituée à `gsc_plot()`. Exemple :

```
program xxxx
integer    connect_gsc
...
call gsc_read_gsc(connect_gsc, status)
....
end
```

si `xgsc` est interfacé avec un catalogue (voir man xgsc), on peut donner un identificateur d'objet à la place des coordonnée alpha et delta. Exemple :

```
struct gop_connect *connect_gsc;

if (gsc_client(connect_gsc, actual_status, option, delay, repeat)
              != GOP_OK){...}

gsc_set_identifier(connect_gsc, "HD12345");
gsc_set_magnitudes(connect_gsc, 9., 12.);
gsc_set_scale(connect_gsc, "15min");
gsc_set_year(connect_gsc, "1900");
gsc_plot(connect_gsc)
...
```

1.4 Lecture de position

La fonction `gsc_get_cursor()` permet la lecture de coordonnées équatorialles alors que `gsc_get_xy()` permet la lecture de coordonnées écrans. Dans les deux cas `xgsc` répond au client lorsque :

1. le bouton central de la souris a été cliqué. Le curseur se centre sur l'objet la plus proche et renvoie les coordonnées de cet objet. `OK=1`.
2. le bouton de droite de la souris a été cliqué, l'opération est annulée, `OK=0`.
3. la barre d'espacement du clavier a été enfoncée, le curseur reste immobile et ce sont les coordonnées du curseur qui sont renvoyée. `OK=1`.

Le boutton de gauche sert à lire les paramètres des objets du champs tout en restant dans le mode lecture. Exemple :

```
struct gop_connect *connect_gsc;
float               alpha, delta, magnitude;
char                calpha[20], cdelta[20];
char                message[80];
int                 ok;
...
if (gsc_client(connect_gsc, actual_status, option, delay, repeat)
              != GOP_OK){...}
...
if(gsc_plot(connect_gsc) != GOP_OK) {...}
...
if(gsc_get_cursor(connect_gsc, &alpha, &delta, calpha, cdelta, &magnitude,
           message, &ok) != GOP_OK) {...}
...
```

Cet appel retourne alpha et delta en degré d'arc ainsi qu'en `DegMinSec` et `HeuMinSec` dans `calpha et cdelta`, la magnitude, le flag de sortie ansi qu'un message complet formatté contenant la position et la magnitude.

L'utilisation de `gsc_get_xy()` permet de récupérer les coordonnées X–Y du curseur en postition pixel, ainsi que les paramètres de l'équation de la projection plan–tangent. (voir src/subgsc/coord.f). Exemple :

```
struct gop_connect *connect_gsc;
float              x, y, ax, bx, ay, by, x0, y0, a0, d0;
int                ok;

if (gsc_client(connect_gsc, actual_status, option, delay, repeat)
              != GOP_OK){...}
...
if(gsc_plot(connect_gsc) != GOP_OK) {...}
...
if(gsc_get_xy(connect_gsc, &x, &y, &ax, &bx, &ay, &by,
              &x0, &y0, &a0, &d0, &ok) != GOP_OK) {...}
```

1.5 Recherche des objets centraux

La fonction `gsc_multi_get()` retourne les coordonnées écran, les magnitudes et les distance au centre (en pixels) des `"n"` objets les plus proche du centre du champ, triés selon leur distance au centre. Exemple :

```
program xxxx

integer    connect_gsc
integer    n, size
parameter  (n=100)
integer    x(n), y(n)
real      mag(n), dist(n)
integer    status
...


call gsc_multi_get(connect, n, size, x, y, mag, dist, status)
if(status.ne.0) ...
....
end
```

1.6 Recherche d'une coordonnée selon la position écran

La fonction `gsc_get_adm()` retourne alpha et delta en degré d'arc ainsi qu'en `DegMinSec` et `HeuMinSec` dans `calpha et cdelta`, la magnitude ansi qu'un message complet formatté contenant la position et la magnitude d'une position écran. Exemple :

```
struct gop_connect *connect_gsc;
int        ix, iy;
float               alpha, delta, magnitude;
char                calpha[20], cdelta[20];
char                message[80];
...
if (gsc_client(connect_gsc, actual_status, option, delay, repeat)
              != GOP_OK){...}
...
if(gsc_plot(connect_gsc) != GOP_OK) {...}
...
if(gsc_get_adm(connect_gsc, ix, iy, &alpha, &delta, calpha, cdelta,
               &magnitude, message) != GOP_OK) {...}
...
```

1.7 Position de l'image

La fenêtre image se positionne avec la fonction `gsc_position_image()`. Exemple :

```
program xxxx

integer    connect_gsc
integer    xpos, ypos

...
call gsc_position_image(connect_gsc, xpos, ypos, status)
if(status.ne.0) ...
...
end
```

1.8 Taille de l'image

On donne la taille de la fenêtre image se positionne avec la fonction `gsc_set_size()`. Exemple :

```
      program xxxx

      integer    connect_gsc
      integer    xsiz, ysiz

      ...
      call gsc_set_size(connect_gsc, xsiz, ysiz, status)
      if(status.ne.0) ...
      ...
      end
```

1.9 Ouverture et fermeture

La fonction `gsc_iconify()` met `xgsc` sous forme d'icone. La fonction `gsc_deiconify()` ouvre l'icone. La fonction `gsc_hide_image()` dépunaise l'image. La fonction `gsc_show_image()` faire réaparaître l'image dépunaisée. Exemple

```
main()
{
   struct gop_connect *connect_gsc;
   int    hscroll, vscroll;
   ...
   if(gsc_iconify(connect_gsc) != GOP_OK)
       {...}
   if(gsc_deiconify(connect_gsc) != GOP_OK)
       {...}
   if(gsc_hide_image(connect_gsc) != GOP_OK)
       {...}
   if(gsc_show_image(connect_gsc) != GOP_OK)
       {...}
   ...
}
```

1.10 Effacement

L'image s'efface avec la fonction `gsc_clear_image()`. Exemple :

```
program xxxx

integer    connect_gsc
...
call gsc_clear_image(connect_gsc, status)
if(status.ne.0) ...
...
end
```

1.11 Traitement des erreurs

Un status de retour différent de zéro indique une erreur. Le numéro d'une erreur de transmission est lut avec la fonction `gsc_get_error_number()`, le message avec `gsc_get_error_string()`. On peut également tester si l'erreur est due à une interruption de la communication (broken pipe) avec la fonction `gsc_is_broken_pipe()`. Exemple :

```
program xxxx

integer       connect_gsc
integer       pipe, no, ilen
character*80  message
...
call gsc_mget(connect_gsc, ...)
if(status.ne.0) then
  call gsc_is_broken_pipe(pipe)
  if(pipe)then
     write(*,*) deconnection avec xgsc
     actual_status = 0
     goto 8888
  else
     call gsc_get_error_number(no)
     call gsc_get_error_string(message, ilen)
     write(*,'(i,a)') no, message(1:ilen)
     goto 9999
  endif
endif
...
end
```

1.12 Verbosité de GOP

Le passage des messages entre le client et `xgsc` peut être visualisé avec la fonction `gsc_set_verbose()` en donnant un niveau de verbosité : 0, 1, 2 et 3 à 9 (voir libgop.a).
Exemple :

```
program xxxx

integer       connect_gsc
...
call gsc_set_verbose(connect_gsc, 2)
...
end
```