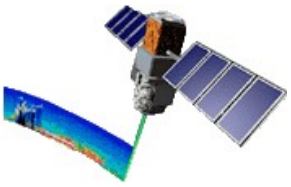


Introduction à l'utilisation des fichiers HDF

	Nom et Fonction	Date	Signature
Préparé par	**N. PASCAL**	**18/10/2005**	

DIFFUSION		
DIFFUSION INTERNE		**DIFFUSION EXTERNE**
CGTD		Libre

Historique et révisions	
18/10/05	v1.0 : Création

1 Installation de la librairie HDF

La procédure d'installation, le téléchargement des binaires, et la documentation sont disponibles ici : http://hdf.ncsa.uiuc.edu/hdf4.html

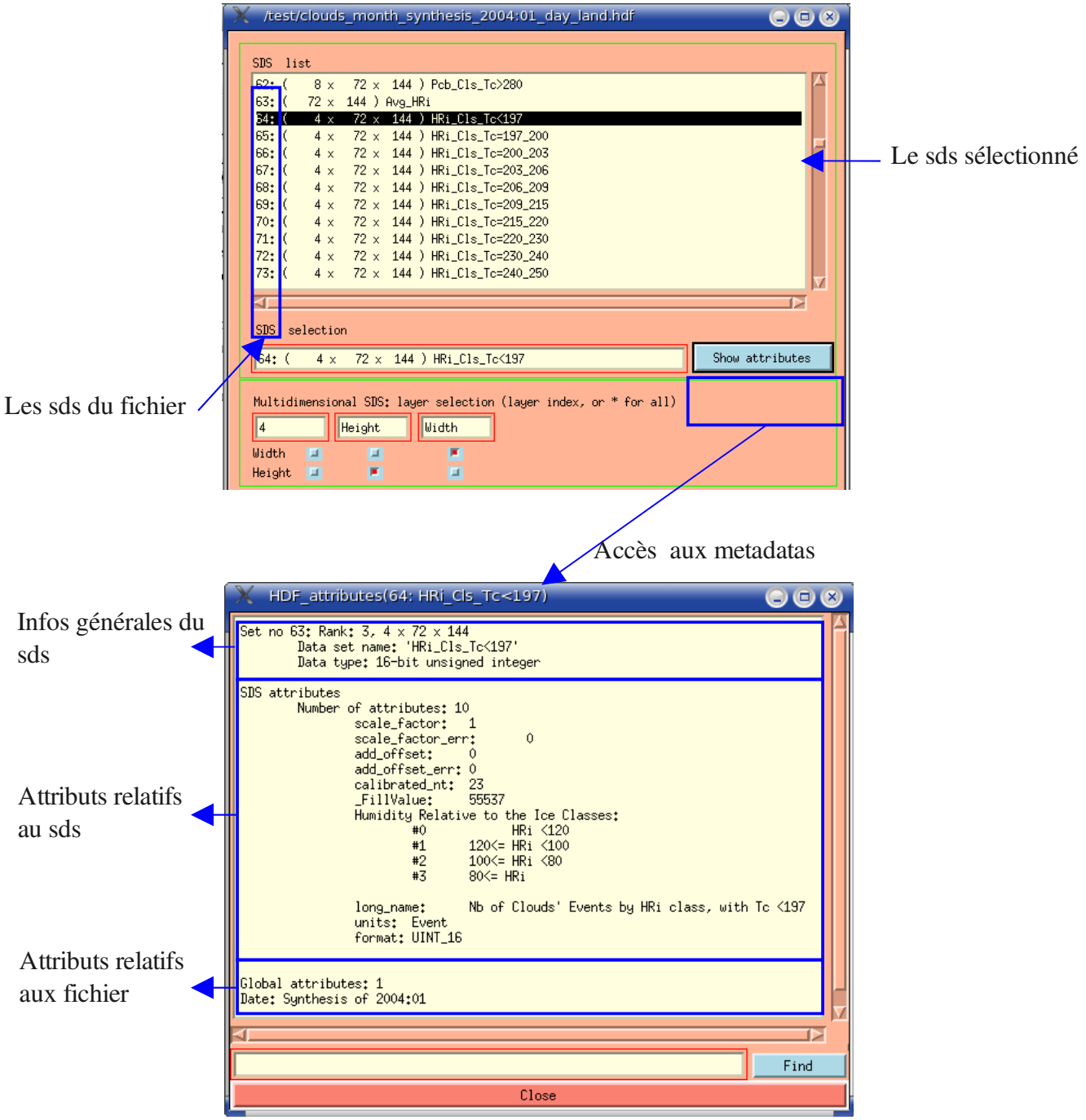
2 Quelques définitions :

sds : Un fichier HDF est composé de plusieurs tableaux de données. Chaque tableau est appelé un *sds,* acronyme de *Scientific Data Set.*

Metadata : Un sds est lui-même composé de données binaires, les datas, mais aussi d'informations sur les données, comme par exemple l'unité des données, les incertitudes, éventuellement la description de ce que représentent les données du sds... Ces « *données sur les données* » sont appelées metadatas.

Pour plus de détails sur ces concepts, on peut se référer à la documentation officielle en général, et au fichier *HDF42r0_UserGd.pdf* en particulier.

Graphiquement, cela donne :



3 Lecture des données HDF

Voici différentes routines pour lire les données d'un fichier HDF, que ce soit en C, en FORTRAN, ou en utilisant une librairie de plus haut niveau développée par le CGTD-ICARE.

3.1 en C :

```
#include "mfhdf.h"
#define FILE_NAME "my_file.hdf"
#define X_LENGTH 5
#define Y_LENGTH 16

main( ) {
  /*************************** Variable declaration ***************************/
  int32 sd_id, sds_id, sds_index;
  intn status;
  int32 start[2];
  int32 edges[2];
  int32 data[Y_LENGTH][X_LENGTH];
  int i, j;
  /********************** End of variable declaration **********************/
  /* Open the file for reading and initialize the SD interface. */
  sd_id = SDstart (FILE_NAME, DFACC_READ);
  /* Select the first data set. */
  sds_index = 0;
  sds_id = SDselect (sd_id, sds_index);
  /*Set elements of array start to 0, elements of array edges
  * to SDS dimensions,and use NULL for the argument stride in SDreaddata
  * to read the entire data.
  * BE CAREFUL : the start and edges parametres use this convention {...,Z,Y,X}
  */
  start[0] = 0; start[1] = 0; /* the start position of the datas to be read */
  edges[0] = Y_LENGTH; edges[1] = X_LENGTH; /* the size of the datas to be read */
  /*Read entire data into data array.
  * REM : the parametre that is NULL is the stride parametre :
  *          it defines the step between 2 read values along each direction.
  *          When set to NULL, all datas are read
  *          <=> the step is 1 along each direction
  *          <=> int32 stride[2]; stride[0]=1; stride[1]=1;
  */
  status = SDreaddata (sds_id, start, NULL, edges, (VOIDP)data);
  /* Print 10th row;
  the following numbers should be displayed. * * 10 1000 12 13 14 */
  for (j = 0; j < X_LENGTH; j++)
   printf ("%d ", data[9][j]);
  printf ("\n");
  /* Terminate access to the data set. */
  status = SDendaccess (sds_id);
  /* Terminate access to the SD interface and close the file. */
  status = SDend (sd_id);
};
```

Quelques remarques sur ce code :

- la librairie HDF redéfinit un certain nombre de types C. Ainsi, int32 est un entier sur 32bits, VOIDP est un void*,
- Quand on lit les datas avec **Sdreaddata**, data doit être passé en VOIDP (void*) car on ne connaît pas à priori le type des données qui vont être lues dans le sds (cela peut être des entiers, des flottants...) Tout dépend de comment les données ont été définies quand elles ont été écrites,
- La plupart des méthodes renvoient un status pour tester si l'exécution s'est déroulée sans problème.

1.1 en FORTRAN

```
      program read_data
      implicit none
C
C Parameter declaration.
C
      character*7 FILE_NAME
      integer X_LENGTH, Y_LENGTH
      parameter (FILE_NAME = #my_file.hdf#,
     + X_LENGTH = 5,
     + Y_LENGTH = 16)
      integer DFACC_READ, DFNT_INT32
      parameter (DFACC_READ = 1,
     + DFNT_INT32 = 24)
C
C Function declaration.
C
      integer sfstart, sfselect, sfrdata, sfendacc, sfend
C
C**** Variable declaration ********************************************
C
      integer sd_id, sds_id, sds_index, status
      integer start(2), edges(2), stride(2)
      integer data(X_LENGTH, Y_LENGTH)
      integer j
C
C**** End of variable declaration *************************************
C
C
C Open the file and initialize the SD interface.
C
      sd_id = sfstart(FILE_NAME, DFACC_READ)
C
C Select the first data set.
C
      sds_index = 0
      sds_id = sfselect(sd_id, sds_index)
C
C Set elements of the array start to 0, elements of the array edges to
C SDS dimensions, and elements of the array stride to 1 to read the
C entire data.
C
      start(1) = 0
      start(2) = 0
      edges(1) = X_LENGTH
      edges(2) = Y_LENGTH
      stride(1) = 1
      stride(2) = 1
C
C Read entire data into data array. Note that sfrdata is used
```

```
C to read the numeric data. If the datas would be characters, the method sfrcdata should have been used
C
status = sfrdata(sds_id, start, stride, edges, data)
C
C Print 10th column; the following numbers are displayed:
C
C 10 1000 12 13 14
C
write(*,*) (data(j,10), j = 1, X_LENGTH)
C
C Terminate access to the data set.
C
status = sfendacc(sds_id)
C
C Terminate access to the SD interface and close the file.
C
status = sfend(sd_id)
end
```

1.2 avec la librairie HDFFileData

Cette librairie offre 2 façons de lire les données contenues dans un fichier HDF : une qui renvoie un « vector » de la STL plus orienté c++, et une qui permet de remplir un buffer statique avec les valeurs lues plus orienté C.

1.2.1 style C++

```
#include "mfhdf.h"
#include <string>
#include <vector>
#include "HDFFileData.h"

#define FILE_NAME "my_file.hdf"
#define SDS_NAME "my_sds"
#define X_LENGTH 5
#define Y_LENGTH 16

int main(int argc, char *argv[]) {
   string filename(FILE_NAME);
   /* Open and create the access to the file  */
   HDFFileData hfd(filename);

   /* Define the limits  of the datas selection */
   int32 start[] = {0,0};
   int32 edges[] = {Y_LENGTH,X_LENGTH};
   int32 rank = 2; // number of dimensions of the sds (and so, length of start and edges)

   /* Read the datas and put it in a 2D vector
    * This time, the sds is accessed via its name, not via its number.
   */
   vector < vector<int32> > *data = hfd.get_value_2D<int32>(SDS_NAME,start,NULL,edges,rank);
  /* Print 10th row */
  for (j = 0; j < X_LENGTH; j++)
   printf ("%d ", (*data)[9][j]);
  printf ("\n");
  /* The vector must be deleted by the caller, because it has been allocated dynamically by the get_value_2D
      method.
  */
   delete v;
   return 0;
}
```

1.2.2 style C

```
#include "mfhdf.h"
#include "HDFFileData.h"

#define FILE_NAME "my_file.hdf"
#define SDS_NAME "my_sds"
#define X_LENGTH 5
#define Y_LENGTH 16

main() {
   /* Open and create the access to the file  */
   HDFFileData hfd(FILE_NAME);

   /* Define the limits  of the datas selection */
   int32 start[] = {0,0};
   int32 edges[] = {Y_LENGTH,X_LENGTH};
   int32 rank = 2; // number of dimensions of the sds (and so, length of start and edges)

   /* Read the datas and put it in a 2D array
    * This time, the sds is accessed via its name, not via its number.
   */
   int32 data[Y_LENGTH][X_LENGTH]; // the datas of the read sds are supposed to have a int32 type.
   hfd.read_data((VOIDP)data,SDS_NAME,start,NULL,edges,rank);

   /* Print the datas row by row*/
   for (int j = 0; j < Y_LENGTH ; j++) {
      for (int i = 0; i < X_LENGTH; i++)
         printf ("%e ", data[j][i]);
      printf ("\n");
   }
   return 0;
}
```

2 Références :

Vous trouverez plus de détails, et des possibilités que je n'ai pas abordées (écriture d'un fichier HDF...) dans les documents suivants

2.1 Documentation officielle

- HDF42r0_UserGd.pdf : C'est surtout le chapitre 3 qui contient les informations intéressantes. Vous y trouverez toutes les possibilités d'utilisation d'un sds, sous forme de **tutoriel**, avec des **exemples de code**. Une bonne partie des codes ci-dessus sont issus de ce document.
- HDF42r0_RefMan.pdf : Fonctions par fonctions, contient le détail de tout ce qui est disponible dans la librairie HDF. Plutôt pour un utilisateur avancé, ou pour des **informations sur une méthode précise**.
- hdf_spec_and_developper_guide.pdf : **Concepts** et **détails d'implémentation**. Plutôt à l'usage des informaticiens. Il ne devrait pas contenir d'informations qui vous intéressent, mais je l'ai ajouté au cas où, et pour que la documentation soit complète.

1.1 Outils et Liens :

- HDFLook : outil de visualisation des fichiers HDF, développé au LOA. Assez complet. Toute la documentation et les sources sont disponibles ici :
  http://www-loa.univ-lille1.fr/Hdflook/index.html
- hdfview : outil de visualisation de fichiers HDF en JAVA. Simple à installer. Très pratique pour visualiser les valeurs d'un sds. La documentation et le logiciel sont disponibles ici :
  http://hdf.ncsa.uiuc.edu/hdf-java-html/hdfview