GPM/TRMM
データ読み込みプログラムガイド
(C 言語編)

2019/01/28

第四版

本書は全球降雨観測衛星(GPM/TRMM)のデータを読み込むプログラム（C 言語）の作成方法についてまとめたものです。
本書で解説するサンプルプログラムは、GPM/TRMM はプロダクトバージョン０6、GSMaP はプロダクトバージョン０４で動作を確認しています。

目次

1. はじめに

本書は GPM/TRMM データに対して C 言語を用いて読み込む方法について解説します。
TRMM バージョン 8 相当プロダクトは、GPM バージョン 06 とフォーマットを統一し、GPM/TRMM バージョン 06 としてリリースされました。本サンプルプログラムにた同様に読むことができます。
GPM/TRMM データを読み込むにはＣ言語の他にも表 1.1 に示すような方法があります。どの方法で読み込むかについては、次頁の「読み込み方法判断フロー」を参考にして判断してください。
また、本資料で使用しているサンプルプログラムの動作を確認したＯＳの一覧を表 1.2 に示します。

表 1.1 データ読み込み方法

	データ読み込み方法	資料名	備考
1	THOR を使用する	GPM/TRMM データ読み込みプログラムガイド(THOR 編)	
2	IDL を使用する	GPM/TRMM データ読み込みプログラムガイド(IDL 編)	
3	C を使用する	GPM/TRMM データ読み込みプログラムガイド(C 言語編)	
4	FORTRAN を使用する	GPM/TRMM データ読み込みプログラムガイド(FORTRAN 編)	
5	Python を使用する	GPM/TRMM データ読み込みプログラムガイド(Python 編)	

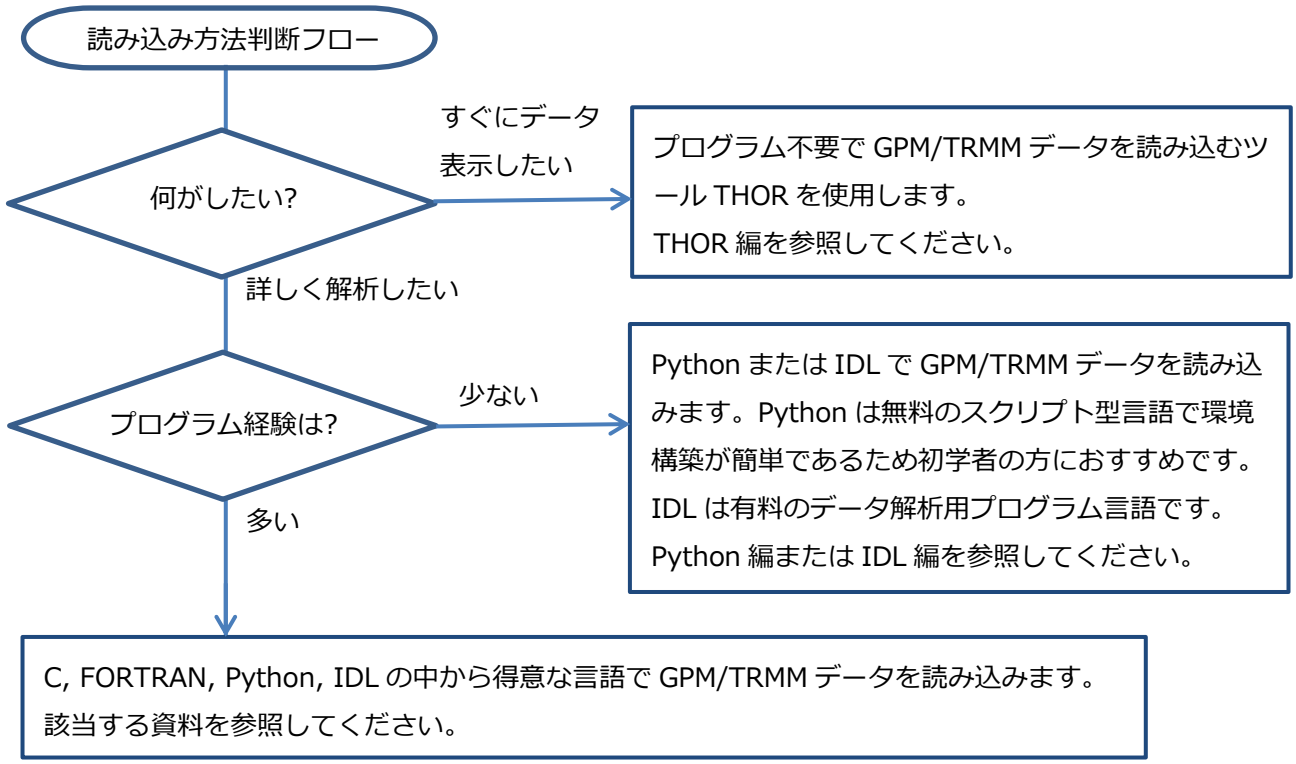
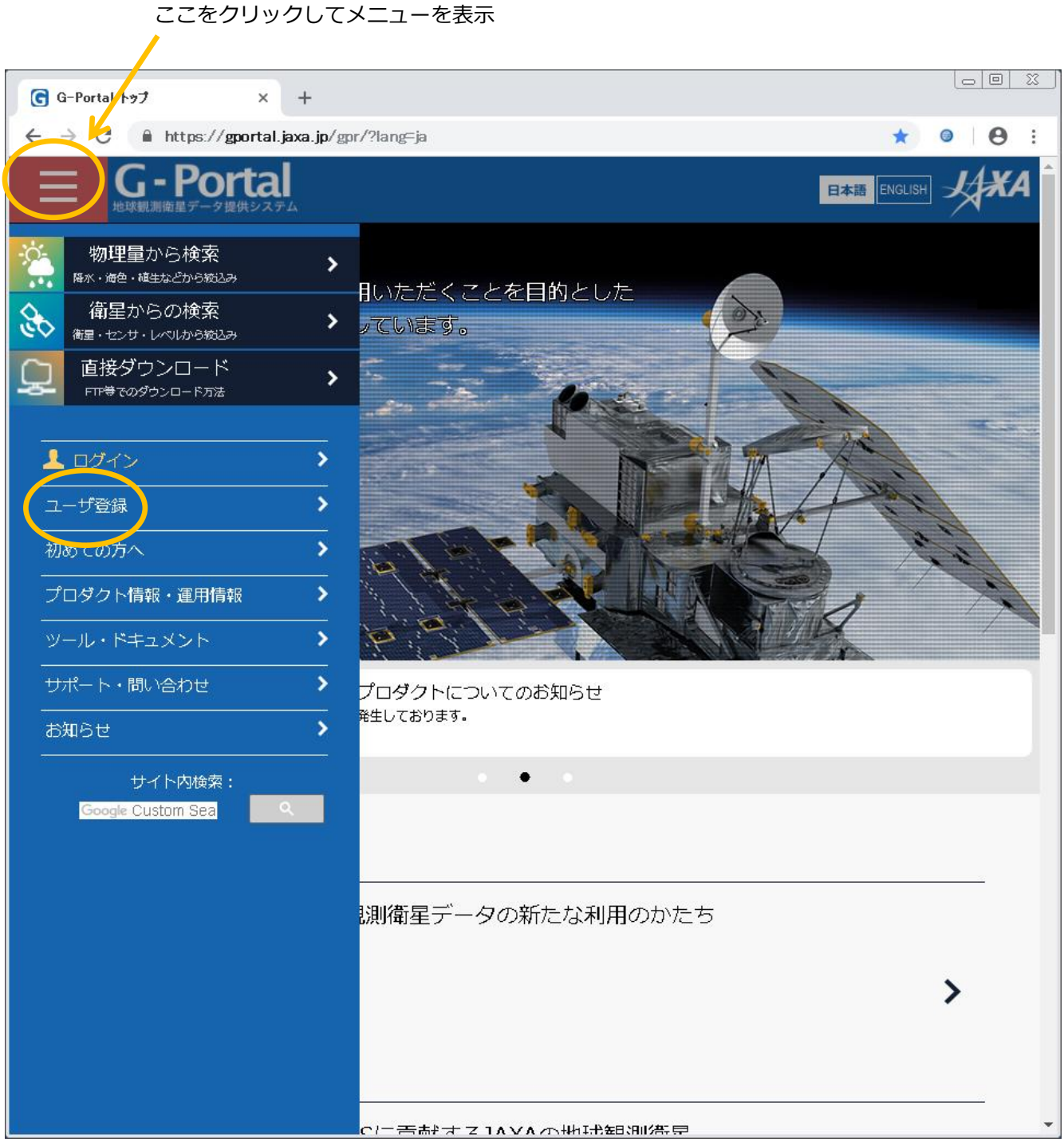


表 1.2 サンプルプログラム動作確認表

	サンプルプログラム	Linux	Windows	備考
1	C	○	—	
2	Fortran	○	—	
3	Python	○	○	
4	IDL	○	○	

2. GPM/TRMM データの入手方法

GPM/TRMM データは、G-Portal のサイト(https://www.gportal.jaxa.jp/gp/top.html)から取得することができます。 取得の際にはユーザ登録が必要になりますので、G-Portal のサイトのメニューから「ユーザ登録」を選択してユーザ登録を行ってください。



規約を読み「同意して次へ」をクリックします。

ユーザ登録画面になりますので、ユーザ登録を行います。

以降の手順や、ユーザ登録後のデータ取得方法については、「GPM データ利用ハンドブック」の「5.2 データ提供サービスの使い方」を参照してください。「GPM データ利用ハンドブック」の入手方法については「3. 関連文書、サンプルプログラムの入手方法」を参照してください。

3. 関連文書、サンプルプログラムの入手方法

GPM/TRMM データの関連文書には、データ利用に関する文書と、プロダクトに関する文書があります。どちらも全球降水観測計画 GPM のサイト(https://www.eorc.jaxa.jp/GPM/index.html)のトップページ ＞ 資料を読む ＞ その他 からダウンロードできます。また、本書で解説しているサンプルコードについてもこちらからダウンロードできます。

GPM データ利用に関する文書には以下のものがあります。

GPM データ利用ハンドブック

ファイル命名規約

「TRMM/GPM V06」をクリックするとプロダクトバージョン 06 の文書一覧が表示されます。Format Specification は各プロダクトのデータ仕様が記載されたドキュメントです。

本書で解説するプロダクトとプログラム、サンプルデータは以下の通りです。

表 3.1 サンプルプログラム一覧

プロダクト	サンプルプログラム	サンプルデータ
L1Ku	sample_L1_Ku_C.c	GPMCOR_KUR_1708202148_2320_019762_1BS_DUB_05A.h5
L2DPR	sample_L2_DPR_C.c	GPMCOR_DPR_1512282046_2218_010412_L2S_DD2_06A.h5
	sample_HDF5_L2_DPR_C.c	
	sample_h5dump_L2_C.c	
L3DPR	sample_L3_DPR_C.c	GPMCOR_DPR_1407_M_L3S_D3M_06A.h5
	sample_HDF5_L3_DPR_C.c	
	sample_h5dump_L3_C.c	
L2GMI	sample_L2_GMI_C.c	GPMCOR_GMI_1709252152_2324_020322_L2S_GL2_05A.h5
L3GMI	sample_L3_GMI_C.c	GPMCOR_GMI_1707_M_L3S_GL3_05A.h5
L2CMB	sample_L2_CMB_C.c	GPMCOR_CMB_1901082158_2331_027633_L2S_CL2_06A.h5
L3CMB	sample_L3_CMB_C.c	GPMCOR_CMB_1812_M_L3S_CL3_06A.h5
L1PR	sample_L1_PR_C.c	GPMTRM_PR1_1503251901_2032_098882_1BS_PU1_8b21.h5
L2PR	sample_L2_PR_C.c	GPMTRM_PR1_1503251901_2032_098882_L2S_PU2_8b21.h5
L3PR	sample_L3_PR_C.c	GPMTRM_KUR_1406_M_D3M_06A.h5
GSMaP	sample_GSMaP_HDF5_C.c	GPMMRG_MAP_1709242300_H_L3S_MCH_04D.h5
	sample_HDF5_GSMaP_C.c	

4. ライブラリ・ツールのインストール

C 言語で GPM データを読み込むには、表 4.1 で示すように 3 種類の方法があり、方法によってはツールをインストールする必要があります。本書ではそれぞれについてプログラム作成の解説を行います。

表 4.1 GPM データ読み込み方法

	GPM データ読み込み方法	必要なライブラリ、ツール	備考
1	PPS Toolkit(TKIO)	HDF5、PPS TKIO	表 4.2 参照
2	HDF5 ライブラリ	HDF5	
3	h5dump	HDF5	

確認したプロダクトバージョンと PPS ツールキットのバージョンは以下の通りです。

表 4.2 プロダクトバージョンと PPS Toolkit(TKIO)の対応バージョン

プロダクト	プロダクトバージョン	PPS ツールキットのバージョン	備考
L1Ku	05	3．80．44	
L2DPR	06	3．80．44	
L3DPR	06	3．80．44	
L2GMI	05	3．80．26	
L3GMI	05	3．80．26	
L2CMB	06	3．90．0	
L3CMB	06	3．90．0	
L1PR	05	3．80．25	
L2PR	06	3．80．25	
L3PR	06	3．80．25	
GSMaP	04	3．80．10	

注）PPS Toolkit(TKIO)は基本的には上位互換ですが、一部で正常に読み込めない場合があります。
その場合は「5.5 PPS Toolkit(TKIO)のバージョンについて」を参照してください。

本書のサンプルプログラムは以下の環境で動作確認を行っています。

表 4.3 動作環境

項目	環境
計算機	Intel(R) Xeon(R) CPU ES-2665　2.4GHz
OS	Red Hat Enterprise Linux Server release 6.4
C コンパイラ	gcc 4.4.7 icc 14.0.1
HDF5	Hdf5-1.8.9
PPS TKIO	tkio-x.xx.x

4.1　HDF5 のインストール

4.1.1 ダウンロード

The HDF Group ホームページ(http://www.hdfgroup.org/)から HDF5 のソースインストール版の圧縮ファイルをダウンロードします。

※以下では hdf5-1.8.9.tar.gz をダウンロードしたものとして説明します。

4.1.2 解凍

適当な作業ディレクトリで圧縮ファイルを解凍します。 以下のコマンドで解凍できます。

```
$ tar -xzvf hdf5-1.8.9.tar.gz
```

解凍すると、hdf5-1.8.9 のようなディレクトリが作成されるので、その配下へ移動します。

```
$ cd hdf5-1.8.9
```

4.1.3 コンパイルとインストール

以下のコマンドを順番に実行して、コンパイルとインストールを行います。

--prfix=には、インストール先ディレクトリを指定します。

※この例の場合、hdf5 のバージョンは 1.8.9 なので、hdf5_1.8.9 としています。

バージョン文字部分は実際に使用するバージョンに置き換えてください。

<HDF5 の FORTRAN ライブラリを使用しない場合>

```
$ ./configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.9
 --with-szlib=/home/user1/util/szip_2.1
$ make
$ make install
```

<HDF5 の FORTRAN ライブラリを使用する場合>

```
$ ./configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.9
 --with-szlib=/home/user1/util/szip_2.1 --enable-fortran FC=ifort
$ make
$ make install
```

4.2　PPS Toolkit(TKIO)のインストール

PPS Toolkit(TKIO)とは、GPM の HDF5 ファイルを読み込むプログラムを作成する際に使用するライブラリです。HDF5 ライブラリを使用して読み出す場合や、h5dump を使用して読み出す場合にはインストールする必要はありません。

4.2.1 ダウンロード

以下の URL から 、自分の環境に合った圧縮ファイルをダウンロードします。

https://gpmweb2https.pps.eosdis.nasa.gov/pub/PPStoolkit/GPM/

4.2.2 解凍

適当な作業ディレクトリを作成してダウンロードしたファイルを移し、圧縮ファイルを解凍します。以下のコマンドで解凍できます。

```
$ mkdir tikio.xxx
$ mv tikio.xxx.tar.gz tikio.xxx/.
$ cd tikio.xxx
$ tar zxf tikio.xxx.tar.gz
```

4.2.3 前提条件の確認

docs ディレクトリにある tkioINSTALL.txt ファイルを参照し、ダウンロードした PPS Toolkit(TKIO)が動作する前提条件を確認します。必要なライブラリがインストールされていない場合や、バージョンが古い場合はインストールを行います。

・libxml2 ライブラリのインストール

```
必要なバージョンは docs/tkioINSTALL.txt を確認！
./configure --prefix=[インストール DIR]
make
make install
```

・zlib ライブラリのインストール

```
./configure --prefix=[インストール DIR]
make
make install
```

・jpeg ライブラリのインストール

```
./configure --prefix=[インストール DIR] --enable-shared
make
make install
make install-lib
```

・hdf4 ライブラリのインストール

```
./configure --prefix=[インストール DIR] --with-zlib=[zlib インストール DIR]
--with-jpeg=[jpeg インストール DIR] --with-szlib=[szlib インストール DIR] CC=icc F77=ifort CXX=icpc
make
make install
```

・hdf5 ライブラリのインストール

```
./configure --prefix=[インストール DIR] --enable-fortran --with-zlib=[zlib インストール DIR]
--with-szlib=[szip インストール DIR] CC=icc CXX=icpc FC=ifort
make
make install
```

4.2.4 環境設定ファイルの編集

環境変数を定義するファイルを作成します。以下に作成例を示します。自分の環境に合った環境変数を定義してください。

```
 1:unlimit
 2:setenv TKDEBUG "-g"
 3:
 4:setenv TKIO /home/tool/tkio-x.xx.x_HDF4/tkio
 5:setenv HDF_INC /export/trmm5/tool/x86_64/HDF4.2r1/include
 6:setenv HDF_LIB /export/trmm5/tool/x86_64/HDF4.2r1/lib
 7:setenv HDF4_INC /export/trmm5/tool/x86_64/HDF4.2r1/include
 8:setenv HDF4_LIB /export/trmm5/tool/x86_64/HDF4.2r1/lib
 9:setenv HDF5_INC /export/trmm5/tool/x86_64/hdf5-1.8.9_gcc/include
10:setenv HDF5_LIB /export/trmm5/tool/x86_64/hdf5-1.8.9_gcc/lib
11:setenv CLASSPATH $TKIO/classes
12:
13:setenv SZIP_INC /home/tool/szip-2.1/include
14:setenv SZIP_LIB /home/tool/szip-2.1/lib
15:setenv xml2 /usr/include/libxml2
16:
17:setenv LD_LIBRARY_PATH ${HDF5_LIB}:${LD_LIBRARY_PATH}
18:
19:setenv CC icc
20:setenv CFLAGS '-fPIC -mcmodel=medium'
21:setenv CXXFLAGS '-fPIC -mcmodel=medium'
22:setenv FFLAGS '-fPIC -mcmodel=medium'
23:setenv FC ifort
24:setenv F77 ifort
25:setenv F90 ifort
26:setenv FORTC ifort
27:
28:setenv PATH ./:/home/tool/hdf5-1.8.9/bin:$PATH
```

4.2.5 環境設定ファイルの読み込み

以下のコマンドで環境設定ファイルを読み込みます。

$ source 環境設定ファイル名

4.2.6 コンパイル

以下のコマンドでコンパイルを実行します。

$ ./INSTALL.pl compileJAVA

$ ./INSTALL.pl buildRW

$ ./INSTALL.pl compileRW

5. PPS Toolkit(TKIO) で GPM/TRMM データ読み込み

PPS Toollit(TKIO)を使用した C 言語プログラムの作成方法について説明します。PPS Toollit(TKIO)を使用する場合は、予め PPS Toollit(TKIO)をインストールしておく必要があります。

また、PPS Toollit(TKIO)を使用してプログラムを作成する場合、予めアルゴリズム ID を知っておく必要があります。アルゴリズム ID とはプロダクト（データの種類）毎にある ID で、HDF5 ファイルのファイルヘッダに格納されています。主なプロダクトのアルゴリズム ID と使用する TKIO ヘッダファイル例を以下に示します。なお、PPS Viewer THOR でファイルヘッダの情報を確認することもできます。

表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応

レベル	プロダクト	アルゴリズム ID	TKIO ヘッダファイル	備考
1	L1Ku	1BKu	TK_1BKu.h	
	L1PR	1BPR	TK_1BPR.h	
2	L2DPR	2ADPR	TK_2ADPR.h	
	L2GMI	2AGPROFGMI	TK_2AGPROFGMI.h	
	L2CMB	2BCMB	TK_2BCMB.h	
	L2PR	2APR	TK_L2APR.h	
3	L3DPR	3DPR	TK_3DPR.h	
	L3GMI	3GPROF	TK_3GPROF.h	
	L3CMB	3CMB	TK_3CMB.h	
	L3PR	3PR	TK_3PR.h	
	GSMaP	3GSMAPH	TK_3GSMAP.h	

プログラムを作成する際、読み込むデータにあわせて格納する領域を定義する必要があります。GPM/TRMM データは、スキャン、アングルビン、レンジビンという名称の次元で構成されています。このスキャン、アングルビン、レンジビンの関係については、「GPM/TRMM データ読み込みプログラムガイド(付録)」の「1.2 シーン定義」を参照してください。また、読み込むデータの構成については 「3. 関連文書、サンプルプログラムの入手方法」で示したサイトから「GPM/DPR TRMM/PR L1 プロダクトフォーマット説明書」/ 「GPM/DPR TRMM/PR L2/L3 プロダクトフォーマット説明書」をダウンロードして参照してください。

次項よりデータ読み込みプログラムの作成例を示します。

プログラムの説明は以下のように色分けしています。

赤字の解説はサンプルプログラムについて説明しています。

青字の解説は PPS Toolkit または衛星基礎知識について説明しています。

5.1 L1 データ読み込み

5.1.1 ソースプログラム

以下は L1Ku を読み込むプログラム例です。ジョブ名と、L1Ku の HDF5 ファイル名を引数として、引数として指定されたファイルから、日時情報、緯度経度情報、echoPower、noisePower というデータを読み込んでいます。

```
 1:#include "TKheaders.h"
 2:#include "TK_1BKu.h"
 3:

 4:#define NRAY 49   /* number of Anglebin in one scan */
 5:#define NBIN 260  /* number of Rangebin in one Anglebin */
 6:
 7:int main(int argc, char *argv[]){
 8:
 9:    /* declare Data struct */
10:    TKINFO granuleHandle1BKu;
11:    L1BKu_NS  L1BKu;
12:
13:    /* declare character */
14:    char job[256];
15:    char inputfile[256];
16:
17:    strcpy(job, argv[1]);
18:    strcpy(inputfile, argv[2]);
19:
20:    /* declare variables */
21:    int year, month, dayOfMon, hour, min;
22:    int i, j, k;
23:    int numOfScan;
24:    int status;
25:
26:    /* declare array */
27:    float lat[NRAY], lon[NRAY];
28:    short noisePower[NRAY], echoPower[NRAY][NBIN];
29:
30:    /* HDF file open */
31:    status = TKopen(inputfile, "1BKu", TKREAD, "HDF5", job, &granuleHandle1BKu,
1);
32:    if(status != TK_SUCCESS) {
33:        fprintf(stderr, "error:file open error[%s] input1BKu¥n", inputfile);
34:    }
35:
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。L1Ku はアルゴリズム ID が"1BKu"なので"TK_1BKu.h"をインクルードします。

granuleHandle1BKu は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L1BKu は HDF5 ファイルの構造体です。1 スキャン分のデータを格納します。

job は Tkopen で指定する引数に使用します。

読み込んだ 1 スキャン分のデータを格納する領域を定義しています。ファイル全体を格納する場合は＊スキャン数分必要です。

HDF5 ファイルのオープン
inputfile:HDF5 ファイル名(引数で指定した文字列)
1BKu:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
job:ジョブ名(引数で指定した文字列)
&granuleHandle1BKu : ファイルポインタ(TKINFO 構造体を指定)
１：ファイルの内部圧縮(1 を指定)

```
36:    /* META data read */
37:    status = TKgetMetaInt(&granuleHandle1BKu, "SwathHeader",
38:                          "NumberScansGranule", &numOfScan);
39:    if(status != TK_SUCCESS) {
40:         fprintf(stderr, "error:file read error[%s]¥n", inputfile);
41:    }
42:
43:    /* Scan data read */
44:    for (i=0; i<numOfScan; i++){
45:       status = TKreadScan( &granuleHandle1BKu, &L1BKu);
46:
47:       /* Scantime Read */
48:       year      = L1BKu.ScanTime.Year;
49:       month     = L1BKu.ScanTime.Month;
50:       dayOfMon  = L1BKu.ScanTime.DayOfMonth;
51:       hour      = L1BKu.ScanTime.Hour;
52:       min       = L1BKu.ScanTime.Minute;
53:
54:       for (j=0; j<NRAY; j++){
55:          /* Latitude and Longitude Read */
56:          lat[j]  = L1BKu.Latitude[j];
57:          lon[j]  = L1BKu.Longitude[j];
58:
59:           /* noisePower Read */
60:           noisePower[j] = L1BKu.Receiver.noisePower[j];
61:
62:           /* echoPower Read */
63:           for (k=0; k<NBIN; k++){
64:               echoPower[j][k] = L1BKu.Receiver.echoPower[j][k];
65:           }
66:
67:           /* Print the value */
68:           if(i == 3946 && j == 19){
69:              printf("Scan=%d, Angle=%d¥n", i, j);
70:              printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
71:              printf("echoPower[259]= %d ¥n", echoPower[j][259]);
72:              printf("noisePower= %d ¥n", noisePower[j]);
73:           }
74:       }
75:    }
76:
77:    /* HDF File close*/
78:    status = TKclose(&granuleHandle1BKu);
79:
80:    return 0;
81:}
```

メタデータ読み込み（スキャン数読み出し）
&granuleHandle1BKu：ファイルポインタ(TKINFO 構造体を指定)
"SwathHeader"：HDF5 ファイルにある項目名で、読み出すデータの情報が格納されています。予め項目名を「L1 プロダクトフォーマット説明書」か PPS Viewer THOR で確認しておく必要があります。
"NumberScansGranule"：スキャン数を指定
&numOfScan：読み出したスキャン数を格納するアドレス

スキャン数分ループ

スキャンデータ読み込み
&granuleHandle1BKu:ファイルポインタ(TKINFO 構造体を指定)
&L1BKu：読み出したデータを格納するアドレス

スキャン日時の読み込み

緯度経度情報読み込み

noisePower 読み込み

echoPower 読み込み

正しく読み込めているか確認するため、一部分（このケースは、スキャンが 3947 番目のアングル 20、レンジビン 260 のデータ）を出力しています。

HDF ファイルのクローズ
&granuleHandle1BKu：ファイルポインタ(TKINFO 構造体を指定)

5.1.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_L1_Ku_C
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/ccode ¥
12:     -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:     -L$(HDF5_LIB) ¥
16:     -L$(TKIO)/lib ¥
17:     -L$(SZIP_LIB)
18:
19:LIBES = -ltkctkTSDIS -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:     -ltkchdf5algs -ltkchdf5 ¥
21:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
22:
23:$(MAIN): $(MAIN).o
24:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).c
27:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
28:
29:clean:
30:     rm -f *.o $(MAIN)
```

コンパイラにインテルコンパイラを使用する場合は、1,2 行目を使用し 4,5 行目はコメントとします。

5.1.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の C 言語用のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.1.3 実行結果

5.1.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_L1_Ku_C "cc" "GPMCOR_KUR_1708202148_2320_019762_1BS_DUB_05A.h5"
Scan=3946, Angle=19
lat=64.885056, lon=-160.390869
echoPower[259]= -29999
noisePower= -11076
$
```

第 1 引数の"cc"はジョブ名です。(任意の文字列で構いません)

第 2 引数は HDF5 ファイル名です。

5.2 L2 データ読み込み

5.2.1 ソースプログラム

以下は L2DPR を読み込むプログラム例です。ジョブ名と、L2DPR の HDF5 ファイル名を引数として、引数として指定されたファイルから、日時情報、緯度経度情報、precipRateESurface、zFactorCorrected というデータを読み込んでいます。

```
 1:#include "TKheaders.h"
 2:#include "TK_2ADPR.h"
 3:
 4:#define NRAY 49   /* number of Anglebin in one scan */
 5:#define NBIN 176  /* number of Rangebin in one Anglebin */
 6:
 7:int main(int argc, char *argv[]){
 8:
 9:    /* declare Data struct */
10:    TKINFO granuleHandle2ADPR;
11:    L2ADPR_SWATHS  L2ADPR;
12:
13:    /* declare character */
14:    char job[256];
15:    char inputfile[256];
16:
17:    strcpy(job, argv[1]);
18:    strcpy(inputfile, argv[2]);
19:
20:
21:    /* declare variables */
22:    int year, month, date, dayOfMon, hour, min;
23:    int i, j, k;
24:    int numOfScan;
25:    int status;
26:
27:    /* declare array */
28:    float lat[NRAY], lon[NRAY], precipESurf[NRAY];
29:    float zFactorCor[NRAY][NBIN];
30:
31:    /* HDF file open */
32:    status = TKopen(inputfile, "2ADPR", TKREAD, "HDF5", job, &granuleHandle2ADPR, 1);
33:    if(status != TK_SUCCESS) {
34:        fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
35:    }
36:
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。L2DPR はアルゴリズム ID が"2ADPR"なので"TK_2ADPR.h"をインクルードします。

granuleHandle2ADPR は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L2ADPR は HDF5 ファイルの構造体です。1 スキャン分のデータを格納します。

job は Tkopen で指定する引数に使用します。

読み込んだ 1 スキャン分のデータを格納する領域を定義しています。ファイル全体を格納する場合は＊スキャン数分必要です。

HDF5 ファイルのオープン
inputfile:HDF5 ファイル名(引数で指定した文字列)
2ADPR:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
job:ジョブ名(引数で指定した文字列)
&granuleHandle2ADPR：ファイルポインタ(TKINFO 構造体を指定)
1 ：ファイルの内部圧縮(1 を指定)

メタデータ読み込み（スキャン数読み出し）
&granuleHandle2ADPR：ファイルポインタ(TKINFO 構造体を指定)
"NS_SwathHeader"：HDF5 ファイルにある項目名で、読み出すデータの情報が格納されています。予め項目名を「L1 プロダクトフォーマット説明書」か PPS Viewer THOR で確認しておく必要があります。
"NumberScansGranule”：スキャン数を指定
&numOfScan：読み出したスキャン数を格納するアドレス

```
37:    /* META data read */
38:    status = TKgetMetaInt(&granuleHandle2ADPR, ”NS_SwathHeader”,
39:                          ”NumberScansGranule”, &numOfScan);
40:    if(status != TK_SUCCESS) {
41:        fprintf(stderr, ”error:file read error[%s]¥n”, inputfile);
42:    }
43:
44:    /* Scan data read */
45:    for (i=0; i<numOfScan; i++){
46:       status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);
47:
48:       /* Scantime Read */
49:       year      = L2ADPR.NS.ScanTime.Year;
50:       month     = L2ADPR.NS.ScanTime.Month;
51:       dayOfMon  = L2ADPR.NS.ScanTime.DayOfMonth;
52:       hour      = L2ADPR.NS.ScanTime.Hour;
53:       min       = L2ADPR.NS.ScanTime.Minute;
54:
55:       for (j=0; j<NRAY; j++){
56:          /* Latitude and Longitude Read */
57:          lat[j]  = L2ADPR.NS.Latitude[j];
58:          lon[j]  = L2ADPR.NS.Longitude[j];
59:
60:          /* precipRateESurface Read */
61:          precipESurf[j] = L2ADPR.NS.SLV.precipRateESurface[j];
62:          for (k=0; k<NBIN; k++){
63:             zFactorCor[j][k] = L2ADPR.NS.SLV.zFactorCorrected[j][k];
64:          }
65:
66:          /* Print the value */
67:          if(i == 3048 && j == 10){
68:             printf(”Scan=%d, Angle=%d¥n”, i, j);
69:             printf(”lat=%f, lon=%f¥n”, lat[j], lon[j]);
70:             printf(”precipESurf= %f (mm/hr)¥n”, precipESurf[j]);
71:          }
72:       }
73:    }
74:
75:    /* HDF File close*/
76:    status = TKclose(&granuleHandle2ADPR);
77:    return 0;
78:}
```

スキャン数分ループ

スキャンデータ読み込み
&granuleHandle2ADPR：ファイルポインタ(TKINFO 構造体を指定)
&L2ADPR：読み出したデータを格納するアドレス

スキャン日時の読み込み

緯度経度情報読み込み

precipRateESurface 読み込み

zFactorCorrected 読み込み

正しく読み込めているか確認するため、一部分（このケースは、スキャンが 3049 番目のアングル 11 のデータ）を出力しています。

HDF ファイルのクローズ
&granuleHandle2ADPR：ファイルポインタ(TKINFO 構造体を指定)

5.2.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:cc=icc
 2:CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:#cc=gcc
 5:#CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_L2_DPR_C
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/ccode ¥
12:     -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:     -L$(HDF5_LIB) ¥
16:     -L$(TKIO)/lib ¥
17:     -L$(SZIP_LIB)
18:
19:LIBES = -ltkctkTSDIS -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:     -ltkchdf5algs -ltkchdf5 ¥
21:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
22:
23:$(MAIN): $(MAIN).o
24:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).c
27:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
28:
29:clean:
30:     rm -f *.o $(MAIN)
```

コンパイラに GNU コンパイラを使用する場合は、4,5 行目を使用し 1,2 行目はコメントとします。

5.2.1 のプログラムの名前です。

HDF4/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の C 言語用のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.2.3 実行結果

5.2.1 で説明したプログラムの実行結果を示します。

第 1 引数の”cc”はジョブ名です。(任意の文字列で構いません)

第 2 引数は HDF5 ファイル名です。

```
$ ./sample_L2_DPR_C ”cc” ”GPMCOR_DPR_1512282046_2218_010412_L2S_DD2_06A.h5”
Scan=3048, Angle=10
lat=43.165447, lon=-88.955727
precipESurf= 20.550385 (mm/hr)
$
```

5.3 L3 データ読み込み

5.3.1 ソースプログラム

以下は L3DPR 読み込みプログラム例です。ジョブ名と、L3DPR の HDF5 ファイル名を引数として、引数として指定されたファイルから、precipRateESurface.mean というデータを読み込んでいます。

```
 1:#include "TKheaders.h"
 2:#include "TK_3DPR.h"
 3:
 4:#define LTL 28  /* number of low resolution 5° grid intervals of latitude from 70°S to 70°N.
*/
 5:#define LNL 72  /* number of low resolution 5° grid intervals of longitude from 180°W to 180°E.
*/
 6:#define CHN  5  /* number of channels: KuNS, KaMS, KaHS, DPRMS, KuMS. */
 7:#define RT   3  /* number of rain types: stratiform, convective, all */
 8:#define ST   3  /* number of surface types: ocean, land, all */
 9:
10:int main(int argc, char *argv[]){
11:
12:    /* declare Data struct */
13:    TKINFO granuleHandle3DPR;
14:    L3DPR_GRIDS  L3DPR;
15:
16:    /* declare character */
17:    char job[256];
18:    char inputfile[256];
19:
20:    strcpy(job, argv[1]);
21:    strcpy(inputfile, argv[2]);
22:
23:    /* declare variables */
24:    int i, j, k, l, m, n;
25:    int status;
26:
27:    /* declare array */
28:    float  precipESurf[ST]RT][CHN][LHL][LTL];
29:
30:    /* HDF file open */
31:    status = TKopen(inputfile, "3DPR", TKREAD, "HDF5", job, &granuleHandle3DPR, 1);
32:    if(status != TK_SUCCESS) {
33:        fprintf(stderr, "error:file open error[%s] input3DPR¥n", inputfile);
34:    }
35:
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。L3DPR はアルゴリズム ID が"3DPR"なので"TK_3DPR.h"をインクルードします。

granuleHandle2BCMB は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L3DPR は HDF5 ファイルの構造体です。1 スキャン分のデータを格納します。

job は Tkopen で指定する引数に使用します。

読み込んだデータを格納する領域を定義しています。

HDF5 ファイルのオープン
inputfile:HDF5 ファイル名(引数で指定した文字列)
3DPR:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
job:ジョブ名(引数で指定した文字列)
&granuleHandle2BCMB：ファイルポインタ(TKINFO 構造体を指定)
1：ファイルの内部圧縮(1 を指定)

```
 36:    /* Grid data read */
 37:    status = TKreadGrid( &granuleHandle3DPR, &L3DPR);
 38:
 39:    /* precipRateESurface.mean data read */
 40:    for (j=0; j<ST; j++){
 41:       for (k=0; k<RT; k++){
 42:          for (l=0; l<CHN; l++){
 43:             for (m=0; m<LNL; m++){
 44:                for (n=0; n<LTL; n++){
 45:                   precipESurf[j][k][l][m][n] =
L3DPR.G1.precipRateESurface.mean[j][k][l][m][n];
 46:                   if( m==63 && n==14 && l==4 )
 47:                      printf("precipRateESurface.mean[%d][%d][%d][%d][%d}= %f (mm/hr)¥n",
 48:                          j, k, l, m, n, precipESurf[j][k][l][m][n]);
 49:                }
 50:             }
 51:          }
 52:       }
 53:    }
 54:
 55:    /* HDF File close*/
 56:    status = TKclose(&granuleHandle3DPR);
 57:
 58:    return 0;
 59:}
```

グリッドデータ読み込み
&granuleHandle3DPR：ファイルポインタ(TKINFO 構造体を指定)
&L3DPR：読み出したデータを格納するアドレス

precipRateESurface.mean 読み込み

正しく読み込めているか確認するため、一部分（このケースは、経度グリッド間隔が 64 番目、緯度グリッド間隔が 15 番目で、チャネルが 5 番目のデータ）を出力しています。

HDF ファイルのクローズ
&granuleHandle3DPR：ファイルポインタ(TKINFO 構造体を指定)

5.3.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:cc=icc
 2:CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:#cc=gcc
 5:#CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_L3_DPR_C
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/ccode ¥
12:     -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:     -L$(HDF5_LIB) ¥
16:     -L$(TKIO)/lib ¥
17:     -L$(SZIP_LIB)
18:
19:LIBES = -ltkctkTSDIS -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:     -ltkchdf5algs -ltkchdf5 ¥
21:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
22:
23:$(MAIN): $(MAIN).o
24:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).c
27:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
28:
29:clean:
30:     rm -f *.o $(MAIN)
```

コンパイラに GNU コンパイラを使用する場合は、4,5 行目を使用し 1,2 行目はコメントとします。

5.5.1 のプログラムの名前です。

HDF4/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の C 言語用のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.3.3 実行結果

5.3.1 で説明したプログラムの実行結果を示します。

第 1 引数の”cc”はジョブ名です。(任意の文字列で構いません)

第 2 引数は HDF5 ファイル名です。

```
$ ./sample_L3_DPR_C "cc" "GPMCOR_DPR_1407_M_D3M_06A.h5"
precipRateESurface.mean[0][0][4][63][14]= 3.844850 (mm/hr)
precipRateESurface.mean[0][1][4][63][14]= 32.549240 (mm/hr)
precipRateESurface.mean[0][2][4][63][14]= 7.556628 (mm/hr)
precipRateESurface.mean[1][0][4][63][14]= 0.000000 (mm/hr)
precipRateESurface.mean[1][1][4][63][14]= 0.000000 (mm/hr)
precipRateESurface.mean[1][2][4][63][14]= 0.000000 (mm/hr)
precipRateESurface.mean[2][0][4][63][14]= 3.844850 (mm/hr)
precipRateESurface.mean[2][1][4][63][14]= 32.549240 (mm/hr)
precipRateESurface.mean[2][2][4][63][14]= 7.556628 (mm/hr)
$
```

5.4 GSMaP_HDF5 データ読み込み

5.4.1 ソースプログラム

以下は GSMaP HDF5 読み込みプログラム例です。ジョブ名と、GSMaP の HDF5 ファイル名を引数として、引数として指定されたファイルから、hourlyPrecipRateGC というデータを読み込んでいます。

```
 1:#include "TKheaders.h"
 2:#include "TK_3GSMAPH.h"
 3:int main(int argc, char *argv[]){
 4:
 5:    /* declare Data struct */
 6:    TKINFO granuleHandle3GSMAPH;
 7:    L3GSMAPH_GRID  L3GSMAPH;
 8:
 9:    /* declare character */
10:    char job[256];
11:    char inputfile[256];
12:
13:    strcpy(job, argv[1]);
14:    strcpy(inputfile, argv[2]);
15:
16:    /* declare variables */
17:    int nlon, nlat;
18:    int status;
19:
20:    /* declare array */
21:    float  hourlyPrecipRateGC[3600][1800];
22:

23:    /* HDF file open */
24:    status = TKopen(inputfile, "3GSMAPH", TKREAD, "HDF5", job,
&granuleHandle3GSMAPH, 1);
25:    if(status != TK_SUCCESS) {
26:        fprintf(stderr, "error:file open error[%s] input3GSMAPH¥n", inputfile);
27:    }
28:

29:    /* Grid data read */
30:    status = TKreadGrid( &granuleHandle3GSMAPH, &L3GSMAPH);
31:
32:    /* surfacePrecipitation data read */
33:    for (nlon=0; nlon<3600; nlon++){
34:      for (nlat=0; nlat<1800; nlat++){
35:        hourlyPrecipRateGC[nlon][nlat] =
L3GSMAPH.hourlyPrecipRateGC[nlon][nlat];
36:        if( nlon==3599 && nlat==1799 ) printf("hourlyPrecipRateGC[%d][%d]= %f
(mm/hr)¥n",
37:                                    nlon, nlat, hourlyPrecipRateGC[nlon][nlat]);

38:      }
39:    }
40:
41:    /* HDF File close*/
42:    status = TKclose(&granuleHandle3GSMAPH);
43:    return 0;
44:}
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。GSMaP はアルゴリズム ID が" 3GSMAPH"なので"TK_ 3GSMAPH.h"をインクルードします。

granuleHandle3GSMAPH は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L3GSMAPH は HDF5 ファイルの構造体です。1 スキャン分のデータを格納します。

job は Tkopen で指定する引数に使用します。

読み込んだデータを格納する領域を定義しています。

HDF5 ファイルのオープン
inputfile:HDF5 ファイル名(引数で指定した文字列)
3GSMAPH:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
job:ジョブ名(引数で指定した文字列)
&granuleHandle3GSMAPH : ファイルポインタ(TKINFO 構造体を指定)
1 ：ファイルの内部圧縮(1 を指定)

グリッドデータ読み込み
&granuleHandle3GSMAPH : ファイルポインタ(TKINFO 構造体を指定)
&L3GPROF: 読み出したデータを格納するアドレス

hourlyPrecipRateGC 読み込み

正しく読み込めているか確認するため、一部分を出力しています。

HDF ファイルのクローズ
&granuleHandle3GSMAPH : ファイルポインタ(TKINFO 構造体を指定)

5.4.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_GSMaP_HDF5_C
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/ccode ¥
12:     -I$(xml2) ¥
13:     -I$(SZIP_INC)
14:
15:LIB = -L$(HDF4_LIB) ¥
16:     -L$(HDF5_LIB) ¥
17:     -L$(TKIO)/lib ¥
18:     -L/usr/lib64 ¥
19:     -L$(SZIP_LIB)
20:
21:LIBES = -ltkctkTSDIS -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
22:     -ltkchdf5algs -ltkchdf5 ¥
23:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
24:
25:$(MAIN): $(MAIN).o
26:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
27:
28:$(MAIN).o: $(MAIN).c
29:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
30:
31:clean:
32:     rm -f *.o $(MAIN
```

コンパイラにインテルコンパイラを使用する場合は、1,2 行目を使用し 4,5 行目はコメントとします。

5.8.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の C 言語用のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.4.3 実行結果

5.4.1 で説明したプログラムの実行結果を示します。

第 1 引数の”hh”はジョブ名です。(任意の文字列で構いません)

第 2 引数の”/GSMaP/MCD…….h5”は HDF5 ファイル名です。

```
$ ./sample_GSMaP_HDF5_C "hh" "/GSMaP/MCD/STD/L3_MVK_CMB_HH/GPMMRG_MAP_sample.h5"
hourlyPrecipRateGC[3599][1799]= -9999.900391 (mm/hr)
$
```

5.5 PPS Toolkit(TKIO)のバージョンについて

インストールした PPS Toolkit(TKIO)のバージョンと、HDF5 ファイルを作成したバージョンが異なると正常に読み込めない場合があります。その場合、HDF5 ファイルのバージョンを調べ、バージョンに合わせたヘッダファイル、アルゴリズム ID にプログラムを変更する必要があります。

HDF5 ファイルのバージョンを調べるには PPS Viewer THOR を使用して HDF5 ファイルの FileInfo を読出し、「DataFormatVersion」と「TKCodeBuildVersion」の値を確認します。

DataFormatVersion=bk

TKCodeBuildVersion=2

の場合、バージョンは bk2 となります。

プログラムの変更は以下の 3 箇所です。

1) インクルードするヘッダファイル名
2) アルゴリズム ID
3) ヘッダファイルの参照箇所

ヘッダファイルとアルゴリズム ID の変更はバージョンの表記を追加します。ヘッダファイルが「TK_2ADPR.h」、アルゴリズム ID が「2ADPR」の場合、ヘッダファイルは「TK_2DPR_bk2.h」、アルゴリズム ID は「2ADPR_bk2」となります。

ヘッダファイルの変更に伴い、ヘッダファイルの内容を参照している箇所も変更後のヘッダファイルに合わせた内容に変更する必要があります。

以下に L2DPR データ読み込みサンプルプログラムの修正例を示します。

```
1:#include "TKheaders.h"
2:#include "TK_2ADPR.h"
3:


4:int main(int argc, char *argv[]){
5:
6:    /* declare Data struct */
7:    TKINFO granuleHandle2ADPR;
8:    L2ADPR_SWATHS L2ADPR;
9:
10:    /* declare character */
11:    char job[256];
12:    char inputfile[256];
13:
14:    strcpy(job, argv[1]);
15:    strcpy(inputfile, argv[2]);
16:
17:    /* declare variables */
18:    int year, month, date, dayOfMon, hour, min;
19:    int i, j, k;
20:    int numOfScan;
21:    int status;
```

インクルードするヘッダファイルで、バージョンに合わせて変更するのはこのファイルです。（TK_xxxx.h）
“TK_2ADPR_bk2.h”に変更します。

TK_2ADPR.h の内容を参照しているのはこの部分です。
“TK_2ADPR_bk2.h”の内容を調べて対応する定義の
“L2ADPR_SWATHS_bk2”に変更します。

```
 22:
 23:    /* declare array */
 24:    float lat[49], lon[49], precipESurf[49];
 25:    float zFactorCor[49][176];
 26:
 27:    /* HDF file open */
 28:    status = TKopen(inputfile, "2ADPR", TKREAD, "HDF5", job, &granuleHandle2ADPR,
1);
 29:    if(status != TK_SUCCESS) {
 30:       fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
 31:    }
 32:
 33:    /* META data read */
 34:    status = TKgetMetaInt(&granuleHandle2ADPR, "NS_SwathHeader",
 35:                          "NumberScansGranule", &numOfScan);
 36:    if(status != TK_SUCCESS) {
 37:        fprintf(stderr, "error:file read error[%s]¥n", inputfile);
 38:    }
 39:
 40:    /* Scan data read */
 41:    for (i=0; i<numOfScan; i++){
 42:      status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);
 43:
 44:      /* Scantime Read */
 45:      year      = L2ADPR.NS.ScanTime.Year;
 46:      month     = L2ADPR.NS.ScanTime.Month;
 47:      dayOfMon  = L2ADPR.NS.ScanTime.DayOfMonth;
 48:      hour      = L2ADPR.NS.ScanTime.Hour;
 49:      min       = L2ADPR.NS.ScanTime.Minute;
 50:
 51:      for (j=0; j<49; j++){
 52:           /* Latitude and Longitude Read */
 53:       lat[j]  = L2ADPR.NS.Latitude[j];
 54:       lon[j]  = L2ADPR.NS.Longitude[j];
 55:
 56:      /* precipRateESurface,zFactorCorrected Read */
 57:       precipESurf[j] = L2ADPR.NS.SLV.precipRateESurface[j];
 58:      for (k=0; k<176; k++){
 59:         zFactorCor[j][k] = L2ADPR.NS.SLV.zFactorCorrected[j][k];
 60:      }
 61:
 62:      /* Print the value */
 63:      if(i == 3946 && j == 19){
 64:         printf("Scan=%d, Angle=%d¥n", i, j);
 65:         printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
 66:         printf("precipESurf= %f (mm/hr)¥n", precipESurf[j]);
 67:      }
 68:     }
 69:    }
 70:
 71:    /* HDF File close*/
 72:    status = TKclose(&granuleHandle2ADPR);
 73:
 74:    return 0;
 75:}
```

アルゴリズム ID を指定しているのは、この部分です。バージョンを追加して“2ADPR_bk2”に変更します。

6. HDF ライブラリで GPM/TRMM データ読み込み

HDF ライブラリを使用した C プログラムの作成方法について説明します。

赤字の解説はサンプルプログラムについて説明しています。

青字の解説は HDF ライブラリまたは衛星基礎知識について説明しています。

6.1 L2DPR データ読み込み

6.1.1 ソースプログラム

以下のサンプルプログラムは、filename で指定されたファイルから、precipRateESurface というデータを読み込んでいます。

```
 1:#include <stdlib.h>
 2:#include <string.h>
 3:#include "hdf5.h"
 4:int main(){
 5:
 6:    hid_t  file_id, dataset_id;
 7:
 8:    /* declare character */
 9:    char inputfile[64]= "/DPR/STD/L2/2A.GPM.DPR.sample.HDF5";
10:    char dsetname[32]= "/NS/SLV/precipRateESurface";
11:
12:    /* declare variables */
13:    int ret;
14:
15:    /* declare array */
16:    float precipRateESurface[7932][49];
17:
18:    /* HDF file open */
19:    file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);
20:
21:    if(file_id < 0 ) {
22:        fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
23:    }
24:    else
```

HDF ライブラリ使用時は必ずインクルードします。

file_id、dataset_id は、それぞれファイルの識別子、データの識別子として使用します。

inputfile は読み込む HDF5 のファイル名を設定しています。

dsetname は HDF5 のファイルから読み出すデータ名を設定しています。

読み込んだデータを格納する領域を定義しています。

HDF5 ファイルオープン
inputfile：HDF5 ファイル名
H5F_ACC_RDONLY：アクセス指定（読込のみ）
H5P_DEFAULT：ファイルアクセスプロパティリスト（通常は H5P_DEFAULT を使用します）

オープンに失敗した場合、file_id にマイナスの値が設定されます。成功した場合、file_id は H5Dopen で使用します。

```
 25:    {
 26:      /* dataset open */
 27:      dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);
 28:      if(dataset_id < 0 ) {
 29:        fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
 30:      }
 31:      else
 32:      {
 33:        /* precipRateESurface read   */
 34:        ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);
 35:        if(ret < 0) {
 36:          fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
 37:        }
 38:        else
 39:        {
 40:          /* file,dataset print */
 41:          printf("file name= %s ¥n", inputfile);
 42:          printf("data set name= %s ¥n", dsetname);
 43:          /* precipRateESurface print */
 44:          printf("precipRateESurface[3763][9])= %f (mm/hr)¥n",
precipRateESurface[3763][9]);
 45:          printf("precipRateESurface[5635][10]= %f (mm/hr)¥n",
precipRateESurface[5635][10]);
 46:        }
 47:        /* detaset close   */
 48:        ret = H5Dclose(dataset_id);
 49:      }
 50:      /* HDF file close   */
 51:      ret = H5Fclose(file_id);
 52:    }
 53:    return 0;
 54:}
```

データセットのオープン
file_id：H5Fopen の戻り値を指定します。
dsetname：読み込むデータの名前です。
H5P_DEFAULT：ファイルアクセスプロパティリスト（通常は H5P_DEFAULT を使用します）

オープンに失敗した場合、dataset_id にマイナスの値が設定されます。成功した場合、file_id は H5Dread で使用します。

データ読み込み
dataset_id：H5Dopen の戻り値を指定します。
H5T_NATIVE_FLOAT：メモリタイプ ID(データの型を指定します)
H5S_ALL,：メモリスペース ID（H5S_ALL を指定します）
H5S_ALL, ：ファイルスペース ID
H5P_DEFAULT ：転送プロパティリスト（H5P_DEFAULT を指定）
precipRateESurface：読み込んだデータを格納する領域を指定します。

読み出しに失敗した場合、ret にマイナスの値が設定されます。

正しく読み込めているか確認するため、一部分を出力しています。

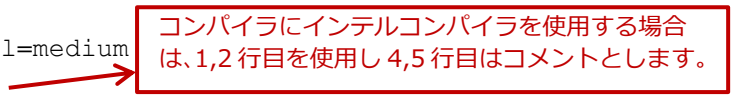
データセットのクローズ
dataset_id：H5Dopen で取得した値を指定します。

HDF5 ファイルのクローズ
file_id：H5Fopen で取得した値を指定します。

6.1.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_HDF5_L2_DPR_C
 8:
 9:INC = -I$(HDF5_INC) ¥
10:     -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥
13:     -L$(SZIP_LIB)
14:
15:LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:     rm -f *.o $(MAIN)
```



6.1.3 実行結果

6.1.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_HDF5_L2_DPR_C
file name= /DPR/STD/L2/2A.GPM.DPR.sample.HDF5
data set name= /NS/SLV/precipRateESurface
precipRateESurface[3763][9]= 2.790208 (mm/hr)
precipRateESurface[5635][10]= 2.090051 (mm/hr)
$
```

6.2 L3DPR データ読み込み

6.2.1 ソースプログラム

以下のサンプルプログラムは、filename で指定されたファイルから、precipRateESurface というデータを読み込んでいます。

```
 1:#include <stdlib.h>
 2:#include <string.h>
 3:#include "hdf5.h"
 4:int main(){
 5:
 6:    hid_t  file_id, dataset_id;
 7:
 8:    /* declare character */
 9:    char inputfile[64]= "/DPR/STD/L3/3A-MO.GPM.DPR.sample.HDF5";

10:    char dsetname[64]= "/Grids/G1/precipRateESurface/mean";
11:
12:    /* declare variables */
13:    int ret;
14:
15:    /* declare array */
16:    float precipRateESurface[3][3][5][72][28];
17:

18:    /* HDF file open */
19:    file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);

20:    if(file_id < 0 ) {
21:      fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
22:    }
23:    else{

24:      /* dataset open */
25:      dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);

26:      if(file_id < 0 ) {
27:        fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
28:      }
```

HDF ライブラリ使用時は必ずインクルードします。

file_id、dataset_id は、それぞれファイルの識別子、データの識別子として使用します。

inputfile は読み込む HDF5 のファイル名を設定しています。

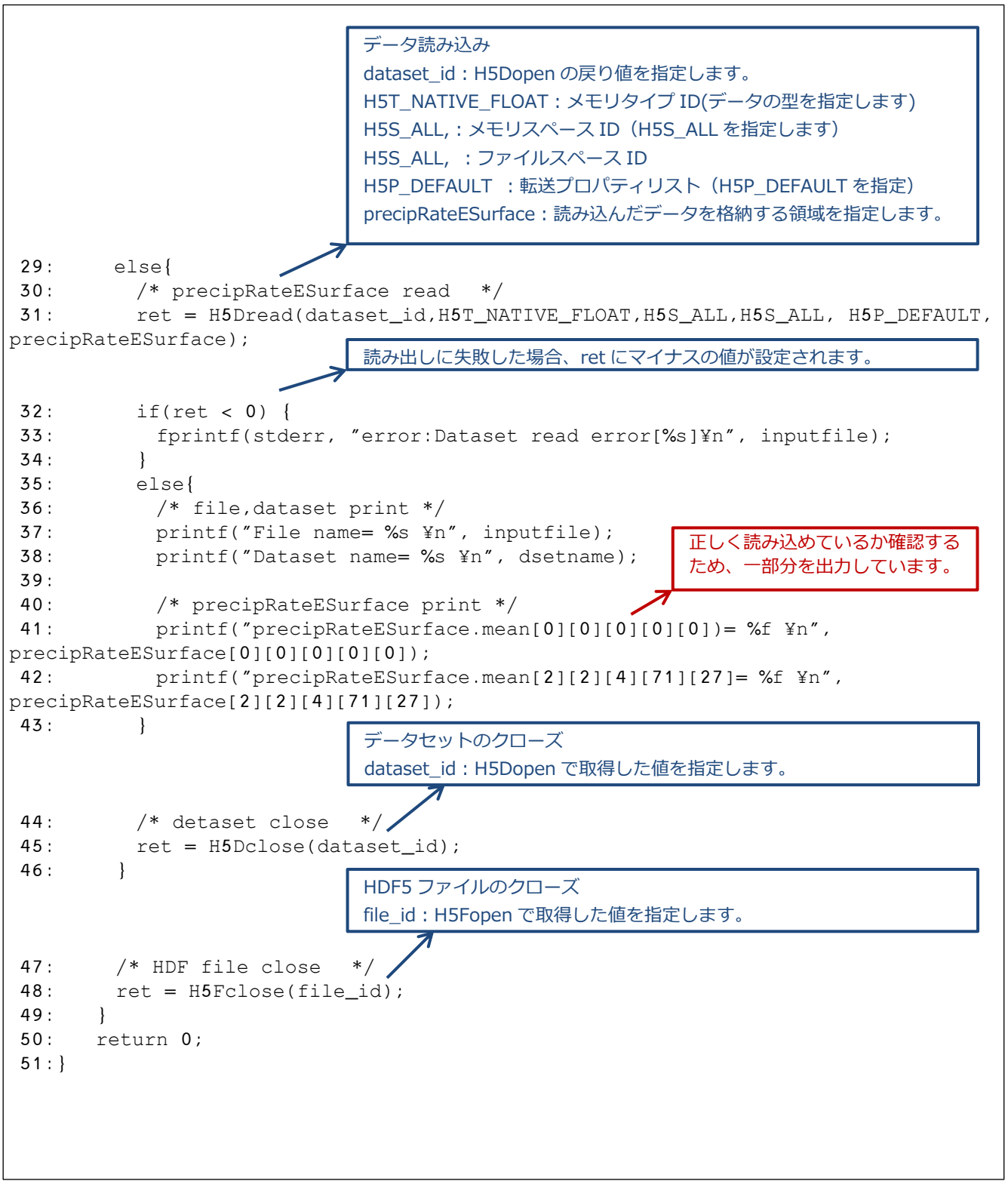
dsetname は HDF5 のファイルから読み出すデータ名を設定しています。

HDF5 ファイルオープン
inputfile：HDF5 ファイル名
H5F_ACC_RDONLY：アクセス指定（読込のみ）
H5P_DEFAULT：ファイルアクセスプロパティリスト（通常は H5P_DEFAULT を使用します）

オープンに失敗した場合、file_id にマイナスの値が設定されます。成功した場合、file_id は H5Dopen で使用します。

データセットのオープン
file_id：H5Fopen の戻り値を指定します。
dsetname：読み込むデータの名前です。
H5P_DEFAULT：ファイルアクセスプロパティリスト（通常は H5P_DEFAULT を使用します）

オープンに失敗した場合、dataset_id にマイナスの値が設定されます。成功した場合、file_id は H5Dread で使用します。

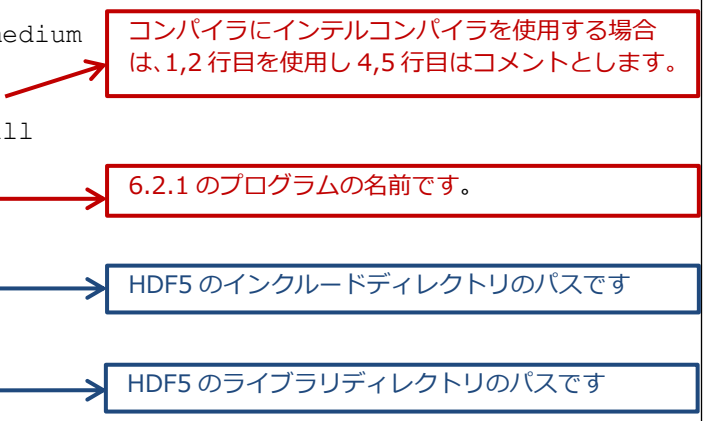



```
 29:     else{
 30:       /* precipRateESurface read   */
 31:       ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);
 32:       if(ret < 0) {
 33:         fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
 34:       }
 35:       else{
 36:         /* file,dataset print */
 37:         printf("File name= %s ¥n", inputfile);
 38:         printf("Dataset name= %s ¥n", dsetname);
 39:
 40:         /* precipRateESurface print */
 41:         printf("precipRateESurface.mean[0][0][0][0][0])= %f ¥n",
precipRateESurface[0][0][0][0][0]);
 42:         printf("precipRateESurface.mean[2][2][4][71][27]= %f ¥n",
precipRateESurface[2][2][4][71][27]);
 43:       }
 44:       /* detaset close   */
 45:       ret = H5Dclose(dataset_id);
 46:     }
 47:     /* HDF file close   */
 48:     ret = H5Fclose(file_id);
 49:   }
 50:   return 0;
 51:}
```

6.2.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_HDF5_L2_DPR_C
 8:
 9:INC = -I$(HDF5_INC) ¥
10:     -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥
13:     -L$(SZIP_LIB)
14:
15:LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:     rm -f *.o $(MAIN)
```



6.2.3 実行結果

6.2.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_HDF5_L3_DPR_C
file name= /DPR/STD/L3/3A-MO.GPM.DPR.sample.HDF5
data set name= /Grids/G1/precipRateESurface/mean
precipRateESurface.mean[0][0][0][0][0])= 0.488317
precipRateESurface.mean[2][2][4][71][27]= 1.369588
$
```

6.3 GSMaP_HDF5 データ読み込み

6.3.1 ソースプログラム

以下のサンプルプログラムは、inputfile で指定されたファイルから、hourlyPrecipRateGC というデータを読み込んでいます。

```
 1:#include <stdlib.h>
 2:#include <string.h>
 3:#include "hdf5.h"
 4:int main(){
 5:
 6:    hid_t  file_id, dataset_id;
 7:
 8:    /* declare character */
 9:    char inputfile[64]= "/GSMaP/MCD/STD/GPMMRG_MAP_sample.h5";

10:    char dsetname[64]= "/Grid/hourlyPrecipRateGC";
11:
12:    /* declare variables */
13:    int ret;
14:
15:    /* declare array */
16:    float hourlyPrecipRateGC[3600][1800];
17:

18:    /* HDF file open */
19:    file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);

20:    if(file_id < 0 ) {
21:        fprintf(stderr, "error:File open error[%s] ¥n", inputfile);
22:    }
23:    else{

24:      /* dataset open */
25:      dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);

26:      if(file_id < 0 ) {
27:        fprintf(stderr, "error:Dataset open error[%s] ¥n", inputfile);
28:      }
```

HDF ライブラリ使用時は必ずインクルードします。

file_id、dataset_id は、それぞれファイルの識別子、データの識別子として使用します。

inputfile は読み込む HDF5 のファイル名を設定しています。

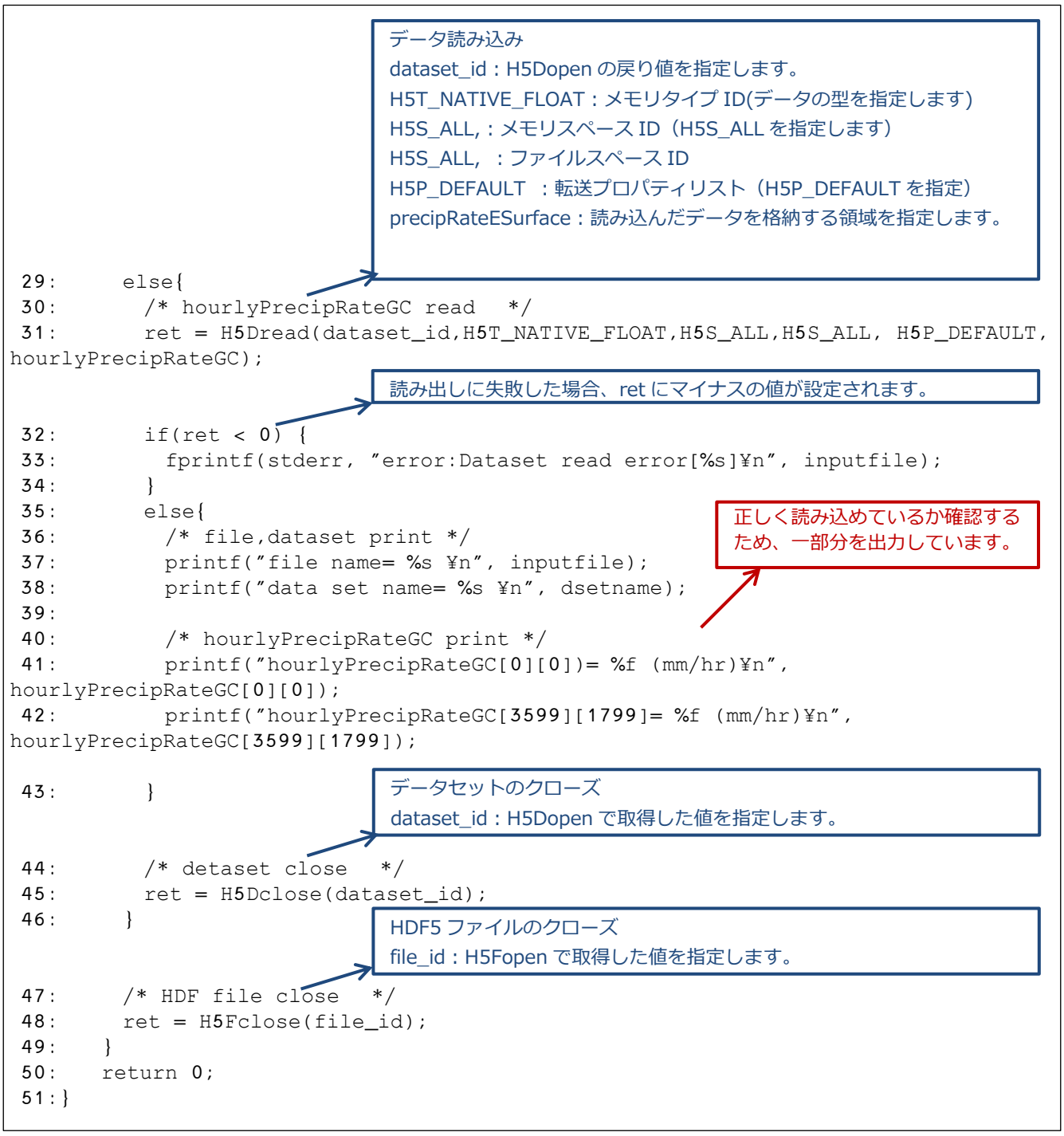
dsetname は HDF5 のファイルから読み出すデータ名を設定しています。

HDF5 ファイルオープン
inputfile：HDF5 ファイル名
H5F_ACC_RDONLY：アクセス指定（読込のみ）
H5P_DEFAULT：ファイルアクセスプロパティリスト（通常は H5P_DEFAULT を使用します）

オープンに失敗した場合、file_id にマイナスの値が設定されます。成功した場合、file_id は H5Dopen で使用します。

データセットのオープン
file_id：H5Fopen の戻り値を指定します。
dsetname：読み込むデータの名前です。
H5P_DEFAULT：ファイルアクセスプロパティリスト（通常は H5P_DEFAULT を使用します）

オープンに失敗した場合、dataset_id にマイナスの値が設定されます。成功した場合、file_id は H5Dread で使用します。

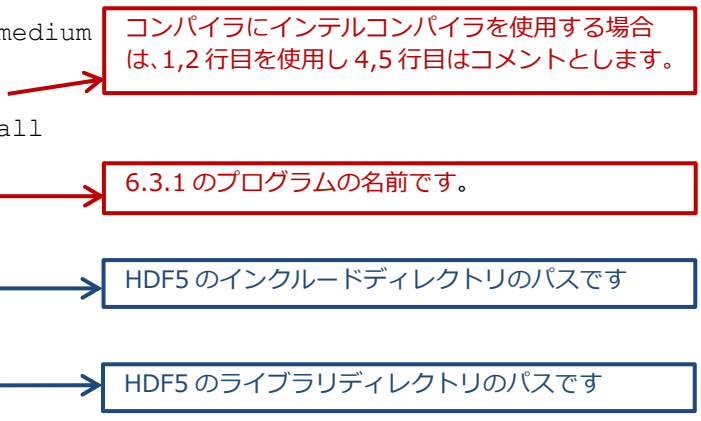



```
 29:      else{
 30:        /* hourlyPrecipRateGC read   */
 31:        ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
hourlyPrecipRateGC);
 32:        if(ret < 0) {
 33:          fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
 34:        }
 35:        else{
 36:          /* file,dataset print */
 37:          printf("file name= %s ¥n", inputfile);
 38:          printf("data set name= %s ¥n", dsetname);
 39:
 40:          /* hourlyPrecipRateGC print */
 41:          printf("hourlyPrecipRateGC[0][0])= %f (mm/hr)¥n",
hourlyPrecipRateGC[0][0]);
 42:          printf("hourlyPrecipRateGC[3599][1799]= %f (mm/hr)¥n",
hourlyPrecipRateGC[3599][1799]);
 43:        }
 44:        /* detaset close   */
 45:        ret = H5Dclose(dataset_id);
 46:      }
 47:      /* HDF file close   */
 48:      ret = H5Fclose(file_id);
 49:    }
 50:    return 0;
 51:}
```

6.3.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_HDF5_GSMaP_C
 8:
 9:INC = -I$(HDF5_INC) ¥
10:     -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥
13:     -L$(SZIP_LIB)
14:
15:LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:     $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:     $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:     rm -f *.o $(MAIN)
```



6.3.3 実行結果

6.3.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_HDF5_GSMaP_C
file name= /GSMaP/MCD/STD/GPMMRG_MAP_sample.h5
data set name= /Grid/hourlyPrecipRateGC
hourlyPrecipRateGC[0][0])= -9999.900391 (mm/hr)
hourlyPrecipRateGC[3599][1799]= -9999.900391 (mm/hr)
$
```

7. h5dump で GPM/TRMM データ読み込み

h5dump を使用して、HDF5 ファイルから読み込みたいデータのバイナリファイルを作成し、そのバイナリファイルを読み出す C プログラムの作成方法について説明します。

赤字の解説はサンプルプログラムについて説明しています。

青字の解説は HDF ライブラリまたは衛星基礎知識について説明しています。

7.1 L2 データ読み込み

7.1.1 バイナリファイルの作成

h5dump を使用してバイナリファイルの作成を行うシェルの作成例を示します。

```
  1:#!/bin/tcsh -f
  2:
  3:set h5dump_bin=/home/tool/hdf5-1.8.9/bin/h5dump
  4:set file_dir=/DPR/STD/L2/
  5:set OUTPUT=/h5dump_samplecode/L2/binfile/
  6:
  7:cd ${file_dir}
  8:set file_in=(`ls *008435* |sed 's/.HDF5/ /' `)
  9:mkdir -p ${OUTPUT}
 10:cd ${OUTPUT}
 11:
 12:foreach file ($file_in)
 13:echo ${file}
 14:$h5dump_bin -d NS/SLV/precipRateESurface -b -o ${file}.precipRateESurface
${file_dir}/${file}.HDF5
 15:end
```

h5dump のフルパスを指定しています。

HDF5 ファイルが存在するディレクトリを指定しています。

バイナリファイルの出力先のディレクトリを指定しています。

HDF5 ファイルが存在するディレクトリで処理するファイルを絞込む場合指定します。"*008435*"はファイル名に"008435"が含まれているファイルのみ対象とする意味です。

ファイル名の拡張子の部分を削除しています。

対象ファイル数分ループ

バイナリファイル作成
読み込んだファイルから-d で指定されたデータを、-b-o で指定されたファイル名でバイナリファイルを作成しています。

上記のシェルを実行すると以下のように表示され、OUTPUT で指定したディレクトリにバイナリファイルが作成されます。

```
$ ./dump_L2.sh
2A.GPM.DPR.sample.HDF5 "/DPR/STD/L2/2A.GPM.DPR.sample.HDF5" {
DATASET "NS/SLV/precipRateESurface" {
   DATATYPE  H5T_IEEE_F32LE
   DATASPACE  SIMPLE { ( 7932, 49 ) / ( H5S_UNLIMITED, 49 ) }
   DATA {
   }
}
}
```

7.1.2 ソースプログラム

以下のサンプルプログラムは、inputfile で指定されたバイナリファイルから情報を読み込んでいます。

```
  1:#include <stdio.h>
  2:#include <string.h>
  3:
  4:int main(){
  5:
  6:    /* declare variables */
  7:    float precipRateESurface[7932][49];
  8:    FILE  *fp;
  9:
 10:    /* binary file declare */
 11:    char inputfile[128]
="/h5dump_samplecode/L2/binfile/2A.GPM.DPR.sample.precipRateESurface";
 12:
 13:    /* binary file open */
 14:    fp = fopen(inputfile, "rb");
 15:    if(fp == NULL ) {
 16:      printf("error:file open error[%s] ¥n", inputfile);
 17:    }
 18:    else
 19:    {
 20:      /* file read */
 21:      fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);
 22:
 23:      /* precipRateESurface print */
 24:     printf("filename=%s¥n",inputfile);
 25:      printf("precipRateESurface[3763][9])= %f (mm/hr)¥n",
precipRateESurface[3763][9]);
 26:      printf("precipRateESurface[5635][10]= %f (mm/hr)¥n",
precipRateESurface[5635][10]);
 27:
 28:      /* binary file close   */
 29:      fclose(fp);
 30:    }
 31:    return 0;
 32:}
```

データを読み込む領域を定義しています。

7.1.1 で作成したバイナリファイルを指定しています。

バイナリファイルのオープン
2 番目の引数には"rb"(バイナリ)を指定してください。

バイナリファイルの読み込み
1 回で全データを precipRateESurface に読み込んでいます。

7.1.3 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_h5dump_L2_C
 8:
 9:INC = -I$(HDF5_INC) ¥
10:   -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥
13:   -L$(SZIP_LIB)
14:
15:LIBES =   -lm -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:   $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:   $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:   rm -f *.o $(MAIN)
```

7.1.4 実行結果

7.1.2 で説明したプログラムの実行結果を示します。

```
$ ./sample_h5dump_L2_C
filename=/h5dump_samplecode/L2/binfile/2A.GPM.DPR.sample.precipRateESurface
precipRateESurface[3763][9])= 2.790208 (mm/hr)
precipRateESurface[5635][10]= 2.090051 (mm/hr)
$
```

7.2 L3 データ読み込み

7.2.1 バイナリファイルの作成

h5dump を使用してバイナリファイルの作成を行うシェルの作成例を示します。

```
 1:#!/bin/tcsh -f
 2:
 3:set h5dump_bin=/home/tool/hdf5-1.8.9/bin/h5dump
 4:set file_dir=/DPR/STD/L3/
 5:set OUTPUT=/h5dump_samplecode/L3/binfile/
 6:
 7:cd ${file_dir}
 8:set file_in=(`ls *150801* |sed 's/.HDF5/ /' `)
 9:cd ${OUTPUT}
10:
11:foreach file ($file_in)
12:echo ${file}
13:$h5dump_bin -d /Grids/G1/precipRateESurface/mean -b -o
${file}.precipRateESurface_mean ${file_dir}/${file}.HDF5
14:end
```

h5dump のフルパスを指定しています。

HDF5 ファイルが存在するディレクトリを指定しています。

バイナリファイルの出力先のディレクトリを指定しています。

HDF5 ファイルが存在するディレクトリで処理するファイルを絞込む場合指定します。"*150801*"はファイル名に"150801"が含まれているファイルのみ対象とする意味です。

ファイル名の拡張子の部分を削除しています。

対象ファイル数分ループ

バイナリファイル作成
読み込んだファイルから-d で指定されたデータを、-b-o で指定されたファイル名でバイナリファイルを作成しています。

上記のシェルを実行すると以下のように表示され、OUTPUT で指定したディレクトリにバイナリファイルが作成されます。

```
$ ./dump_L3.sh
3A-MO.GPM.DPR.sample.HDF5 "/DPR/STD/L3/3A-MO.GPM.DPR.sample.HDF5" {
DATASET "/Grids/G1/precipRateESurface/mean" {
   DATATYPE  H5T_IEEE_F32LE
   DATASPACE  SIMPLE { ( 3, 3, 5, 72, 28 ) / ( 3, 3, 5, 72, 28 ) }
   DATA {
   }
}
}
$
```

7.2.2 ソースプログラム

以下のサンプルプログラムは、inputfile で指定されたバイナリファイルから情報を読み込んでいます。

```
  1:#include <stdio.h>
  2:#include <string.h>
  3:int main(){
  4:
  5:    /* declare variables */
  6:    float precipRateESurface[3][3][5][72][28];
  7:    FILE  *fp;
  8:    int   size;
  9:
 10:    /* binary file name */
 11:    char inputfile[128]
="/h5dump_samplecode/L3/binfile/3A-MO.GPM.DPR.sample.precipRateESurface_mean";
 12:
 13:    /* binary file open */
 14:    fp = fopen(inputfile, "rb");
 15:    if(fp == NULL ) {
 16:      printf("error:file open error[%s] ¥n", inputfile);
 17:    }
 18:    else
 19:    {
 20:      /* file read */
 21:      size = fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);
 22:
 23:      /* precipRateESurface print */
 24:      printf("filename=%s¥n",inputfile);
 25:      printf("precipRateESurface.mean[0][0][0][0][0]= %f ¥n",
precipRateESurface[0][0][0][0][0]);
 26:      printf("precipRateESurface.mean[2][2][4][71][27]= %f ¥n",
precipRateESurface[2][2][4][71][27]);
 27:
 28:      /* binary file close   */
 29:      fclose(fp);
 30:    }
 31:    return 0;
 32:}
```

データを読み込む領域を定義しています。

7.2.1 で作成したバイナリファイルを指定しています。

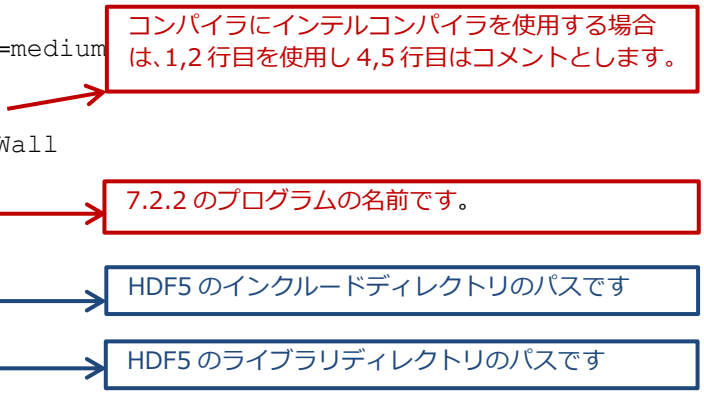
バイナリファイルのオープン
2 番目の引数には"rb"(バイナリ)を指定してください。

バイナリファイルの読み込み
1 回で全データを precipRateESurface に読み込んでいます。

7.2.3 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:#cc=icc
 2:#CFLAGS=-g -shared-intel -mcmodel=medium
 3:
 4:cc=gcc
 5:CFLAGS=-g -mcmodel=medium -fpic -Wall
 6:
 7:MAIN=./sample_h5dump_L3_C
 8:
 9:INC = -I$(HDF5_INC) ¥
10:     -I$(SZIP_INC)
11:LIB = -L$(HDF5_LIB) ¥
12:     -L$(SZIP_LIB)
13:
14:LIBES =   -lm -ljpeg -lz -lxml2
15:
16:$(MAIN): $(MAIN).o
17:   $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
18:
19:$(MAIN).o: $(MAIN).c
20:   $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
21:
22:clean:
23:   rm -f *.o $(MAIN)
```



7.2.4 実行結果

7.2.2 で説明したプログラムの実行結果を示します。

```
$ ./sample_h5dump_L3_C
read size=1
filename=/
h5dump_samplecode/L3/binfile/3A-MO.GPM.DPR.sample.precipRateESurface_mean
precipRateESurface.mean[0][0][0][0][0]= 0.488317
precipRateESurface.mean[2][2][4][71][27]= 1.369588
$
```

改版履歴

版数	日付	改版内容	備考
1	2016/1/26		
2	2017/9/13	1. はじめに : 表 1.1 に python の記載を追加、それに伴いフローチャート修正。 表 1.2 サンプルコード動作確認表を追加。 4. ライブラリ・ツールのインストール:表 4.2 プロダクトバージョンと PPS Toolkit(TKIO)の対応バージョンを追加。 表 4.3 動作環境の tkio-3.70.7 と記載している箇所を tkio-x.xx.x に変更 4.2.4 環境設定ファイルの編集: tkio-3.70.7 と記載している箇所を tkio-x.xx.x に変更。	
3	2018/3/15	3. 関連文書、サンプルプログラムの入手方法 : 表 3.1 サンプルプログラム一覧を追加	
4	2019/1/25	1.～3 . TRMM 追加及び GPM サイトリニューアルに伴う修正 ４. 表 4.2 プロダクトバージョンと TKIO 対応バージョンをプロダクト毎に記載するように修正 5. アルゴリズム ID の一覧表を追加、またサンプルプログラムはレベル毎に１つ記載するように変更	