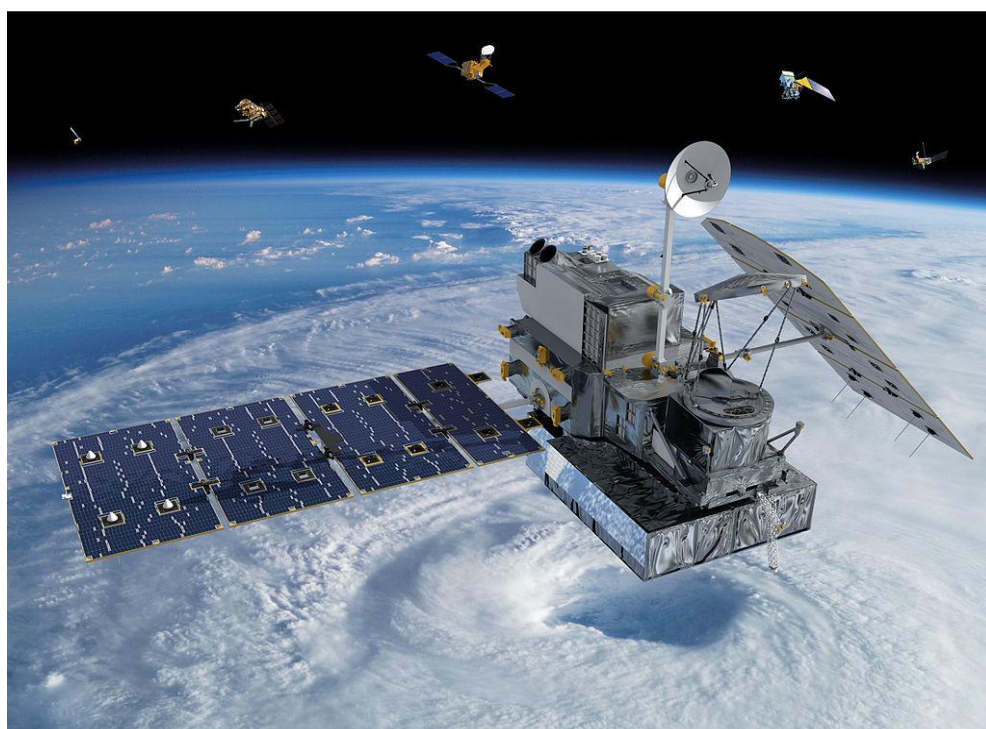


GPM/TRMM データ読み込みプログラムガイド (C 言語編)



2026/06/01

第七版

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 08、GSMaP はプロダクトバージョン 05 で動作を確認しています。

目次

1. はじめに.....	3
2. GPM/TRMM データの入手方法	5
3. 関連文書、サンプルプログラムの入手方法.....	8
4. ライブラリ・ツールのインストール.....	10
4.1 HDF5 のインストール	11
4.2 NetCDF のインストール.....	11
4.3 PPS Toolkit(TKIO)のインストール	11
5. PPS Toolkit(TKIO) で GPM/TRMM データ読み込み	14
5.1 L1 データ読み込み.....	16
5.2 L2 データ読み込み.....	19
5.3 L3 データ読み込み.....	22
5.4 PPS Toolkit(TKIO)のバージョンについて.....	25
6. HDF ライブラリで GPM/TRMM データ読み込み	27
6.1 L2DPR データ読み込み	27
6.2 L3DPR データ読み込み	30
7. h5dump で GPM/TRMM データ読み込み	33
7.1 L2 データ読み込み.....	33
7.2 L3 データ読み込み.....	36
8. 改訂履歴.....	39

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。

GPM 及び TRMM はバージョン 06 プロダクト (TRMM バージョン 8 相当) からフォーマットを統一しており、最新のアルゴリズムはバージョン 08 となっています。本サンプルプログラムにて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	QGIS や Panoply、 HDFView などの市販ツール (COTS) を使用する	GPM/TRMM データ読み込みプログラムガイド(COTS 編)	
3	GeoTIFF を使用する	GPM/TRMM データ読み込みプログラムガイド(GeoTIFF 変換編)	
4	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)	
5	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)	
6	FORTTRAN を使用する	GPM/TRMM データ読み込みプログラムガイド(FORTTRAN 編)	
7	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)	

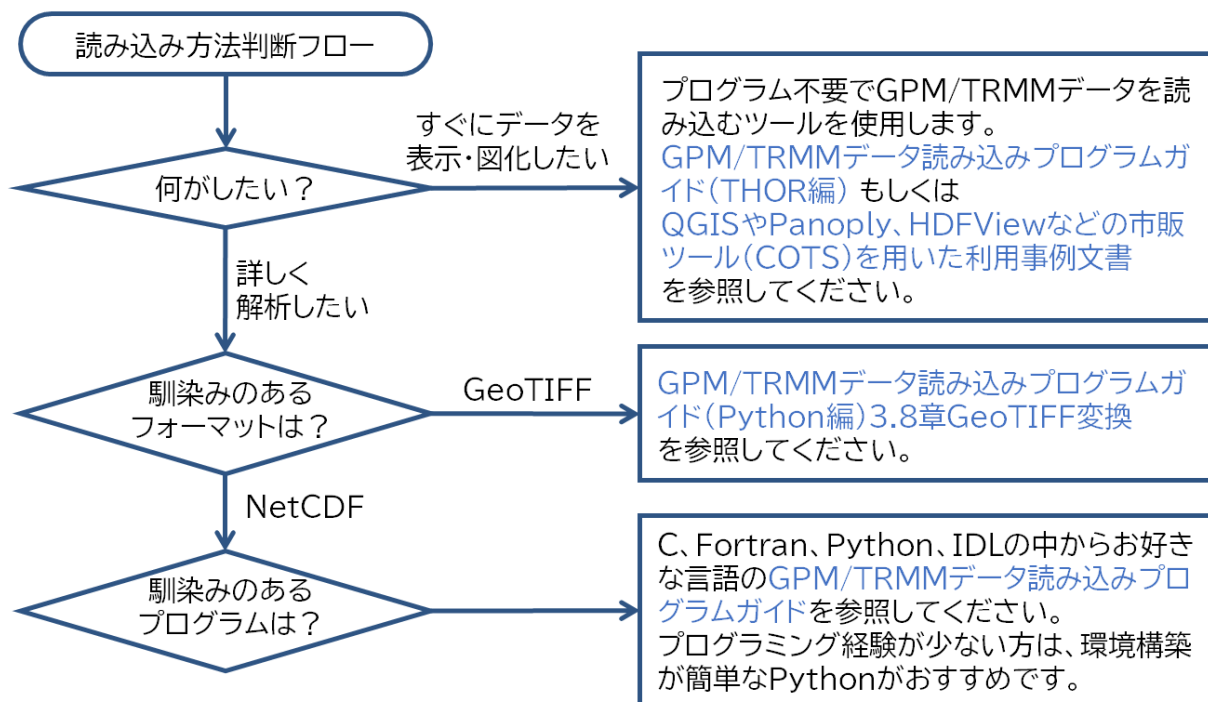


表 1.2 サンプルプログラム動作確認表

	サンプルプログラム	Linux	Windows	備考
1	C	○	—	
2	FORTTRAN	○	—	
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://gportal.jaxa.jp/gpr/>)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。

ここをクリックしてメニューを表示



規約を読み「同意して次へ」をクリックします。



G-Portal ユーザ登録

https://gportal.jaxa.jp/gpr/user/regist1

日本語 ENGLISH JAXA

1 2 3 4 5
利用規約 登録情報入力 登録内容確認 仮登録完了 本登録完了

ユーザ登録 STEP1/5: G-Portal 利用規約

G-Portalからプロダクトをダウンロードするには、ユーザー登録が必要です。以下のご利用規約を確認の上、次のステップへお進みください。

G-Portal

2. 個人情報保護および個人情報の取り扱い

JAXAは、ご登録いただいた個人情報（氏名、メールアドレス、所属機関、所属部署、国または地域名、利用目的）を、個人情報保護に関する法令、およびEU一般データ保護規則（General Data Protection Regulation : GDPR）を含むその他の規範、また機構にて別途定める「個人情報保護に関する規程」に則り、適切に取り扱います。詳細は [JAXA | 個人情報保護](#) をご確認ください。

JAXAは、ご登録いただいた個人情報をG-Portalに関する目的以外には使用いたしません。

(使用用途)

- サービス利用状況の把握
- G-Portalの向上を目的とするユーザ意向調査・アンケート・周知の実施
- ユーザからの問い合わせ対応

また、JAXAがG-Portalに係る業務の一部（システム管理、ユーザ管理、ヘルプデスク業務等）を委託する場合、委託業務に必要な範囲に限り、ご登録いただいた個人情報を受託者に利用させるものとします。

3. アカウントおよびパスワードの管理

ユーザアカウント、およびパスワードの管理・使用はユーザが全ての責任を持つものとし、第三者の不正使用等から生

上記の利用規約に同意する

同意して次へ 同意しません

ユーザ登録画面になりますので、ユーザ登録を行います。

G-Portal
地球観測衛星データ提供システム

日本語 ENGLISH JAXA

1 2 3 4 5
利用規約 登録情報入力 登録内容確認 仮登録完了 本登録完了

ユーザ登録 STEP2/5: G-Portal 登録情報入力

以下の項目を全てを入力し、「登録確認画面へ」ボタンを押してください。

ユーザアカウント (必須):

パスワード (必須) ①:

パスワード (確認) (必須):

氏名 (必須):

メールアドレス (必須) ①:

メールアドレス(確認) (必須):

所属機関:

所属部署:

国名:

メール使用言語 (必須) ①: 日本語 English

利用目的 (必須): データ解析 アルゴリズム開発 データ検証 応用研修 教育 校正 注文生産 その他

準備完了通知メールの受信設定 (必須) ①: オータ単位 準備完了単位

*メールアドレスの取扱い

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 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はトップページ > 観測データを使う 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

The screenshot shows the '資料を読む' (Read Materials) page for TRMM/GPM V08. The page title is 'TRMM/GPM Products (Version08)'. Below the title, there is a table listing various products and their versions. The table has columns for 'TRMM' and 'GPM' products, and their respective versions. The table content is as follows:

		TRMM		GPM
	PR/DPR L1B	V08	V08	2026/03/01-現在 V08
	PR/DPR L2/L3	V08	V08	2026/03/01-現在 V08
	SLH	V08	V08	2026/03/01-現在 V08
NASA	PR/DPR comb.(CSH)	V08	V08	---
	VIRS/TMI/GMI	V08	V08	---

Below the table, there is a section for 'データダウンロード' (Data Download) and 'データ利用' (Data Usage). The 'データ利用' section includes links to the 'データ利用ハンドブック' (Data Usage Handbook), 'プロダクトに関する文書はこちら' (Documents related to products), 'プロダクトに関する論文はこちら' (Papers related to products), 'GSMaP利用ガイドライン (PDF)' (GSMaP Usage Guidelines (PDF)), and 'GSMaP利用ガイドライン スライド (ppt)' (GSMaP Usage Guidelines Slides (ppt)).

「TRMM/GPM V08」をクリックするとプロダクトバージョン 08 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_2604010211_0345_068594_1BS_DUB_08A.nc
L2DPR	sample_L2_DPR_C.c	GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
	sample_HDF5_L2_DPR_C.c	
	sample_h5dump_L2_C.c	
L3DPR	sample_L3_DPR_C.c	GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
	sample_HDF5_L3_DPR_C.c	
	sample_h5dump_L3_C.c	
L2GMI	sample_L2_GMI_C.c	GPMCOR_GMI_2604010211_0345_068594_L2S_GL2_08A.nc
L3GMI	sample_L3_GMI_C.c	GPMCOR_GMI_2603_M_L3S_GL3_08A.nc
L2SLP	sample_L2_SLP_C.c	GPMCOR_DPR_2604010211_0345_068594_L2S_SLP_08A.nc
L3SLM	sample_L3_SLM_C.c	GPMCOR_DPR_2603_M_L3S_SLM_08A.nc
L3SLG	sample_L3_SLG_C.c	GPMCOR_DPR_2604010211_0345_068594_L3S_SLG_08A.nc

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、NetCDF、PPS TKIO	表 4.2 参照
2	HDF5 ライブラリ	HDF5、NetCDF	
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	0 8	3. 1 0 2	
L2DPR	0 8	3. 1 0 2	
L3DPR	0 8	3. 1 0 2	
L2GMI	0 8	3. 1 0 2	
L3GMI	0 8	3. 1 0 2	
L2SLP	0 8	3. 1 0 2	
L3SLM	0 8	3. 1 0 2	
L3SLG	0 8	3. 1 0 2	

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「5.4 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 E5-2665 0 @ 2.40GHz
OS	AlmaLinux release 8.8
C コンパイラ	gcc (GCC) 8.5.0 20210514 (Red Hat 8.5.0-28) icx (ICX) Intel(R) oneAPI DPC++/C++ Compiler 2025.3.2 (2025.3.2.20260112)
HDF5	hdf5-1.10.5
NetCDF	netcdf-4.7.0
PPS TKIO	tkio-3.102

4.1 HDF5 のインストール

以下コマンドを実行し、インストールを行います。

```
$ dnf install hdf5 hdf5-devel
```

※HDF5 ライブラリが提供されているリポジトリが有効になっていることを、事前に確認してください。サンプルプログラムでは、EPEL リポジトリにて提供されているライブラリを使用しています。

4.2 NetCDF のインストール

以下コマンドを実行し、インストールを行います。

```
$ dnf install netcdf netcdf-devel
```

※NetCDF ライブラリが提供されているリポジトリが有効になっていることを、事前に確認してください。サンプルプログラムでは、EPEL リポジトリにて提供されているライブラリを使用しています。

4.3 PPS Toolkit(TKIO)のインストール

PPS Toolkit(TKIO)とは、GPM の NetCDF 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。

4.3.1 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/>

4.3.2 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 tkio.xxx
$ mv tkio.xxx.tar.gz tkio.xxx/
$ cd tkio.xxx
$ tar xzf tkio.xxx.tar.gz
```

4.3.3 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
```

4.3.4 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。

```

ulimit -s unlimited
export TKDEBUG="-g"

export TKIO=/home/user1/util/tkio-3.102/tkio_v8
export HDF5_INC=/usr/include
export HDF5_LIB=/usr/lib64
export NETCDF_INC=/usr/include
export NETCDF_LIB=/usr/lib64
export CLASSPATH=$TKIO/classes
export JPEG_INC=/usr/include
export JPEG_LIB=/usr/lib64
export ZLIB=/usr/lib64
export LIBXML2_INC=/usr/include/libxml2
export LIBXML2_LIB=/usr/lib64

export
LD_LIBRARY_PATH=/home/user1/util/jdk1.8.0_60/lib:${ZLIB}:/home/user1/util/libtool-2.4/lib:/home/user1/util/flex-2.5.39/lib:${LD_LIBRARY_PATH}

export
PATH=./:/home/user1/util/jdk1.8.0_60/bin:/home/user1/util/byacc-20150711/bin:/home/user1/util/libtool-2.4/bin:/home/user1/util/flex-2.5.39/bin:${PATH}

export CC=gcc
export CFLAGS='-fPIC -mmodel=medium'
export CXXFLAGS='-fPIC -mmodel=medium'
export FFLAGS='-fPIC -mmodel=medium'
export FC=gfortran
export F77=gfortran
export F90=gfortran
export FORTC=gfortran

export PERL5LIB=/usr/share/perl5/vendor_perl/CPAN

```

4.3.5 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
```

4.3.6 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 Toolkit(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 Toolkit(TKIO)を使用する場合は、予め PPS Toolkit(TKIO)をインストールしておく必要があります。

また、PPS Toolkit(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、NetCDF 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	
	L2SLH	2HSLH	TK_2HSLH.h	
	L2LHP	2HSLHT	TK_2HSLHT.h	
3	L3DPR	3DPR	TK_3DPR.h	
	L3GMI	3GPROF	TK_3GPROF.h	
	L3CMB	3CMB	TK_3CMB.h	
	L3PR	3PR	TK_3PR.h	
	L3HSLH	3HSLH	TK_3HSLH.h	
	L3GSLH	3GSLH	TK_3GSLH.h	
	L3HSLHT	3HSLHT	TK_3HSLHT.h	
	L3GSLHT	3GSLHT	TK_3GSLHT.h	
	GSMaP	3GSMAPH5	TK_3GSMAPH5.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 NetCDF 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。

```
#include "TKheaders.h"
#include "TK_1BKu.h"

#define NRAY 49 /* number of Anglebin in one scan */
#define NBIN 260 /* number of Rangebin in one Anglebin */

int main(int argc, char *argv[]){

    /* declare Data struct */
    TKINFO granuleHandle1BKu;
    L1BKu_FS L1BKu;

    /* declare character */
    char job[256];
    char inputfile[256];

    strcpy(job, argv[1]);
    strcpy(inputfile, argv[2]);

    /* declare variables */
    int year, month, dayOfMon, hour, min;
    int i, j, k;
    int numOfScan;
    int status;

    /* declare array */
    float lat[NRAY], lon[NRAY];
    short noisePower[NRAY], echoPower[NRAY][NBIN];

    /* NetCDF file open */
    status = TKopen(inputfile, "1BKu", TKREAD, "NETCDF4", job, &granuleHandle1BKu, 1);
    if(status != TK_SUCCESS) {
        fprintf(stderr, "error:file open error[%s] input1BKu¥n", inputfile);
    }
}
```

→ 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 NetCDF 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 NetCDF 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。

NetCDF ファイルのオープン
inputfile: NetCDF 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
NETCDF4: 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)

```

/* META data read */
status = TKgetMetaInt(&granuleHandle1BKu, "SwathHeader",
                    "NumberScansGranule", &numOfScan);
if(status != TK_SUCCESS) {
    fprintf(stderr, "error:file read error[%s]¥n", inputfile);
}

/* Scan data read */
for (i=0; i<numOfScan; i++){
    status = TKreadScan( &granuleHandle1BKu, &L1BKu);

    /* Scantime Read */
    year      = L1BKu.ScanTime.Year;
    month     = L1BKu.ScanTime.Month;
    dayOfMon  = L1BKu.ScanTime.DayOfMonth;
    hour      = L1BKu.ScanTime.Hour;
    min       = L1BKu.ScanTime.Minute;

    for (j=0; j<NRAY; j++){
        /* Latitude and Longitude Read */
        lat[j] = L1BKu.Latitude[j];
        lon[j] = L1BKu.Longitude[j];

        /* noisePower Read */
        noisePower[j] = L1BKu.Receiver.noisePower[j];

        /* echoPower Read */
        for (k=0; k<NBIN; k++){
            echoPower[j][k] = L1BKu.Receiver.echoPower[j][k];
        }

        /* Print the value */
        if(i == 3946 && j == 19){
            printf("Scan=%d, Angle=%d¥n", i, j);
            printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
            printf("echoPower[259]= %d ¥n", echoPower[j][259]);
            printf("noisePower= %d ¥n", noisePower[j]);
        }
    }
}

/* NetCDF File close*/
status = TKclose(&granuleHandle1BKu);

return 0;
}

```

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" : NetCDF 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

スキャン数分ループ

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。

NetCDF 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。

```

CC=icx
CFLAGS=-O0 -g -mcmmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmmodel=medium -fpic -Wall

MAIN=./sample_L1_Ku_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(TKIO)/inc/ccode ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB) ¥
      -L$(TKIO)/lib

LIBES = -ltkselect ¥
        -ltkchdf5algs -ltkchdf5 ¥
        -ltknetcdf4 -ltknetcdf4algs -ltk¥
        -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
        $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
        $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
        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 3,4 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。

NetCDF 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)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
TEST ../data/GPM/DPR/GPMCOR_KUR_2604010211_0345_068594_1BS_DUB_08A.nc

Scan=3946, Angle=19
lat=64.764145, lon=57.453876
echoPower[259]= -29999
noisePower= -11130
    
```

第 1 引数の"TEST"はジョブ名です。(任意の文字列で構いません)

第 2 引数は NetCDF 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 NetCDF 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。

```
#include "TKheaders.h"
#include "TK_2ADPR.h"

#define NRAY 49 /* number of Anglebin in one scan */
#define NBIN 176 /* number of Rangebin in one Anglebin */

int main(int argc, char *argv[]){

    /* declare Data struct */
    TKINFO granuleHandle2ADPR;
    L2ADPR_SWATHS L2ADPR;

    /* declare character */
    char job[256];
    char inputfile[256];

    strcpy(job, argv[1]);
    strcpy(inputfile, argv[2]);

    /* declare variables */
    int year, month, date, dayOfMon, hour, min;
    int i, j, k;
    int numOfScan;
    int status;

    /* declare array */
    float lat[NRAY], lon[NRAY], precipESurf[NRAY];
    float precipWater[NRAY][NBIN];

    /* NetCDF file open */
    status = TKopen(inputfile, "2ADPR", TKREAD, "NETCDF4", job, &granuleHandle2ADPR, 1);
    if(status != TK_SUCCESS) {
        fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
    }
}
```

→ 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 NetCDF 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 NetCDF 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。

NetCDF ファイルのオープン
inputfile: NetCDF 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
NETCDF4: 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)

```

/* META data read */
status = TKgetMetaInt(&granuleHandle2ADPR, "FS_SwathHeader",
                    "NumberScansGranule", &numOfScan);
if(status != TK_SUCCESS) {
    fprintf(stderr, "error:file read error[%s]\n", inputfile);
}

/* Scan data read */
for (i=0; i<numOfScan; i++){
    status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);

    /* Scantime Read */
    year    = L2ADPR.FS.ScanTime.Year;
    month   = L2ADPR.FS.ScanTime.Month;
    dayOfMon = L2ADPR.FS.ScanTime.DayOfMonth;
    hour    = L2ADPR.FS.ScanTime.Hour;
    min     = L2ADPR.FS.ScanTime.Minute;

    for (j=0; j<NRAY; j++){
        /* Latitude and Longitude Read */
        lat[j] = L2ADPR.FS.Latitude[j];
        lon[j] = L2ADPR.FS.Longitude[j];

        /* precipRateESurface Read */
        precipESurf[j] = L2ADPR.FS.SLV.precipRateESurface[j];
        for (k=0; k<NBIN; k++){
            presipWater[j][k] = L2ADPR.FS.SLV.precipWater[j][k];
        }

        /* Print the value */
        if(i == 5210 && j == 43){
            printf("Scan=%d, Angle=%d\n", i, j);
            printf("lat=%f, lon=%f\n", lat[j], lon[j]);
            printf("precipRateESurface= %f (mm/hr)\n", precipESurf[j]);
            printf("precipWater[43][170]= %f (g/m~3)\n", precipWater[j][170]);
        }
    }
}

/* NetCDF File close*/
status = TKclose(&granuleHandle2ADPR);
return 0;
}

```

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)
 "FS_SwathHeader" : NetCDF 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
 &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

precipWater 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 5210 番目のアングル 43 のデータ) を出力しています。

NetCDF 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。

```

CC=icx
CFLAGS=-O0 -g -mcmmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmmodel=medium -fpic -Wall

MAIN=./sample_L2_DPR_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(TKIO)/inc/ccode ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB) ¥
      -L$(TKIO)/lib

LIBES = -ltkselect ¥
        -ltkchdf5algs -ltkchdf5 ¥
        -ltknetcdf4 -ltknetcdf4algs -ltk ¥
        -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
        $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
        $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
        rm -f *.o $(MAIN)
    
```

コンパイラに GNU コンパイラを使用する場合は、3,4 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。

NetCDF 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)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。

```

$ ./sample_L2_DPR_C
TEST ../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
Scan=5210, Angle=43
lat=32.052860, lon=132.983185
precipRateESurface= 2.261208 (mm/hr)
precipWater[43][170]= 0.128261 (g/m~3)
    
```

第 1 引数の"TEST"はジョブ名です。(任意の文字列で構いません)

第 2 引数は NetCDF ファイル名です。

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 NetCDF 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。

```
#include "TKheaders.h"
#include "TK_3DPR.h"

#define LTL 28 /* number of low resolution 5° grid intervals of latitude from 70°S to 70°N. */
#define LNL 72 /* number of low resolution 5° grid intervals of longitude from 180°W to 180°E. */
#define CHN 3 /* number of channels: KuFS, KaMS, KaHS, DPRMS, KuMS, KaFS, DPRFS */
#define RT 3 /* number of rain types: all, stratiform, convective */
#define ST 3 /* number of surface types: all, ocean, land */

int main(int argc, char *argv[]){

    /* declare Data struct */
    TKINFO granuleHandle3DPR;
    L3DPR_GRIDS L3DPR;

    /* declare character */
    char job[256];
    char inputfile[256];

    strcpy(job, argv[1]);
    strcpy(inputfile, argv[2]);

    /* declare variables */
    int i, j, k, l, m, n;
    int status;
    float lat, lon;
    /* declare array */
    float precipESurf[ST][RT][CHN][LHL][LTL];

    /* NetCDF file open */
    status = TKopen(inputfile, "3DPR", TKREAD, "NETCDF4", job, &granuleHandle3DPR, 1);
    if(status != TK_SUCCESS) {
        fprintf(stderr, "error:file open error[%s] input3DPR¥n", inputfile);
    }
}
```

→ 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。

→ granuleHandle3DPR は NetCDF 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 NetCDF 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。

NetCDF ファイルのオープン
inputfile: NetCDF 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
NETCDF4: 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)

```

/* Grid data read */
status = TKreadGridMulti( &granuleHandle3DPR, &L3DPR, "L3DPR_FS");

/* precipRateESurface.mean data read */
for (j=0; j<ST; j++){
    for (k=0; k<RT; k++){
        for (l=0; l<CHN; l++){
            for (m=0; m<LNL; m++){
                for (n=0; n<LTL; n++){
                    precipESurf[j][k][l][m][n] =
L3DPR.G1.precipRateESurface.mean[j][k][l][m][n];
                    if( m==63 && n==14 && l==0 && j==0 ) {
                        lat = (140.0/28.0) * n - 70.0 + (140.0/28.0/2);
                        lon = (360.0/72.0) * m - 180.0 + (360.0/72.0/2);
                        printf("lat=%f, lon=%f¥n", lat, lon);
                        printf("precipRateESurface.mean[%d][%d][%d][%d][%d]= %f (mm/hr)¥n",
                            j, k, l, m, n, precipESurf[j][k][l][m][n]);
                    }
                }
            }
        }
    }
}

/* NetCDF File close*/
status = TKclose(&granuleHandle3DPR);

return 0;
}

```

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
"L3DPR_FS": 読み出すデータ

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、チャンネルが 1 番目のデータ）を出力しています。

NetCDF 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。

```
CC=icx
CFLAGS=-O0 -g -mcmmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmmodel=medium -fpic -Wall

MAIN=./sample_L3_DPR_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(TKIO)/inc/ccode ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB) ¥
      -L$(TKIO)/lib

LIBES = -ltkselect ¥
        -ltkchdf5algs -ltkchdf5 ¥
        -ltknetcdf4 -ltknetcdf4algs -ltk ¥
        -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
    $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
    $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
    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。

5.3.1 のプログラムの名前です。

NetCDF 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)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。

```
./sample_L3_DPR_C TEST ../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.189663 (mm/hr)
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][1][0][63][14]= 1.616606 (mm/hr)
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][2][0][63][14]= 2.714188 (mm/hr)
```

第 1 引数の"TEST"はジョブ名です。(任意の文字列で構いません)

第 2 引数は NetCDF ファイル名です。

5.4 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、NetCDF 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、NetCDF 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。

NetCDF ファイルのバージョンを調べるには PPS Viewer THOR を使用して NetCDF 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_2ADPR_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 L2ADPR 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。

```
#include "TKheaders.h"
#include "TK_2ADPR.h"

int main(int argc, char *argv[]){

    /* declare Data struct */
    TKINFO granuleHandle2ADPR;
    L2ADPR_SWATHS L2ADPR;

    /* declare character */
    char job[256];
    char inputfile[256];

    strcpy(job, argv[1]);
    strcpy(inputfile, argv[2]);

    /* declare variables */
    int year, month, date, dayOfMon, hour, min;
    int i, j, k;
    int numOfScan;
    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。

```

/* declare array */
float lat[49], lon[49], precipESurf[49];
float zFactorCor[49][176];

/* NetCDF file open */
status = TKopen(inputfile, "2ADPR", TKREAD, "NETCDF4", job, &granuleHandle2ADPR, 1);
if(status != TK_SUCCESS) {
    fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
}

/* META data read */
status = TKgetMetaInt(&granuleHandle2ADPR, "NS_SwathHeader",
    "NumberScansGranule", &numOfScan);
if(status != TK_SUCCESS) {
    fprintf(stderr, "error:file read error[%s]¥n", inputfile);
}

/* Scan data read */
for (i=0; i<numOfScan; i++){
    status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);

    /* Scantime Read */
    year      = L2ADPR.NS.ScanTime.Year;
    month     = L2ADPR.NS.ScanTime.Month;
    dayOfMon  = L2ADPR.NS.ScanTime.DayOfMonth;
    hour      = L2ADPR.NS.ScanTime.Hour;
    min       = L2ADPR.NS.ScanTime.Minute;

    for (j=0; j<49; j++){
        /* Latitude and Longitude Read */
        lat[j] = L2ADPR.NS.Latitude[j];
        lon[j] = L2ADPR.NS.Longitude[j];

        /* precipRateESurface, zFactorCorrected Read */
        precipESurf[j] = L2ADPR.NS.SLV.precipRateESurface[j];
        for (k=0; k<176; k++){
            zFactorCor[j][k] = L2ADPR.NS.SLV.zFactorCorrected[j][k];
        }

        /* Print the value */
        if(i == 3946 && j == 19){
            printf("Scan=%d, Angle=%d¥n", i, j);
            printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
            printf("precipESurf= %f (mm/hr)¥n", precipESurf[j]);
        }
    }
}

/* NetCDF File close*/
status = TKclose(&granuleHandle2ADPR);

return 0;
}

```

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。

```
#include <stdlib.h>
#include <string.h>
#include "hdf5.h"
int main(){
    hid_t file_id, dataset_id;

    /* declare character */
    char inputfile[64]= "../data/GPM/DPR/
GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc";

    char dsetname[32]= "/FS/SLV/precipRateESurface";

    /* declare variables */
    int ret;

    /* declare array */
    float precipRateESurface[7988][49];

    /* NetCDF file open */
    file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);

    if(file_id < 0 ) {
        fprintf(stderr, "error:file open error[%s] %n", inputfile);
    }
    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 NetCDF のファイル名を設定しています。

dsetname は NetCDF 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。

NetCDF ファイルオープン
inputfile : NetCDF 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。

```

{
  /* dataset open */
  dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);

  if(dataset_id < 0 ) {
    fprintf(stderr, "error:file open error[%s] %n", inputfile);
  }
  else
  {

    /* precipRateESurface read */
    ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);

    if(ret < 0) {
      fprintf(stderr, "error:Dataset read error[%s]%n", inputfile);
    }
    else
    {
      /* file,dataset print */
      printf("file name= %s %n", inputfile);
      printf("data set name= %s %n", dsetname);
      /* precipRateESurface print */
      printf("precipRateESurface[5210][43])= %f (mm/hr)%n",
precipRateESurface[5210][43]);
    }

    /* dataset close */
    ret = H5Dclose(dataset_id);

  }
  /* NetCDF file close */
  ret = H5Fclose(file_id);
}
return 0;
}

```

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。

NetCDF 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。

```
CC=icx
CFLAGS=-O0 -g -mcmmodel=medium -fpic

#CC=gcc
#CFLAGS=-g -mcmmodel=medium -fpic -Wall

MAIN=./sample_HDF5_L2_DPR_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
          $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
           $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
      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。

6.1.1 のプログラムの名前です。

NetCDF のインクルードディレクトリのパスです

NetCDF のライブラリディレクトリのパスです

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= ../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
data set name= /FS/SLV/precipRateESurface
precipRateESurface[5210][43]= 2.261208 (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。

```
#include <stdlib.h>
#include <string.h>
#include "hdf5.h"
int main(){

    hid_t file_id, dataset_id;

    /* declare character */
    char inputfile[64]= "../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc";

    char dsetname[64]= "/FS/G1/precipRateESurface/mean";

    /* declare variables */
    int ret;
    float lat,lon;
    /* declare array */
    float precipRateESurface[3][3][3][72][28];

    /* NetCDF file open */
    file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);

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

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

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

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 NetCDF のファイル名を設定しています。

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

NetCDF ファイルオープン
inputfile : NetCDF ファイル名
H5F_ACC_RDONLY : アクセス指定 (読込みのみ)
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。

```

else{
  /* precipRateESurface read */
  ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);

  if(ret < 0) {
    fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
  }
  else{
    /* file,dataset print */
    printf("File name= %s ¥n", inputfile);
    printf("Dataset name= %s ¥n", dsetname);

    /* lat/lon print */
    lat = (140.0/28.0) * 14 - 70.0 + (140.0/28.0/2);
    lon = (360.0/72.0) * 63 - 180.0 + (360.0/72.0/2);
    printf("lat=%f, lon=%f¥n",lat,lon);

    /* precipRateESurface print */
    printf("precipRateESurface.mean[0][0][0][63][14]= %f ¥n",
precipRateESurface[0][0][0][63][14]);
    printf("precipRateESurface.mean[0][1][0][63][14]= %f ¥n",
precipRateESurface[0][1][0][63][14]);
  }

  /* dataset close */
  ret = H5Dclose(dataset_id);
}

/* NetCDF file close */
ret = H5Fclose(file_id);
}
return 0;
}

```

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。

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

6.2.2 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。

```

CC=icx
CFLAGS=-O0 -g -mmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mmodel=medium -fpic -Wall

MAIN=./sample_HDF5_L3_DPR_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
          $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
           $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
      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。

6.2.1 のプログラムの名前です。

NetCDF のインクルードディレクトリのパスです

NetCDF のライブラリディレクトリのパスです

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= ../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
data set name= /FS/G1/precipRateESurface/mean
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.189663
precipRateESurface.mean[0][1][0][63][14]= 1.616606

```

7. h5dump で GPM/TRMM データ読み込み

h5dump を使用して、NetCDF 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例を示します。

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。

```

$ h5dump -d FS/SLV/precipRateESurface -b -o
./GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipRateESurface
./GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
    
```

出力ファイルパス

入力ファイルパス

上記コマンドを実行すると以下のように表示され、出力ファイルパスにバイナリファイルが作成されます。

```

HDF5 "/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc" {
  DATASET "FS/SLV/precipRateESurface" {
    DATATYPE H5T_IEEE_F32LE
    DATASPACE SIMPLE { ( 7988, 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。

```

#include <stdio.h>
#include <string.h>

int main(){
    /* declare variables */
    float precipRateESurface[7988][49];
    FILE *fp;

    /* binary file declare */
    char inputfile[128] = "../data/GPM/DPR/
GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipRateESurface";

    /* binary file open */
    fp = fopen(inputfile, "rb");
    if(fp == NULL ) {
        printf("error:file open error[%s] %n", inputfile);
    }
    else
    {
        /* file read */
        fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);

        /* precipRateESurface print */
        printf("filename=%s\n",inputfile);
        printf("precipRateESurface[5210][43]= %f (mm/hr)\n",
precipRateESurface[5210][43]);
        printf("precipRateESurface[5210][44]= %f (mm/hr)\n",
precipRateESurface[5210][44]);

        /* binary file close */
        fclose(fp);
    }
    return 0;
}

```

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。

```

CC=icx
CFLAGS=-O0 -g -mmodel=medium -fPIC
#CC=gcc
#CFLAGS=-g -mmodel=medium -fpic -Wall

MAIN=./sample_h5dump_L2_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
          $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
           $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
      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。

7.1.2のプログラムの名前です。

NetCDFのインクルードディレクトリのパスです

NetCDFのライブラリディレクトリのパスです

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=./data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipR
ateESurface
precipRateESurface[5210][43]= 2.261208 (mm/hr)
precipRateESurface[5210][44]= 2.951939 (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。

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。

```
$ h5dump -d FS/G1/precipRateESurface/mean -b -o
./GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean
./GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
```

出力ファイルパス

入力ファイルパス

上記コマンドを実行すると以下のように表示され、出力ファイルパスにバイナリファイルが作成されます。

```
HDF5 "/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc" {
DATASET "FS/G1/precipRateESurface/mean" {
  DATATYPE  H5T_IEEE_F32LE
  DATASPACE  SIMPLE { ( 3, 3, 3, 72, 28 ) / ( 3, 3, 3, 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。

```

#include <stdio.h>
#include <string.h>
int main(){

    /* declare variables */
    float precipRateESurface[3][3][3][72][28];
    FILE *fp;
    int i, j, k, m, n, size;
    float lat,lon;
    /* binary file name */
    char inputfile[128]
=" ../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean";

    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            for(k=0; k<3; k++){
                for(m=0; m<72; m++){
                    for(n=0; n<28; n++){
                        precipRateESurface[i][j][k][m][n]=0;
                    }
                }
            }
        }
    }

    /* binary file open */
    fp = fopen(inputfile, "rb");
    if(fp == NULL ) {
        printf("error:file open error[%s] %n", inputfile);
    }
    else
    {
        /* file read */
        size = fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);

        /* precipRateESurface print */
        printf("filename=%s\n",inputfile);
        lat = (140.0/28.0) * 14 - 70.0 + (140.0/28.0/2);
        lon = (360.0/72.0) * 63 - 180.0 + (360.0/72.0/2);
        printf("lat=%f, lon=%f\n",lat,lon);
        printf("precipRateESurface.mean[0][0][0][63][14]= %f %n",
precipRateESurface[0][0][0][63][14]);
        printf("precipRateESurface.mean[0][1][0][63][14]= %f %n",
precipRateESurface[0][1][0][63][14]);

        /* binary file close */
        fclose(fp);
    }
    return 0;
}

```

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。

```

CC=icx
CFLAGS=-O0 -g -mmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mmodel=medium -fpic -Wall

MAIN=./sample_h5dump_L3_C

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
          $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
            $(CC) $(CFLAGS) $(INC) -c $< -o $@

clean:
      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。

7.2.2のプログラムの名前です。

NetCDFのインクルードディレクトリのパスです

NetCDFのライブラリディレクトリのパスです

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=./data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.189663
precipRateESurface.mean[0][1][0][63][14]= 1.616606

```

8. 改訂履歴

改版履歴

版数	日付	改版内容	備考
1	2016/1/26		
2	2017/9/13	<p>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。</p> <p>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</p> <p>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。</p>	
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	<p>1.~ 3. TRMM 追加及び GPM サイトリニューアルに伴う修正</p> <p>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</p> <p>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</p>	
5	2021/12/24	<p>1. GSMP プロダクトバージョン 5、GPM/TRMM プロダクトバージョン 7 に修正。</p> <p>3. 関連文書とサンプルプログラムの入手方法修正</p> <p>4. TKIO ダウンロードの URL 修正</p> <p>3. ~ 5. SLP/SLM/SLG/LHP/LHM/LHG のサンプルコード追加に伴い表 3.1、表 4.2、表 5.1 に記載を追加</p> <p>5. プログラム中の V7 変更部分を修正</p>	
6	2022/1/21	5.~ 7. サンプルプログラム修正 (表示位置見直しや表示項目に lat/lon 追加等) に伴う修正	
7	2026/6/1	<p>サンプルデータを V8 に更新</p> <p>コードの説明を V8、NetCDF に合わせて修正</p>	