平成 15 年 3 月 31 日
気 象 庁 予 報 部

配信資料に関する技術情報（気象編）第129号

～FTP 方式の 30 分毎の解析雨量及び降水短時間予報 GPV の
サンプルデータの提供について～

配信資料に関する技術情報（気象編）第128号において、毎正時及び正時+30分を初期時刻とする解析雨量及び降水短時間予報について、全国分のデータを1ファイルとしてFTP方式での提供を開始すること及びそのフォーマットについてお知らせしました。

本日、支援センターにこれらGPVのサンプルファイルを提供しましたのでお知らせします。

これらGPVの解読（デコード）処理の参考となる技術情報を添付します。

FTP形式で提供する解析雨量及び降水短時間予報GPVの解読処理に関する技術情報

本技術情報の全部又は一部を利用することは問題ありません。ただし、本技術情報の一部又は全部を利用したことにより、利用者が被った直接的または間接的ないかなる損害についても、気象庁は一切責任を負いません。本技術情報に関する個別の対応は行いかねますので、ご容赦願います。

利用方法

1. **「make」コマンドによりコンパイルすることにより実行ファイル「grib2_dec」が作成される。**
   - ANSI 準拠の c コンパイラでコンパイルできます。ただし、拡張機能 unsigned long long を利用しています。UNIX（HP-UX，AIX，HI-UX/WE2）及び Linux（RedHat）での動作を確認しています。
   - リトルエンディアンマシンの場合、ヘッダーファイル「sample_decode.h」の 5 行目のコメントをはずしてください。

2. **次のコマンドを入力することにより、GRIB2各節の内容が端末に表示されると共に、xpm形式の画像イメージファイルが作成される。**

   grib2_dec ｛サンプルデータファイル名｝ -xpm

   - サンプルデータファイル名は仮名（実際の提供時には名前を変えることがあります。）です。本技術情報はこの仮名のファイル名により解析雨量と降水短時間予報を区別しておりますので、留意下さい。

3. **データを4バイト整数の配列で出力したい場合は、次のコマンドを実行する。**

   grib2_dec ｛サンプルデータファイル名｝

※UNIX は、X/Open Company Limited が独占的にライセンスしている米国ならびに他の国における登録商標です。
※Linux は、Linus Torvalds の米国及びその他の国における登録商標あるいは商標です。
※AIX は、米国における米国 Internatinal Business Machines Corp.の登録商標です。
※HP-UX は、米国 Hewlett-Packard Company のオペレーティングシステムの名称です。
※HI-UX/WE2 は、（株）日立製作所の登録商標です。
※RedHat は、米国 Red Hat Software,Inc.の登録商標です。

Makefike

```
CC       = cc
CFLAGS   = -O
MODULE   = grib2_dec
OBJS     = sample_grib2_dec.o rlencmp.o i2pix.o
HEADER  = sample_decode.h prr_template.h pmf_template.h

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

$(MODULE) : $(OBJS)
          $(CC) $(OBJS) -o $(MODULE)

clean :
          rm $(OBJS) $(MODULE)
```

sample_decode.h

```
/* ************************************* */
/* sample grib2 decode program include file */
/* ************************************* */

/* #define LITTLE_ENDIAN */

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

pmf_template.h

```
/* ************************************************************** */
/*  template for short range precipitation forecast               */
/* ************************************************************** */
char sc_pmf[][40]={
  "42118",
  "412211121111111",
  "41141121141414444444144441",
  "41221111121141141142111111411141488821n",
  "41421221n",
  "411",
  "41n",
  "4",
  ""
};
```

prr_template.h

```
/* ***************************************************** */
/*  template for analisys precipitation                  */
/* ***************************************************** */
char sc_prr[][40]={
  "42118",
  "412211121111111",
  "41141121141414444444144441",
  "41221111121141141142111111414111414888",
  "41421221n",
```

```
    "411",
    "41n",
    "4",
    ""
  };
```

sample_grib2_dec.c

```
/* ---------------------------------------------------------------------- */
/*  Last Updated Ver.1.2    2003.03.27                                    */
/*        Copyright (C) 2003 Japan Meteorological Agency  All rights reserved */
/* ---------------------------------------------------------------------- */

#include "sample_decode.h"
#include "prr_template.h"
#include "pmf_template.h"

#ifdef LITTLE_ENDIAN
  #define Fread fread_little_endian
#else
  #define Fread fread
#endif

typedef struct sect {
  int num;
  int *len;
  unsigned char **v;
} ST_SECT;

fread_little_endian(void *d, int len, int num, FILE *fp)
{
unsigned char uc[8], *ud;
int i, j, k;

  ud = d;
  if (len==1) fread(d, len, num, fp);
  else {
    for(i=0, k=0; i<num; i++, k+=len) {
      fread(uc, len, 1, fp);
      for(j=0; j<len; j++)  *(ud+k+j) = uc[len-1-j];
    }
  }
}

void init_sect(ST_SECT ss[], int af)
{
int i, j, k;
char c[2];

  if (af==0) {
    k=0; while(strlen(sc_prr[k])!=0) k++;
    for(i=0; i<k; i++) {
      ss[i].num = strlen(sc_prr[i]);
      ss[i].len = (int *)malloc(sizeof(int *)*ss[i].num);
      for(j=0; j<ss[i].num; j++) {
        strncpy(c, &sc_prr[i][j], 1); *(ss[i].len+j) = atoi(c);
      }
      ss[i].v = (unsigned char **)malloc(sizeof(unsigned char *)*ss[i].num);
    }
  } else {
    k=0; while(strlen(sc_pmf[k])!=0) k++;
    for(i=0; i<k; i++) {
      ss[i].num = strlen(sc_pmf[i]);
```

```
56          ss[i].len = (int *)malloc(sizeof(int *)*ss[i].num);
57          for(j=0; j<ss[i].num; j++) {
58            strncpy(c, &sc_pmf[i][j], 1); *(ss[i].len+j) = atoi(c);
59          }
60          ss[i].v = (unsigned char **)malloc(sizeof(unsigned char *)*ss[i].num);
61          for(j=0; j<ss[i].num; j++)  *(ss[i].v+j) = NULL;
62        }
63      }
64    }
65
66    int read_sect(ST_SECT ss[], FILE *fp)
67    {
68    int i, j, nn, mm, si;
69    unsigned int  slen;
70    unsigned short *us, ud;
71    unsigned char sn;
72
73      Fread(&slen, 4, 1, fp);
74      if (memcmp(&slen, "7777", 4)==0) {
75        si=8;
76        return(si);
77    #ifdef LITTLE_ENDIAN
78      } else if (memcmp(&slen, "BIRG", 4)==0) {
79    #else
80      } else if (memcmp(&slen, "GRIB", 4)==0) {
81    #endif
82        si=0; slen = 16;
83        Fread(&ud, 2, 1, fp);
84        *(ss[si].v+0)=(unsigned char *)realloc(*(ss[si].v+0),sizeof(unsigned int));
85        *(ss[si].v+1)=(unsigned char *)realloc(*(ss[si].v+1),sizeof(unsigned short));
86        memcpy(*(ss[si].v+0), &slen, 4);
87      } else {
88        Fread(&sn, 1, 1, fp);
89        si = (int)sn;
90        if (si>=3) si--;
91        *(ss[si].v+0)=(unsigned char *)realloc(*(ss[si].v+0),sizeof(unsigned int));
92        *(ss[si].v+1)=(unsigned char *)realloc(*(ss[si].v+1),sizeof(unsigned char));
93        memcpy(*(ss[si].v+0), &slen, 4);
94        memcpy(*(ss[si].v+1), &sn, 1);
95      }
96
97      for(i=2; i<ss[si].num; i++) {
98        if (*(ss[si].len+i)==0) {
99          us = (unsigned short *)(*(ss[si].v+i-2));
100         nn = (*(ss[si].len+i-2)==4) ? sizeof(unsigned char)
101                                     : sizeof(unsigned short);
102         mm = (*(ss[si].len+i-2)==4) ? slen-5 : *us;
103       } else {
104         nn = *(ss[si].len+i);
105         mm = 1;
106       }
107       *(ss[si].v+i) = (unsigned char *)realloc(*(ss[si].v+i),(size_t)nn*mm);
108       Fread(*(ss[si].v+i), nn, mm, fp);
109     }
110
111   return(si);
112   }
113
114   int dec_data(ST_SECT ss[], int **lv)
115   {
116   int nin, nout, maxv, nbit, rt;
117   unsigned int *ui;
```

```
118   unsigned short *us;
119
120     ui = (unsigned int *)*(ss[4].v+2);   nout = *ui;
121     nbit = **(ss[4].v+4);
122     us = (unsigned short *)*(ss[4].v+5); maxv = *us;
123     ui = (unsigned int *)*(ss[6].v+0);   nin = *ui-5;
124
125     *lv = (int*)malloc(sizeof(int)*nout);
126     rt = decode_rlen_nbit(*lv, sizeof(int), *(ss[6].v+2), nin, nout, maxv, nbit);
127
128   return(rt);
129   }
130
131   void print_info(ST_SECT ss[], int sn)
132   {
133   int i, j, k;
134   unsigned long long *ull;
135   int *ii;
136   unsigned short *us, *n;
137
138     printf("========== SECTION %1.1d ==========¥n", (sn>=2) ? sn+1 : sn);
139     for(i=0, j=1; i<ss[sn].num; j+=*(ss[sn].len+i), i++) {
140       if (*(ss[sn].len+i)==1) {
141         if (**(ss[sn].v+i)==0xff)
142           printf("    %3d    : 0x%2.2x¥n", j, **(ss[sn].v+i));
143         else
144           printf("    %3d    : %d¥n", j, **(ss[sn].v+i));
145       } else if (*(ss[sn].len+i)==8) {
146         if (sn==0) {
147           ull = (unsigned long long *)*(ss[sn].v+i);
148           printf("%4d --%4d: %d¥n", j,j+*(ss[sn].len+i)-1, (unsigned int)ull);
149         } else {
150           printf("%4d --%4d: ", j,j+*(ss[sn].len+i)-1);
151           for(k=0; k<8; k++) printf("%2.2x", *(*(ss[sn].v+i)+k)); printf("¥n");
152         }
153       } else if (*(ss[sn].len+i)==4) {
154         ii = (int *)*(ss[sn].v+i);
155         if (*ii>=0 || (i==12 && sn==3)) /* only fcst_time is signed int */
156           printf("%4d --%4d: %d¥n", j,j+*(ss[sn].len+i)-1, *ii);
157         else
158           printf("%4d --%4d: 0x%8.8x¥n", j,j+*(ss[sn].len+i)-1, *ii);
159       } else if (*(ss[sn].len+i)==2) {
160         us = (unsigned short *)*(ss[sn].v+i);
161         if (*us==0xffff)
162           printf("%4d --%4d: 0x%4.4x¥n", j,j+*(ss[sn].len+i)-1, *us);
163         else
164           printf("%4d --%4d: %d¥n", j,j+*(ss[sn].len+i)-1, *us);
165       } else if (sn==3 || sn==4) {
166         n = (unsigned short *)*(ss[sn].v+i-2);
167         us = (unsigned short *)*(ss[sn].v+i);
168         for(k=0; k<*n; k++) {
169           if (*(us+k)==0xffff)
170             printf("%4d --%4d: 0x%4.4x¥n", j+2*k,j+2*k+1, *(us+k));
171           else
172             printf("%4d --%4d: %d¥n", j+2*k,j+2*k+1, *(us+k));
173         }
174       }
175     }
176     fflush(stdout);
177   }
178
179   int main(int argc, char *argv[])
```

```
{
ST_SECT ss[8];
FILE *fp, *fpo;
char fname[160], gname[160], suffix[160], fcs[160], ffm[160];
int sn, *lv, gn, sff=0, *xs, *ys, maxv, fcnt=0, af, ll;
unsigned short *us_maxv;

  if (argc==1) {
    fprintf(stderr, "¥n¥nusage: grib2_dec ***_grib2.bin (-xpm)¥n¥n");
    fprintf(stderr, "    This program decodes the grib2 file named ***_grib2.bin, and prints the¥n
value of each section in GRIB2.  Also, this program puts out a raw (4 byte¥n  integer) data file
***_int.bin as a rectangle grid dimension.  In case of¥n  specifying -xpm options, an output file
is to a picture image file ***.xpm¥n  formatted as X-pixmap.¥n¥n");
    exit(1);
  } else if (argc==3 && strcmp(argv[2],"-xpm")==0) {
    strcpy(suffix, ".xpm");  sff = 1;
  } else
    strcpy(suffix, ".bin");

  strcpy(fname, argv[1]);
  if ((fp=fopen(fname,"r"))==NULL) {
    fprintf(stderr, "grib2 file <%s> open error!!¥n", fname); exit(1);
  }
  af = (strstr(fname, "_ANAL")==NULL) ? 1 : 0;

  init_sect(ss, af);


  while((sn=read_sect(ss, fp))!=8) {
    if (sn==6) {
      print_info(ss, sn);
      gn = dec_data(ss, &lv);
      if (gn>0) {
        ll = strlen(fname)-strlen(strstr(fname,"_grib2.bin"));
        strncpy(gname, fname, ll); gname[ll] = '¥0';
        sprintf(fcs, "_%1d", fcnt); strcpy(ffm, (sff==1) ? "" : "_int");
        strcat(gname, strcat(fcs, strcat(ffm, suffix)));
        if ((fpo=fopen(gname,"w"))==NULL) {
          fprintf(stderr, "output file <%s> open error!!¥n", gname); exit(1);
        }
        if (sff==0)
          fwrite(lv, sizeof(int), gn, fpo);
        else {
          xs = (int *)*(ss[2].v+14);
          ys = (int *)*(ss[2].v+15);
          us_maxv = (unsigned short *)*(ss[4].v+6);
          maxv = (int)*us_maxv;
          i2pix(lv, *xs, *ys, maxv, fpo);
        }
        fclose(fpo);
        fcnt++;
      }
      free(lv);
    } else {
      print_info(ss, sn);
    }
  }

  fclose(fp);
}
```

rlencmp.c

```
1   /* ---------------------------------------------------------------------- */
2   /*  Last Updated Ver.1.2    2003.03.27                                    */
3   /*        Copyright (C) 2003 Japan Meteorological Agency  All rights reserved */
4   /* ---------------------------------------------------------------------- */
5
6   #include "sample_decode.h"
7
8   int ipow(const int i, const int j)
9   {
10  int k, l;
11    for(k=0, l=1; k<j; k++) l*=i;
12  return(l);
13  }
14
15  void endian_conv4(void *idat)
16  {
17  unsigned char *uc, c;
18
19    uc = (unsigned char *)idat;
20    c=*(uc+0); *(uc+0)=*(uc+3); *(uc+3)=c;
21    c=*(uc+1); *(uc+1)=*(uc+2); *(uc+2)=c;
22
23  }
24
25  int nbit_unpack(unsigned char din[], int nin, unsigned int dout[],
26                    int nout, int nbit)
27  {
28
29  unsigned int wi, n_b2s, *ui;
30  int i, bitp, dp;
31
32    n_b2s = ipow(2,nbit)-1;
33
34    i = 0; dp = 0; bitp = 0;
35    while(dp<nin)  {
36      bitp += nbit;
37      memcpy(&wi, &din[dp], 4);
38  #ifdef LITTLE_ENDIAN
39      dout[i] = wi & n_b2s;
40  #else
41      dout[i] = (wi >> (32-bitp)) & n_b2s;
42  #endif
43      while (bitp>=8)  { dp++; bitp-=8; }
44      if (++i>nout)  return(-1);
45    }
46
47  return(i);
48  }
49
50  /* ---------------------------------------------------------------------- */
51  int decode_rlen_nbit(void *udata, size_t utype, unsigned char *din, int nin,
52                         int nout, int maxv, int nbit)
53  /* ---------------------------------------------------------------------- */
54  /* decode ranlength compress ( nbit data version )                        */
55  /*   output:   udata  = user data for put                                 */
56  /*   input:    utype  = user data type :sizeof(int or short or unsigned char) */
57  /*             din    = compressed data                                   */
58  /*             nin    = compressed data size (byte)                       */
59  /*             nout   = number of grid point                              */
60  /*             maxv   = maximum value of user data                        */
```

```
61  /*            nbit   = number of bit used for a compressed data             */
62  /*   return:  >=0    = number of decoded data                               */
63  /*            -4     = uncompressed data size exceeds nout                  */
64  /*            -6     = first user data is out of the data range             */
65  /* ---------------------------------------------------------------------- */
66  {
67  int i, j, k, l, m, n, v, p, cf=0, *doi, ninb;
68  short *dos;
69  unsigned char *d, *doc;
70  unsigned int *wd, *ww;
71
72    doi = (int *)udata; dos = (short *)udata; doc = (unsigned char *)udata;
73    wd = (unsigned int *)malloc(sizeof(unsigned int)*nout);
74    ww = (unsigned int *)malloc(sizeof(unsigned int)*nout);
75    ninb = nbit_unpack(din, nin, wd, nout, nbit);
76    if (ninb<0)  return(-4);
77    l = ipow(2,nbit)-1-maxv;
78    v = (int)(*wd);
79    if (v<0 || v>maxv)  return(-6);
80
81    i = 0; k = 0; p = -1;
82    while(i<ninb) {
83      v = (int)(*(wd+i++));
84      if (v<=maxv) {
85        if (p>=0) {
86          for(j=0; j<m; j++) {
87            if (k==nout) { cf=1; break; }
88            *(ww+k++) = p;
89          }
90          if (cf==1) break;
91        }
92        p = v;
93        m = 1;
94        n = 0;
95      } else {
96        m += ipow(l,n++)*(v-maxv-1);
97      }
98    }
99    for(j=0; j<m; j++) {
100     if (k==nout) break;
101     *(ww+k++) = p;
102   }
103   switch(utype) {
104     case 1: for(j=0; j<k; j++) *(doc+j) = (unsigned char)*(ww+j);  break;
105     case 2: for(j=0; j<k; j++) *(dos+j) = (short)*(ww+j);          break;
106     case 4: for(j=0; j<k; j++) *(doi+j) = (int)*(ww+j);            break;
107   }
108
109   free(wd); free(ww);
110 return(k);
111 }
```

i2pix.c

```
1   /* ---------------------------------------------------------------------- */
2   /*  Last Updated Ver.1.2    2003.03.27                                    */
3   /*        Copyright (C) 2003 Japan Meteorological Agency  All rights reserved */
4   /* ---------------------------------------------------------------------- */
5
6   #include <stdio.h>
```

```
#include <stdlib.h>

void i2pix(
  int *fd,    /* orignal data array    */
  int ix,     /* size of x-axis        */
  int iy,     /* size of y-axis        */
  int maxval, /* maximum value of data */
  FILE *fp    /* output file pointer   */
) {
int c, i, j, k, l, m;
int r, g, b;
char *line, *p;

  line = (char *)malloc(sizeof(char)*ix+1);
  fprintf(fp,"static char *no_xpm[] = {¥n");
  fprintf(fp,"¥"%d %d %d 1¥",¥n", ix, iy, maxval+1);
  fprintf(fp,"¥"# c #000000¥",¥n");
  fprintf(fp,"¥"$ c #888888¥",¥n");
  fprintf(fp,"¥"%c c #FFFFFF¥",¥n", '%');
  for(i=2; i<maxval; i++) {
    if (i<=maxval/4)        r = 0;
    else if (i<=maxval/2)   r = (i-maxval/4)*4;
    else                    r = maxval;
    if (i<=maxval/4)        g = i*4;
    else if (i<=maxval/2)   g = maxval;
    else if (i<=maxval*3/4) g = maxval-(i-maxval/2)*4;
    else                    g = 0;
    if (i<=maxval/4)        b = maxval;
    else if (i<=maxval/2)   b = maxval-(i-maxval/4)*4;
    else if (i<=maxval*3/4) b = 0;
    else                    b = (i-maxval*3/4)*4;
    if (r<0) r = 0; r *= 255.0/maxval;
    if (g<0) g = 0; g *= 255.0/maxval;
    if (b<0) b = 0; b *= 255.0/maxval;
    fprintf(fp,"¥"%c c #%02X%02X%02X¥",¥n", i+'$', r, g, b);
  }

  for(j=0; j<iy-10; j++) {
    p = line;
    for(i=0; i<ix; i++) {
      k = i+j*ix;
      if (fd[k]==0) *p++ = '$';
      else          *p++ = (char)((int)'$'+fd[k]);
    }
    *p = '¥0';
    fprintf(fp,"¥"%s¥",¥n", line);
  }

/* color bar */

  for(j=iy-10; j<iy; j++) {
    p = line;
    for(i=0; i<ix; i++) *p++ = (char)((int)'$'+i*(maxval+1)/ix);
    *p = '¥0';
    fprintf(fp,"¥"%s¥",¥n", line);
  }

  fprintf(fp,"};¥n");
  free(line);
}
```