GMT

CDB2XY

Programme d'extraction de données de la Base intégrée dans GMT (côtes, rivières et frontières)

(Version 1.0)

Germinal GABALDA
Laboratoire de Géophysique - Orstom - Bondy

Août 1995

CDB2XY permet extraire sous forme de doublets xy des coordonnées de lignes de côtes, de rivières et de frontières de la Base de Données incluse dans GMT (répertoire LIBDIR). Version modifiée du programme pscoast, il est écrit en Langage C et utilise les ressources du produit GMT (Generic Mapping Tool) et les librairies netCDF. Se reporter à l'*annexe 1* pour l'installation de cdb2xy et de son manuel en ligne.

L'exécution de la commande sans arguments donne un rapide résumé mais il est également possible d'obtenir une description complète en affichant le manuel en ligne (**man cdb2xy**, *Annexe 2*). Les différentes options possibles sont néanmoins développées plus longuement dans la partie *UTILISATION* .

Pour obtenir de plus amples renseignements sur la ''GMT High-Resolution Coastline database'' il est conseillé de se reporter au document *Technical Reference and Cookbook - Appendix K* dont les références sont citées en annexe 4.

Pour toute suggestion ou renseignement complémentaire

Germinal GABALDA
Orstom - Laboratoire de Géophysique
32, av. Henri Varagnat - 93143 - BONDY CEDEX FRANCE
Email: gabalda@gravi.bondy.orstom.fr

La Base de Données du produit GMT contient trois types d'informations (lignes de côtes, rivières et frontières) et pour chacun d'eux cinq résolutions (grossière, basse, intermédiaire, haute et pleine), soit quinze fichiers dans le répertoire **LIBDIR**. En fonction du niveau de précision et du type de données sélectionnés, la commande **cdb2xy** extrait de la base les informations demandées et les écrit sous forme de doublet xy (**longitude latitude** par défaut) dans le (ou les) fichier(s) de sortie. Les données en sortie sont organisées en segments et ceux-ci sont séparés par l'enregistrement '**> niveau_hiérarchique**'. Vous pouvez changer le format d'écriture des données en sortie en modifiant le paramètre **D_FORMAT** dans le fichier **.gmtdefaults**. La syntaxe complète de la commande est la suivante:

cdb2xy -Ggénérique -Rouest/est/sud/nord [-Aaire_min/niveau_max] [-Drésolution] [-Irivière] [-Nfrontière] [-V] [-W] [-:]

-G **générique** est le nom générique du (ou des) fichier(s) de sortie. Le nom du (ou des) fichier(s) de sortie dépend du *type* de données sélectionné et du *niveau* de résolution choisie et il est de la forme:

générique_type_niveau

Type = **shore** (ligne de côtes), **river** (rivière) ou **border** (frontière)
Niveau = **c**, **l**, **i**, **h** ou **f** (*voir option D*)

-R **ouest/est/sud/nord** permet de définir les limites géographiques de la région qui nous intéresse. Les limites peuvent être indiquées en degrés et minutes [et secondes] avec le format **dd:mm[:ss].**

OPTIONS

-A Les polygones d'une surface inférieure à **aire_min** en km^2 ou de niveau hiérarchique supérieur à **niveau_max** ne sont pas extraient de la base [0/4 (tous les objets) par défaut].

-D Pour éviter d'extraire des polygones qui compte tenu de la zone géographique délimité ne seront pas visibles, il existe cinq niveaux de résolutions (**c**, **l**, **i**, **h** et **f**) [l par défaut].

c (*crude* = grossière) : polygones > 500 km^2
l (*low* = basse) : polygones > 100 km^2
i (*intermediate* = intermédiaire) : polygones > 20 km^2
h (*hight* = haute) : polygones > 1 km^2
f (*full* = pleine) : tous les polygones

-I pour extraire des rivières. Choisissez le type de rivières dans la liste ci-dessous. Il faut répéter l'option -I pour sélectionner différents types.

1 = Rivières importantes
2 = Rivières additionnelles
3 = Rivières additionnelles (importantes)
4 = Petites rivières
5 = Rivières larges (2 traits de côtes)
6 = Rivières importantes non permanentes
7 = Rivières additionnelles non permanentes
8 = Petites rivières non permanentes
9 = Canaux importants
10 = Petits canaux

a = Toutes les rivières et tous les canaux (1-10)
r = Toutes les rivières permanentes(1-4)
i = Toutes les rivières non-permanentes (6-8)
c = Tous les canaux (9-10)

-N pour extraire des frontières politiques. Choisissez le type de frontières dans la liste ci-dessous. Il faut répéter l'option -N pour sélectionner différents types.

1 = Frontières nationales
2 = Frontières d'Etats (Amériques)
3 = Frontières marines

a = Toutes les frontières (1-3)

-V Affichage de message.

-W pour extraire les lignes de côtes.

-: Pour inverser l'écriture en sortie des coordonnées. Par défaut (longitude latitude).

EXEMPLE

Pour extraire en haute résolution et pour l'Afrique les lignes de côtes, les frontières nationales et les plus importantes rivières il faut taper la commande:

cdb2xy -Gafrica -R-30/3/640/40 -Dh -I1 -I2 -N1 -W

Création de trois fichiers en sortie:

africa_shore_h pour les lignes de côtes
africa_border_h pour les frontières
africa_river_h pour les rivières

1. GMT - The Generic Mapping Tool

GMT est un ensemble d'outils Unix qui permettent de manipuler des données et de produire des dessins postscripts. GMT peut être obtenu gratuitement sur internet ainsi que les librairies netCDF qui sont utilisées par plusieurs fonctions de GMT.

- ftp anonyme à partir de la machine **kiawe.soest.hawaii.edu** (128.171.151.16) sous le répertoire pub/gmt
- www à l'adresse **http://www.soest.hawaii.edu/soest/gmt.html**.

2. Préparation

- Créez un répertoire sous le répertoire contenant les sources GMT
- Copiez les fichiers **cdb2xy.c**, **cdb2xy.man** et **makefile** dans ce répertoire
- Editez le makefile et adaptez à votre site les variables **PRODUC**, **NETCDF**, **LIBDIR**, **BINDIR** et **MANDIR**

  PRODUC = cdb2xy
  NETCDF = Chemin pour atteindre netCDF
  LIBDIR = Chemin pour atteindre les librairies GMT
  BINDIR = Répertoire d'installation du binaire cdb2xy
  MANDIR = Répertoire d'installation du man

3. Installation de CDB2XY

- Pour générer et installer le programme, installer le manuel en ligne et nettoyer le répertoire, tapez **make all**
- Pour générer le programme, tapez **make bin**
- Pour installer le programme, tapez **make install**
- Pour installer le manuel en ligne, tapez **make man**
- Pour nettoyer le répertoire, tapez **make clean**

Si BINDIR et MANDIR ne sont pas des répertoires standards alors pensez à modifier les variables **PATH** et **MANPATH**.

4. Makefile

```
#       Follow the instructions in this makefile to customize your setup.
#       To compile/link/install the program,install man pages and clean, try "make all".
#       To compile/link the program, try "make bin".
#       To install the program, try "make install".
#       To install man pages, try "make man".
#       To clean out directory with "make clean".
#
#       Author:         Germinal Gabalda
#                       ORSTOM - Geophysique
#                       Bondy - 93143 - FRANCE
#       Date:           1-AUG-1995
#-----------------------------------------------------------------------
#   Set PRODUC,NETCDF, BINDIR, LIBDIR, and MANDIR before making anything!
#-----------------------------------------------------------------------
#       PRODUC          -> The name of the program
#       NETCDF          -> Where to find netcdf sub-directories lib and include
#       BINDIR          -> Where to install executable code
#       LIBDIR          -> Where GMT support data files are installed.  This path
#                          must be hard, i.e., starting from root /
#       MANDIR  -> Where GMT manual pages are installed
#-----------------------------------------------------------------------
PRODUC          = cdb2xy
NETCDF          = /usrx/NETCDF-2.3.2
BINDIR          = /usrx/GMT-3.0/bin
LIBDIR          = /usrx/GMT-3.0/lib
MANDIR          = /usrx/GMT-3.0/man/manl
#------------------------------------------------ stop here ------------------
NETCDFLIB       = $(NETCDF)/lib
NETCDFINC       = $(NETCDF)/include
#-----------------------------------------------------------------------
SHELL=/bin/sh
CC      = cc
SI      = -DSI
INSTALL = install -s
CFLAGS  = -O -I.. -I$(NETCDFINC) -DLIBDIR=\"$(LIBDIR)\"  $(SI)
CDF     = -L$(NETCDFLIB) -lnetcdf
PS      = -lpsl
GMT     = -lgmt
#-----------------------------------------------------------------------
.SUFFIXES:
all: bin install man clean

# BINARY FILES ------------------------------------------------------
bin:    $(PRODUC)

$(PRODUC):      $(PRODUC).c
                $(CC) $(CFLAGS) $? -L.. -lgmt $(CDF) -lm -o $@
# INSTALL -----------------------------------------------------------
install: $(BINDIR)/ $(PRODUC)

$(BINDIR)/ $(PRODUC): $(PRODUC)
                $(INSTALL) $? $(BINDIR)

# MANS FILES --------------------------------------------------------
man:    $(MANDIR)/ $(PRODUC).l

$(MANDIR)/ $(PRODUC).l: $(PRODUC).man
        cp $? $@

# CLEAN -------------------------------------------------------------
clean:
                rm -f *.o $(PRODUC)
```

CDB2XY(l) MISC. REFERENCE MANUAL PAGES CDB2XY(l)

NAME

cdb2xy - To extract coastlines, borders and rivers from Coastline DataBase

SYNOPSIS

cdb2xy -Goutfile -Rwest/east/south/north [-Amin area/max level] [-Dresolution] [-Iriver] [-Nborder] [-V] [-W] [-:]

DESCRIPTION

cdb2xy extract data from a hight-resolution coastline database. Three data sets are considered: shores, borders, and rivers. The datafiles come in 5 different resolutions: (f)ull, (h)igh, (i)ntermediate, (l)ow, and (c)rude. The full resolution files provide great detail; for maps of larger geographical extent it is more economical to use one of the other resolutions. cdb2xy writes out xy-doublets in ASCII format to output file(s). Segments are separated by the record '> level'.

-G outfile the generic name of output file(s).

-R west, east, south, and north specify the Region of interest. To specify boundaries in degrees and minutes [and seconds], use the dd:mm[:ss] format.

OPTIONS

-A Features with an area smaller than min area in km^2 or of hierarchical level higher than max level will not be extracted [Default is 0/4 (all features)].

-D Selects the resolution of the data set to use (f,h,i,l, and c). The resolution drops off by 80% between data sets. [Default is l].

-I To extract rivers. Specify the type of rivers. Choosefrom the list of river types below. Repeat option -I as often as necessary.

1 = Permanent major rivers
2 = Additional rivers
3 = Additional major rivers
4 = Minor rivers
5 = Double lined rivers
6 = Intermittent rivers - major
7 = Intermittent rivers - additional
8 = Intermittent rivers - minor
9 = Major canals
10 = Minor canals

a = All rivers and canals (1-10)
r = All permanent rivers (1-4)
i = All intermittent rivers (6-8)
c = All canals (9-10)

Sun Release 4.1 Last change: 1 August 1995 1

-N To extract political boundaries. Specify the type of boundary. choose from tle list of boundaries below. Repeat option -N as often as necessary.

1 = National boundaries
2 = State boundaries within the Americas
3 = Marines boundaries
a = All boundaries (1-3)

-V Selects verbose mode, which will send progress reports to stderr [Default runs «silently»]

-W To extract coastlines.

-: Toggles between (longitude,latitude) and (latitude,longitude) output [Default is (longitude,latitude)].

EXAMPLES

To select the hight-resolution and write Africa shoreline to output file 'africa_shore_h', permanent major rivers and additional major rivers to output file 'africa_river_h' and national borders to output file 'africa_border_h', try

cdb2xy -Gafrica -R-30/3/-40/40 -Dh -I1 -I2 -N1 -W

DATABASE INFORMATION

See pscoast and the GMT Cookbook and Technical Reference Appendix K for further details.

BUGS

See pscoast for details.

SEE ALSO

gmtdefaults, gmt, pscoast

REFERENCES

Wessel, P., and W. H. F. Smith, 1995, The Generic Mapping Tools (GMT) version 3.0 Technical Reference & Cookbook, SOEST/NOAA.
Wessel, P., and W. H. F. Smith, 1991, Free software helps map and display data, EOS Trans. AGU, 72, 441.

AUTHOR

Germinal Gabalda (gabalda@gravi.bondy.orstom.fr)
Geophysic Laboratory
ORSTOM - (F - 93143 Bondy Cedex)

Based on pscoast.c (version 3.0) of Paul Wessel

Sun Release 4.1 Last change: 1 August 1995 1

```
/*
 * cdb2xy extract data from a hight-resolution coastline database. Three
 * data sets are considered: shores, borders, and rivers. Each of these
 * data sets come in 5 different resolutions. The lower resolution files
 * are derived from the full resolution data using the Douglas-Peucker (DP)
 * line reduction algorithm. By giving a tolerance in km, the algorithm
 * will remove points that do not depart more than the tolerance from the
 * straight line that would connect the points if the point in question was
 * removed. The resolutions are:
 *
 * full            - The complete World Vector Shoreline + CIA data base
 * high            - DP reduced with tolerance = 0.2 km
 * intermediate    - DP reduced with tolerance = 1 km
 * low             - DP reduced with tolerance = 5 km
 * crude  - DP reduced with tolerance = 25 km
 *
 * If selected, cdb2xy will open the desired binned shoreline file and the
 * shoreline may be extract. Political boundaries and rivers may be extracted too.
 * cdb2xy writes out xy-doublets in ASCII format to output file(s). Segments are
 * separated by the record '> level'.
 *
 * Author:         Germinal Gabalda - Orstom (Geophysic Laboratory)
 *                 Email: gabalda@gravi.bondy.orstom.fr
 * Date:           1-AUG-1995
 * History:        Based on pscoast.c (version 3.0) of Paul Wessel
 * Version:        1.0
 *
 */

#include "gmt.h"
#include "gmt_shore.h"                  /* Defines shore structures */

struct SHORE c;
struct BR b, r;

char *shore_resolution[5] = {"full", "high", "intermediate", "low", "crude"};

main (argc, argv)
int argc;
char **argv; {
        int i, j, n, np, nb_in, ind, bin, base = 3;
        int max_level = MAX_LEVEL, k;
        int blevels[N_BLEVELS], n_blevels = 0, rlevels[N_RLEVELS], n_rlevels = 0;

        BOOLEAN         error = FALSE, got_fh[2];
        BOOLEAN         coast = FALSE, river = FALSE, border = FALSE;

        BOOLEAN selec_region = TRUE;

        double    west = 0.0, east = 0.0, south = 0.0, north = 0.0;
        double min_area = 0.0, step = 0.01;

        char res = 'l', key[5], *string, *outfile, file[80];

        struct POL *p;

        FILE *fp;

        memset ((char *)rlevels, 0, N_RLEVELS * sizeof (int));
        memset ((char *)blevels, 0, N_BLEVELS * sizeof (int));

        project_info.region = TRUE;

        outfile = CNULL;
```

```
/* Check and interpret the command line arguments */

for (i = 1; i < argc; i++) {
        if (argv[i][0] == '-') {
                switch(argv[i][1]) {

                        /* Common parameters */

                        case '\0':
                                gmt_quick = TRUE;
                                break;
                        case ':':
                                gmtdefs.xy_toggle = TRUE;
                                break;
                        case 'R':
                                selec_region = TRUE;
                                error += get_arg_R (argv[i], &west, &east, &south, &north);
                                break;
                        case 'V':
                                gmtdefs.verbose = TRUE;
                                break;
                        case 'G':
                                outfile = &argv[i][2];
                                break;
                        case 'A':
                                n = sscanf (&argv[i][2], "%lf/%d", &min_area, &max_level);
                                if (n == 1) max_level = MAX_LEVEL;
                                break;
                        case 'D':
                                res = argv[i][2];
                                switch (res) {
                                        case 'f':
                                                base = 0;
                                                break;
                                        case 'h':
                                                base = 1;
                                                break;
                                        case 'i':
                                                base = 2;
                                                break;
                                        case 'l':
                                                base = 3;
                                                break;
                                        case 'c':
                                                base = 4;
                                                break;
                                        default:
                                                fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR
                                                        -D option:  Unknown modifier %c\n", argv[i][2]);
                                                break;
                                }
                                break;

                        case 'N':
                                if (!argv[i][2]) {
                                        fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR:  -N
                                        option takes two arguments\n");
                                        error++;
                                        continue;
                                }
                                border = TRUE;
                                strcpy (key, &argv[i][2]);
                                string = CNULL;
```

```
                        switch (key[0]) {
                                case 'a':
                                        for (k = 0; k < N_BLEVELS; k++) blevels[k] = TRUE;
                                        break;
                                default:
                                        k = atoi (key) - 1;
                                        if (k < 0 || k >= N_BLEVELS) {
                                                fprintf (stderr, "cdb2xy: GMT SYNTAX
                                                    ERROR -N option: Feature not in list!\n");
                                                error++;
                                        }
                                        else blevels[k] = TRUE;
                                        break;
                        }
                        break;

                case 'I':
                        if (!argv[i][2]) {
                                fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR:  -I option
                                                                takes two arguments\n");
                                error++;
                                continue;
                        }
                        river = TRUE;
                        strcpy (key, &argv[i][2]);
                        string = CNULL;

                        switch (key[0]) {
                                case 'a':
                                        for (k = 0; k < N_RLEVELS; k++) rlevels[k] = TRUE;
                                        break;
                                case 'r':
                                        for (k = 0; k < 4; k++) rlevels[k] = TRUE;
                                        break;
                                case 'i':
                                        for (k = 5; k < 8; k++) rlevels[k] = TRUE;
                                        break;
                                case 'c':
                                        rlevels[8] = rlevels[9] = TRUE;
                                        break;
                                default:
                                        k = atoi (key) - 1;
                                        if (k < 0 || k >= N_RLEVELS) {
                                                fprintf (stderr, "cdb2xy: GMT SYNTAX
                                                    ERROR -I option: Feature not in list!\n");
                                                error++;
                                        }

                                        else rlevels[k] = TRUE;
                                        break;
                        }
                        break;

                case 'W':
                        coast = TRUE;
                        break;

                default:        /* Options not recognized */
                        error = TRUE;
                        gmt_default_error (argv[i][1]);
                        break;
                }
        }
}
```

```
find_resolutions (got_fh);      /* Check to see if full &| high resolution coastlines are installed */

if (argc == 1 || gmt_quick) {  /* Display usage */
        fprintf (stderr,"cdb2xy - Extracts shorelines, rivers, and borders from Coastline DataBase\n\n");
        fprintf (stderr,"usage: cdb2xy -G<outfile> -R<west>/<east>/<south>/<north>
                                                        [-A<min_area>[/<max_level>]] \n");
        fprintf (stderr, "      [-D<resolution>] [-I<feature>] [-N<feature>] [-V] [-W] [-:]\n");

        if (gmt_quick) exit (-1);

        fprintf (stderr, "     -G specifies generic name of output file(s)\n");
        fprintf (stderr, "     -R specifies the min/max coordinates of data region in user units.\n");
        fprintf (stderr, "      Use dd:mm[:ss] format for regions given in degrees and minutes [and seconds].\n");
        fprintf (stderr, "\n\tOPTIONS:\n");
        fprintf (stderr, "-A features smaller than <min_area> (in km^2) or of higher level (0-4) than
                                                                <max_level>\n");
        fprintf (stderr, " will be skipped [0/4 (4 means lake inside island inside lake)]\n");
        fprintf (stderr, " -D Choose one of the following resolutions:\n");
        if (got_fh[0]) fprintf (stderr, "               f - full resolution (may be very slow for large regions)\n");
        if (got_fh[1]) fprintf (stderr, "               h - high resolution (may be slow for large regions)\n");
        fprintf (stderr, "        i - intermediate resolution\n");
        fprintf (stderr, "        l - low resolution [Default]\n");
        fprintf (stderr, "        c - crude resolution, for busy plots that need crude continent outlines only\n");
        fprintf (stderr, "     -I extracts rIvers. Choose from the features below. Repeat the -I option as many
                                                                times as needed\n");
        fprintf (stderr, "          1 = Permanent major rivers\n");
        fprintf (stderr, "          2 = Additional major rivers\n");
        fprintf (stderr, "          3 = Additional rivers\n");
        fprintf (stderr, "          4 = Minor rivers\n");
        fprintf (stderr, "          5 = Double lined rivers\n");
        fprintf (stderr, "          6 = Intermittent rivers - major\n");
        fprintf (stderr, "          7 = Intermittent rivers - additional\n");
        fprintf (stderr, "          8 = Intermittent rivers - minor\n");
        fprintf (stderr, "          9 = Major canals\n");
        fprintf (stderr, "         10 = Minor canals\n");
        fprintf (stderr, "          a = All rivers and canals (1-10)\n");
        fprintf (stderr, "          r = All permanent rivers (1-4)\n");
        fprintf (stderr, "          i = All intermittent rivers (6-8)\n");
        fprintf (stderr, "          c = All canals (9-10)\n");
        fprintf (stderr, "     -N extracts boundaries. Choose from the features below. Repeat the -N option as
                                                                many times as needed\n");
        fprintf (stderr, "      Choose from the features below.  Repeat the -N option as many times as needed\n");
        fprintf (stderr, "          1 = National boundaries\n");
        fprintf (stderr, "          2 = State boundaries within the Americas\n");
        fprintf (stderr, "          3 = Marine boundaries\n");
        fprintf (stderr, "          a = All boundaries (1-3)\n");
        fprintf (stderr, "     -V Run in verbose mode [%s].\n", gmt_choice[gmtdefs.verbose]);
        fprintf (stderr,"      -W extracts shorelines.\n");
        fprintf (stderr, "     -: lat/lon output rather than lon/lat [%s].\n",gmt_choice[gmtdefs.xy_toggle]);
        exit (-1);
}
/* Check that the options selected are mutually consistant */

if (!selec_region) {
        fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR:  Must specify -R option\n");
        error++;
}
if (!outfile) {
        fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR -G option:  Must specify output file\n");
        error++;
}
if (!(coast || border || river)) {
        fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR:  Must specify at least one of -I, -N, and -W\n");
        error++;
}
```

```
if (river) {
        for (k = 0; k < N_RLEVELS; k++) {
                if (!rlevels[k]) continue;
                rlevels[n_rlevels] = k + 1;
                n_rlevels++;
        }
}

if (border) {
        for (k = 0; k < N_BLEVELS; k++) {
                if (!blevels[k]) continue;
                blevels[n_blevels] = k + 1;
                n_blevels++;
        }
}

if (error) exit (-1);

region_init (west, east, south, north);

if (coast && gmt_init_shore (res, &c, project_info.w, project_info.e, project_info.s, project_info.n))  {
        fprintf (stderr, "cdb2xy: %s resolution shoreline data base not installed\n", shore_resolution[base]);
        coast = FALSE;
}

if (border && gmt_init_br ('b', res, &b, project_info.w, project_info.e, project_info.s, project_info.n)) {
        fprintf (stderr, "cdb2xy: %s resolution political boundary data base not installed\n",
                                                                                shore_resolution[base]);
        border = FALSE;
}

if (river && gmt_init_br ('r', res, &r, project_info.w, project_info.e, project_info.s, project_info.n)) {
        fprintf (stderr, "cdb2xy: %s resolution river data base not installed\n", shore_resolution[base]);
        river = FALSE;
}

if (! (coast || border || river)) {
        fprintf (stderr, "cdb2xy: No databases available - aborts\n");
        exit (-1);
}

if (project_info.w < 0.0 && project_info.e <= 0.0) {        /* Temporarily shift boundaries */
        project_info.w += 360.0;
        project_info.e += 360.0;
}


if (coast) {                /* Extracts shorelines */

        sprintf (file, "%s_shore_%c\0", outfile, res);
        if ((fp = fopen (file, "w")) == NULL) {
                fprintf(stderr,"cdb2xy: Cannot open w %s\n", file);
                exit (-1);
        }

        if (gmtdefs.verbose) fprintf (stderr, "cdb2xy: Adding Shorelines...\n");

        for (ind = 0; ind < c.nb; ind++) {        /* Loop over necessary bins only */

                bin = c.bins[ind];
                gmt_get_shore_bin (ind, &c, min_area, max_level);

                if (gmtdefs.verbose) fprintf (stderr, "cdb2xy: Working on block # %5d\r", bin);

                if (c.ns == 0) continue;        /* Extracts shorelines, no need to assemble polygons */
```

```
                np = ggp_assemble_shore (&c, &p);

                for (i = 0; i < np; i++) copy_segment (fp, gmtdefs.xy_toggle, i, p);

                /* Free up memory */

                free_polygons (p, np);
                free ((char *)p);

                free_shore (&c);

        }
        shore_cleanup (&c);
        fprintf(fp, ">\n");
        fclose (fp);
}

if (gmtdefs.verbose) fprintf (stderr, "\n");

if (river) {            /* Extracts rivers */

        sprintf (file, "%s_river_%c\0", outfile, res);
        if ((fp = fopen (file, "w")) == NULL) {
                fprintf(stderr,"cdb2xy: Cannot open w %s\n", file);
                exit (-1);
        }

        if (gmtdefs.verbose) fprintf (stderr, "cdb2xy: Adding Rivers...");

        for (ind = 0; ind < r.nb; ind++) {          /* Loop over necessary bins only */

                bin = r.bins[ind];
                gmt_get_br_bin (ind, &r, rlevels, n_rlevels);

                if (r.ns == 0) continue;

                np = ggp_assemble_br (&r, &p);

                for (i = 0; i < np; i++) copy_segment (fp, gmtdefs.xy_toggle, i, p);

                /* Free up memory */

                free_br (&r);
                for (k = 0; k < np; k++) {
                        free ((char *)p[k].lon);
                        free ((char *)p[k].lat);
                }
                free ((char *)p);
        }
        br_cleanup (&r);
        fprintf(fp, ">\n");
        fclose (fp);

        if (gmtdefs.verbose) fprintf (stderr, "\n");
}

if (border) {           /* Extracts borders */

        if (gmtdefs.verbose) fprintf (stderr, "cdb2xy: Adding Borders...");

        /* Must resample borders because some points may be too far apart and look like 'jumps' */

        step = MAX (fabs(project_info.w - project_info.e), fabs (project_info.n - project_info.s)) / 4.0;
```

```
                sprintf (file, "%s_border_%c\0", outfile, res);
                if ((fp = fopen (file, "w")) == NULL) {
                        fprintf(stderr,"cdb2xy: Cannot open w %s\n", file);
                        exit (-1);
                }

                for (ind = 0; ind < b.nb; ind++) {      /* Loop over necessary bins only */

                        bin = b.bins[ind];
                        gmt_get_br_bin (ind, &b, blevels, n_blevels);

                        if (b.ns == 0) continue;

                        np = ggp_assemble_br (&b, &p);

                        for (i = 0; i < np; i++) copy_segment (fp, gmtdefs.xy_toggle, i, p);

                        /* Free up memory */

                        free_br (&b);
                        for (k = 0; k < np; k++) {
                                free ((char *)p[k].lon);
                                free ((char *)p[k].lat);
                        }
                        free ((char *)p);
                }
                br_cleanup (&b);
                fprintf(fp, ">\n");
                fclose (fp);

                if (gmtdefs.verbose) fprintf (stderr, "\n");
        }
}

int get_arg_R (item, w, e, s, n)
char *item;
double *w, *e, *s, *n; {
    char *text, string[100];

    /* get_arg_R interprets the command line for the unique option -R  */

    int i, error = 0;
    double *p[4], ddmmss_to_degree ();

    switch (item[1]) {
        case 'R':
            p[0] = w;    p[1] = e;    p[2] = s;    p[3] = n;

            i = 0;
            strcpy (string, &item[2]);
            text = strtok (string, "/");
            while (text) {
                *p[i] = ddmmss_to_degree (text);
                i++;
                text = strtok (CNULL, "/");
            }
            if ((i < 4) || (*p[0] >= *p[1]) || (*p[2] >= *p[3]) ) {
                error++;
                fprintf (stderr, "cdb2xy: GMT SYNTAX ERROR -R option.  Correct syntax:\n");
                fprintf (stderr, "\t-R<xmin>/<xmax>/<ymin>/<ymax>], dd:mm format ok\n");
            }
            break;
    }
        return (error);
}
```

```
int whereiam (lon, lat)
double lon, lat; {

        int where=0;

        if (project_info.w >= 0.0) {
                if (lon < project_info.w) {
                        where=4;
                        if (lat > project_info.n) where=1;
                        if (lat < project_info.s) where=6;
                }
                else    if (lon > project_info.e) {
                                where=5;
                                if (lat > project_info.n) where=3;
                                if (lat < project_info.s) where=8;
                        }
                        else {
                                if (lat > project_info.n) where=2;
                                if (lat < project_info.s) where=7;
                        }
        }
        else {
                if (lon > 180.0) {
                        if (lon < 360.0 + project_info.w) {
                                where=4;
                                if (lat > project_info.n) where=1;
                                if (lat < project_info.s) where=6;
                        }
                        else {
                                if (lat > project_info.n) where=2;
                                if (lat < project_info.s) where=7;
                        }
                }
                else    if (lon > project_info.e) {
                                where=5;
                                if (lat > project_info.n) where=3;
                                if (lat < project_info.s) where=8;
                        }
                        else {
                                if (lat > project_info.n) where=2;
                                if (lat < project_info.s) where=7;
                        }
        }

    return (where);
}
```

```
int copy_segment (fp, toggle, ind, p)
FILE *fp;
BOOLEAN toggle;
struct POL *p;
int ind; {
        double lon_b, lat_b;
        int i=0, where=0, last_where, point_in=0, point_out=0, begin_seg=1;

        for (i=0; i < p[ind].n; i++) {

                point_in++;
                last_where = where;
                where = whereiam (p[ind].lon[i], p[ind].lat[i]);

                if (where != 0) { /* point out the region */

                        if (point_in > 1) {             /* last point in the region */

                                border_point (where, ind , i, i-1, p, &lon_b, &lat_b);
                                if (toggle) fprintf(fp, "%lf %lf\n", lat_b, lon_b);
                                else fprintf(fp, "%lf %lf\n", lon_b, lat_b);

                                begin_seg = 1;
                        }
                        point_in = 0;
                        point_out = 1;
                }
                else { /* point in the region */

                        if (begin_seg) { /* first point of the segment */
                                fprintf(fp, "> %d\n", p[ind].level);
                                begin_seg = 0;
                        }

                        if (point_out) { /* last point out the region */
                                border_point (last_where, ind, i-1, i, p, &lon_b, &lat_b);

                                if (toggle) fprintf(fp, "%lf %lf\n", lat_b, lon_b);
                                else fprintf(fp, "%lf %lf\n", lon_b, lat_b);

                                point_in++;
                                point_out = 0;
                        }

                        if (toggle) fprintf(fp, "%lf %lf\n", p[ind].lat[i], p[ind].lon[i]);
                        else fprintf(fp, "%lf %lf\n", p[ind].lon[i], p[ind].lat[i]);
                }
        }
}

int border_point (where, i , i_out, i_in, p, x_border, y_border)
double *x_border, *y_border;
struct POL p[];
int where, i , i_out, i_in; {

        double west, x_out, y_out, x_in, y_in;

        x_out = p[i].lon[i_out];
        y_out = p[i].lat[i_out];

        x_in = p[i].lon[i_in];
        y_in = p[i].lat[i_in];

        if (project_info.w < 0.0) west = 360.0 + project_info.w; else west = project_info.w;
```

```
        if (where==1 || where==4 || where==6) {
                *y_border = y_in + ((y_out-y_in)/(x_out-x_in)*(west-x_in));
                *x_border = west;
        }
        if (where==3 || where==5 || where==8) {
                *y_border = y_in + ((y_out-y_in)/(x_out-x_in)*(project_info.e-x_in));
                *x_border = project_info.e;
        }
        if (where==6 || where==7 || where==8) {
                *x_border = x_in + ((x_out-x_in)/(y_out-y_in)*(project_info.s-y_in));
                *y_border = project_info.s;
        }
        if (where==1 || where==2 || where==3) {
                *x_border = x_in + ((x_out-x_in)/(y_out-y_in)*(project_info.n-y_in));
                *y_border = project_info.n;
        }
}

int ggp_assemble_shore (c, pol)
struct SHORE *c;
struct POL *pol[];
{
        struct POL *p;
        int id, P = 0;

        p = (struct POL *) memory (CNULL, c->ns, sizeof (struct POL), "ggp_assemble_shore");

        for (id = 0; id < c->ns; id++) {
                p[id].lon = (double *) memory (CNULL, (int)c->seg[id].n, sizeof (double), "ggp_assemble_shore");
                p[id].lat = (double *) memory (CNULL, (int)c->seg[id].n, sizeof (double), "ggp_assemble_shore");
                p[id].n = copy_to_shore_path (p[id].lon, p[id].lat, c, id);
                p[id].level = c->seg[id].level;
                p[id].interior = FALSE;
        }

        *pol = p;
        return (c->ns);
}

int ggp_assemble_br (c, pol)
struct BR *c;
struct POL *pol[];
{
    struct POL *p;
    int id;

    p = (struct POL *) memory (CNULL, c->ns, sizeof (struct POL), "ggp_assemble_br");

    for (id = 0; id < c->ns; id++) {
        p[id].lon = (double *) memory (CNULL, (int)c->seg[id].n, sizeof (double), "ggp_assemble_br");
        p[id].lat = (double *) memory (CNULL, (int)c->seg[id].n, sizeof (double), "ggp_assemble_br");
        p[id].n = copy_to_br_path (p[id].lon, p[id].lat, c, id);
        p[id].level = c->seg[id].level;
    }

    *pol = p;
    return (c->ns);
}
```

```
int region_init (west, east, south, north)
double west, east, south, north; {

	if (west == east && south == north) {
		fprintf (stderr, "cdb2xy: GMT Fatal Error: No region selected\n");
		exit (-1);
	}

	if (south < -90.0) south = -90.0;
	if (north > 90.0) north = 90.0;

	if (west < -720.0) west = -720.0;
	if (east < -720.0) east = -720.0;

	if (west < -360.0) {
		west += 360.0;
		east += 360.0;
	}

	if (west > 720.0) west = 720.0;
	if (east > 720.0) east = 720.0;

	if (east > 360.0) {
		west -= 360.0;
		east -= 360.0;
	}

	if (fabs (west - east) >= 360.0) {
		west = 0;
		east = 360;
	}

	project_info.w = west;  project_info.e = east;  project_info.s = south; project_info.n = north;

}
```

- Wessel, P., and W. H. F. Smith, New version of the Generic Mapping Tools released, *EOS Trans. Amer. Geophys. U.*, vol. 76, pp. 329, 1995.

- Wessel, P., and W. H. F. Smith, 1995, The Generic Mapping Tools (GMT) version 3.0 Technical Reference & Cookbook, SOEST/NOAA.

- Wessel, P., and W. H. F. Smith, New version of the Generic Mapping Tools released, *EOS Trans. Amer. Geophys. U. electronic supplement*, http://www.agu.org/eos_elec95154e.html,1995.

- Wessel, P., and W. H. F. Smith, Free software helps map and display data, *EOS Trans. Amer. Geophys. U.*, vol. 72, pp. 441, 445-446, 1991.