NOTES TECHNIQUES

SCIENCES DE LA TERRE

GÉOLOGIE-GÉOPHYSIQUE

N° 26

2002

Diffusion automatique de données GPS, sismologiques et magnétiques par autodrm : une première mise en œuvre au Centre IRD de Nouméa

Pierre LEBELLEGARD

Institut de recherche
pour le développement

/Lebellegard, P.

Diffusion automatique de données GPS, sismologiques et magnétiques par autodrm : une première mise en œuvre au Centre IRD de Nouméa

Nouméa : IRD. Décembre 2002. 50 p.
Notes Tech. : Sci. Terre ; Géol.-Géophys. ; 26

DONNEES NUMÉRIQUES ; DIFFUSION ; SISMOLOGIE ; GPS. SYSTEME DE POSITIONNEMENT GLOBAL ; MAGNETISME / NOUVELLE CALEDONIE ; FUTUNA

Imprimé par la Service de Reprographie
Centre IRD de Nouméa
Décembre 2002

1 - Introduction

Le nombre de stations d'acquisitions de données en continu, opérant dans le Sud-Ouest Pacifique, et gérées à partir du centre IRD de Nouméa, n'a cessé de croître avec le temps. Le type des données s'est également diversifié : actuellement, le Centre IRD gère des données sismologiques, issues des deux stations de Port-Laguerre (PLG) et du mont Dzumac (DZM). Il gère également des données géodésiques (GPS) provenant des quatre stations permanentes de Lifou (LPIL), Koumac (KOUC), Nouméa (NOUM, cette station appartenant au Territoire de la Nouvelle-Calédonie, mais l'IRD a en charge la diffusion des données), et de Futuna (FTNA). Le système de diffusion décrit dans cette note technique permet en outre la diffusion des données des observatoires magnétiques de l'IRD à M'Bour (MBO, Sénégal), et Bangui (BNG, Centrafrique).

La présente note technique a pour objet de présenter le système de diffusion de données à la demande (Automatic Data Request Manager, Autodrm) mis en œuvre au centre IRD de Nouméa, d'en décrire l'utilisation pour la récupération de données, et de présenter les détails de sa mise en œuvre, afin d'en permettre la maintenance et les développements futurs.

1.1 – Disponibilité des données

Le choix du système de mise à disposition des données a été dicté par des impératifs liés à la sismologie : lorsqu'un séisme se produit, la localisation du séisme, la détermination de sa magnitude et de son mécanisme au foyer nécessitent les données de plusieurs stations sismologiques. Il fallait donc développer un système opérationnel en permanence, qui permette aux stations sismologiques de transmettre leurs données de manière automatique, et dans un délai aussi proche du temps réel que permis par les moyens de communication mis en œuvre.

En ce qui concerne les données GPS, la forme sous laquelle elles sont mises à disposition sur Internet est standard : il s'agit de fichiers de 24 heures (de 0 à 24h TU). On les trouve sur Internet dans un délai de un à quatre jours (il s'agit du délai concernant les stations permanentes de l'IGS[2]). Quant aux données magnétiques, les fichiers à mettre à disposition sont des fichiers mensuels. La contrainte de disponibilité temps réel ou quasi temps réel ne se pose donc que pour les données de sismologie.

1.2 – Mise à disposition automatique de données : Automatic Data request Manager (Autodrm).

Le système que l'on rencontre de plus en plus fréquemment dans les échanges de données entre stations sismologiques est l'autodrm. A notre avis, ce système combine simplicité, fiabilité, et sécurité des données et des systèmes qui les hébergent. Le principe en est simple : il s'agit d'un programme d'interprétation de messages électroniques (courriels). Ceux-ci contiennent des requêtes qui suivent une syntaxe particulière (cf. plus bas), et qui précisent tous les paramètres permettant d'obtenir les données souhaitées. A chaque minute[3], l'autodrm examine sa boîte aux lettres, dépouille les messages, et exécute les traitements spécifiés par chaque requête valide.

---

[2] International Geodetic Service, voir http://www.igscb.jpl.nasa.gov

[3] Il s'agit ici de l'intervalle le plus court permis par la crontab, puisque notre implémentation est réalisée sur des stations de travail SUN (Solaris) : la planification des tâches (crontab) ne permettant pas de préciser la seconde, une tâche ne peut être exécutée au plus que chaque minute.

2 – Syntaxe et envoi des requêtes Autodrm

Sont essentiellement détaillées ici les commandes modifiées dans cette première mise en œuvre, par rapport à l'autodrm « standard » original[4], pour adapter le comportement de l'autodrm à nos besoins propres. Ce sont les commandes qui principalement permettent de modifier le type des données souhaitées, le format de celles-ci, d'obtenir la liste des stations disponibles, de choisir la liste des stations, l'intervalle de temps considéré, etc.

Un message autodrm doit être envoyé à l'adresse suivante : autodrm@ird.nc; de manière assez répandue, les adresses des différents autodrm suivent cette syntaxe, autodrm@machine.domaine. Un message autodrm doit être envoyé en texte seul, sans caractère de contrôle. Une des principales causes d'échec est l'envoi de messages HTML, ou « texte enrichi ». Si l'on utilise un programme tel que Eudora ou Outlook Express, il faut le configurer pour envoyer des messages en **texte seul**. Le problème ne se pose pas a priori si les messages sont envoyés d'une station de travail UNIX. Un message contient un (ou plusieurs) ensemble(s) de requêtes, ou blocs. Un bloc est délimité par les requêtes BEGIN et STOP.

Principales requêtes :

BEGIN : Doit être la première ligne de toute requête.

E-MAIL <adresse.de.la.réponse> : Adresse courriel à laquelle la réponse doit être expédiée. Par défaut, la réponse est envoyée à l'expéditeur du message, mais par cette requête celui-ci peut demander à ce qu'elle soit envoyée à un tiers.

E-MAIL <adresse.de.la.réponse> : Idem.

EMAIL <adresse.de.la.réponse> : Idem.

AIDE : Envoie un fichier d'aide (manuel utilisateur), en français, sur les commandes disponibles.

GUIDE : Envoie un fichier d'aide (manuel utilisateur), en anglais, sur les commandes disponibles.

HELP : Idem.

INFOR : Idem.

DATA_TYPE <type> : Permet de choisir le type de données auxquelles on s'intéresse. Actuellement, trois types sont acceptés:

SIS (défaut) : données sismologiques,
GPS : données géodésiques (GPS)
MAG : données magnétiques.

[4] La version de base que nous avons adaptée a été téléchargée du site du Service Sismologique Suisse, à l'adresse http://seismo.ethz.ch/autodrm/; elle consiste principalement en un programme de dépouillement des messages, écrit en Fortran. La taille de celui-ci, environ 4.000 lignes, n'a pas permis de l'inclure dans cette note technique.

A chaque exécution de cette commande, la liste des stations disponibles (voir **SLIST**) est actualisée, mais **la liste des stations choisies est remise à zéro.** Il faut donc refaire une exécution de **STA_LIST** (voir **STA_LIST**).

DATA_FORMAT <format> : Une fois le type de données spécifié, permet de choisir le format de sortie. Actuellement, sont disponibles:

SISMALP (défaut) et **SAC** pour les données sismologiques,
RINEX et **HATANAKA** (défaut) (format codé et compressé) pour les données GPS,
BIN (défaut) pour les données magnétiques.

TIME [DateDeDebut] **TO** [DateDeFin] : Définition des dates de début et de fin, selon le format: 1994/02/24 16:23:50.20. Si DateDeFin est manquante, la date courante est prise par défaut. Au lieu de la commande **TIME**, vous pouvez également utiliser les commandes **DATE1** et **DATE2**. Les valeurs par défaut sont la date courante pour DateDeFin, et DateDeFin-24heures pour DateDeDebut.

DATE1 aaaamsjjhhmm : Définit le début de l'intervalle de temps, où 'aaaa' précise l'année, 'ms' le mois, 'jj' le jour, 'hh' l'heure, et 'mm' la minute. (Exemple: 24 février 1992 à 15h46 s'écrira: **DATE1** 199202241546). Valeur par défaut: la date courante-24 heures.

DATE2 aaaamsjjhhmm : Définit la fin de l'intervalle de temps, où 'aaaa' précise l'année, 'ms' le mois, 'jj' le jour, 'hh' l'heure, et 'mm' la minute. (Exemple: 24 février 1992 à 15h46 s'écrira: **DATE2** 199202241546). Valeur par défaut: la date courante.

STA_LIST ABC [,DEFG] [,HIJK]: Définit la liste des stations souhaitées (séparées par des virgules). Pas de valeur par défaut.

TITLE <votre sujet>: Le sujet du courriel que vous recevrez en réponse. Un sujet est spécifié par défaut (IRD Nouméa AUTO_DRM Response).

SUBJE <votre sujet> : Idem.

DETEC : Envoie la liste des détections effectuées dans l'intervalle de temps spécifié (dans le cas des données sismologiques, pour chaque détection, nous enregistrons une minute de pré-événement, et trois minutes de post-événement). Cette commande doit être précédée soit de la commande **TIME**, soit des commandes **DATE1** et **DATE2**. La réponse est envoyée par courriel, si sa taille est inférieure à 4 mégaoctets; sinon elle est placée dans un répertoire ftp (si la taille résultante est supérieure à 10 mégaoctets, la requête est rejetée), et une notification est envoyée par courriel à l'utilisateur, lui précisant comment récupérer le résultat de sa requête.

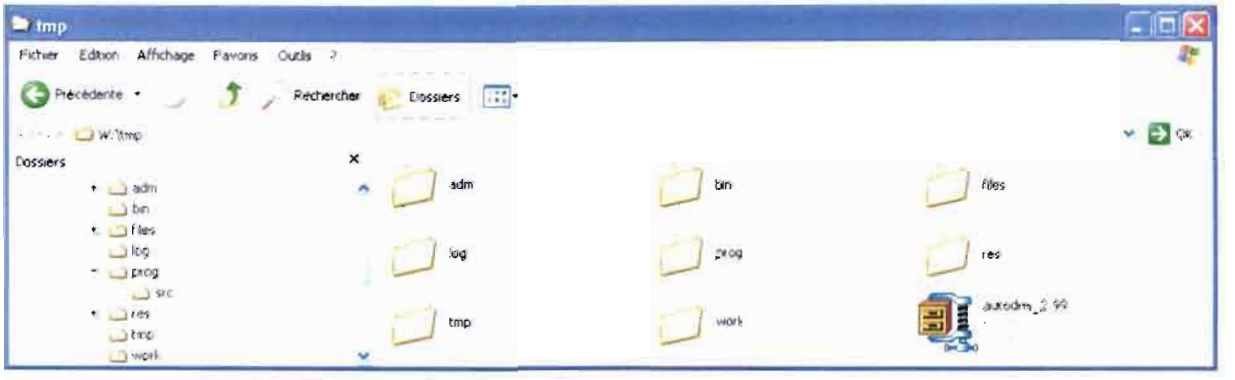
**** **IMPORTANT** **** Dans le cas de données géodésiques (GPS), ou magnétiques, on ne peut bien sûr pas parler de détections. Dans ces cas, ce sont les données elles-mêmes qui sont envoyées, si elles correspondent à l'intervalle de temps spécifié. Les conditions de taille de la réponse précisées plus haut s'appliquent également. Dans le cas des données géodésiques (GPS), les valeurs heure/minute précisées par les commandes **DATE1** et **DATE2** ou **TIME** ne sont pas prises en compte: elles sont considérées comme égales à zéro, puisque les fichiers GPS fournis sont de toute façon datés de 0 heures GMT à 24 heures GMT.

SLIST : Envoie une liste des stations pour lesquelles on dispose de données. La liste des stations contient également leurs coordonnées.

STOP : Doit être la dernière ligne de toute requête.

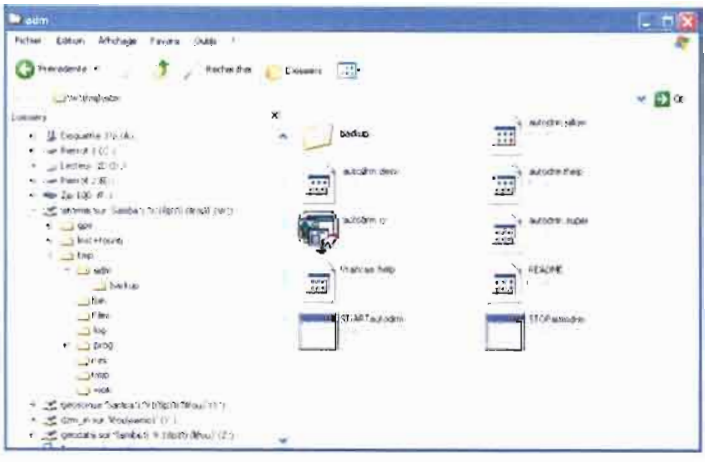
3 – Organisation des fichiers source et fonctionnement interne de l'autodrm

Pour pouvoir définir une planification des tâches (« crontab »), et donc une exécution répétée du programme de dépouillement, il a donc fallu créer un pseudo utilisateur sur une des machines SUN de l équipe de géophysique (« santo », en l'occurrence). C'est vers ce pseudo utilisateur que sont envoyés tous les courriels entrants. Nous employons l'expression de pseudo utilisateur, car à la différence des utilisateurs « normaux » connus sur tout le réseau grâce au NIS[5], le pseudo utilisateur autodrm n'est connu que sur la machine sur laquelle il est installé, ce qui renforce la sécurité des données dont il est propriétaire : pas plus que autodrm, ces données ne sont visibles de l 'extérieur, et **on ne peut y avoir accès qu'en utilisant les requêtes de l'autodrm décrites plus haut**. Le répertoire racine (« home directory ») de autodrm a la structure suivante :



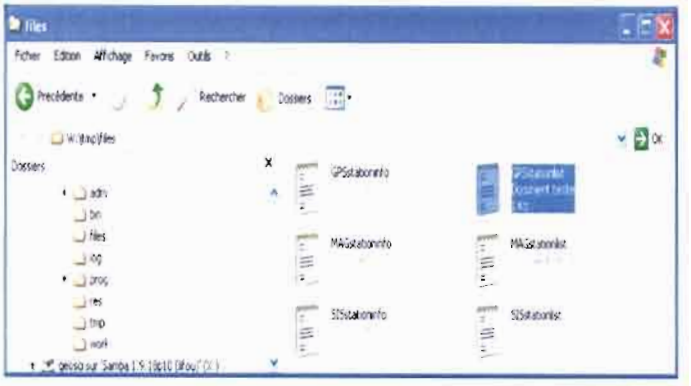
Le fichier autodrm_2.99 est l'archive de l'ensemble des fichiers nécessaires à l'installation, telle que trouvée sur le site web mentionné plus haut. Il contient entre autres fichiers, un fichier script install, qui permet de paramétrer l'installation en fonction de la version et du type de système UNIX avec lequel o travaille (HP-UX, Solaris, etc.).

Examen des différents sous-répertoires autodrm :

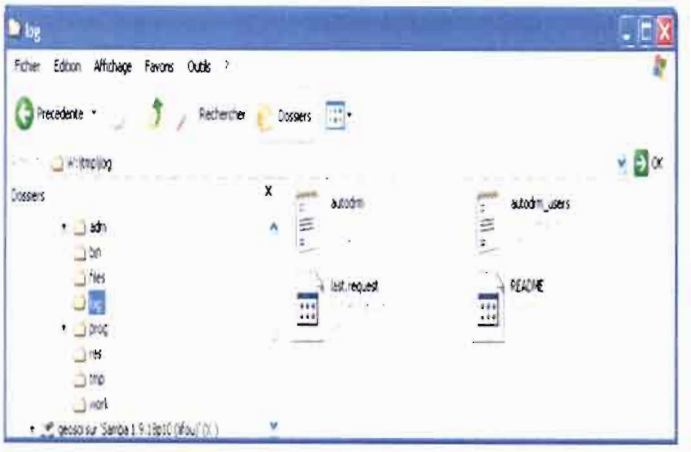


- Le répertoire **adm** contient la copie du fichier d'aide original (en anglais). Nous avons traduit ce fichier en français. Les autres fichiers permettent de limiter l'accès à l'autodrm.

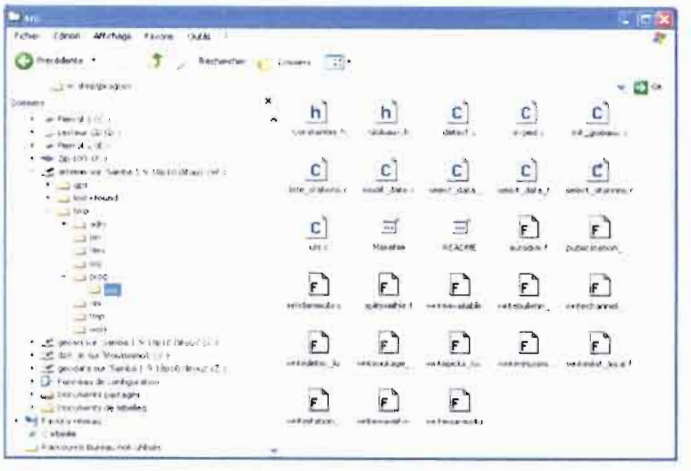
[5] Network Information Service, anciennement YP (« Pages Jaunes »).



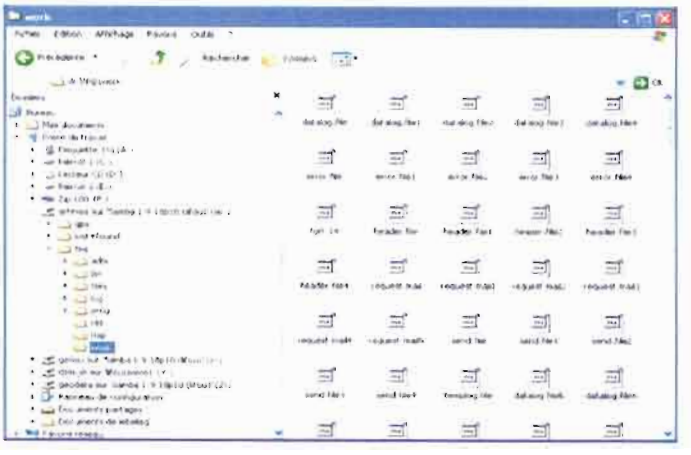
- Nous avons rajouté le répertoire **files** à la version originale : par exemple, le fichier **GPSstationlist** contient la liste (un par ligne) des acronymes des stations disponibles. **GPSstationinfo** contient quelques informations basiques (localisation, coordonnées) sur ces mêmes stations. De même pour les stations sismologiques et magnétiques.



- Le répertoire **log** contient essentiellement les statistiques d'utilisation de l'autodrm : requêtes reçues, réponses renvoyées, et quantité de données transmises.



- Le répertoire **prog\src** contient le code source (Fortran pour l'original, et C pour nos procédures) de l'autodrm, ainsi que le makefile qui permet de régénérer les exécutables après modification d'un fichier source



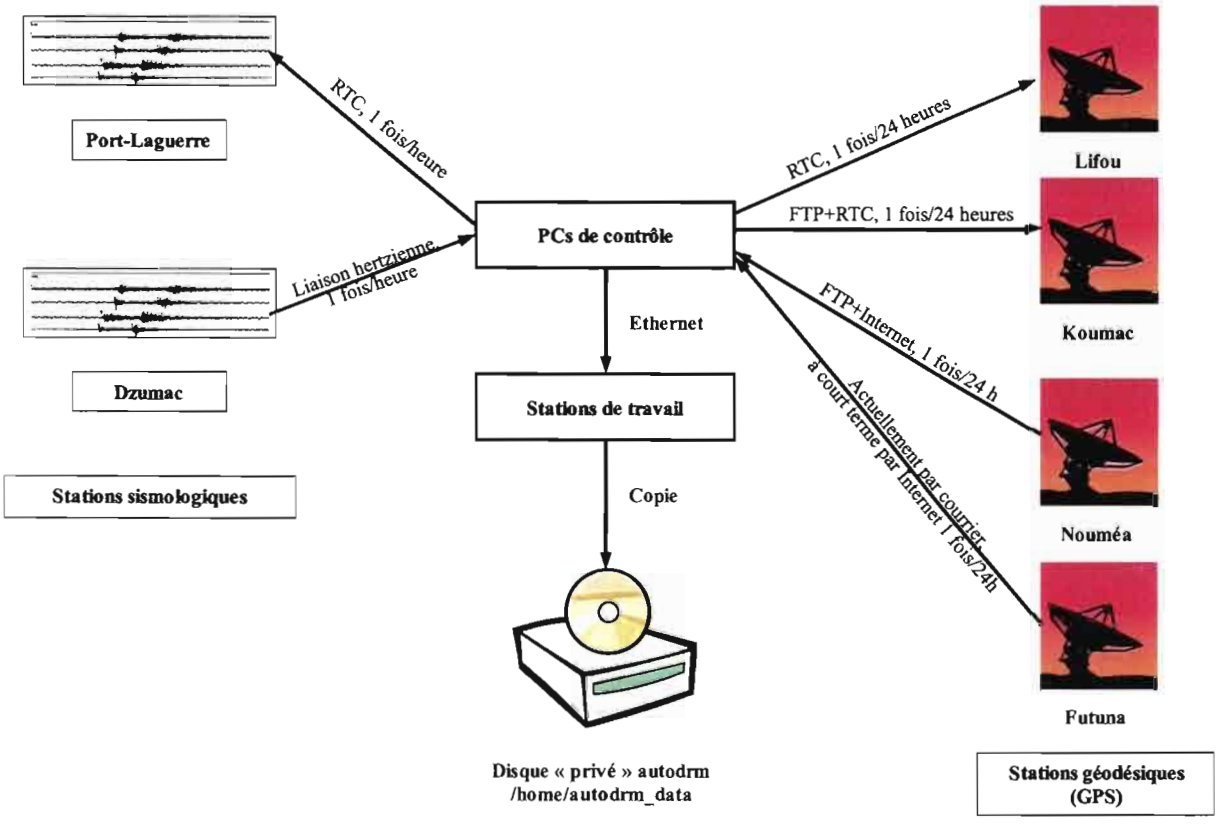
- Le répertoire de travail **work** renseigne sur les requêtes arrivées, leur traitement, et les éventuelles erreurs.

Enfin, le planificateur de tâches (« crontab ») permet de vérifier chaque minute la présence de nouveaux messages :

```
#! /bin/sh
# $HOME/cron/autodrm_1
#
# Run AutoDRM every minute, every day (if there is new mail for AutoDRM):
#
* * * * * `echo $HOME/prog/autodrm.com`
#
# check, if an old AutoDRM lock-file is here; if yes, remove it:
#
* * * * * `echo $HOME/prog/autodrm_watcher.com` >/dev/null 2>/dev/null
#
```

4 – Flot de données et gestion des données par l'autodrm

Le schéma suivant précise les données récupérées directement depuis le centre IRD de Nouméa (données GPS et sismologiques) :



Le sens de la flèche indique l'initiateur du transfert ; par exemple, pour le GPS de Lifou, l'initiateur du transfert est le PC de contrôle. Les données sont transférées directement vers les disques dédiés des stations de travail, pour utilisation directe, et elles sont copiées, sous le format le plus compact, sur le disque dédié de l'autodrm (/home/autodrm_data). Ainsi, quelle que soit l'opération effectuée avec l'autodrm, l'intégrité des données est assurée, ce qui est particulièrement important dans un système où on met des données à disposition de l'extérieur. Les données sont disponibles par l'autodrm au même moment que pour les

utilisateurs des stations de travail, c'est-à-dire moins de une heure après la fermeture du fichier pour les fichiers GPS, et moins de une heure après l'événement pour les séismes détectés.

Pour des raisons de gain de place, les données sont stockées sous un seul format, même si l'autodrm en propose plusieurs pour chaque type de données. Bien évidemment, a été choisi celui qui minimise l'espace disque, à savoir SISMALP pour les données sismologiques, et RINEX codé selon l'algorithme de Hatanaka, puis compressé, pour les données GPS.

5 – Conclusion

Il a été présenté ici un sous-ensemble, adapté à nos besoins, de ce que permet – en théorie – l'autodrm « standard », autrement dit, les requêtes de l'autodrm qui font l'objet d'un traitement spécifique (au moyen des procédures écrites en C). Le but était de permettre la mise à disposition le plus rapidement possible l'ensemble des données acquises sur le Centre IRD de Nouméa, tout en assurant au maximum la sécurité des données. On pourra par la suite étoffer la liste des requêtes acceptées, par exemple la requête FTP qui permet à l'autodrm d'envoyer directement sur un site ftp externe la liste des données demandées, ce qui permettra encore plus d'automatisation dans les traitements.

Signalons enfin qu'est en cours de développement un système utilisant cet autodrm pour mettre aussitôt à disposition sur le site web du centre IRD les tracés des événements sismiques détectés, quasiment en temps réel.

ANNEXE 1

Code source des procédures C

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <time.h>
#include "Constantes.h"
#include "Globaux.h"

void detect_()
{
	int	i, Status, InNumber;
	int	NbDetections;
	int	qt, aa, ms, jj;
	char	ch[MAXLEN];
	char	orig[MAXLEN];
	char	root[MAXLEN];
	char	*pointe;
	struct tm	*pt;
	struct tm	curr;
	time_t current;
	int	year, year_beg, year_end;
	DIR	*repertoire;
	struct dirent *ptfic;

	getcwd(orig, MAXLEN);

	switch (data_type)
	{
		/*
		 * Detection of seismological events is supposed to be stored in
SISMALP format,
		 * in yearly subdirectories. SISMALP means a couple of .ndx (index)
file and .sis file.
		 * For example, if an event occurs on july, 26, 2002 at 14:27, the
corresponding files
		 * will be 07261427.ndx and 07261427.sis, and will be found in the
following
		 * directory: SIS\DZM\2002, if requested station is DZM, and so on.
		 */
		case SIS:
			sprintf(ch, "### DETECTION ON SEISMOLOGICAL DATA *****\n");
			PSTANDARD(ch);

			pt = localtime (&begin);
			sprintf(ch, "### BEGIN: %s %d %s, %d %02d:%02d\n",
				ascii_day[pt->tm_wday],
				pt->tm_mday,
				ascii_month[pt->tm_mon],
				1900+pt->tm_year,
				pt->tm_hour,
				pt->tm_min);
			PSTANDARD(ch);
			year_beg = 1900+pt->tm_year;

			pt = localtime (&end);
			sprintf(ch, "### END: %s %d %s, %d %02d:%02d\n",
				ascii_day[pt->tm_wday],
				pt->tm_mday,
				ascii_month[pt->tm_mon],
				1900+pt->tm_year,
				pt->tm_hour,
				pt->tm_min);
			PSTANDARD(ch);
			year_end = 1900+pt->tm_year;

			if (NbSelectedStations == 0)
			{
				sprintf(ch, "### NO STATION SELECTED\n");
				PERROR(ch);
				Status = chdir(orig);
				return;
			}

			sprintf(ch, "### STATIONS SELECTED:\n");
			PSTANDARD(ch);
			for (i=0; i<NbSelectedStations; i++)
			{
				sprintf(ch, "### %s\n",
SelectedStations[i]);
				PSTANDARD(ch);
			}

			NbMails = 0;

			for (i=0; i<NbSelectedStations; i++)
			{
				NbDetections = 0;
				Status = chdir(DETECTIONS);
				if (Status != 0)
				{
					sprintf(ch, "### NO DETECTIONS
AVAILABLE FOR STATION %s\n",
						SelectedStations[i]);
					PERROR(ch);
					Status = chdir(orig);
					return;
				}
				Status = chdir("SIS");
				if (Status != 0)
				{
					sprintf(ch, "### NO DETECTIONS
AVAILABLE FOR STATION %s\n",
						SelectedStations[i]);
					PERROR(ch);
					Status = chdir(orig);
					return;
				}
```

```
                        Status = chdir(SelectedStations[i]);
                        if (Status != 0)
                        {
                                sprintf(ch, "### NO DETECTIONS AVAILABLE FOR
STATION %s\n",
                                        SelectedStations[i]);
                                PERROR(ch);
                                Status = chdir(orig);
                                return;
                        }
                        /*
                         * Fix root dir for that station
                         */
                        getcwd(root, MAXLEN);

                        for (year = year_beg; year <= year_end; year++)
                        {
                                sprintf(ch, "%04d", year);
                                repertoire = opendir(ch);
                                if (repertoire == NULL)
                                {
                                        sprintf(ch, "### NO DETECTIONS
AVAILABLE FOR STATION %s FOR YEAR %04d\n",
                                                SelectedStations[i], year);
                                        PERROR(ch);
                                        continue;
                                }
                                else
                                {
                                        sprintf(ch, "### DETECTIONS ARE
AVAILABLE FOR STATION %s FOR YEAR %04d\n",
                                                SelectedStations[i], year);
                                        PSTANDARD(ch);
                                }
                                for (;;)
                                {
                                         ptfic = readdir(repertoire);
                                         if (ptfic == NULL)
                                        {
                                                closedir(repertoire);
                                                break;
                                        }
                                        /*
                                         * Test only on ndx files
                                         */
                                        pointe = ptfic->d_name;
                                        if (strlen(pointe) != (size_t) 12)
                                                continue;
                                        if (*(pointe+8) != '.')     /* Le
point */
                                                continue;
                                        if (strstr(pointe, "ndx") == NULL)
                                                continue;
                                        /*
                                         * Test if this file fits the
requested interval
                                         */
                                        curr.tm_year   =       year-
1900;
                                        /*
                                         * Month
                                         */
                                        strcpy(ch, pointe); ch[2] =
'\0';
                                        curr.tm_mon    =
                                                atoi(ch) - 1;
                                        /*
                                         * Day
                                         */
                                        strcpy(ch, pointe+2); ch[2] =
'\0';
                                        curr.tm_mday   =
                                                atoi(ch);
                                        /*
                                         * Hour
                                         */
                                        strcpy(ch, pointe+4); ch[2] =
'\0';
                                        curr.tm_hour   =
                                                atoi(ch);
                                        /*
                                         * Minute
                                         */
                                        strcpy(ch, pointe+6); ch[2] =
'\0';
                                        curr.tm_min    =
                                                atoi(ch);
                                        /*
                                         * Seconde
                                         */
                                        curr.tm_sec    =       0;
                                        current = mktime (&curr);

                                        /*
                                         * Check interval
                                         */
                                        if (current < begin)
                                                continue;
                                        if (current > end)
                                                continue;

                                        /*
                                         * Current file fits the
interval
                                         */
```

```
                                        strcpy(event[NbMails].Station,
SelectedStations[i]);
                                        event[NbMails].year = year;
                                        strcpy(event[NbMails].NFile, pointe);
                                        if (NbMails > MAXEVENTS)
                                                goto finished;
                                        else ++NbMails;
                                }
                        }
                }
finished:;              if (NbMails == 0)
                {
                        sprintf(ch, "### NO SEISMIC EVENT DETECTED.\n");
                        PSTANDARD(ch);
                }
                else    if (NbMails >= MAXEVENTS)
                {
                        sprintf(ch, "###\n");
                        PSTANDARD(ch);
                        sprintf(ch, "###\t***** ERROR *****\n");
                        PSTANDARD(ch);
                        sprintf(ch, "###\n");
                        PSTANDARD(ch);
                        sprintf(ch, "### THE RESULT OF YOUR REQUEST EXCEEDS
MAXIMUM\n");
                        PSTANDARD(ch);
                        sprintf(ch, "### NUMBER (%d) OF FILES ALLOWED.\n",
MAXEVENTS);
                        PSTANDARD(ch);
                        sprintf(ch, "### WE SUGGEST YOU SELECT A SMALLER TIME
INTERVAL, OR\n");
                        PSTANDARD(ch);
                        sprintf(ch, "### A SMALLER NUMBER OF STATIONS.\n");
                        PSTANDARD(ch);
                }
                else
                {
                        sprintf(ch, "### %d SEISMIC EVENT(S) DETECTED.\n",
NbMails);
                        PSTANDARD(ch);
                        sprintf(ch, "### THEY WILL BE SENT TO YOU BY
SUBSEQUENT E-MAILS.\n");
                        PSTANDARD(ch);
                }
                break;
        case GPS:
                sprintf(ch, "### DISTRIBUTION OF GPS DATA *****\n");
                PSTANDARD(ch);

                pt = localtime (&begin);
                sprintf(ch, "### BEGIN: %s %d %s, %d %02d:%02d\n",
                        ascii_day[pt->tm_wday],
                        pt->tm_mday,
                        ascii_month[pt->tm_mon],
                        1900+pt->tm_year,
                        pt->tm_hour,
                        pt->tm_min);
                PSTANDARD(ch);
                year_beg = 1900+pt->tm_year;

                pt = localtime (&end);
                sprintf(ch, "### END: %s %d %s, %d %02d:%02d\n",
                        ascii_day[pt->tm_wday],
                        pt->tm_mday,
                        ascii_month[pt->tm_mon],
                        1900+pt->tm_year,
                        pt->tm_hour,
                        pt->tm_min);
                PSTANDARD(ch);
                year_end = 1900+pt->tm_year;

                if (NbSelectedStations == 0)
                {
                        sprintf(ch, "### NO STATION SELECTED\n");
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }

                sprintf(ch, "### STATIONS SELECTED:\n");
                PSTANDARD(ch);
                for (i=0; i<NbSelectedStations; i++)
                {
                        sprintf(ch, "### %s\n",
SelectedStations[i]);
                        PSTANDARD(ch);
                }

                NbMails = 0;

                for (i=0; i<NbSelectedStations; i++)
                {
                        NbDetections = 0;
                        Status = chdir(DETECTIONS);
                        if (Status != 0)
                        {
                                sprintf(ch, "### NO GPS DATA
AVAILABLE FOR STATION %s\n",
                                        SelectedStations[i]);
                                PERROR(ch);
                                Status = chdir(orig);
                                return;
                        }
                        Status = chdir("GPS");
                        if (Status != 0)
                        {
```

```
                        sprintf(ch, "### NO GPS DATA AVAILABLE FOR
STATION %s\n",
                                SelectedStations[i]);
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }
                Status = chdir(SelectedStations[i]);
                if (Status != 0)
                {
                        sprintf(ch, "### NO GPS DATA AVAILABLE FOR
STATION %s\n",
                                SelectedStations[i]);
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }
                /*
                 * Fix root dir for that station
                 */
                getcwd(root, MAXLEN);

                for (year = year_beg; year <= year_end; year++)
                {
                        sprintf(ch, "%04d", year);
                        repertoire = opendir(ch);
                        if (repertoire == NULL)
                        {
                                sprintf(ch, "### THERE IS NO GPS DATA
AVAILABLE FOR STATION %s FOR YEAR %04d\n",
                                        SelectedStations[i], year);
                                PERROR(ch);
                                continue;
                        }
                        else
                        {
                                sprintf(ch, "### THERE IS GPS DATA
AVAILABLE FOR STATION %s FOR YEAR %04d\n",
                                        SelectedStations[i], year);
                                PSTANDARD(ch);
                        }
                        for (;;)
                        {
                                ptfic = readdir(repertoire);
                                if (ptfic == NULL)
                                {
                                        closedir(repertoire);
                                        break;
                                }
                                /*
                                 * Test only on Hatanaka coded and
gzipped files
                                 */
```

```
is, length = 15


(size_t) 15)

/* Dot */

"D.gz"))

requested interval

1900;

actual aa, mm, dd

'\0';

&jj);
```

```
                pointe = ptfic->d_name;
                /*
                 * e.g: KOUC1230.02D.gz, that
                 *      |    |    |    |
                 *      012345678901234
                 */
                if (strlen(pointe) !=

                        continue;
                if (*(pointe+8) != '.')

                        continue;
                if (strcmp(pointe+11,

                        continue;
                /*
                 * Test if this file fits the
                 */
                curr.tm_year   =       year-

                /*
                 * Convert julian day to
                 */
                strcpy(ch, pointe+4); ch[3] =

                qt      =       atoi(ch);
                i_qt(year%100, qt, &aa, &ms,

                /*
                 * Month
                 */
                curr.tm_mon    =       ms;
                /*
                 * Day
                 */
                curr.tm_mday   =       jj;
                /*
                 * Hour
                 */
                curr.tm_hour   =       0;
                /*
                 * Minute
                 */
                curr.tm_min    =       0;
                /*
                 * Seconde
                 */
                curr.tm_sec    =       0;
                current = mktime (&curr);
```

```
                                        /*
                                         * Check interval
                                         */
                                        if (current < begin)
                                                continue;
                                        if (current > end)
                                                continue;

                                        /*
                                         * Current file fits the interval
                                         */
                                        strcpy(event[NbMails].Station,
SelectedStations[i]);
                                        event[NbMails].year = year;
                                        strcpy(event[NbMails].NFile, pointe);
                                        if (NbMails > MAXEVENTS)
                                                goto hopla;
                                        else ++NbMails;
                                }
                        }
                }
hopla:;         if (NbMails == 0)
                {
                        sprintf(ch, "### NO GEODETIC DATA FOUND.\n");
                        PSTANDARD(ch);
                }
                else    if (NbMails >= MAXEVENTS)
                {
                        sprintf(ch, "###\n");
                        PSTANDARD(ch);
                        sprintf(ch, "###\t***** ERROR *****\n");
                        PSTANDARD(ch);
                        sprintf(ch, "###\n");
                        PSTANDARD(ch);
                        sprintf(ch, "### THE RESULT OF YOUR REQUEST EXCEEDS
MAXIMUM\n");
                        PSTANDARD(ch);
                        sprintf(ch, "### NUMBER (%d) OF FILES ALLOWED.\n",
MAXEVENTS);
                        PSTANDARD(ch);
                        sprintf(ch, "### WE SUGGEST YOU SELECT A SMALLER TIME
INTERVAL, OR\n");
                        PSTANDARD(ch);
                        sprintf(ch, "### A SMALLER NUMBER OF STATIONS.\n");
                        PSTANDARD(ch);
                }
                else
                {
                        sprintf(ch, "### %d GPS FILE(S) FOUND.\n", NbMails);
                        PSTANDARD(ch);
                        sprintf(ch, "### THEY WILL BE SENT TO YOU BY
SUBSEQUENT E-MAILS.\n");
                        PSTANDARD(ch);
                }

                        break;
                case MAG:
                        sprintf(ch, "### DATA DISTRIBUTION NOT YET
AVAILABLE FOR MAGNETIC DATA\n");
                        PERROR(ch);
                        break;
        }

        Status = chdir(orig);
        return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "time.h"
#include "Constantes.h"
#include "Globaux.h"

/*
 * Send mails with all requested data
 */
void exped_(recipient)
char *recipient;
{
        char    orig[MAXLEN];
        char    ch[MAXLEN];
        char    ch2[MAXLEN];
        char    dest[MAXLEN];
        char    tarfilename[MAXLEN];
        char    *pointe;
        FILE    *fich;
        int     i, Status;
        char    *pti, *pto;
        struct  stat    buf;
        DIR     *repertoire;
        struct  dirent  *ptfic;
        time_t  curr;
        struct  tm *p_curr;


        pti = recipient;
        pto = dest;

        for (;;)
        {
                if (isspace(*pti) !=0)
                {
                        *pto = '\0';
                        break;
                }
                else
                {
                        *pto = *pti;
                        ++pti;
                        ++pto;
                }
        }

        /*
         * Addresses not to reply to

        if (strstr(dest, "lebelleg"))
                return;
        if (strstr(dest, "LEBELLEG"))
                return;
        if (strstr(dest, "autodrm"))
                return;
        if (strstr(dest, "AUTODRM"))
                return;
         */

        if (NbMails == 0)
                return;

        if (NbMails >= MAXEVENTS)
                return;

        getcwd(orig, MAXLEN);

        /*
         * Clean up temporary directory
         */
        Status = chdir(TEMP_DIR);
        if (Status != 0)
                return;

        system("rm -rf *");

        switch (data_type)
        {
                case SIS:
                        for (i=0; i<NbMails; i++)
                        {
                                Status = chdir(DETECTIONS);
                                if (Status != 0)
                                        return;

                                Status = chdir("SIS");
                                if (Status != 0)
                                        return;

                                Status = chdir(event[i].Station);
                                if (Status != 0)
                                        return;

                                sprintf(ch, "%04d", event[i].year);
                                Status = chdir(ch);
                                if (Status != 0)
                                        return;

                                strncpy(ch2, event[i].NFile, 8);
                                ch2[8] = '\0';

                                sprintf(ch, "cp %s.ndx %s/%s%s.ndx", ch2,
TEMP_DIR, event[i].Station, ch2); system(ch);
```

```
                        sprintf(ch, "cp %s.sis %s/%s%s.sis", ch2, TEMP_DIR,
event[i].Station, ch2); system(ch);
                }
                break;
        case GPS:
                for (i=0; i<NbMails; i++)
                {
                        Status = chdir(DETECTIONS);
                        if (Status != 0)
                                return;

                        Status = chdir("GPS");
                        if (Status != 0)
                                return;

                        Status = chdir(event[i].Station);
                        if (Status != 0)
                                return;

                        sprintf(ch, "%04d", event[i].year);
                        Status = chdir(ch);
                        if (Status != 0)
                                return;

                        strncpy(ch2, event[i].NFile, 8);
                        ch2[8] = '\0';

                        sprintf(ch, "cp %s.* %s", ch2, TEMP_DIR); system(ch);
                }
                break;
        case MAG:
                return;
                break;
    }

    /*
     * All files to send are now in TEMP_DIR
     */
    Status = chdir(TEMP_DIR);
    if (Status != 0)
            return;

    time (&curr);
    p_curr = gmtime(&curr);
    /*
     * Generate tarfile name: yymmddhhmmss
     */
     sprintf(tarfilename, "%02d%02d%02d%02d%02d%02d.tar",
            p_curr->tm_year%100,
            p_curr->tm_mon+1,
            p_curr->tm_mday,
            p_curr->tm_hour,
            p_curr->tm_min,
```



```
            p_curr->tm_sec);

    switch (data_type)
    {
            case SIS:
                    /*
                     * If SAC format is requested, convert SISMALP
files
                     * to SAC format before tar. A temporary directory
is used
                     */
                    if (data_format == SAC)
                    {
                            sprintf(ch, "/home/geosci/bin/sis2sac . -
bs"); system(ch);
                            sprintf(ch, "tar cf %s *.SAC", tarfilename);
system(ch);
                    }
                    else
                    {
                            sprintf(ch, "tar cf %s *.sis *.ndx",
tarfilename); system(ch);
                            system(ch);
                    }
                    break;
            case GPS:
                    /*
                     * If RINEX format is requested, convert HATANAKA
files
                     * to RINEX format before tar. A temporary
directory is used
                     */
                    sprintf(ch, "gunzip *"); system(ch);
                    if (data_format == RINEX)
                    {
                            repertoire = opendir(".");
                            if (repertoire != NULL)
                            {
                                    for (;;)
                                    {
                                            ptfic = readdir(repertoire);
                                            if (ptfic == NULL)
                                            {
                                                    closedir(repertoire);
                                                    break;
                                            }
                                            /*
                                             * Test only on .yyD files
                                             */
                                            pointe = ptfic->d_name;
                                            if (strlen(pointe) !=
(size_t) 12)
                                                    continue;
```

```
                                        if (*(pointe+8) != '.') /* Dot */
                                                continue;
                                        if (*(pointe+11) != 'D') /* This is
Hatanaka-coded RINEX file */
                                                continue;
                                        sprintf(ch,
"/home/geosci/bin/crx2rnx %s", pointe); system(ch);
                                        unlink(pointe);
                                }
                        }
                        sprintf(ch, "tar cf %s *o", tarfilename); system(ch);
                }
                else
                {
                        sprintf(ch, "tar cf %s *D", tarfilename); system(ch);
                        system(ch);
                }
                break;
        case MAG:
                return;
                break;
}

/*
 * Compress tarfile
 */
sprintf(ch, "gzip %s", tarfilename); system(ch);

/*
 * Uuencode zipped tarfile
 */
sprintf(ch, "uuencode %s.gz %s.gz > fich_encode", tarfilename,
tarfilename);
system(ch);

/*
 * Check size of encoded mail. It must be less than 4 megabytes.
 */
stat("fich_encode", &buf);

/* If less than 4 megabytes, send it by mail */
if (buf.st_size < (4*MEGA))
{
        /*
         * Send file by mail
         */
        sprintf(ch, "mailx -s \"Events detected by IRD Noumea AutoDRM\" %s
< fich_encode", dest); system(ch);
}
/*
 * If greater than 4 megabytes, place it in an outgoing FTP directory and
 * notify the user.
 */
```



```
else
{
        /*
         * Check size of gzipped tarfile. It must not exceed 10
megabytes.
         */
        sprintf(ch, "%s.gz", tarfilename);
        stat(ch, &buf);

        if (buf.st_size < (10*MEGA))
        {
                fich = fopen(TEMP_FILE, "w");
                fprintf(fich, "####\n");
                fprintf(fich, "#### THE MAIL FILE RESULTING FROM
YOUR REQUEST EXCEEDS\n");
                fprintf(fich, "#### MAXIMUM SIZE ALLOWED (4
MEGABYTES) TO BE SENT BY E-MAIL\n");
                fprintf(fich, "####\n");
                fprintf(fich, "#### THE CORRESPONDING TARFILE WILL
BE PLACED INSTEAD IN\n");
                fprintf(fich, "#### THE FOLLOWING DIRECTORY,
ACCESSIBLE VIA FTP:\n");
                fprintf(fich, "####\n");
                fprintf(fich, "#### SITE: FTP.IRD.NC,\n");
                fprintf(fich, "#### USER: anonymous,
PASSWORD: %s\n", dest);
                fprintf(fich, "#### DIRECTORY:
/PUB/OUTGOING/GEOPHY/AUTODRM/%s\n", dest);
                fprintf(fich, "####\n");
                fprintf(fich, "#### DATA WILL BE ACCESSIBLE IN A
MAXIMUM DELAY OF 30 MINUTES\n");
                fprintf(fich, "####\n");
                fprintf(fich, "#### AND WILL BE KEPT ONE DAY.\n");
                fprintf(fich, "####\n");
                fclose(fich);
                sprintf(ch, "mailx -s \"IRD Noumea AutoDRM
notification\" %s < %s", dest, TEMP_FILE); system(ch);
                sprintf(ch, "mkdir %s/%s", FTP_DIR, dest);
system(ch);
                sprintf(ch, "chmod 777 %s/%s", FTP_DIR, dest);
system(ch);
                sprintf(ch, "cp %s.gz %s/%s", tarfilename, FTP_DIR,
dest); system(ch);
        }
        else
        {
                fich = fopen(TEMP_FILE, "w");
                fprintf(fich, "####\n");
                fprintf(fich, "#### THE TARFILE RESULTING FROM YOUR
REQUEST EXCEEDS\n");
                fprintf(fich, "#### MAXIMUM SIZE ALLOWED (10
MEGABYTES) TO BE PLACED\n");
```

```
                fprintf(fich, "#### ON OUR FTP SERVER.\n");
                fprintf(fich, "####\n");
                fprintf(fich, "#### WE SUGGEST YOU SELECT A SMALLER TIME
INTERVAL, OR\n");
                fprintf(fich, "#### A SMALLER NUMBER OF STATIONS.\n");
                fprintf(fich, "####\n");
                fclose(fich);
                sprintf(ch, "mailx -s \"IRD Noumea AutoDRM error
notification\" %s < %s", dest, TEMP_FILE); system(ch);
              }
        }

        /*
         * Unlink files
         */
        sprintf(ch, "rm -f *");
        system(ch);

        Status = chdir(orig);
        return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "Constantes.h"

/*
 * Start of time interval
 */
time_t  begin;
struct  tm      *p_begin;

/*
 * End of time interval
 */
time_t  end;
struct  tm      *p_end;

/*
 * Current date
 */
time_t  now;
struct  tm      *p_now;

/*
 * Existing stations
 */
char    ExistingStations[MAXSTATIONS][MAXLEN];
int     NbExistingStations;

/*
 * Selected stations
 */
char    SelectedStations[MAXSTATIONS][MAXLEN];
int     NbSelectedStations;

/*
 * Format GSE2.0 or not
 */
int     gse2;

/*
 * Data selected (default: SIS)
 */
int     data_type = SIS;

/*
 * Data format selected (seismological data, default: SISMALP)
 */
int     sis_data_format = SISMALP;

/*
 * Data format selected (geodetic data, default: RINEX)
 */
int     gps_data_format = RINEX;

/*
 * Data format selected (magnetic data, default: BIN)
 */
int     mag_data_format = BIN;

/*
 * Data format selected
 */
int     data_format = SISMALP;

/*
 * Short name of months
 */
char    *abbrev_month[12] =
        {
                "jan",  "feb",  "mar",
                "apr",  "may",  "jun",
                "jul",  "aug",  "sep",
                "oct",  "nov",  "dec"
        };

/*
 * Name of months
 */
char    *ascii_month[12] =
        {
                "january",      "february",     "march",
                "april",        "may",          "june",
                "july",         "august",       "september",
                "october",      "november",     "december"
        };

/*
 * Name of days
 */
char    *ascii_day[7] =
        {
                "sunday",       "monday",       "tuesday",      "wednesday",
                "thursday",     "friday",       "saturday"
        };

/*
 * E-mail address to send responses to
 */
char    emailaddress[MAXLEN];

/*
 * Number of mails to send
 */
int     NbMails=0;
```

```
/*
 * Event detected
 */
struct
{
        char    Station[MAXLEN];
        int     year;
        char    NFile[MAXLEN];
} event [MAXEVENTS];

void init_globaux_(format)
char *format;
{
        int     i, Status;
        FILE    *fich;
        char    ch[MAXLEN];
        char    orig[MAXLEN];
        char    *homedir;

        getcwd(orig, MAXLEN);

        /*
         * Check format GSE2.0 or not
         */
        if (!strcmp(format, "GSE2.0"))
                gse2 = TRUE;
        else
                gse2 = FALSE;

        time (&begin);
        now = begin;
        end = begin-DAY;
        p_begin = gmtime(&begin);
        p_now = gmtime(&now);
        p_end = gmtime(&end);

        /*
         * Read existing station names, using environment variables EXIST_STA
         */
        NbExistingStations = 0;
        /*
         * Selected stations: no default
         */
        NbSelectedStations = 0;

        homedir = getenv("HOME");
        Status = chdir(homedir);

        Status = chdir(AUTODRMFILES);
        if (Status != 0)
                goto the_end;

        switch(data_type)
        {
                case SIS:
                        fich = fopen(SIS_EXIST_STA, "r");
                        break;
                case GPS:
                        fich = fopen(GPS_EXIST_STA, "r");
                        break;
                case MAG:
                        fich = fopen(MAG_EXIST_STA, "r");
                        break;
        }

        if (fich == NULL)
                goto the_end;

        for (;;)
        {
                if (fgets(ch, MAXLEN, fich) == NULL)
                {
                        fclose(fich);
                        break;
                }
                /*
                 * Don't include carriage return
                 */
                ch[strlen(ch)-1] = '\0';

                strcpy(ExistingStations[NbExistingStations++], ch);
        }
the_end:;
        Status = chdir(orig);
        return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "Constantes.h"
#include "Globaux.h"

void liste_stations_(is,il,ie)
int *is;
int *il;
int *ie;
{
	FILE	*fich;
	char	orig[MAXLEN];
	char	ch[MAXLEN];
	char	ch2[MAXLEN];
	char	liste[MAXLEN];
	char	*homedir;
	int	i, Status = 0;

	getcwd(orig, MAXLEN);

	switch(data_type)
	{
		case SIS:
			strcpy(liste, SIS_STA_LIST);
			break;
		case GPS:
			strcpy(liste, GPS_STA_LIST);
			break;
		case MAG:
			strcpy(liste, MAG_STA_LIST);
			break;
	}

	homedir = getenv("HOME");
	Status = chdir(homedir);

	Status = chdir(AUTODRMFILES);
	if (Status == 0)
	{
		fich = fopen(liste, "r");
		if (fich == NULL)
		{
			if (gse2 == TRUE)
			{
				sprintf(ch, "\n*** +----------------------------
+\n");
				PERROR(ch);
				sprintf(ch, "*** | No station list available. |\n");
				PERROR(ch);
				sprintf(ch, "*** +----------------------------
+\n\n");
				PERROR(ch);
			}
			else
			{
				sprintf(ch, "\n*** +------------------------
----+\n");
				PERROR(ch);
				sprintf(ch, "*** | No station list available.
|\n");
				PERROR(ch);
				sprintf(ch, "*** +------------------------
--+\n\n");
				PERROR(ch);
			}
		}
		else
		{
			for (;;)
			{
				if (fgets(ch, MAXLEN, fich) == NULL)
				{
					sprintf(ch, "###\n");
					PSTANDARD(ch);
					fclose(fich);
					break;
				}
				sprintf(ch2, "### %s", ch);
				PSTANDARD(ch2);
			}
		}
	}

	Status = chdir(orig);
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "Constantes.h"
#include "Globaux.h"

void select_begin_date_(format)
char *format;
{
	int	i, Status, InNumber;
	char	ch[MAXLEN];
	char	orig[MAXLEN];
	char	date[MAXLEN];
	char	*pto = format;
	char	*ptd = date;
	struct tm	struct_date, *pt;

	getcwd(orig, MAXLEN);

	InNumber = FALSE;
	for (;;)
	{
		if (isdigit(*pto) != 0)
		{
			InNumber = TRUE;
			*ptd++ = *pto++;
		}
		else
		{
			if (InNumber == TRUE)
			{
				*ptd = '\0';
				break;
			}
			++pto;
		}
	}

	strncpy(ch, date, 4); ch[4] = '\0';
	struct_date.tm_year = atoi(ch) - 1900;
	strncpy(ch, date+4, 2); ch[2] = '\0';
	struct_date.tm_mon = atoi(ch) - 1;
	strncpy(ch, date+6, 2); ch[2] = '\0';
	struct_date.tm_mday = atoi(ch);
	strncpy(ch, date+8, 2); ch[2] = '\0';
	struct_date.tm_hour = atoi(ch);
	strncpy(ch, date+10, 2); ch[2] = '\0';
	struct_date.tm_min = atoi(ch);
	struct_date.tm_sec = 0;

	begin = mktime(&struct_date);
	pt = localtime(&begin);

	Status = chdir(orig);
	return;
}

void select_end_date_(format)
char *format;
{
	int	i, Status, InNumber;
	char	ch[MAXLEN];
	char	orig[MAXLEN];
	char	date[MAXLEN];
	char	*pto = format;
	char	*ptd = date;
	struct tm	struct_date, *pt;

	getcwd(orig, MAXLEN);

	InNumber = FALSE;
	for (;;)
	{
		if (isdigit(*pto) != 0)
		{
			InNumber = TRUE;
			*ptd++ = *pto++;
		}
		else
		{
			if (InNumber == TRUE)
			{
				*ptd = '\0';
				break;
			}
			++pto;
		}
	}

	strncpy(ch, date, 4); ch[4] = '\0';
	struct_date.tm_year = atoi(ch) - 1900;
	strncpy(ch, date+4, 2); ch[2] = '\0';
	struct_date.tm_mon = atoi(ch) - 1;
	strncpy(ch, date+6, 2); ch[2] = '\0';
	struct_date.tm_mday = atoi(ch);
	strncpy(ch, date+8, 2); ch[2] = '\0';
	struct_date.tm_hour = atoi(ch);
	strncpy(ch, date+10, 2); ch[2] = '\0';
	struct_date.tm_min = atoi(ch);
	struct_date.tm_sec = 0;

	end = mktime(&struct_date);
	pt = localtime(&end);

	Status = chdir(orig);
	return;
```

```
}

void select_interval_(dated, seconded, datef, secondef)
char *dated;
char *seconded;
char *datef;
char *secondef;
{
        int     i, Status, InNumber;
        char    ch[MAXLEN];
        char    orig[MAXLEN];
        char    begs[MAXLEN];
        char    ends[MAXLEN];
        char    *pto;
        char    *ptd;

        getcwd(orig, MAXLEN);

        select_begin_date_(dated);
        select_end_date_(datef);

        pto = seconded; ptd = begs;
        InNumber = FALSE;
        for (;;)
        {
                if (isdigit(*pto) != 0)
                {
                        InNumber = TRUE;
                        *ptd++ = *pto++;
                }
                else
                {
                        if (InNumber == TRUE)
                        {
                                *ptd = '\0';
                                break;
                        }
                        ++pto;
                }
        }

        pto = secondef; ptd = ends;
        InNumber = FALSE;
        for (;;)
        {
                if (isdigit(*pto) != 0)
                {
                        InNumber = TRUE;
                        *ptd++ = *pto++;
                }
                else
                {
                        if (InNumber == TRUE)
                        {
                                *ptd = '\0';
                                break;
                        }
                        ++pto;
                }
        }

        begin += atoi(begs);
        end += atoi(ends);

        Status = chdir(orig);
        return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "Constantes.h"
#include "Globaux.h"

/*
 * Parameter to procedure is a string containing desired
 * stations, separated by commas
 */
void select_data_format_(string)
char *string;
{
        char    orig[MAXLEN];
        char    ch[MAXLEN];
        char    format[MAXLEN];
        char    *homedir;
        char    *pt, *pt2;
        FILE    *fich;
        int     Status, i;

        getcwd(orig, MAXLEN);

        /*
         * Only non blank characters
         */
        pt = string;

        for (;;)
        {
                if (!isalpha(*pt))
                        ++pt;
                else
                        break;
        }

        pt2 = pt;

        for (;;)
        {
                if (isalpha(*pt2))
                        ++pt2;
                else
                        break;
        }

        *pt2 = '\0';
        strcpy(format, pt);

        /*
          * Test if selected data format is compatible
          * with current data type
          */
        if (!strcmp(format, "SISMALP"))
                if (data_type == SIS)
                {
                        sis_data_format = SISMALP;
                        data_format = SISMALP;
                        sprintf(ch, "### DATA FORMAT SELECTED FOR
SEISMOLOGICAL DATA: SISMALP.\n");
                        PSTANDARD(ch);
                        Status = chdir(orig);
                        return;
                }
                else
                {
                        sprintf(ch, "### SISMALP DATA FORMAT IS FOR
SEISMOLOGICAL DATA ONLY.\n");
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }

        if (!strcmp(format, "SAC"))
                if (data_type == SIS)
                {
                        sis_data_format = SAC;
                        data_format = SAC;
                        sprintf(ch, "### DATA FORMAT SELECTED FOR
SEISMOLOGICAL DATA: SAC.\n");
                        PSTANDARD(ch);
                        Status = chdir(orig);
                        return;
                }
                else
                {
                        sprintf(ch, "### SAC DATA FORMAT IS FOR
SEISMOLOGICAL DATA ONLY.\n");
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }

        if (!strcmp(format, "HATANAKA"))
                if (data_type == GPS)
                {
                        gps_data_format = HATANAKA;
                        data_format = HATANAKA;
                        sprintf(ch, "### DATA FORMAT SELECTED FOR GEODETIC
DATA: HATANAKA.\n");
                        PSTANDARD(ch);
                        Status = chdir(orig);
                        return;
                }
                else
                {
```

```
                        sprintf(ch, "### HATANAKA DATA FORMAT IS FOR GEODETIC DATA
ONLY.\n");
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }

        if (!strcmp(format, "RINEX"))
                if (data_type == GPS)
                {
                        gps_data_format = RINEX;
                        data_format = RINEX;
                        sprintf(ch, "### DATA FORMAT SELECTED FOR GEODETIC DATA:
RINEX.\n");
                        PSTANDARD(ch);
                        Status = chdir(orig);
                        return;
                }
                else
                {
                        sprintf(ch, "### RINEX DATA FORMAT IS FOR GEODETIC DATA
ONLY.\n");
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }

        if (!strcmp(format, "BIN"))
                if (data_type == MAG)
                {
                        mag_data_format = BIN;
                        data_format = BIN;
                        sprintf(ch, "### DATA FORMAT SELECTED FOR MAGNETIC DATA:
BIN.\n");
                        PSTANDARD(ch);
                        Status = chdir(orig);
                        return;
                }
                else
                {
                        sprintf(ch, "### RINEX DATA FORMAT IS FOR MAGNETIC DATA
ONLY.\n");
                        PERROR(ch);
                        Status = chdir(orig);
                        return;
                }

        sprintf(ch, "### DATA FORMAT UNKNOWN: %s.\n", format);
        PERROR(ch);
        sprintf(ch, "###\n");
        PSTANDARD(ch);

        Status = chdir(orig);
                return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "Constantes.h"
#include "Globaux.h"

/*
 * Parameter to procedure is a string containing desired
 * stations, separated by commas
 */
void select_data_type_(string)
char *string;
{
        char    orig[MAXLEN];
        char    ch[MAXLEN];
        char    type[MAXLEN];
        char    *homedir;
        char    *pt;
        FILE    *fich;
        int     Status;

        getcwd(orig, MAXLEN);

        /*
         * Only first 3 non blank characters
         */
        pt = string;
        while (isspace(*pt))
                if (*pt++ == '\0')
                        continue;

        strncpy(type, pt, 3);
        type[4] = '\0';

        if (!strcmp(type, "SIS"))
        {
                data_type = SIS;
                sprintf(ch, "### Seismological data (SIS) selected.\n");
                PSTANDARD(ch);

        }
        else if (!strcmp(type, "GPS"))
        {
                data_type = GPS;
                sprintf(ch, "### Geodetic data (GPS) selected.\n");
                PSTANDARD(ch);
        }
        else if (!strcmp(type, "MAG"))
        {
                data_type = MAG;
                sprintf(ch, "### Magnetic data (MAG) selected.\n");
                PSTANDARD(ch);
        }
        else
```



```
        {
                data_type = SIS;
                sprintf(ch, "### Unknown type: %s.\n", type);
                PERROR(ch);
                sprintf(ch, "### Default type (SIS) will be selected.\n");
                PERROR(ch);
        }

        /*
         * Read existing station names, using environment variables
EXIST_STA
         */
        NbExistingStations = 0;
        /*
         * Selected stations: no default
         */
        NbSelectedStations = 0;

        switch(data_type)
        {
                case SIS:
                        sprintf(ch, "%s/%s",
                                AUTODRMFILES, SIS_EXIST_STA);
                        fich = fopen(ch, "r");
                        break;
                case GPS:
                        sprintf(ch, "%s/%s",
                                AUTODRMFILES, GPS_EXIST_STA);
                        fich = fopen(ch, "r");
                        break;
                case MAG:
                        sprintf(ch, "%s/%s",
                                AUTODRMFILES, MAG_EXIST_STA);
                        fich = fopen(ch, "r");
                        break;
        }

        if (fich == NULL)
        {
                Status = chdir(orig);
                return;
        }

        for (;;)
        {
                if (fgets(ch, MAXLEN, fich) == NULL)
                {
                        fclose(fich);
                        break;
                }
                /*
                 * Don't include carriage return
                 */
```

```
            ch[strlen(ch)-1] = '\0';

            strcpy(ExistingStations[NbExistingStations++], ch);
        }

        sprintf(ch, "### %d station(s) available.\n", NbExistingStations);
        PSTANDARD(ch);
        sprintf(ch, "###\n");
        PSTANDARD(ch);

        Status = chdir(orig);
        return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "Constantes.h"
#include "Globaux.h"

/*
 * Parameter to procedure is a string containing desired
 * stations, separated by commas
 */
void select_stations_(string)
char *string;
{
        FILE    *sortie;
        char    orig[MAXLEN];
        char    ch[MAXLEN];
        char    station[MAXLEN];
        char    *pt, *pts;
        int     i, j, Status, Present = 0;

        getcwd(orig, MAXLEN);

        pt = string;
        pts = station;
        for (;;)
        {
                if (isalnum(*pt)!=0)
                        *pts++ = *pt;
                else
                {
                        *pts = '\0';
                        if (strlen(station)>0)
                        {
                                if (NbSelectedStations == 0)
                                {
                                        strcpy(SelectedStations[0], station);
                                        NbSelectedStations = 1;
                                }
                                else
                                {
                                        Present = FALSE;
                                        for (i=0; i<NbSelectedStations; i++)
                                                if (!strcmp(SelectedStations[i],
station))

                                                {
                                                        Present = TRUE;
                                                        break;
                                                }
                                        if (Present == FALSE)

        strcpy(SelectedStations[NbSelectedStations++], station);
                                }
                        }
                        if (*pt == '\0')
                                break;
                        pts = station;
                }
                ++pt;
        }

        for (i=0; i<NbSelectedStations; i++)
        {
                Present = FALSE;
                for (j=0; j<NbExistingStations; j++)
                        if (!strcmp(SelectedStations[i],
ExistingStations[j]))
                        {
                                Present = TRUE;
                                break;
                        }

                if (Present == TRUE)
                {
                        sprintf(ch, "### STA_LIST: %s: OK\n",
SelectedStations[i]);
                        PSTANDARD(ch);
                }
                else
                {
                        sprintf(ch, "### STA_LIST: %s: not available (see
SLIST)\n", SelectedStations[i]);
                        PERROR(ch);
                }
        }

        sprintf(ch, "###\n");
        PSTANDARD(ch);

        Status = chdir(orig);
}
```

```
/*
 * From 0 to PIVOT-1: 20xx; from PIVOT to 99: 19xx; e.g. 05 = 2005,
 * and 85 = 1985
 */
#define PIVOT       70

int     nbj[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

/*
 * Number of days in the year
 */
int dysize(annee)
int annee;
{
        if (annee % 4 != 0)
                return (365);
        if (annee % 400 == 0)
                return (366);
        if (annee % 100 == 0)
                return (365);
        /*
         * Else leap year
         */
        return (366);
}

/*
 * Century on 2 digits; if year > 70, century is supposed to be 19 (20th...)
 */
int
siecle(annee)
        int             annee;
{
        if (annee >= 100)
                return (20);
        if (annee < PIVOT)
                return (20);
        return (19);
}


/*
 * Rank of the day from 1st of january
 */
int
qt(a, m, d)
int             a, m, d;
{
        int i;
        int quant = 0;

        if (dysize(siecle(a) * 100 + a) == 366)
                nbj[1] = 29;

        for (i = 0; i < m - 1; i++)
                quant += nbj[i];
        quant += d;

        return (quant);
}

/*
 * Actual day from the yday
 */
void
i_qt(a, qt, aa, ms, jj)
int a, qt;
int *aa, *ms, *jj;
{
        int i;

        if (dysize(siecle(a) * 100 + a) == 366)
                nbj[1] = 29;

        *aa = a;
        *ms = 0;
        *jj = 0;

        for (i = 0; i < 12; i++) {
                if (qt <= nbj[i]) {
                        *jj = qt;
                        return;
                }
                ++*ms;
                qt -= nbj[i];
        }

        return;
}
```

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

/*
 * Start of time interval
 */
extern time_t  begin;
extern struct tm      *p_begin;

/*
 * End of time interval
 */
extern time_t  end;
extern struct tm      *p_end;

/*
 * Current date
 */
extern time_t  now;
extern struct tm      *p_now;

/*
 * Existing stations
 */
extern char    ExistingStations[MAXSTATIONS][MAXLEN];
extern int     NbExistingStations;

/*
 * Selected stations
 */
extern char    SelectedStations[MAXSTATIONS][MAXLEN];
extern int     NbSelectedStations;

/*
 * Format GSE2.0 or not
 */
extern int     gse2;

/*
 * Data selected (default: SIS)
 */
extern int     data_type;

/*
 * Data format selected (seismological data, default: SISMALP)
 */
extern int     sis_data_format;

/*
 * Data format selected (geodetic data, default: RINEX)
 */
extern int     gps_data_format;

/*
 * Data format selected (magnetic data, default: BIN)
 */
extern int     mag_data_format;

/*
 * Data format selected
 */
extern int     data_format;

/*
 * Short name of months
 */
extern char    *abbrev_month[12];

/*
 * Name of months
 */
extern char    *ascii_month[12];

/*
 * Name of days
 */
extern char    *ascii_day[7];

/*
 * E-mail address to send responses to
 */
extern char    emailaddress[MAXLEN];

/*
 * Number of mails to send
 */
extern int     NbMails;

/*
 * Event detected
 */
extern struct
{
        char    Station[MAXLEN];
        int     year;
        char    NFile[MAXLEN];
} event[MAXEVENTS];
```

```
#define MAXLEN  1024

#define ERROR_OUTPUT   4
#define STANDARD_OUTPUT        6

#define MAXSTATIONS    10

#define MAXEVENTS      150

#define SECOND  1
#define MINUTE  (60*SECOND)
#define HOUR    (60*MINUTE)
#define DAY     (24*HOUR)
#define WEEK    (7*DAY)

#define KILO    (1024)
#define MEGA    (1024*KILO)
#define GIGA    (1024*MEGA)

/* Approximate values for one month and one year... */
#define MONTH   (30*DAY)
#define YEAR    (365*DAY)

/*
 * Environment files for external C AutoDRM procedures. These procedures are
 * called by fortran routines in autodrm.f
 */
#define AUTODRMFILES    "/export/home/local/autodrm/files"

#define SIS_STA_LIST    "SISstationinfo.txt"
#define SIS_EXIST_STA   "SISstationlist.txt"

#define GPS_STA_LIST    "GPSstationinfo.txt"
#define GPS_EXIST_STA   "GPSstationlist.txt"

#define MAG_STA_LIST    "MAGstationinfo.txt"
#define MAG_EXIST_STA   "MAGstationlist.txt"

/*
 * Print (unbuffered) on standard output
 */
#define PSTANDARD(ch)   write(STANDARD_OUTPUT, ch, strlen(ch))

/*
 * Print (unbuffered) on error output
 */
#define PERROR(ch)      write(ERROR_OUTPUT, ch, strlen(ch))

enum {
        TRUE, FALSE
};

enum {
/*
 * Data types available on autodrm
 */
        SIS, GPS, MAG
};

enum {
/*
 * Data formats available for seismic data
 */
        SISMALP, SAC,
/*
 * Data formats available for geodetic data
 */
       HATANAKA, RINEX,
/*
 * Data formats available for magnetic data
 */
        BIN
};

/*
 * Where to put files in order they are accessible by user via FTP
 */
#define FTP_DIR         "/home/geophy/outgoing/autodrm"

/*
 * Where detections are
 */
#define DETECTIONS      "/home/autodrm_data"

/*
 * Temporary directory
 */
#define TEMP_DIR        "/home/autodrm_data/tmp"

/*
 * Temporary file
 */
#define TEMP_FILE       "temp.txt"
```

ANNEXE 2

Exemples de réponses de l'autodrm

Réponse à une requête d'une journée d'observations GPS (notée en gras). La taille de la réponse est de l'ordre de 300 kilooctets (inférieure à 4 mégaoctets), celle-ci est immédiatement envoyée par messagerie:

```
From: autodrm@noumea.ird.nc
Date: Wed, 20 Nov 2002 10:00:01 +1100 (GMT)
To: lebelleg@noumea.ird.nc
Subject: IRD Noumea AutoDRM Response
X-Sanitizer: Anomy Sanitizer - McAfee Virusscan
X-Sanitizer-URL: http://mailtools.anomy.net/
X-Sanitizer-Rev: $Id: Sanitizer.pm,v 1.54 2002/02/15 16:59:07 bre Exp $

### Sortie generee automatiquement par:
### IRD Noumea AutoDRM: Automatic Data Request Manager (V2.99)  en service a:
### Laboratoire de Geosciences, IRD Centre de Noumea, New Caledonia
### Aide en francais: AIDE
### ------------------------------------------------------------------------
### This output was generated fully automatically by the
### IRD Noumea AutoDRM: Automatic Data Request Manager (V2.99)  running at the
### Laboratoire de Geosciences, IRD Centre de Noumea, New Caledonia
### Help: INFOR/HELP/GUIDE
### ------------------------------------------------------------------------
###
### Your request arrived at : Nov 20 09:59:34 2002 local time
### The  AutoDRM started at : Nov 20 10:00:01 2002 local time
###
### DATA_TYPE log:
### Got command: BEGIN
### Request comes from:  Pierre Lebellegard <lebelleg@noumea.ird.nc>
### Got command: DATA_TYPE GPS
### Got command: STA_LIST KOUC
### sta_list after check is:   KOUC
### Got command: DATE1 200210010000
### Got command: DATE2 200210012359
### Got command: DETEC
### Got command: STOP
###
### (A copy of your request mail is included at the end)
###

### Voici le resultat de votre commande
### (a moins d'une erreur serieuse!):

### Below follows what you requested
### (unless your request contained a severe error!):

### Geodetic data (GPS) selected.
### 4 station(s) available.
###
### STA_LIST: KOUC: OK
###
### DISTRIBUTION OF GPS DATA *****
### BEGIN: tuesday 1 october, 2002 00:00
### END: tuesday 1 october, 2002 23:59
### STATIONS SELECTED:
### KOUC
### THERE IS GPS DATA AVAILABLE FOR STATION KOUC FOR YEAR 2002
### 1 GPS FILE(S) FOUND.
### THEY WILL BE SENT TO YOU BY SUBSEQUENT E-MAILS.

### Below follows the request we received from you:
```

>From lebelleg@noumea.ird.nc Wed Nov 20 09:59:34 2002

<…>

Date: Wed, 20 Nov 2002 09:53:20 +1100 (NCT)
From: Pierre Lebellegard <lebelleg@noumea.ird.nc>
Message-Id: <200211192253.gAJMrKhE028651@artemis.ird.nc>
To: autodrm@noumea.ird.nc
Content-Length: 83

BEGIN
DATA_TYPE GPS
STA_LIST KOUC
DATE1 200210010000
DATE2 200210012359
DETEC
STOP

Réponse reçue :

From: autodrm@noumea.ird.nc
Date: Wed, 20 Nov 2002 10:00:16 +1100 (GMT)
To: lebelleg@noumea.ird.nc
Subject: Events detected by IRD Noumea AutoDRM
X-Sanitizer: Anomy Sanitizer - McAfee Virusscan
X-Sanitizer-URL: http://mailtools.anomy.net/
X-Sanitizer-Rev: $Id: Sanitizer.pm,v 1.54 2002/02/15 16:59:07 bre Exp $


021119230012.tar.gz

Réponse à une requête d'une semaine d'observations GPS pour 3 stations GPS (notée en gras). La taille de la réponse étant supérieure à 4 mégaoctets, celle-ci est mise à disposition sur le site ftp de l'IRD dans un délai maximum de une demi-heure:

```
From: autodrm@noumea.ird.nc
Date: Wed, 27 Nov 2002 15:19:01 +1100 (GMT)
To: lebelleg@noumea.ird.nc
Subject: IRD Noumea AutoDRM Response
X-Sanitizer: Anomy Sanitizer - McAfee Virusscan
X-Sanitizer-URL: http://mailtools.anomy.net/
X-Sanitizer-Rev: $Id: Sanitizer.pm,v 1.54 2002/02/15 16:59:07 bre Exp $

### Sortie generee automatiquement par:
### IRD Noumea AutoDRM: Automatic Data Request Manager (V2.99)  en service a:
### Laboratoire de Geosciences, IRD Centre de Noumea, New Caledonia
### Aide en francais: AIDE
### ------------------------------------------------------------------------
### This output was generated fully automatically by the
### IRD Noumea AutoDRM: Automatic Data Request Manager (V2.99)  running at the
### Laboratoire de Geosciences, IRD Centre de Noumea, New Caledonia
### Help: INFOR/HELP/GUIDE
### ------------------------------------------------------------------------
###
### Your request arrived at : Nov 27 15:18:59 2002 local time
### The  AutoDRM started at : Nov 27 15:19:00 2002 local time
###
### DATA_TYPE log:
### Got command: BEGIN
### Request comes from:  Pierre Lebellegard <lebelleg@noumea.ird.nc>
### Got command: DATA_TYPE GPS
### Got command: DATA_FORMAT HATANAKA
### Got command: SLIST
### Got command: STA_LIST FTNA, LPIL, KOUC
### sta_list after check is:   FTNA, LPIL, KOUC
### Got command: TIME 2002/11/01 00:00 TO 2002/11/10 23:59
### StartTime: 200211010000 0.00
### EndTime  : 200211102359 0.00
### Got command: DETEC
### Got command: STOP
###
### (A copy of your request mail is included at the end)
###

### Voici le resultat de votre commande
### (a moins d'une erreur serieuse!):

### Below follows what you requested
### (unless your request contained a severe error!):

### Geodetic data (GPS) selected.
### 4 station(s) available.
###
### DATA FORMAT SELECTED FOR GEODETIC DATA: HATANAKA.
### +=========================================================================================+
### |             AVAILABLE GEODETIC STATIONS AT IRD, CENTRE DE NOUMEA, NEW CALEDONIA         |
### +=========================================================================================+
### | Station name |              Location              |              Coordinates              |
### +--------------+------------------------------------+---------------------------------------+
### |     KOUC     | Meteo, Koumac, New Caledonia       | 20°33'31"S, 164°17'14"E, Height: 84.08m |
### +--------------+------------------------------------+---------------------------------------+
### |     LPIL     | LAS, Lifou island, New Caledonia   | 20°55'05"S, 167°15'50"E, Height: 73.68m |
### +--------------+------------------------------------+---------------------------------------+
### |     NOUM     | Nouville, Noumea, New Caledonia    | 22°16'11"S, 166°24'37"E, Height: 83.08m |
### +--------------+------------------------------------+---------------------------------------+
### |     FTNA     | Meteo, Futuna island, Wallis&Futuna | 14°18'28"S, 178°07'15"W, Height: 85.58m |
### +--------------+------------------------------------+---------------------------------------+
```

```
###
### STA_LIST: FTNA: OK
### STA_LIST: LPIL: OK
### STA_LIST: KOUC: OK
###
### DISTRIBUTION OF GPS DATA *****
### BEGIN: friday 1 november, 2002 00:00
### END: sunday 10 november, 2002 23:59
### STATIONS SELECTED:
### FTNA
### LPIL
### KOUC
### THERE IS GPS DATA AVAILABLE FOR STATION FTNA FOR YEAR 2002
### THERE IS GPS DATA AVAILABLE FOR STATION LPIL FOR YEAR 2002
### THERE IS GPS DATA AVAILABLE FOR STATION KOUC FOR YEAR 2002
### 22 GPS FILE(S) FOUND.
### THEY WILL BE SENT TO YOU BY SUBSEQUENT E-MAILS.

### Below follows the request we received from you:

>From lebelleg@noumea.ird.nc Wed Nov 27 15:18:59 2002

<...>

Date: Wed, 27 Nov 2002 15:12:40 +1100 (NCT)
From: Pierre Lebellegard <lebelleg@noumea.ird.nc>
Message-Id: <200211270412.gAR4CeJ9016401@artemis.ird.nc>
To: autodrm@noumea.ird.nc
Content-Length: 126

BEGIN
DATA_TYPE GPS
DATA_FORMAT HATANAKA
SLIST
STA_LIST FTNA, LPIL, KOUC
TIME 2002/11/01 00:00 TO 2002/11/10 23:59
DETEC
STOP

Réponse reçue :

From: autodrm@noumea.ird.nc
Date: Wed, 27 Nov 2002 15:20:26 +1100 (GMT)
To: lebelleg@noumea.ird.nc
Subject: IRD Noumea AutoDRM notification
X-Sanitizer: Anomy Sanitizer - McAfee Virusscan
X-Sanitizer-URL: http://mailtools.anomy.net/
X-Sanitizer-Rev: $Id: Sanitizer.pm,v 1.54 2002/02/15 16:59:07 bre Exp $

####
#### THE MAIL FILE RESULTING FROM YOUR REQUEST EXCEEDS
#### MAXIMUM SIZE ALLOWED (4 MEGABYTES) TO BE SENT BY E-MAIL
####
#### THE CORRESPONDING TARFILE WILL BE PLACED INSTEAD IN
#### THE FOLLOWING DIRECTORY, ACCESSIBLE VIA FTP:
####
#### SITE: FTP.IRD.NC,
#### USER: anonymous, PASSWORD: lebelleg@noumea.ird.nc
#### DIRECTORY: /PUB/OUTGOING/GEOPHY/AUTODRM/lebelleg@noumea.ird.nc
####
#### DATA WILL BE ACCESSIBLE IN A MAXIMUM DELAY OF 30 MINUTES
####
#### AND WILL BE KEPT ONE DAY.
```

####

En allant sur le site ftp de l'IRD :

```
ftp -v -n ftp.ird.nc
Connected to ftpub.ird.nc.
220 FTP Server (Public IRD FTP Server) [ftpub.ird.nc]
ftp> user anonymous lebelleg@ird.nc
331 Anonymous login ok, send your complete email address as your password.
230-_____
            Bienvenue sur le serveur du Centre IRD de Noumea

<...>

 Local time in New Caledonia (GMT+11) : Wed Nov 27 15:42:59 2002
230 Anonymous access granted, restrictions apply.
ftp> cd pub/outgoing/geophy/autodrm
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
drwxrwxrwx   2 root     ftp           512 Nov 27 04:20 lebelleg@noumea.ird.nc
drwxrwxrwx   2 root     ftp           512 Nov  7 00:34 lebelleg@santo.ird.nc
226 Transfer complete.
ftp> cd lebelleg@noumea.ird.nc
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
-rw-r--r--   1 root     other     5912667 Nov 27 04:20 021127041920.tar.gz
226 Transfer complete.
ftp> quit
221 Goodbye.
```

Réponse à une requête de données sismologiques de la station de Port-Laguerre sur une période de 12 heures (notée en gras). La station de Port-Laguerre étant interrogée par modem toutes les heures, et aussitôt mise à disposition sur l 'autodrm, le délai de mise à disposition des données est donc compris entre 0 et une heure.

```
To: lebelleg@noumea.ird.nc
Subject: IRD Noumea AutoDRM Response

### Sortie generee automatiquement par:
### IRD Noumea AutoDRM: Automatic Data Request Manager (V2.99)  en service a:
### Laboratoire de Geosciences, IRD Centre de Noumea, New Caledonia
### Aide en francais: AIDE
### ------------------------------------------------------------------------
### This output was generated fully automatically by the
### IRD Noumea AutoDRM: Automatic Data Request Manager (V2.99)  running at the
### Laboratoire de Geosciences, IRD Centre de Noumea, New Caledonia
### Help: INFOR/HELP/GUIDE
### ------------------------------------------------------------------------
###
### Your request arrived at : Nov 29 11:25:02 2002 local time
### The  AutoDRM started at : Nov 29 11:26:00 2002 local time
###
### DATA_TYPE log:
### Got command: BEGIN
### Request comes from:  lebelleg@noumea.ird.nc (Pierre Lebellegard)
### Got command: STA_LIST PLG
### sta_list after check is:   PLG
### Got command: TIME 2002/11/28 12:00 TO 2002/11/28 23:59
### StartTime: 200211281200 0.00
### EndTime  : 200211282359 0.00
### Got command: DETEC
### Got command: STOP
###
### (A copy of your request mail is included at the end)
###

### Voici le resultat de votre commande
### (a moins d'une erreur serieuse!):

### Below follows what you requested
### (unless your request contained a severe error!):

### STA_LIST: PLG: OK
###
### DETECTION ON SEISMOLOGICAL DATA *****
### BEGIN: thursday 28 november, 2002 12:00
### END: thursday 28 november, 2002 23:59
### STATIONS SELECTED:
### PLG
### DETECTIONS ARE AVAILABLE FOR STATION PLG FOR YEAR 2002
### 4 SEISMIC EVENT(S) DETECTED.
### THEY WILL BE SENT TO YOU BY SUBSEQUENT E-MAILS.

### Below follows the request we received from you:

>From lebelleg@noumea.ird.nc Fri Nov 29 11:25:02 2002

<...>
```

```
BEGIN
STA_LIST PLG
TIME 2002/11/28 12:00 TO 2002/11/28 23:59
DETEC
STOP

Réponse reçue :
```

To: lebelleg@noumea.ird.nc
Subject: Events detected by IRD Noumea AutoDRM

 021129002612.tar.gz