

© IRD, Nouméa, 2003

/Lebellegard, P.

Réseau SPRINGY :

Station de transmission de donnée sismologiques De Port-Villa (Vanuatu)

Nouméa : IRD. Décembre 2003. 40 p.

Notes Tech. : Sci. Terre ; Géol.-Géophys. ; 27

SPRINGY; TRANSMISSION DE DONNÉES; SISMOLOGIQUE; SÉISME; VANUATU; GÉOLOGIE;
GÉOPHISIQUE; SCIENCES DE LA TERRE; INFORMATIQUE; AUTOMATE DE TRANSMISSION DE
DONNÉES; PROGRAMME; ACQUISITION DE DONNÉES.

I - Introduction

Les 19 et 20 mars 2003, s'est tenue près de Wellington (NZ), dans les locaux de l'IGNS¹ une réunion du groupe de travail intitulé « **South West Pacific Geophysical Networks Working Group** ». Cette réunion a fait suite au symposium France - Nouvelle Zélande sur la coopération dans les sciences de la terre tenu en juillet 2002 à Wellington.

Initialement prévu comme un atelier bilatéral France/Nouvelle-Zélande pour définir des objectifs et une stratégie de réseaux en commun, cet atelier a évolué et grossi vers une réunion internationale où étaient présents des groupes de scientifiques de France et de Nouvelle-Calédonie (IRD), du Japon (NIED), d'Australie (AGSO), et de Nouvelle-Zélande (IGNS). La surveillance sismique était le thème principal de la réunion.

Le groupe de l'IRD représentait leurs partenaires du Service des Mines et de la Géologie de Vanuatu.

Les participants se sont rapidement accordés sur le principe d'une mise en œuvre, à court terme, d'échanges de données gérés par automates en cas d'événements régionaux majeurs. Un consensus s'est dégagé vers cette solution minimale qui nécessite l'installation de machines autodrm (et éventuellement des logiciels de traitement de données sismologiques) dans chaque sous réseau en fonctionnement. Cette première étape permettra une meilleure gestion des crises sismiques, par l'échange en temps quasi-réel des données nécessaires à la détermination des paramètres de l'événement principal et des répliques les plus importantes. Il a donc été proposé :

- ❖ de créer un groupe régional de coopération, regroupant les pays participants, notamment : Australie, Nouvelle-Zélande, Vanuatu, Fidji, Tonga, France (N.-C.). L'acronyme proposé pour cet organisme a été **SPRINGY**, pour **Southwest Pacific Regional Integrated Network for GeophYsics**.
- ❖ S'agissant d'un organisme multinational à but non lucratif, il a été décidé de réserver aussitôt le nom de domaine Internet le plus adapté, à savoir <http://www.springy.org>; ce type de nom de domaine offre l'avantage de ne pas faire apparaître explicitement la maîtrise d'œuvre comme étant le fait d'un pays donné.
- ❖ L'IRD a proposé de mettre en place à Vanuatu pour les stations sismologiques de Port-Vila (BKM, PVC), Devil Point (DVP), Tanna (TAN), Santo (SAN) le système d'autoDRM² développé par P. Lebellegard à Nouméa, et opérationnel depuis 2002 ; ultérieurement et lorsque les financements correspondants auront été dégagés, il est souhaité de mettre en place le même système à Fidji et Tonga ;

La présente note technique a pour but de détailler le système mis en place à Port-Vila (Vanuatu), pour automatiser le transfert des données sismologiques vers le centre IRD de Nouméa et leur mise à disposition immédiate par l'automate de distribution de données (autoDRM) en opération au centre IRD de Nouméa. **Il s'agit, de fait, de la première contribution au réseau SPRINGY hors de Nouvelle-Calédonie.** Les données des stations sismologiques de Dzumac (DZM), et de Port-Laguerre étant en effet déjà disponible au moyen de l'autoDRM de l'IRD Nouméa.

Outre les détail des différents logiciels mis en place, est aussi détaillée une procédure d'intervention à l'intention du personnel du service des Mines et de la Sismologie de Port-Vila, en cas d'incident sur le système.

¹ Institute for Geological and Nuclear Sciences (<http://www/gns.cri.nz>)

² AutoDRM : Automatic Data Request Manager, c.-à.-d. automate de distribution de données à la demande

- 1) Fenêtre de commande (shell). Elle permet de taper des commandes Linux standard. Ainsi pour arrêter le système, on tapera la commande sync (3 fois) pour forcer l'écriture sur disque des données encore en mémoire vive, puis la commande halt.
- 2) Explorateur de fichiers.
- 3) Navigateur internet ;
- 4) Kmail : il s'agit de l'utilitaire Linux permettant l'envoi et la réception de mails. L'adresse correspondant au PC Linux est : irdvila@vanuatu.com.vu.
- 5) Il s'agit de l'état de la file d'attente des fichiers en partance vers Nouméa. Tant qu'il y aura des fichiers présents dans ce dossier, le système tentera la connexion modem et le transfert des fichiers sur le site ftp de l'IRD Nouméa, à et supposer bien entendu que le transfert n'ait pas été invalidé (cf. 7 et 8).
- 6) Ce répertoire contient les données continues en provenance du PC d'acquisition, rangées par sous-répertoires quotidiens, par exemple 2003-329
- 7) Suspension du mécanisme de transfert automatique des fichiers vers Nouméa. Un seul clic suffit.
- 8) Rétablissement du mécanisme de transfert automatique des fichiers vers Nouméa. Un seul clic suffit.

IV – Conclusion

Après une période de test de deux semaines après installation, le système fonctionne parfaitement, malgré la fiabilité aléatoire de la liaison internet RTC de Port-Vila. Il reste désormais à installer sur ce PC de communication un programme qui fasse de la détection automatique d'événements sismiques : nous aurons alors une disponibilité temps réel des événements, alors qu'avec le système actuel, celle-ci est de l'ordre de 24 heures. Une étape ultime, mais liée aux possibilités financières, est de créer une machine autoDRM locale à Port-Vila, entièrement autonome, est accessible directement par une adresse mail indiciduelle. Mais cela suppose une liaison internet permanente, ce qui a un coût élevé.

ANNEXE 1
Schéma fonctionnel

ANNEXE 2
Transfert des données continues
depuis le PC d'acquisition vers le PC Linux

```

#define LOCAL_DATA "C:\\DATA\\"
#define DISTANT_DATA "/data"
#define DEPOSIT "incoming"
#define DISTANT_PC "192.168.0.43"
#define TEMP_DIR "C:\\IRD\\TMP"
#define TIME_STAMP "OPLCOMP.STP"
#define FLISTE "XFERES.TXT"
#define NEWLIST "XFERNEW.TXT"
#define COMMANDE "COMMANDE.BAT"
#define F_IN "IN.TXT"
#define F_OUT "OUT.TXT"

#define SUF "*.OUT"

#define MINUTE (60)
#define HEURE (60*MINUTE)
#define JOUR (24*HEURE)
#define SEMAINE (7*JOUR)

#define DELAI (12*HEURE)

#define MAXLEN 1024

```



```

Status = chdir(repertoire);
if (Status != 0)
{
    ch.Format("Can't cd to directory:\n%s", repertoire);
    AfxMessageBox(ch);
    exit(1);
}

ifstream Liste(fichier);
if (Liste.bad())
{
    ch.Format("Can't open:\n%s",
             fichier);
    AfxMessageBox(ch);
    exit(1);
}

/*
 * On decompose la chaine passée en paramètre en nom de fichier/longueur de
fichier.
 */
sscanf(chaine, "%s%d", &NfChaine, &LgChaine);

for (;;)
{
    Liste.getline(Tampon, MAXLEN, '\n');
    if (Liste.eof())
    {
        Liste.close();
        return(-1);
    }

    /*
     * On decompose la chaine lue en nom de fichier/longueur de fichier.
     */
    sscanf(Tampon, "%s%d", &NfTampon, &LgTampon);

    if (!strcmp(NfTampon, NfChaine) != NULL)
    {
        if (LgTampon == LgChaine)
        {
            Liste.close();
            return(0);
        }
        else
            continue;
    }
}

Liste.close();
return(-1);
}

/*
 * Recherche la chaîne dans le fichier fichier situé sous le répertoire
 * répertoire; renvoie 0 si présent, -1 sinon.
 */
int present(char *repertoire, char *fichier, char *chaine)
{
    int      Status;
    CString ch;
    char Tampon[MAXLEN];

```



```

fprintf(f_in, "PING %s > %s", DISTANT_PC, F_OUT);
fclose(f_in);
Status = system(COMMANDE);
if (Status == -1)
{
    ch.Format("Ping has failed.");
    AfxMessageBox(ch);
    return(-1);
}

Status = present(TEMP_DIR, F_OUT, "octets");
if (Status == 0)
{
    return(0);
}
else
{
    return(-1);
}
}

/*
 * Crée les sous-répertoires sur la machine distante par l'intermédiaire d'un script
 ftp.
 * Sur le PC distant, on est directement positionné sur le bon répertoire de base. On
 a
 * donc à créer deux niveaux de répertoire, par exemple 2002-001, et les sous-
 répertoires
 * horaires, par exemple 2002-001\00.
 */
int creerep(char *machine_distante, CString repertoire)
{
    int      Status;
    CString  ch;
    FILE    *f_in;
    char     Tampon[MAXLEN];

    getcwd(Tampon, MAXLEN);

    Status = chdir(TEMP_DIR);
    if (Status != 0)
    {
        ch.Format("Can't cd to temp dir:\n%s",
                TEMP_DIR);
        AfxMessageBox(ch);
        exit(1);
    }

    f_in = fopen(F_IN, "w");
    if (f_in == NULL)
    {
        ch.Format("Cree_rep: can't create file:\n%s", F_IN);
        AfxMessageBox(ch);
        fclose(f_in);
        return(-1);
    }
    fprintf(f_in, "user drmdata drmdata\n");
    fprintf(f_in, "mkdir %s\n", repertoire);
    fprintf(f_in, "quit\n");

    fclose(f_in);
    f_in = fopen(COMMANDE, "w");
    if (f_in == NULL)

```

```

    {
        ch.Format("Cree_rep: can't create file:\n%s", COMMANDE);
        AfxMessageBox(ch);
        fclose(f_in);
        return(-1);
    }

    fprintf(f_in, "@ECHO OFF\n");
    fprintf(f_in, "FTP -v -n %s < %s > %s\n", machine_distante, F_IN, F_OUT);
    fclose(f_in);

    Status = system(COMMANDE);
    if (Status == -1)
    {
        ch.Format("Ftp has failed.");
        AfxMessageBox(ch);
        return(-1);
    }

    Status = present(TEMP_DIR, F_OUT, "created");
    if (Status == 0)
    {
        Status = chdir(Tampon);
        if (Status != 0)
        {
            ch.Format("Can't get back to original directory:\n%s",
                Tampon);
            AfxMessageBox(ch);
        }
        return(0);
    }
    else
    {
        Status = chdir(Tampon);
        if (Status != 0)
        {
            ch.Format("Can't get back to original directory:\n%s",
                Tampon);
            AfxMessageBox(ch);
        }
        return(-1);
    }
}

/*
 * The parameter "fich" contains the file name and
 * its length actually transfered in the second field.
 */
void ajouter_liste(char *rep, char *liste, char *fich)
{
    int      Status;
    CString  ch;
    FILE     *Liste;

    Status = chdir(rep);
    if (Status != 0)
    {
        ch.Format("Can't cd to directory:\n%s",
            rep);
        AfxMessageBox(ch);
        exit(1);
    }
}

```

```

Liste = fopen(liste, "a");
if (Liste == NULL)
{
    ch.Format("Can't open: %s (append mode).\n",
             liste);
    AfxMessageBox(ch);
    exit(1);
}
fputs(fich, Liste);
fclose(Liste);
}

long depouille_ftp(char *rep, char *nom)
{
    int          Status;
    CString      ch;
    char Tampon[MAXLEN];
    char *pt;
    long longueur_transferee;

    Status = chdir(rep);
    if (Status != 0)
    {
        ch.Format("Can't cd to directory:\n%s",
                 rep);
        AfxMessageBox(ch);
        exit(1);
    }

    ifstream Listing(nom);
    if (Listing.bad())
    {
        ch.Format("Can't open:\n%s",
                 nom);
        AfxMessageBox(ch);
        exit(1);
    }

    for (;;)
    {
        Listing.getline(Tampon, MAXLEN, '\n');
        if (Listing.eof())
        {
            Listing.close();
            return(-1);
        }
        pt = strstr(Tampon, "octets re");
        if (pt == NULL)
            continue;
        /*
         * Pour pointer sur le nombre d'octets transférés
         */
        pt = Tampon;
        longueur_transferee = atol(pt+6);
        Listing.close();
        return(longueur_transferee);
    }
}

/*
 * Transfert par ftp d'un fichier vers le PC distant.
 *
 * Parametres en entrée:      - nom du fichier;

```



```

        return(-1);
    }

    fprintf(f_in, "@ECHO OFF\n");
    fprintf(f_in, "ftp -v -n %s < %s > %s\n", DISTANT_PC, F_IN, F_OUT);
    fclose(f_in);

    Status = system(COMMANDE);
    if (Status == -1)
    {
        ch.Format("Ftp has failed.");
        AfxMessageBox(ch);
        return(-1);
    }

    Etat = depouille_ftp(TEMP_DIR, F_OUT);

    /*
     * Si la longueur transférée est égale à la longueur à transférer,
     * alors le transfert s'est bien passé
     */
    if (Etat == longueur)
        return(0);
    else
        return(-1);
}

```

```

// Transfert.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "direct.h"
#include <io.h>
#include <ctype.h>
#include <string.h>
#include <time.h>
#include "Transfert.h"
#include "Constantes.h"
#include "Procedures.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// The one and only application object

CWinApp theApp;

using namespace std;

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int          nRetCode = 0;
    CString      ch;
    char  ch0 [MAXLEN];
    char  ch1 [MAXLEN];
    char  ch2 [MAXLEN];
    char  ch3 [MAXLEN];
    char  *pt;

    int          Status, quantieme, annee;
    FILE  *liste, *newliste;

    long        ecart, dcour, maintenant;
    struct tm    *ptemps;
    struct tm    scour;

    struct      _finddata_t outfiles;
    long  hOut;

    // initialize MFC and print and error on failure
    if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0))
    {
        // TODO: change error code to suit your needs
        cerr << _T("Fatal Error: MFC initialization failed") << endl;
        nRetCode = 1;
    }
    else
    {
        /*
        * Procédure automatique de transfert de données depuis le
        * PC d'acquisition (cave sismo à Port-Vila) vers Sopac (pc-linux).
        *
        * Date: 26 novembre 2003 - Auteur: Pierre Lebellegard
        *
        * Ce programme gère une liste des fichiers qui ont déjà été transférés,
        * et transfère les fichiers pas encore transférés sur le PC linux
        (192.168.0.43)
        */
    }
}

```

donnees
pour

```
* en générant un script ftp vers ce PC. Un serveur ftp est supposé
* tourner sur le PC distant, avec le login/mot de passe: drmdata/drmdata.
*
* Structure de données sur le PC local:
*
* Répertoire principal des données: C:\DATA. Dans le PC distant, les
* seront organisées en sous-répertoires: aaaa-jjj, par exemple 2002-001
* le 1er janvier 2002.
* Chacun de ces sous-répertoires contient 24 sous-répertoires de 00 à 23.
*
* La liste des fichiers déjà transférés est XFERES.TXT, directement
* sous le répertoire de base.
*/
Status = chdir(LOCAL_DATA);
if (Status != 0)
{
    ch.Format("Can't cd to data directory:\n%s",
              LOCAL_DATA);
    AfxMessageBox(ch);
    exit(1);
}

Status = chdir(TEMP_DIR);
if (Status != 0)
{
    Status = mkdir(TEMP_DIR);
    if (Status != 0)
    {
        ch.Format("Can't create temp directory:\n%s",
                  TEMP_DIR);
        AfxMessageBox(ch);
        exit(1);
    }
}

/*
* La liste des fichiers déjà transférés
*/
liste = fopen(FLISTE, "r");
if (liste == NULL)
{
    liste = fopen(FLISTE, "w");
    if (liste == NULL)
    {
        ch.Format("Can't create listing file:\n%s",
                  FLISTE);
        AfxMessageBox(ch);
        exit(1);
    }
    else
    {
        fclose(liste);
    }
}
else
{
    fclose(liste);
}

/*
* La liste des fichiers qui vont être transférés
*/
```

```

newliste = fopen(NEWLIST, "w");
if (newliste == NULL)
{
    ch.Format("Can't create new listing file:\n%s",
             FLISTE);
    AfxMessageBox(ch);
    exit(1);
}
else
{
    fclose(newliste);
}

time(&maintenant);
ptemps = gmtime(&maintenant);

/*
 * On ne verifie le transfert que des fichiers plus recents
 * que (DELAI). Au besoin, on cree l'arborescence aaaa-jjj,
 * et le repertoire incoming pour y déposer les fichiers.
 */
Status = chdir(LOCAL_DATA);
if (Status != 0)
{
    ch.Format("Can't cd to data directory:\n%s",
             LOCAL_DATA);
    AfxMessageBox(ch);
    exit(1);
}

if ( (hOut = _findfirst(SUF, &outfiles)) == -1L)
{
    fprintf(stderr, "Pas de fichiers %s\n.", SUF);
    exit(1);
}
else
{
    do
    {
        /*
         * Nom de fichier de la forme aammjjxx
         */
        if ( strlen(outfiles.name) != 12)
            continue;

        /*
         * Maintenant on teste si la date indiquee par le
         * nom du fichier est plus recente que DELAI. On
         * considere uniquement aammjj, on fait le calcul
         * avec 0 heure 0 minute 0 seconde.
         */
        strncpy(ch0, &(outfiles.name[0]), 2); ch0[2] = '\0';
        scour.tm_year = atoi(ch0);
        scour.tm_year += 100;
        strncpy(ch0, &(outfiles.name[2]), 2); ch0[2] = '\0';
        scour.tm_mon = atoi(ch0);
        scour.tm_mon -= 1;
        strncpy(ch0, &(outfiles.name[4]), 2); ch0[2] = '\0';
        scour.tm_mday = atoi(ch0);
        scour.tm_hour = 0;
        scour.tm_min = 0;
        scour.tm_sec = 0;
        dcour = mktime(&scour);
        ecart = maintenant-dcour;
    }
}

```

```

if (ecart<DELAI)
{
    annee = 1900+scour.tm_year;
    quantieme = qt(annee,1+scour.tm_mon,scour.tm_mday);
    sprintf(ch0, "%04d-%03d/%s", annee,quantieme, DEPOSIT);

    /*
    * Il faut regarder si ce fichier a déjà été transféré,
    * s'il est présent dans la liste, sous la forme <nom
    * AVEC LA BONNE LONGUEUR (second paramètre de la
    */
    sprintf(ch3, "%s", LOCAL_DATA);
    sprintf(ch2, "%s\t%d", outfiles.name, outfiles.size);

    if (dejatransfere(ch3, FLISTE, ch2) != 0)
    {
        ch.Format("%04d-%03d", annee,quantieme);
        creerep(DISTANT_PC, ch);
        ch.Format("%04d-%03d/%s", annee,quantieme,
DEPOSIT);

        creerep(DISTANT_PC, ch);
        strcpy(ch1, outfiles.name);
        /*
        * Tres important de rajouter le blanc: le
        * le dernier caractere est bouffe...
        */
        strcat(ch1, " ");
        if (ftp(ch1, outfiles.size, ch0) != 0)
        {
            printf("not done...\n");
            continue;
        }
        else
        {
            printf("OK.\n");
        }
        /*
        * Le transfert est OK, on ajoute repertoire+nom
        * Le second champ contient la longueur du
        */
        sprintf(ch2, "%s %d\n", outfiles.name,
outfiles.size);
        ajouter_liste(ch3, FLISTE, ch2);
    }

    continue;
}
else
{
    continue;
}
} while (_findnext(hOut, &outfiles) == 0);
_findclose(hOut);
return nRetCode;
}
}

```

c'est-à-dire
du fichier>,
chaîne ch3).

à la liste,
fichier
outfiles.size);

```
return nRetCode;  
}
```

ANNEXE 3
Script de déclenchement
du transfert ftp vers Nouméa


```

#!/bin/csh
#
#   Send to Noumea all the files that haven't been sent yet.
#   Things are made more easily sending files one by one:
#   one ftp call for each file; tests are made by grep's on
#   the ftp log file.
#
#   To send a file to Noumea, one just have to place a copy of
#   it in the to_send directory. The content of this directory
#   is regularly checked. Files are deleted after a successful
#   ftp.
#
unset noclobber
#
#   The estimated time necessary to establish modem connection
#
setenv      DELAY 40
#
setenv      HOME /home/geosci
setenv      OUTGO $HOME/outgoing
setenv      EXE  $HOME/bin
#
#   *****
#   ***** THIS SCRIPT'S NAME *****
#   *****
#
#   Reverse, get first field before '/' then reverse again:
#   this way, we get script name (name following last '/').
#
setenv      NAME `echo $0 | rev | cut -d/ -f1 | rev`
#
#   Who is going to receive control e-mails
#
setenv      DEST Pierre.Lebellegard@ird.nc
#
#   To be sure that only one script is running.
#
#-----
#   If script has been crashed by a CTRL-C
#-----
#
onintr TAGADA
#
unalias cd
cd $OUTGO
#
#   Test if $NAME is already running
#
ps -Tu | grep $NAME | grep -v grep > /tmp/xxx
set flag = `cat /tmp/xxx | wc -l`; \rm -f /tmp/xxx
if ($flag > 1) then
#
#   At least one other script is running
#
    exit 0
else
#
#   No other $NAME script currently running
#
endif
#
#   Check if there is something to send (ie there are files
#   in the to_send directory).

```

```

#
ls -Cl to_send/* |& grep -v "Pas de correspondance" |& grep -v "No match" > /tmp/xxx
set flag = `ls -l /tmp/xxx | awk '{ print $5 }'; rm /tmp/xxx
#
if ($flag <= 1) then
    exit 0
endif
#
#    Now, we know that there is something to transfer. First, start the modem
connection
#    using wvdial (background). Memorize process number, and loop waiting for
"Connected"
#    in the wvdial log file.
#
(wvdial >& logdial&)
set NUMJOB = `ps aux | grep wvdial | head -1 | awk '{ print $2 }`
#
#    Wait a "reasonable" time for the modem connection to be established
#
sleep $DELAY
#
#    Wait for "Connected" message in the wvdial log file: if present, break the loop
and
#    transfer files. If connexion is not correctly made, kill the wvdial process,
and exit.
#
while (1)
#
    sleep 1
#
#    Test if wvdial is still running: if not, kill pppd and exit
#
ps -ef | grep wvdial | grep -v grep | grep -v pppd > /tmp/xxx
#
if ( -z /tmp/xxx ) then
#
    wvdial doesn't exist anymore... Retrieve, if exists, pppd PID (not smpppd PID)
#
    \rm -f /tmp/xxx
    ps -ef | grep ppp | grep -v smpppd | grep -v grep > /tmp/xxx
    if ( -z /tmp/xxx ) then
#
        Nothing to kill, just exit
#
        \rm -f /tmp/xxx
        exit 1
    else
        set NJ = `cat /tmp/xxx | head -1 | awk '{ print $2 }`
        kill -HUP $NJ
        exit 1
    endif
endif
#
#
#    Test if pppd is still running: if not, kill wvdial and exit
#
ps -ef | grep ppp | grep -v smpppd | grep -v grep > /tmp/xxx
#
if ( -z /tmp/xxx ) then
#
    pppd doesn't exist anymore... Retrieve, if exists, wvdial PID
#
    \rm -f /tmp/xxx

```

```

ps -ef | grep wvdial | grep -v grep | grep -v pppd > /tmp/xxx
if ( -z /tmp/xxx ) then
#
# Nothing to kill, just exit
#
    \rm -f /tmp/xxx
    exit 1
else
    set NJ = `cat /tmp/xxx | head -1 | awk '{ print $2 }'`
    kill -HUP $NJ
    exit 1
endif
endif
#
if ( -f logdial ) then
    grep "No dial tone" logdial >/tmp/xxx
    set flag = `ls -l /tmp/xxx | awk '{ print $5 }'`; rm /tmp/xxx
    if ($flag > 0) then
        kill -HUP $NUMJOB; \rm logdial
        exit 1
    endif
endif
#
if ( -f logdial ) then
    grep "NO DIAL TONE" logdial >/tmp/xxx
    set flag = `ls -l /tmp/xxx | awk '{ print $5 }'`; rm /tmp/xxx
    if ($flag > 0) then
        kill -HUP $NUMJOB; \rm logdial
        exit 1
    endif
endif
#
if ( -f logdial ) then
    grep "Modem not responding" logdial >/tmp/xxx
    set flag = `ls -l /tmp/xxx | awk '{ print $5 }'`; rm /tmp/xxx
    if ($flag > 0) then
        kill -HUP $NUMJOB; \rm logdial
        exit 1
    endif
endif
#
if ( -f logdial ) then
    grep "Device or resource busy" logdial >/tmp/xxx
    set flag = `ls -l /tmp/xxx | awk '{ print $5 }'`; rm /tmp/xxx
    if ($flag > 0) then
        kill -HUP $NUMJOB; \rm logdial
        exit 1
    endif
endif
#
if ( -f logdial ) then
    grep "Modem not responding." logdial >/tmp/xxx
    set flag = `ls -l /tmp/xxx | awk '{ print $5 }'`; rm /tmp/xxx
    if ($flag > 0) then
        kill -HUP $NUMJOB; \rm logdial
        exit 1
    endif
endif
#
if ( -f logdial ) then
    grep "Disconnecting" logdial >/tmp/xxx
    set flag = `ls -l /tmp/xxx | awk '{ print $5 }'`; rm /tmp/xxx
    if ($flag > 0) then

```



```
cat list >> journal
echo "" >> journal
echo -n "End: " >> journal; $EXE/die >> journal
cat journal | mail $DEST
\rm journal
\rm list
\rm begfil
\rm log
endif
exit 0
#
#   If script has been crashed (CTRL-C)
#
TAGADA:
#
#   Hang up modem connection
#
kill -HUP $NUMJOB; \rm logdial
cd $OUTGO
exit 1
```

ANNEXE 4
Démultiplexage des fichiers Sismalp
et transfert vers l'autoDRM

```

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define      BLKSIZE          1024
#define      MAXLEN          1024
#define      MAXSTATIONS 5

enum
{
    OK, BAD
};

struct
{
    char  name[MAXLEN];
    int   block_ind;
    int   block_cnt;
    int   year;
    int   month;
    int   day;
    int   hour;
    int   min;
    double      sec;
    double      freq;
    FILE  *ndx_out;
    FILE  *sis_out;
} station[MAXSTATIONS];

int  nbstations;
int  istation;
int  icode;

char *comp_out[] = {
    "DVPZ",
    "DVPN",
    "DVPE",
    "RPVZ",
    "RPVN",
    "RPVE",
    "APVZ",
    "APVN",
    "APVE",
    "PVCZ",
    "PVCN",
    "PVCE",
    "BKMZ",
    "BKMN",
    "BKME"
};

char *sta_out[] = {
    "DVP",
    "RPV",
    "APV",
    "PVC",
    "BKM"
};

void error(char *msg)
{
    fprintf(stderr, "%s\n", msg);
}

```

```

    exit(1);
}

int station_index(char *sta_name)
{
    int i, status;
    char ch[MAXLEN];

    status = BAD;
    for (i=0; i<nbstations; i++)
    {
        if (!strcmp(station[i].name, sta_name))
        {
            status = OK;
            break;
        }
    }

    if (status == OK)
    {
        return(i);
    }

    if (nbstations>=MAXSTATIONS)
    {
        sprintf(ch, "Maximum number of stations reached.\n");
        error(ch);
    }

    ++nbstations;
    strcpy(station[nbstations-1].name, sta_name);
    return(nbstations-1);
}

void decode_ndx(char *buf, char *channel, int *npoints, int *iblock, int *nblocks,
double *freq, int *day, int *month, int *year, int *hour, int *min, double *sec)
{
    char ch[MAXLEN];

    strncpy(ch, buf, 8);
    sscanf(ch, "%8s", channel);

    sscanf(buf+8, "%d%d%d%lf%d.%d.%d%d:%d:%lf",
npoints,
iblock,
nblocks,
freq,
day,
month,
year,
hour,
min,
sec);
}

main (int argc, char **argv)
{
    FILE *ndx_in;
    FILE *sis_in;

    char ch[MAXLEN];
    char ch2[MAXLEN];
    char line1[MAXLEN];
    char line2[MAXLEN];
}

```



```

char  ndx_name [MAXLEN];
char  sis_name [MAXLEN];

char  channel [MAXLEN];

int  npoints, iblock, nblocks, day, month, year, hour, min;
double  freq, sec;

int  i, status;

char  buf[1024*BLKSIZE];

if (argc != 2)
{
    sprintf(ch, "%s: Usage: %s <Sismalp file (without suffix)>.\n",
            argv[0], argv[0]);
    error(ch);
}

for (i=0; i<MAXSTATIONS; i++)
{
    station[i].ndx_out = NULL;
    station[i].sis_out = NULL;
}

strcpy(ch, argv[1]);
strcat(ch, ".ndx");
if (access(ch, F_OK) != 0)
{
    strcpy(ch, argv[1]);
    strcat(ch, ".NDX");
    if (access(ch, F_OK) != 0)
    {
        sprintf(ch, "%s: Can't find neither %s.ndx nor %s.NDX.\n",
                argv[0], argv[1], argv[1]);
        error(ch);
    }
}

strcpy(ndx_name, ch);
ndx_in = fopen(ndx_name, "r");
if (ndx_in == NULL)
{
    perror(argv[0]);
    sprintf(ch, "%s: Can't open %s.\n", argv[0], ndx_name);
    error(ch);
}

strcpy(ch, argv[1]);
strcat(ch, ".sis");
if (access(ch, F_OK) != 0)
{
    strcpy(ch, argv[1]);
    strcat(ch, ".SIS");
    if (access(ch, F_OK) != 0)
    {
        sprintf(ch, "%s: Can't find neither %s.sis nor %s.SIS.\n",
                argv[0], argv[1], argv[1]);
        error(ch);
    }
}

strcpy(sis_name, ch);
sis_in = fopen(sis_name, "r");

```

```

if (sis_in == NULL)
{
    perror(argv[0]);
    sprintf(ch, "%s: Can't open %s.\n", argv[0], sis_name);
    error(ch);
}

icomp = -1;
for (;;)
{
    if (fgets(ch, MAXLEN, ndx_in) == NULL)
    {
        fclose(ndx_in);
        break;
    }
    strcpy(line1, ch);

    decode_ndx(
        ch,
        channel,
        &npoints,
        &iblock,
        &nblocks,
        &freq,
        &day,
        &month,
        &year,
        &hour,
        &min,
        &sec);

    if (fgets(ch, MAXLEN, ndx_in) == NULL)
    {
        fclose(ndx_in);
        break;
    }

    i = fread(buf, BLKSIZE, nblocks, sis_in);
    if (i < nblocks)
    {
        perror(argv[0]);
        error("Read error on .sis file.");
    }

    if (!strcmp(channel, "Time"))
        continue;

    strcpy(line2, ch);
    strcpy(ch, line1);
    /*
     * Assumed channel is XXXC, where XXX=channel name, and C is component
name (Z/N/E)
     */
    strcpy(ch, channel); ch[3] = '\0';

    if (!strcmp(ch, "PCV"))
    {
        channel[0] = 'P'; ch[0] = 'P';
        channel[1] = 'V'; ch[1] = 'V';
        channel[2] = 'C'; ch[2] = 'C';
    }

    ++icomp;
}

```

```

istation = station_index(ch);

/*
 * Rename station in SSSyymmddhhmm.sis/ndx
 */
if (station[istation].ndx_out == NULL)
{
    station[istation].block_ind = 1;
    station[istation].block_cnt = nblocks;
    station[istation].year = year;
    station[istation].month = month;
    station[istation].day = day;
    station[istation].hour = hour;
    station[istation].min = min;
    station[istation].sec = sec;
    station[istation].freq = freq;

    sprintf(ch2, "%s%02d%02d%02d%02d.ndx",
            sta_out[istation],
            station[istation].year%100,
            station[istation].month,
            station[istation].day,
            station[istation].hour,
            station[istation].min
            );
    station[istation].ndx_out = fopen(ch2, "w");
    if (station[istation].ndx_out == NULL)
    {
        sprintf(ch, "%s: Can't write demultiplexed NDX file: %s.\n",
                argv[0], ch2);
        error(ch);
    }
    fprintf(station[istation].ndx_out,
            "%8s%6d%6d%6d%8.3lf %02d.%02d.%04d %02d:%02d:%06.3lf\n",
            comp_out[icomp],
            npoints,
            station[istation].block_ind,
            station[istation].block_cnt,
            freq,
            day, month, year,
            hour, min, sec);
    fprintf(station[istation].ndx_out, "%s", line2);
    fclose(station[istation].ndx_out);

    if (station[istation].sis_out == NULL)
    {
        sprintf(ch2, "%s%02d%02d%02d%02d.sis",
                sta_out[istation],
                station[istation].year%100,
                station[istation].month,
                station[istation].day,
                station[istation].hour,
                station[istation].min
                );
        station[istation].sis_out = fopen(ch2, "w");
        if (station[istation].sis_out == NULL)
        {
            sprintf(ch, "%s: Can't write demultiplexed SIS
file: %s.\n", argv[0], ch2);
            perror(argv[0]);
            error(ch);
        }
    }
}

```

```

        i = fwrite(buf, BLKSIZE, station[istation].block_cnt,
station[istation].sis_out);
        if (i<station[istation].block_cnt)
        {
            perror(argv[0]);
            sprintf(ch2, "Write error on %s%s.sis.", sta_out[istation],
argv[1]);
            error(ch2);
        }
        fclose(station[istation].sis_out);
    }
    else
    {
        station[istation].block_ind =
station[istation].block_ind+station[istation].block_cnt;
        station[istation].block_cnt = nblocks;

        sprintf(ch2, "%s%02d%02d%02d%02d.ndx",
            sta_out[istation],
            station[istation].year%100,
            station[istation].month,
            station[istation].day,
            station[istation].hour,
            station[istation].min
        );
        station[istation].ndx_out = fopen(ch2, "a");
        if (station[istation].ndx_out == NULL)
        {
            sprintf(ch, "%s: Can't write demultiplexed NDX file: %s.\n",
argv[0], ch2);
            error(ch);
        }
        fprintf(station[istation].ndx_out,
            "%8s%6d%6d%6d%8.3lf %02d.%02d.%04d %02d:%02d:%06.3lf\n",
            comp_out[icomp],
            npoints,
            station[istation].block_ind,
            station[istation].block_cnt,
            freq,
            day, month, year,
            hour, min, sec);
        fprintf(station[istation].ndx_out, "%s", line2);
        fclose(station[istation].ndx_out);

        sprintf(ch2, "%s%02d%02d%02d%02d.sis",
            sta_out[istation],
            station[istation].year%100,
            station[istation].month,
            station[istation].day,
            station[istation].hour,
            station[istation].min
        );
        station[istation].sis_out = fopen(ch2, "a");
        if (station[istation].sis_out == NULL)
        {
            sprintf(ch, "%s: Can't write demultiplexed SIS file: %s.\n",
argv[0], ch2);
            error(ch);
        }
        i = fwrite(buf, BLKSIZE, station[istation].block_cnt,
station[istation].sis_out);
        if (i<station[istation].block_cnt)
        {
            perror(argv[0]);

```

```
        sprintf(ch2, "Write error on %s%s.sis.", sta_out[istation],  
argv[1]);  
        error(ch2);  
    }  
    fclose(station[istation].sis_out);  
}  
}
```

```
#!/bin/csh
#
#   remove file from ftp server ftpriv.ird.nc
#
set SERVER = ftpriv.ird.nc
set USER = drmdata
set PASS = drmdata
echo "Suppression de "$1
/opt/Socks/bin/rftp -v -n $SERVER << EOT
user $USER $PASS
prompt
bin
hash
cd vila
mdele $1
quit
EOT
```

```

#!/bin/csh

set CATHY          = /home/sismol/vila_in
set EXE            = /home/geosci/bin
set AUTODRMDATA   = /data/artemis/autodrm_data
set INDIR         = /data/autodrm/incoming/vila
set DRMDIR        = $AUTODRMDATA/SIS/detections/
#
# *****
# ***** THIS SCRIPT'S NAME *****
# *****
#
setenv NAME      depouil_vila
#
#       To be sure that only one script is running.
#
#-----
#       If script has been crashed
#-----
#
#       Test if $NAME is already running
#
ps -ef | grep $NAME | grep -v grep | grep -v "sh -c" > /tmp/xxx
set flag = `cat /tmp/xxx | wc -l`
\rm -f /tmp/xxx
if ($flag > 1) then
#
#       At least one other script is running
#
    exit 0
else
#
#       No other $NAME script currently running
#
endif

pushd $INDIR >& /dev/null

#
#       Test if there is any .ZIP file
#
ls -l *.ZIP >& /tmp/xxx
set flag = `cat /tmp/xxx | grep -v "No match" | wc -l`
\rm -f /tmp/xxx
if ($flag > 1) then
#
#       At least one .ZIP file
#
else
#
#       No .ZIP file
#
    exit 1
endif
#
foreach i (*.ZIP)
#
    echo "A" | unzip $i >& /dev/null
#
#       Test if the .ZIP files were valid, i.e. there are .NDX and .SIS files
#
ls -l *.SIS *.NDX >& /tmp/xxx
set flag = `cat /tmp/xxx | grep -v "No match" | wc -l`

```

```

\rm -f /tmp/xxx
if ($flag > 1) then
#
#   At least one valid .ZIP file
#
else
#
#   No valid .ZIP file
#
#   exit 1
endif
#
# Place monostation (demultiplexed) Sismalp files in
# autoDRM directories
#
$EXE/dms `echo $i | cut -c1-8` >& /dev/null
#
#   Test if the .SIS/.NDX files were OK, i.e. they have been demultiplexed
#
ls -l *.sis *.ndx >& /tmp/xxx
set flag = `cat /tmp/xxx | grep -v "No match" | wc -l`
\rm -f /tmp/xxx
#
if ($flag > 1) then
#
#   At least one .sis/.ndx file
#
else
#
#   No .sis/.ndx file
#
#   exit 1
endif
#
foreach station (`ls -C1 *.ndx | cut -d. -f1 | cut -c1-3`) >& /dev/null
  set k = `ls -C1 $station*.ndx | cut -d. -f1 | cut -c4-`
  set year = `echo $k | cut -c1-2`
  set month = `echo $k | cut -c3-4`
  set day = `echo $k | cut -c5-6`
  set hour = `echo $k | cut -c7-8`
  set min = `echo $k | cut -c9-10`
  set jday = `$EXE/jday $month$day$year`
#
#
#   Create DRM directories if necessary
#
mkdir $DRMDIR$station >& /dev/null
mkdir $DRMDIR$station/sismalp >& /dev/null
mkdir $DRMDIR$station/sismalp/20$year >& /dev/null
mkdir $DRMDIR$station/sismalp/20$year/$jday >& /dev/null
#
  \mv -f $station$k.ndx
$DRMDIR$station/sismalp/20$year/$jday/$month$day$hour$min.ndx
  \mv -f $station$k.sis
$DRMDIR$station/sismalp/20$year/$jday/$month$day$hour$min.sis
end
#
# Move original Sismalp files for manual processing
#
set j = `ls -C1 *.NDX | cut -d. -f1`
set k = `ls -C1 *.NDX | cut -d. -f1 | $EXE/majmin`

```



```
mkdir $CATHY >& /dev/null
mkdir $CATHY/20$year >& /dev/null

\mv -f $j.NDX $CATHY/20$year/$k.ndx
\mv -f $j.SIS $CATHY/20$year/$k.sis
#
# Delete original Sismalp ZIP files
#
\rm -f $i
#
# Remove original Sismalp ZIP files from ftp server
#
$EXE/supftp $i
#
#
end
```