INSTITUT DES SCIENCES DE L'INGENIEUR
DE MONTPELLIER

FILIERE INFORMATIQUE ET GESTION

BADORA

BA*nque de* **DO***nnées* **RA***dar*

Rapport de Synthèse

Juin **1990**

Stage de 3ème Année

Etudiant

Thierry **VALERO**

Maître de stage

Thierry **LEBEL**

UNIVERSITÉ DES SCIENCES
ET TECHNIQUES DU LANGUEDOC

INSTITUT DES SCIENCES DE L'INGÉNIEUR DE MONTPELLIER

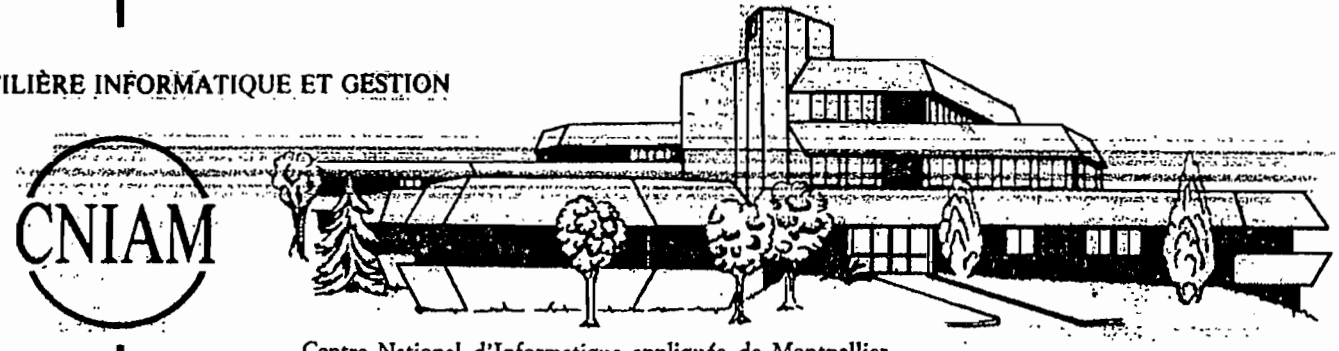
BADORA

BA*nque de* DO*nnées* RA*dar*

Rapport de Synthèse

Juin 1990

FILIÈRE INFORMATIQUE ET GESTION



Centre National d'Informatique appliquée de Montpellier
860, rue de Saint-Priest - 34100 MONTPELLIER - Tél. 67.63.50.65. - Télex USTMONT 490944 F

Ce rapport a été imprimé le 14 juin 1990.

PREFACE

"L'institut français de recherche scientifique pour le développement en coopération risque de n'être plus que ténèbres pour ceux qui ne voudront regarder que lui. C'est dans les temps qui le précédèrent qu'il faut chercher la seule lumière qui puisse l'éclairer".

D'après **Alexis de Tocqueville,**
"L'ancien Régime et la révolution".

REMERCIEMENTS

Je remercie **Thierry LEBEL**, *responsable du projet* **BADORA**, **Michel HOEPFFNER**, *responsable de l'Unité de Recherche 1B (Continent-Atmosphère, Séries climatiques) du Département Terre-Océan-Atmosphère,* **François DELCLAUX** *et* **Alain GIODA** *pour leurs apports tant techniques que matériels, et pour leur disponibilité tout au long de ces trois mois passés en leur compagnie.*

Je remercie également **Valérie THAUVIN**, **Yves ARNAUD** *et* **Philippe CROCHET** *pour leur collaboration active dans le cadre du projet* **EPSAT-Niger**.

Que **Martine CAMPEDEL** *soit remerciée pour sa gentillesse et pour sa livraison quotidienne de caféine.*

Que **Patrick MERDY** *soit remercié pour sa collaboration qui fut agréable et efficace malgré son éloignement.*

Nous tenons à féliciter le personnel pour sa bonne humeur et sa chaleur loin des contrées paradisiaques.

SOMMAIRE

RAPPORT DE SYNTHESE

L'ORSTOM

L'ORSTOM

L'ORSTOM et les pays du tiers monde

L'**ORSTOM**, rebaptisé en 1983 Institut Français de Recherche Scientifique pour le Développement en Coopération, a pour mission de promouvoir la recherche pour le développement des pays et des peuples du tiers monde. Il possède plus de cinquante antennes outre-mer et intervient à la demande sur l'ensemble du globe.

L'office développe, conjointement avec ses partenaires, des programmes de recherche à long terme. Des actions sont menées dans les principaux domaines scientifiques et techniques concernant les activités de ces pays.

L'ORSTOM et Montpellier

L'idée de créer un appui logistique aux équipes outre-mer a donné naissance en 1987 au centre **ORSTOM** de Montpellier. Le centre soutient les recherches dans quatre domaines principaux :

- Santé,
- Hydrologie,
- Biotechnologie,
- Ressources génétiques.

Ces travaux de recherche et de développement sont financés sur la base d'allocations de l'Etat ou de conventions passées avec des entreprises, des pays étrangers, des administrations étrangères et des organismes internationaux (*ONU, FAO, UNESCO,...*).

Le Laboratoire d'Hydrologie

Quatre unités de recherche, dont la plupart des activités se développent à l'étranger, sont représentées dans les secteurs suivants :

- relations Continent-Atmosphère-Séries climatiques,
- géodynamique de l'hydrosphère continentale (qualité de l'eau),
- processus de transformation, fonctionnement et transfert Sol-Eau-Plante-Atmosphère,
- étude et gestion des ressources en eau.

Le laboratoire est chargé d'apporter un appui logistique, informatique et documentaire à la réalisation de ces programmes de recherche.

EPSAT

EPSAT

Le réseau **EPSAT** (**E***stimation des* **P***récipitations par* **SAT***ellite*) a été conçu en 1985 par des chercheurs français en météorologie, afin de développer et comparer des algorithmes de mesure des précipitations par satellite. A cette fin, une expérience spécifique a été mise sur pied en 1988 au Niger dont le but est de valider les algorithmes satellitaires à partir de données spécifiques.

Le satellite ne pouvant fournir de bonnes estimations de précipitation à lui seul, il était nécessaire de mesurer effectivement la pluie au sol avec des pluviographes. Ceci permettra de valider les images satellitaires.

Le pluviographe mesure pour des intervalles de temps variés, les précipitations en un point (plus exactement, sur la surface de réception de son cône, soit 400 cm^2). Afin d'obtenir une évaluation de la pluie sur des surface de plusieurs dizaines de km^2, il a été jugé nécessaire de mettre en place un réseau de pluviographes. Ce réseau de pluviographes couvre la zone géographique étudiée.

Entre le pluviographe qui fournit des mesures ponctuelles mais directes et le satellite qui fournit une mesure moyenne de l'espace-air très indirectement reliée à la pluie, le radar lui, a l'avantage de fournir une mesure indirecte continue dans le temps plus précise que le satellite et avec une meilleur résolution spatiale.

La région expérimentale, autour de Niamey (**NIGER**), fut choisie du fait de l'importance des moyens techniques existants (*radar, réseau de pluviographes accessibles*). De plus, le relief étant peu accidenté et les précipitations étant concentrées sur une courte période, l'utilisation du radar n'en était que plus simple.

LE PROJET

LE PROJET

Afin d'exploiter les données obtenues à partir du radar numérisé de Niamey, il a été nécessaire de réaliser une banque de données.

Au cours du projet de troisième année, auquel je participais avec mes collègues P.MERDY et P.VIALETTO, l'application "Banque de données d'image radar" a été découpée en trois phases :

- **SANAGA** (Système d'Acquisition Numérique pour l'Analyse des Grains Africains), qui numérise et archive les données du radar de Niamey,

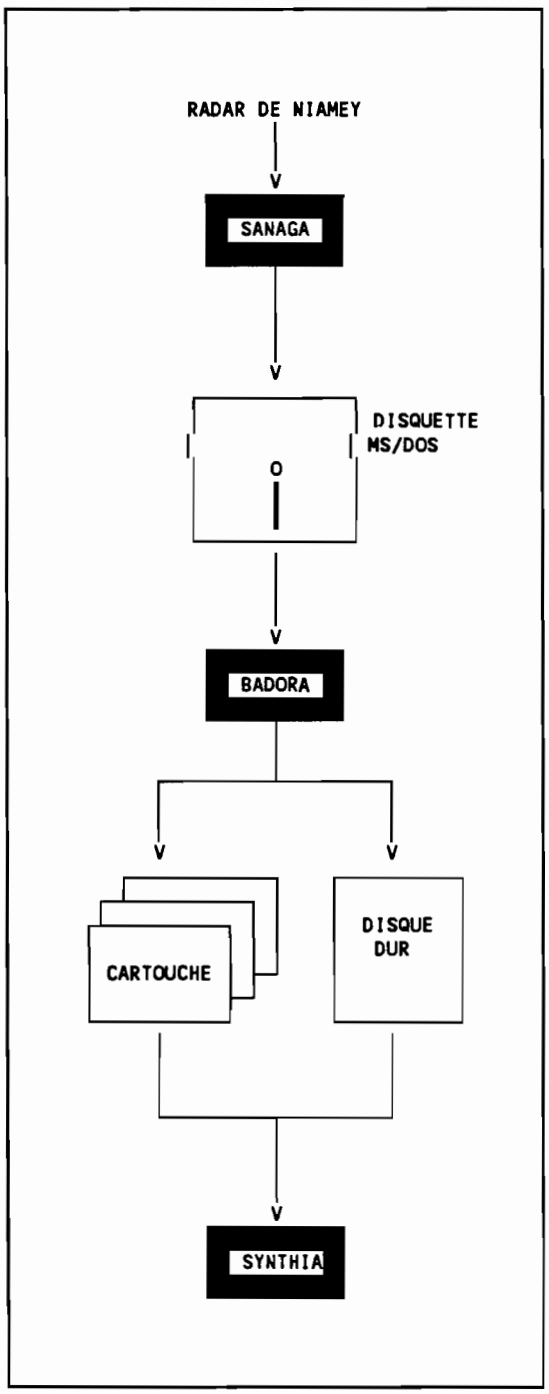
- **BADORA** (**BA**nque de **DO**nnées **RA**dar) qui constitue la banque de données,

- **SYNTHIA** (**SYNTH**èse d'**I**nformations **A**tmosphériques), qui permet au chercheur-utilisateur de visualiser et d'extraire les informations de la banque.

Des options stratégiques ont été définies en collaboration avec le responsable du projet : **BADORA** et **SYNTHIA** seront mises en place sur des stations de travail Apollo, mais ces applications doivent pouvoir être transférées sur d'autres plates-formes dont le système d'exploitation est **UNIX**.

Ma tâche consistait à développer l'application **BADORA** et à la mettre en place sur une station Apollo du Centre ORSTOM de Montpellier.

Etat d'avancement
(Février 1990)



L'application **SANAGA**, réalisée par le Laboratoire d'Aérologie de l'Université Paul Sabatier (Toulouse), est opérationnelle à l'aéroport de Niamey depuis l'été 1989.

BADORA a fait l'objet d'une étude dont les principaux résultats sont l'analyse fonctionnelle, les spécifications détaillées et le choix des outils (**UNIX**, **C** et **C-ISAM**).

Les grandes fonctionnalités de **SYNTHIA** ont été définies (visualisation et extraction des données).

Le point de départ

Je disposais donc des résultats du projet de troisième année, c'est-à-dire :

- du rapport technique (*BADORA - Rapport technique - P.MERDY, P.VIALETTO, T.VALERO - ISIM/ORSTOM - Février 1990*),

- de la maquette qui précisait l'enchaînement des écrans,

- du choix des outils matériel et logiciels.

Champ de l'étude

Mon intervention m'a permis d'aborder les thèmes suivants :

- étude de l'environnement **UNIX,**

- contraintes de portabilité sur le développement en langage C,

- Mise en place d'un environnement de développement croisé à travers un réseau reliant des stations de travail Apollo et des PC,

- Ergonomie de l'application

Modalités et Moyens

T.LEBEL est le responsable du projet, et **F.DELCLAUX** m' a épaulé pour les problèmes informatiques.

L'ORSTOM a laissé à ma disposition les stations de travail, les micro-ordinateurs et les logiciels nécessaires.

LA DEMARCHE

LA DEMARCHE

La mise en place de la banque de données radar restait mon principal objectif, mais ceci ne sera possible qu'à la fin de mon intervention.

Mise à jour des spécifications détaillées

Avant toute chose, il a été nécessaire de mettre à jour les spécifications détaillées afin de prendre en compte une possible évolution du logiciel vers d'autres moyens d'archivage.

Définition des fichiers

Le format des fichiers C-ISAM a été défini, après un premier contact avec cet outil.

L'environnement et les standards de développement

L'application BADORA est le premier projet informatique du Laboratoire d'Hydrologie à être développé en C sous UNIX. A cet effet, deux versions d'UNIX, (System V Release 2 et BSD 4.2) ont été mises en place sur les stations de travail Apollo du Laboratoire.

Il a été nécessaire de définir l'environnement et les standards de développement en tenant compte des contraintes de portabilité et en palliant aux faiblesses du compilateurs C sous Apollo.

Des outils UNIX ont été choisis et il a été décidé de faire un développement croisé : l'application fera d'abord l'objet de compilation sous MS/DOS, puis sera compilée sous UNIX. Pour cela, nous avons dû paramètrer le réseau Apollo.

L'ergonomie de l'application

Les écrans et leurs enchaînements ont été définis dans l'étude précédente (*Rapport, maquette*). Les contraintes de portabilité nous ont obligés à choisir une interface ligne-à-ligne (*télétype*), bien que celle-ci ne soit pas conviviale. Il a donc fallu définir concrètement l'interface en ayant en tête son utilisation quotidienne.

Le développement

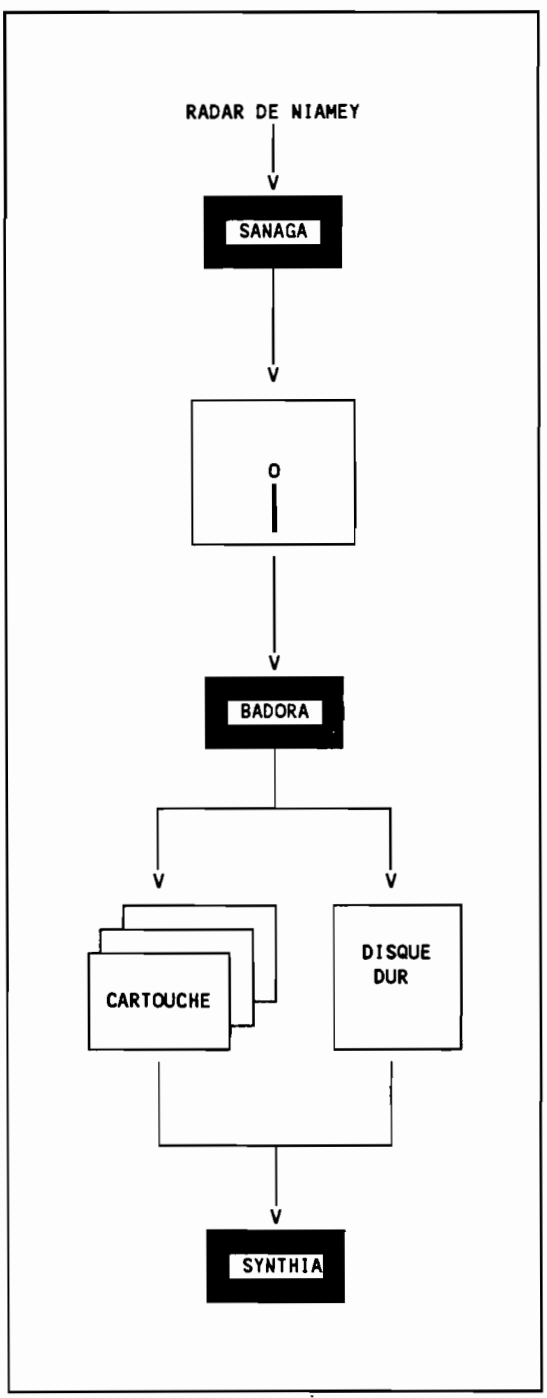
A partir du découpage de l'application précisée dans les spécifications détaillées, les différents modules de l'application ont été écrits. Les plus sensibles ont fait l'objet de programmes de test.

LES RESULTATS

LES RESULTATS

Les résultats se composent de l'application **BADORA** mise en place sur une station Apollo du centre ORSTOM de Montpellier, du rapport technique, et d'un manuel pour les utilisateurs.

Etat d'avancement
(Juin 1990)



L'application **SANAGA**, réalisée par le Laboratoire d'Aérologie de l'Université Paul Sabatier (Toulouse), est opérationnelle à l'aéroport de Niamey depuis l'été 1989.

BADORA est mise en place sur une station Apollo du centre ORSTOM de Montpellier.

La conception de **SYNTHIA** est terminée, et les choix des outils ont été effectués **(UNIX, C, GKS)**.

LES MOYENS MIS EN OEUVRE

LES MOYENS MIS EN OEUVRE

Il est temps de faire un bilan sur le coût et les ressources utilisées pour les projets BADORA et SYNTHIA.

Les coûts n'étant pas les coûts du marché (*coût d'un stagiaire*), il nous a paru indispensable de préciser les ressources.

Les ressources

Conception BADORA

- Novembre 1989 à février 1990
- 3 x 3,5 mois/homme.

Développement de BADORA

- Février 1990 à juin 1990,
- 1 x 5 mois/homme,
- Achat C-ISAM pour Apollo,
- Un aller-retour Montpellier/Niamey.

Conception de SYNTHIA

- Février à juillet 1990,
- 1 x 5 mois/homme,
- Achat GKS pour Apollo,
- Un aller-retour Montpellier/Bordeaux,
- Un aller-retour Montpellier/Niamey.

Développement de SYNTHIA

- Juillet à septembre 1990,
- 1 x 3 mois/homme.

Les Coûts

Conception BADORA

- 3 x 3,5 mois/homme	0,00 F[(1)]

Développement BADORA

- 1 x 5 mois/homme	4 x 5.000,00 F
	1 x 10.000,00 F
- Achat C-ISAM	6.300,00 F
- Aller-retour Montpellier/Niamey	12.000,00 F

Donc, le coût de BADORA est de 48.300,00 F

Conception SYNTHIA

- 1 x 5 mois/homme	2 x 5.000,00 F
	2 x 9.000,00 F
- Achat GKS pour Apollo	1.200,00 F
- Aller-retour Montpellier/Niamey	6.000,00 F
- Aller-retour Montpellier/Bordeaux	1.000,00 F

Développement SYNTHIA

- 1 x 3 mois/homme	3 x 10.000,00 F

Donc, le coût de SYNTHIA est de 52.204,00 F.

Conclusion

Les projets BADORA et SYNTHIA auront donc nécessité 2 années/homme, de novembre 1989 à septembre 1990, et auront coûté environ 100 kF.

---

1 La phase de conception s'est déroulée dans le cadre d'un projet de troisième année de la Filière Informatique et Gestion, de l'Institut des Sciences de l'Ingénieur de Montpellier. Les étudiants n'ont donc pas été rémunérés.

BILAN

BILAN

Vis-à-vis de l'ORSTOM

Ce n'est que lorsque le logiciel sera utilisé à Niamey, que l'on pourra faire un bilan concret sur l'application BADORA.

Apport personnel

Cette intervention m'a permis d'avoir une bonne approche de l'univers UNIX, qui sera, sans conteste, un faire-valoir appréciable dans les années à venir.

Mais surtout, la continuité de mon travail (*stage puis projet*) m'a permis :

- d'avoir un regard critique sur la phase de conception,

- de maîtriser un projet dans sa globalité.

La suite de mon séjour au sein de l'ORSTOM, à l'occasion de l'installation du logiciel à Niamey et de la poursuite du projet de traitement des images radar, me permettra de garder ce même regard critique sur mon travail.

PERSPECTIVES

PERSPECTIVES

A court terme

La réalisation de **BADORA** et l'élaboration du système d'interrogation de la banque de données radar (**SYNTHIA**) doivent être effectives au mois de juillet 1990.

A long terme

BADORA permettra une étude des données pluviométriques pendant cinq ans. Elle posera les jalons du projet **HAPEX** (**H***ydrological* & **A***tmospherical* **P***ilote* **EX***periment*), projet de modélisation des flux de masse (H_2O et CO_2) et d'énergie en évaluant les échanges d'eau et d'énergie entre le sol, la végétation et l'atmosphère en climat tropical sec.

En effet, l'exploitation de cette banque de données permettra d'évaluer la lame d'eau précipitée sur la zone étidiée (100 à 10.000 km²)

INSTITUT DES SCIENCES DE L'INGENIEUR
DE MONTPELLIER

FILIERE INFORMATIQUE ET GESTION

BADORA

BA*nque de* **DO***nnées* **RA***dar*

Rapport Technique

Juin **1990**

Stage de 3ème Année

Etudiant

Thierry **VALERO**

Maître de stage

Thierry **LEBEL**

UNIVERSITÉ DES SCIENCES
ET TECHNIQUES DU LANGUEDOC

INSTITUT DES SCIENCES DE L'INGÉNIEUR DE MONTPELLIER

BADORA

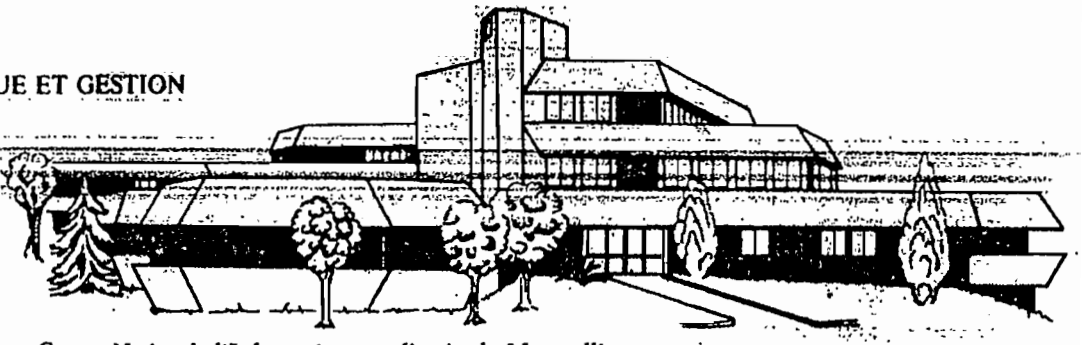
BA*nque de* DO*nnées* RA*dar*

Rapport Technique

Juin 1990

FILIÈRE INFORMATIQUE ET GESTION

CNIAM



Centre National d'Informatique appliquée de Montpellier
860, rue de Saint-Priest - 34100 MONTPELLIER - Tél. 67.63.50.65. - Télex USTMONT 490944 F

Ce rapport a été imprimé le 14 juin 1990.

PREFACE

NOTE DE SYNTHESE

Ce rapport est destiné à être un outil de travail, à la fois pour les personnes qui auront à développer dans un environnement similaire (UNIX,C), et pour ceux qui auront à maintenir l'application.

Sans s'appesantir sur le sujet, une présentation de l'ORSTOM et du projet EPSAT indique dans quel cadre ce stage s'est déroulé (*Présentation de l'ORSTOM, Présentation du projet*).

Le sujet de mon intervention a été défini avec mon tuteur, T.LEBEL (*Définition de la mission*).

Dans un premier temps, il a été nécessaire de modifier quelques points mineurs des spécifications détaillées (*Mise à jour des spécifications détaillées*).

L'application BADORA, a été le premier développement sous UNIX et en langage C, au sein du Laboratoire d'Hydrologie. Les outils nécessaires ont été mis en place, et des standards de travail ont été définis (*Les outils de développement, Les standards de développement, C-ISAM*).

Le format des fichiers a enfin pu être fixé, en tenant compte des contraintes matérielles et logicielles (*Description des fichiers*).

Enfin, l'application a été développée (*Structure de l'application*).

Il est temps de faire le bilan de ce stage (*Bilan*).

TABLE DES MATIERES

CHAPITRE I

PRESENTATION DE L'ORSTOM

SOMMAIRE

PRESENTATION DE L'ORSTOM

1. PREAMBULE

Comme tous les grands organismes scientifiques dont la France se dota dans la période de modernisation de l'après-guerre, l'*ORSTOM* fut créé dans le domaine de responsabilité d'un département ministériel. Sa compétence s'étendit à l'ensemble des domaines scientifiques intéressant le développement économique et le progrès social.

L'existence de l'office a permis à la France de faire face à ses nouvelles responsabilités internationales en se coulant dans les formes nouvelles de la coopération et ceci non seulement vis-à-vis de ses anciennes colonies, mais d'une manière plus générale en prenant sa place dans l'effort international et le dialogue Nord-Sud.

2. UN DEMI SIECLE D'HISTOIRE

2.1. Hier

L'acte de naissance officiel de l'office est la loi du 11 Octobre 1943. Il est né du Centre National de Recherche Scientifique (*C.N.R.S.*). Il fonctionna à ses débuts sous la forme d'un modeste service de Secrétariat d' Etat à la Marine et aux Colonies.

Son rôle était d'orienter, coordonner et contrôler les recherches scientifiques aux colonies. Un des points de départ de l'activité de l'office fut un certain nombre de dossiers de subventions du *C.N.R.S.* portant sur des sujets coloniaux.

Grâce à une expansion géographique importante, l'*ORSTOM* a pu revendiquer le rôle privilégié de porte-parole de la recherche tropicale. L'office a développé, dans ses centres outre-mer, et avec ses partenaires des programmes de recherche à long terme qui sont devenus dans certaines disciplines des références internationales.

2.2. Aujourd'hui

L'*ORSTOM*, Institut Français de Recherche Scientifique pour le Développement en Coopération, dispose de centres, missions ou antennes dans une quarantaine de pays du monde tropical.

L'*ORSTOM* a pour mission de conduire des recherches qui contribuent au développement des régions de la zone intertropicale par l'étude des milieux physiques, biologiques et humains de ces pays. Ces recherches sont conduites en fonction des choix technologiques et scientifiques définis en accord avec des partenaires français et étrangers.

Les programmes de recherches sont conduits par des équipes relevant d'Unités de Recherche regroupées en cinq départements correspondant à cinq champs d'activités :

- Terre, Océan, Atmosphère.
- Milieux et activités agricoles.
- Eaux continentales.
- Santé.
- Sociétés, Développement, Urbanisation.

3. LE CENTRE ORSTOM DE MONTPELLIER

3.1. Dans son ensemble

La construction du centre de Montpellier a été décidée en 1983. Sa vocation est de se consacrer à la mise en valeur des ressources naturelles pour le développement des productions alimentaires tropicales. Cette valorisation des ressources naturelles passe évidemment par leur évaluation et par la connaissance des mécanismes de leur évolution et des conditions de leur exploitation.

Les rôles d'accueil, de formation et de soutien logistique aux programmes développés hors de France sont organisés autour de deux grands domaines scientifiques et une plate-forme technique qui sont :

- Sciences de la vie (*biologie végétale et santé humaine*),

- Sciences de l'environnement (*climatologie, hydrologie, eau, ressources en sol, exploitation des milieux*),

- Laboratoires d'hydrologie, d'analyses biochimiques, insectarium, atelier d'informatique et de traitement d'images.

3.2. Laboratoire d'hydrologie

3.2.1. Axe scientifique

Hydrologie de surface en zone intertropicale.

3.2.2. Objectifs

- La description des régimes hydroclimatiques à l'échelle des grands ensembles géographiques,

- L'étude des mécanismes mis en oeuvre dans le cycle de l'eau *(précipitations, évaporation, ruissellement et infiltration).*

3.2.3. Programmes

Quatre unités de recherche, dont la plupart des activités se développent à l'étranger, sont représentées au laboratoire d'hydrologie.

Leurs recherches sont respectivement orientées dans les secteurs suivants :

- relations Continent-Atmosphère-Séries climatiques,
- géodynamique de l'hydrosphère continentale,
- processus de transformation, fonctionnement et transfert sol-eau-plante-atmosphère,
- étude et gestion des ressources en eau.

Le laboratoire est chargé d'apporter un appui logistique, informatique et documentaire à la réalisation de ses programmes de recherche.

En outre, il assume les fonctions suivantes :

- constituer et gérer une banque de données hydrométriques et pluviométriques, principalement d'Afrique de l'Ouest,
- développer une informatique performante *(traitements statistiques, représentation cartographique, modélisation)*,
- effectuer des recherches en technologie hydrologique. En particulier une chaîne complète de télétransmission de données hydrologiques par satellite *(Systèmes ARGOS et METEOSAT)* est réalisée en collaboration avec les sociétés ELSYDE et CEIS-ESPACE.

CHAPITRE II

PRESENTATION DU PROJET

SOMMAIRE

PRESENTATION DE PROJET

1. LES ZONES SAHELIENNES

L'hydrologie sahélienne au sud du Sahara est quelque peu déconcertante.

D'une part elle offre des facilités parfois exceptionnelles à l'hydrologue :

- saison des pluies très courte permettant à assez peu de frais le maintien de spécialistes sur le terrain,
- formes d'averses simples et peu variées,
- processus de l'écoulement simples avec généralement le ruissellement superficiel prédominant ou parfois seulement l'infiltration,
- couverture végétale intervenant assez peu,
- surfaces cultivées peu importantes,
- types de couches superficielles,
- ciel sans nuage facilitant la télédétection,
- etc

D'autre part, elle présente quelques obstacles redoutables et de dangereux pièges :

- conditions d'accès malaisées,
- différence marquée entre pluie au sol et dans les pluviomètres classiques,
- crues très violentes,
- distributions très asymétriques exigeant de longues séries qui n'existent pas,
- longueur de sécheresse qui défie la statistique et en plus la dégradation hydrographique qui, dès qu'elle est accentuée, perturbe l'écoulement et par suite toutes les conclusions que l'on pourrait déduire de l'examen du bassin versant.

C'est pourtant dans les zones sahéliennes que l'eau est la plus précieuse et c'est surtout là que l'on cherche à en tirer le meilleur parti. Mais on devine aisément que les échecs ont été nombreux : barrages emportés ou au contraire réservoirs qui ne se remplissent que rarement, ponts détruits, etc

Une bonne connaissance du régime hydrologique sahélien s'impose absolument. Très tôt l'*ORSTOM* s'est intéressé tout particulièrement à ce régime et, en coopération avec l'ex service fédéral de l'hydraulique de l'A.O.F., il a aménagé un ensemble de bassins représentatifs et a apporté une contribution très importante à l'aménagement et à l'exploitation des quelques réseaux de stations de jaugeage du Sahel.

2. EPSAT

2.1. Présentation

Le programme *EPSAT* a pour but l'Estimation des Précipitations par SATellite dans la zone sahélo-soudanienne. Il s'agit d'abord de mettre au point les méthodes de mesure, puis d'étudier, à l'aide de ces méthodes, le régime pluviométrique de la zone en question. Les résultats seront ensuite transférés à d'autres régions. La méthode consiste schématiquement à étalonner des mesures par télédétection satellitaire avec des mesures in situ obtenues par un réseau-sol.

Les échelles temporelles et spatiales relativement faibles des systèmes précipitants responsables de la pluie dans la zone concernée ont conduit les membres du réseau *EPSAT* à la conclusion que pour faire avancer les connaissances et mettre au point la méthodologie de mesure, on ne peut se satisfaire de données-sol à faible résolution spatio-temporelle.

Or, des données sol de résolution satisfaisante ne peuvent être obtenues qu'en associant des mesures de champ précipitant par radar à un réseau de pluviographes à faible pas de temps. En l'absence de radar, un réseau de pluviographes de densité prohibitive est nécessaire.

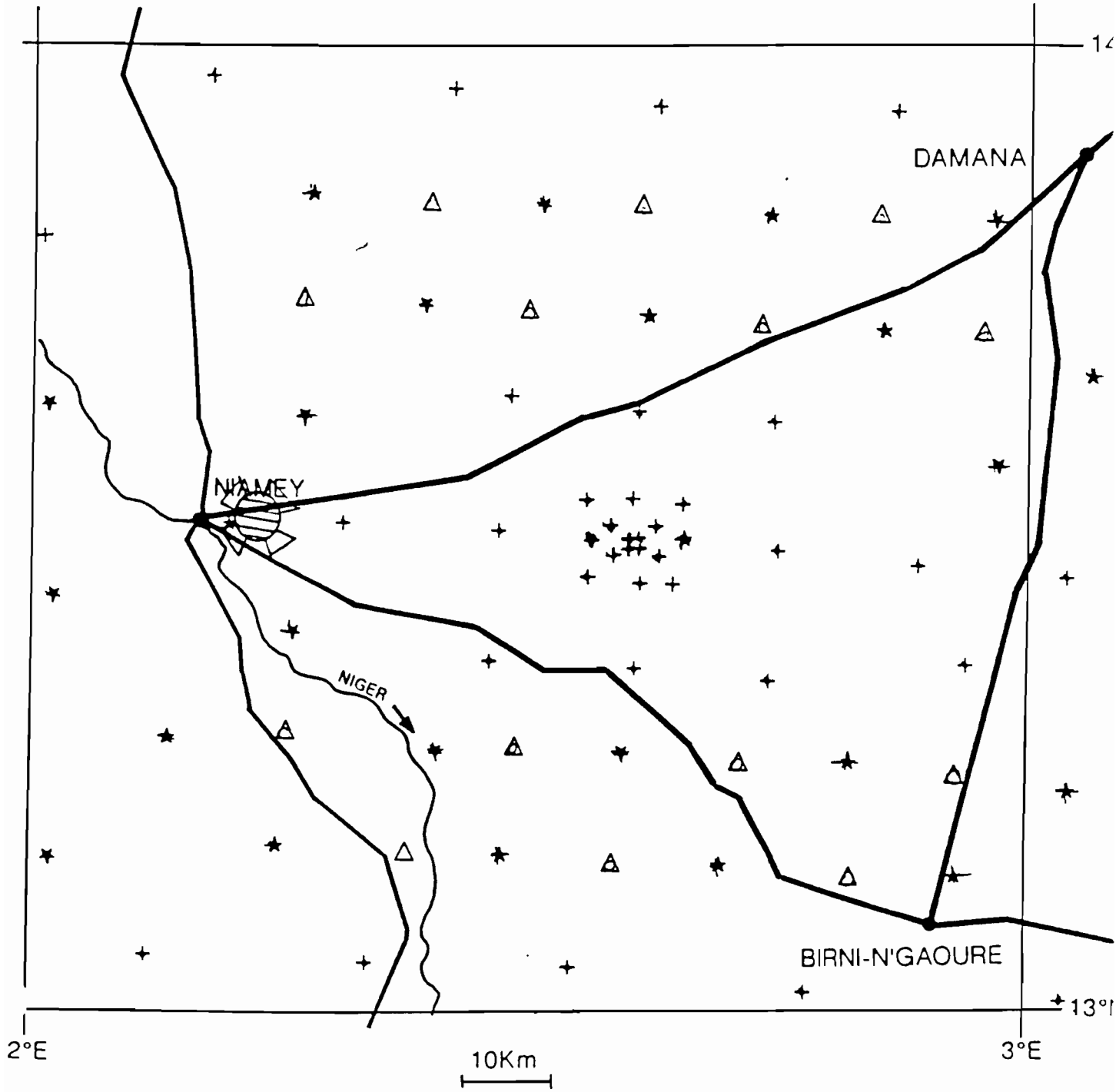
En outre, les champs de réflectivité radar sont utiles pour identifier et classer les divers types de systèmes précipitants responsables des pluies en zone sahélo-soudanienne.

Le présent projet a pour objet la mise à niveau du radar de Niamey pour le projet *EPSAT*. Il vise aussi à répondre aux besoins exprimés par la Direction de la Météorologie du Niger qui souhaite une modernisation du radar et des conditions de son exploitation opérationnelle et notamment une présentation des champs de données sur écran couleur avec mise en mémoire des images des champs passés.

Le système de numérisation-acquisition des données radar développé pour le radar de Niamey a été dénommé SANAGA pour Système d'Acquisition Numérique pour l'Analyse des Grains Africains.

2.2. Zone d'étude

2.2. Zone d'étude



EPSAT-NIGER RECORDING RAIN GAUGE NETWORK

RESEAU PLUVIOGRAPHIQUE D'EPSAT-NIGER

3. LE RADAR

3.1. Présentation

Le radar émet dans l'atmosphère, à intervalles de temps égaux, des impulsions d'énergie électromagnétique puissantes, très brèves et de fréquence élevée.

L'énergie est concentrée en un faisceau de faible ouverture par une antenne directive.

Les cibles de toutes natures, présentes dans le faisceau, interceptent une partie de l'énergie incidente qu'elles absorbent et rayonnent dans diverses directions. La fraction renvoyée vers le radar est le signal utile.

Pour le radar atmosphérique, la cible est constituée par l'atmosphère elle-même et, plus précisément, par les particules de nuages et de précipitations, par des zones ou l'indice de réfraction varie et aussi parfois par les insectes et les oiseaux.

La cible est une région diffusante dont les dimensions sont importantes par rapport à celles du faisceau.

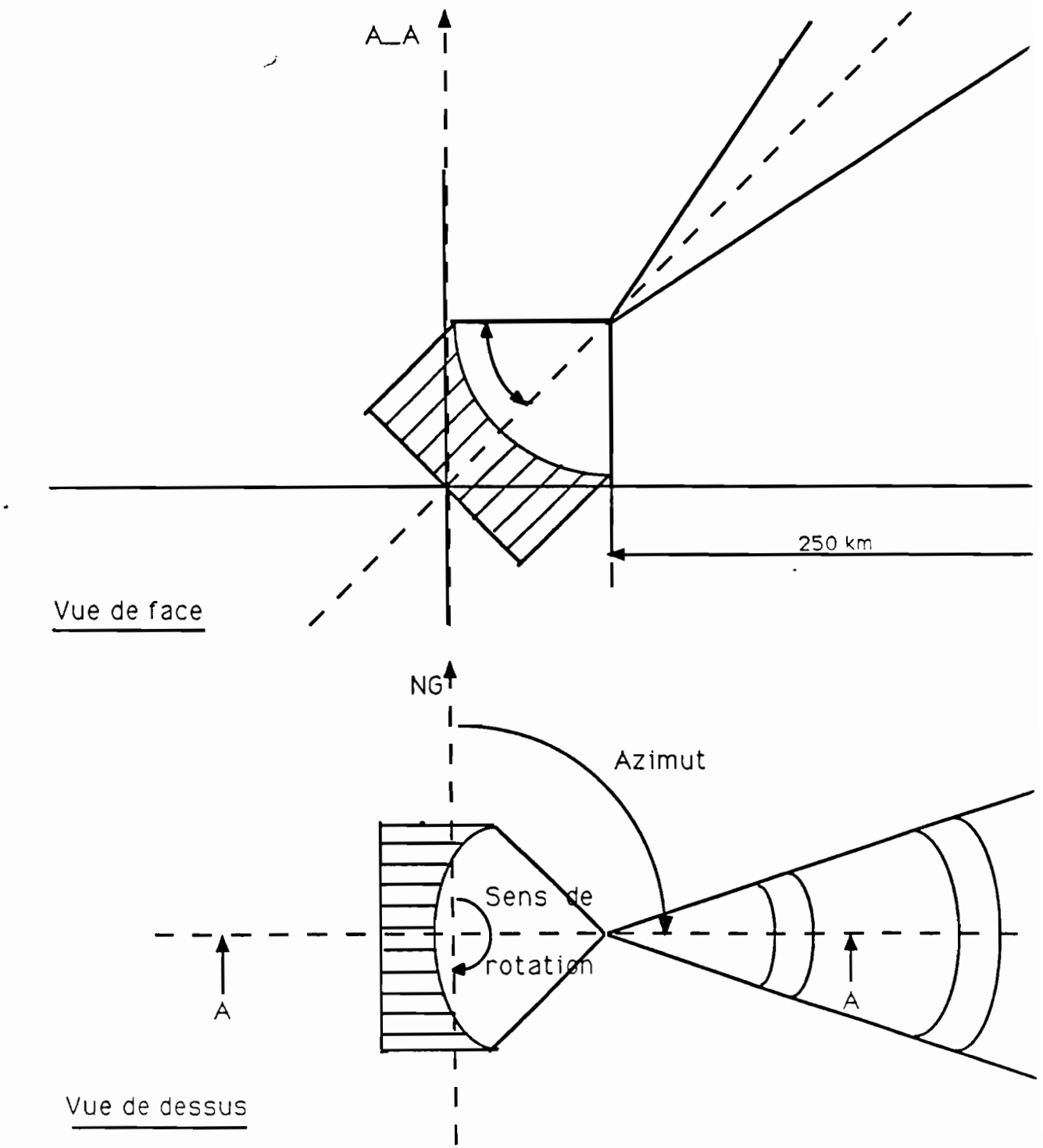
L'observation à toutes les distances dans la direction de visée, combinée aux mouvements de l'antenne permet d'en faire une exploration en volume.

Le radar apporte deux types d'informations :

- d'une part, le signal provenant d'un petit domaine défini par le volume de résolution de l'impulsion, donne des renseignements quantitatifs sur les propriétés du milieu diffusant essentiellement : réflectivité et vitesse,

- d'autre part, l'ensemble des signaux provenant des différents points de l'espace permet de préciser les contours des cibles et, à l'intérieur de celles-ci, la répartition des quantités mesurées, c'est à dire leur structure à petite et moyenne échelle, leur évolution, leur mouvement ainsi que d'autres renseignements.

3.2. Schéma du radar



3.2.1. Les caractéristiques

Le schéma du radar nous amène à définir quelques unes de ses caractéristiques. Tout d'abord, deux rotations possibles, l'une horizontale, l'autre verticale.

La première permet de visualiser toutes les données dans un rayon allant de 250 à 500 kms à site constant.

Ce balayage horizontal, se fait dans le sens des aiguilles d'une montre. A chaque intervalle de tir, une valeur *(Azimut)* exprimée en degré, permet de déterminer l'angle que fait la ligne de visée du radar avec le Nord géographique.

La deuxième exploite les perturbations à des altitudes variables. Le changement d'inclinaison définit un changement de site.

3.2.2. Les constantes des appareils

La chaîne radar est composée de quatre voir cinq sous-ensembles qui sont dans l'ordre : l'antenne, l'émetteur, le récepteur et le système de visualisation.

a. L'antenne

Elle doit posséder les caractéristiques suivantes :

- une très grande directivité,
- un faisceau le plus étroit possible en hauteur ou en site comme d'ailleurs en largeur ou en azimut,
- un aérien de forme parabolique qui doit présenter une grande résistance au vent.

b. L'émetteur

C'est un magnétron ou un auto-oscillateur associé à un maître oscillateur qui produit des signaux électromagnétiques très brefs à intervalles réguliers.

c. Le récepteur

C'est un sous-ensemble capable d'étirer et de mesurer le temps séparant l'émission de la réception.

Au niveau technologique, ce sous-ensemble est très proche d'un appareil radio ordinaire.

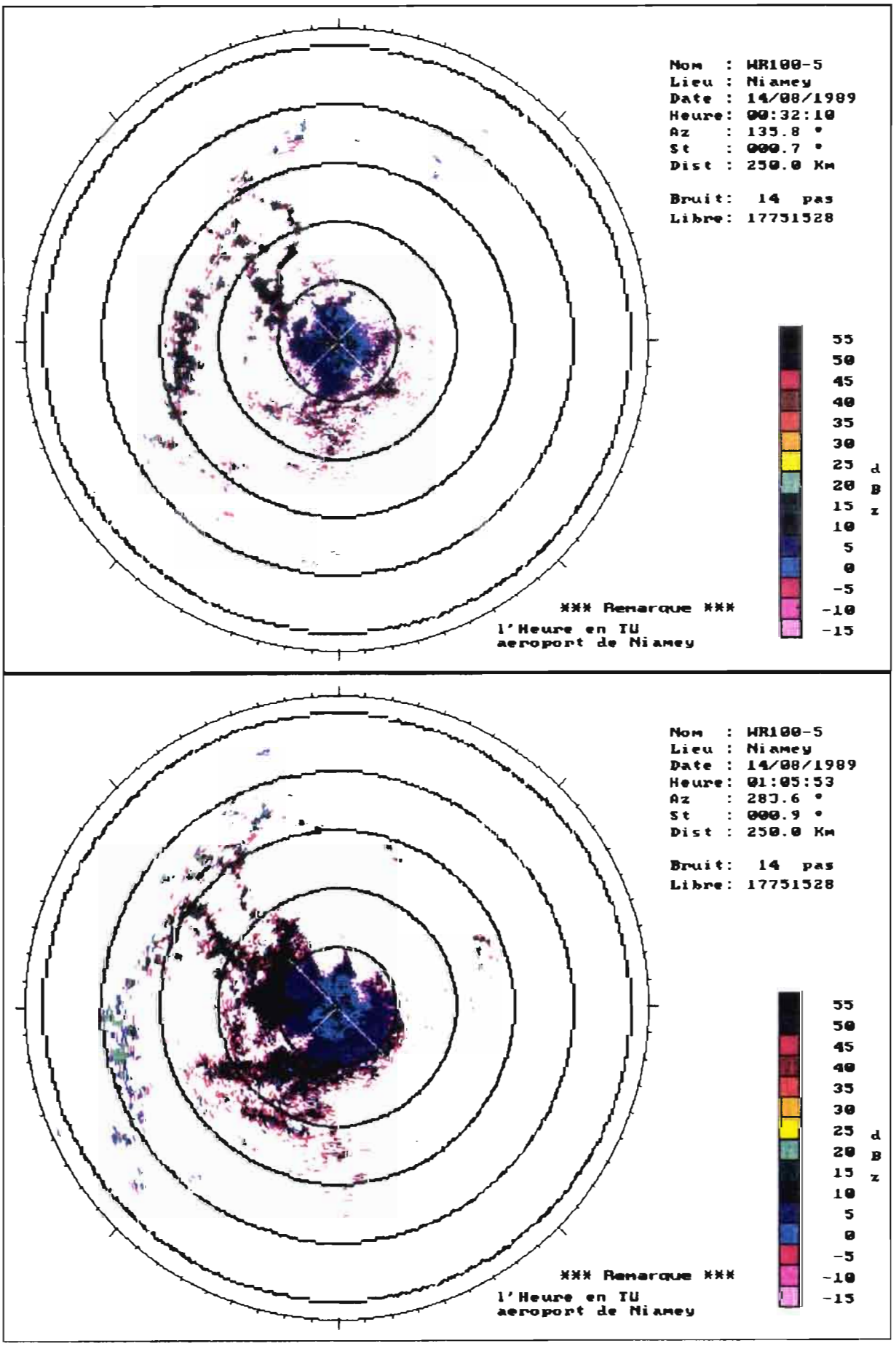
d. Système de visualisation

Ce système s'intègre dans la console de commande et comporte les deux écrans de contrôle suivants dans le cas d'un radar dit analogique :

- un écran de petit diamètre, celui de l'oscilloscope,
- un écran de grand diamètre qui permet de voir les précipitations soit en plan, en position **PPi**, soit en coupe, en position **RHi**.

Dans le projet **BADORA**, nous nous intéresserons aux visualisations en position **PPi**.

3.3. Exemple d'image en PPi



CHAPITRE III

DEFINITION DE LA MISSION

SOMMAIRE

DEFINITION DE LA MISSION

1. INTRODUCTION

1.1. Présentation

Le centre **ORSTOM** de Montpellier (250 personnes) est l'un des derniers éléments de la Technopole **AGROPOLIS** qui comprend 2000 chercheurs.

Dans le cadre de la coopération, le laboratoire d'hydrologie du centre **ORSTOM** de Montpellier mène actuellement une étude des précipitations météorologiques au Niger.

Un radar météorologique installé à Niamey mesure les perturbations atmosphériques. Ces données sont numérisées, compactées, puis archivées sous la forme de fichiers DOS grâce au Système d'Acquisition Numérique pour l'Analyse des Grains Africains (*SANAGA*). Ces fichiers sont ensuite importés sur une station de travail Apollo, en vue de leur exploitation.

L'organisation actuelle des données est peu adaptée à leur exploitation, car il est difficile d'accéder à des données particulières.

Une étude de la future application BADORA, menée par des étudiants de l'Institut des Sciences de l'Ingénieur de Montpellier, a défini les grandes fonctionnalités de l'application BADORA, les choix matériels et logiciels, ainsi que les spécifications détaillées.

1.2. Objectif

Il est nécessaire de mettre au point une chaîne de traitements radar, destinée avant tout à être utilisée dans le cadre de recherches expérimentales.

L'objectif de mon intervention est de développer l'application BADORA, qui mettra à jour la banque de données radar, à partir des fichiers SANAGA.

Les différentes étapes de ma démarche seront :

- Mise à jour de l'étude précédente,
- Etude des outils de développement et définition des standards,
- Développement de l'application BADORA.

2. LA MISSION

2.1. La lettre de mission

Thierry VALERO
Filière Informatique et Gestion

Montpellier le 19 février 1990
à MR Thierry **LEBEL**
Laboratoire d'Hydrologie
ORSTOM - Montpellier

Objet : Définition de la mission
P.J. : Planning prévisionnel

Monsieur,

Suite à notre dernier entretien, je vous présente, ci-dessous, la mission telle que je l'ai perçue.

Dans le cadre de l'exploitation de données informatiques recueillies par le système SANAGA *(radar météorologique installé au Niger)*, je dois développer l'application BADORA, chargée de mettre à jour la banque de données radar.

Dans un premier temps, ma tâche consistera en une mise à jour de l'étude précédente, afin de prendre en compte les modifications conceptuelles et fonctionnelles.

Dans un deuxième temps, l'étude des outils de développement choisis lors de l'étude précédente (*Apollo, UNIX, C-ISAM, C*), me conduira à l'élaboration de la structure physique de l'application et des standards de développement.

Enfin, au terme de ce projet, l'application BADORA sera installée sur un Apollo du Laboratoire d'Hydrologie du centre ORSTOM de Montpellier, et les documents afférents seront livrés.

Les points de départ sont :

- la notice d'utilisation du système **SANAGA,**
- les rapports rédigés au cours du projet BADORA (*février 1990*).

L'intervention se fera sur le site, l'**ORSTOM** met à ma disposition les stations de travail, les logiciels nécessaires, ainsi que les locaux.

Dans l'attente de votre accord, nous vous prions de croire, Monsieur, à l'assurance de nos sentiments dévoués.

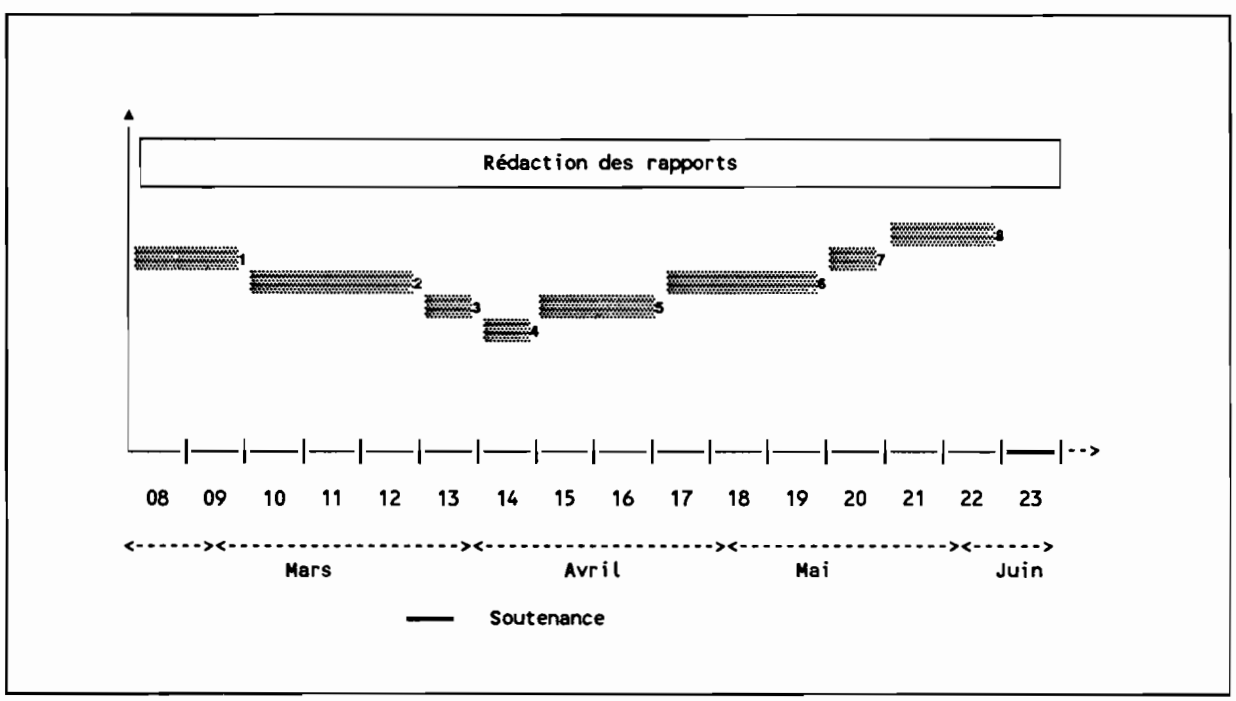
MR. VALERO MR. **LEBEL**

2.2. Les modalités

Les interlocuteurs privilégiés au sein du laboratoire d'hydrologie ont été Messieurs Thierry **LEBEL** et François **DELCLAUX**.

L'intervention s'est faite sur le site, **L'ORSTOM** met à notre disposition les stations de travail, les logiciels nécessaires, ainsi que les locaux.

2.3. Planning prévisionnel



1. Mise à jour de l'étude précédente
2. Formations aux outils de développement
3. Définition physique des fichiers C-ISAM
4. Conception de l'interface BADORA-C
5. Découpage de l'application et plan d'intégration
6. Codage et test
7. Intégration et test
8. Installation et documentation

2.4. Résultats attendus

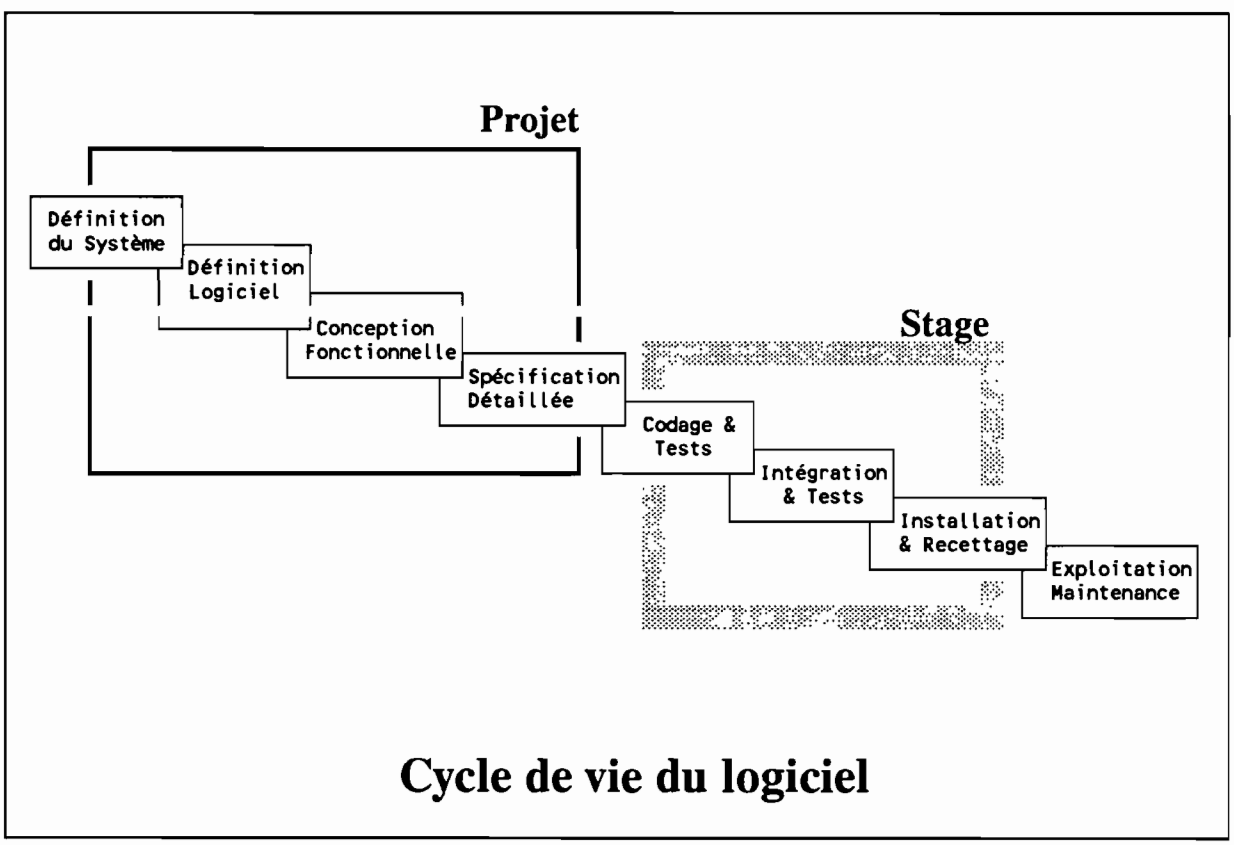
Le logiciel est BADORA installé sur un Apollo du Laboratoire d'Hydrologie du centre ORSTOM de Montpellier.

3. DEMARCHE UTILISEE

Afin de respecter l'*Assurance Qualité* du produit élaboré, nous nous proposons de réaliser les différentes étapes du cycle de vie du logiciel.

La grande partie des étapes sera développée pendant le projet, le reste sera réalisé à la suite d'un stage.

Le schéma ci dessous représente le cycle de vie du logiciel **BADORA** :



Cycle de vie du logiciel

Remarques :

- Chaque étape sera validée avec les utilisateurs avant de passer à la suivante.

- La réalisation du logiciel se fera en deux grandes phases successives :

- Le **projet**, phase regroupant l'analyse et les spécifications,

- Le **stage** : réalisation complète du logiciel.

3.1. Définition du système

On doit définir au début de l'application l'environnement du système, les concepts d'exploitation ainsi que les principales phases du cycle de vie.

Cette étape débouchera sur les résultats suivants :

- Définition et validation des concepts de base de l'exploitation *(homme, machine)*,
- Agencement des principales phases du cycle de vie *(étapes, ressources prévues,...)*

3.2. Définition logiciel

C'est le recensement de l'environnement et des besoins matériels et logiciels.

Les résultats sont les suivants :

- Définition et validation de l'architecture du système *(matériel, logiciel)*,

3.3. Conception fonctionnelle

La conception fonctionnelle du système **BADORA** sera développée dans cette étape. Il en résultera un dossier des spécifications externes *(DSE)* qui comprend :

- Architecture du système,
- Spécification fonctionnelle des traitements et données,
- Enchaînement des écrans du logiciel,
- Maquette fonctionnelle.

Ce dossier mettra en évidence d'une façon fonctionnelle les contraintes et fera ressortir les objectifs.

3.4. Spécification détaillée

L'idée de base de cette étape est la modularité. En effet, afin de maîtriser les domaines de l'application, il est essentiel de diviser d'une façon hiérarchique l'application.

L'application étant ainsi découpée, on spécifiera en détail chaque module, c'est-à-dire les traitements et données, puis les algorithmes et interfaces, ceci dans le but d'avoir une structure de conception suffisamment robuste pour résister au codage et pour être utilisée réellement en intégration et maintenance.

3.5. Codage et tests

Cette étape est constituée par le codage et le test unitaire de chaque module ainsi spécifié. Elle débouche sur les documents suivants :

- code source des modules,
- jeu de test unitaire et résultats,
- rapport d'erreurs,
- bibliothèque des modules,
- informations pour l'intégration.

3.6. Intégration et tests

Le modules ainsi codés et testés unitairement seront assemblés dans cette étape pour donner des ensembles de modules qui seront testés dans leur environnement simulé.

3.7. Recettage et installation

Le logiciel ainsi achevé, l'étape de recettage est importante afin de vérifier avec les utilisateurs la correspondance entre ce qu'ils avaient exprimé *(cahier des charges)* et le logiciel réalisé.

Ensuite, il sera nécessaire d'installer ce logiciel dans l'environnement définitif, et de procéder aux derniers tests.

3.8. Exploitation et maintenance

Les utilisateurs pourront exploiter ce logiciel grâce au manuels de référence et au manuel utilisateur.

Enfin, ils pourront maintenir facilement ce logiciel à l'aide des divers dossiers fournis.

CHAPITRE IV

MISE A JOUR DES SPECIFICATIONS DETAILLEES

SOMMAIRE

MISE A JOUR DES SPECIFICATION DETAILLEE

1. INTRODUCTION

Il est important, à cette phase de l'étude, de mettre à jour les spécifications détaillées, en fonction des contraintes techniques et des besoins de l'utilisateur apparus depuis la fin de l'étude précédente.

Les contraintes techniques sont :

- L'application BADORA devant être portable sur un autre système UNIX, nous devons renoncer à utiliser les types **float** ou **double**, car leurs implémentations dépendent du système. Nous préfèrerons le type **decimal**, défini par C-ISAM qui est indépendant du sytème.
  (*Modification de la table radar*).

- Il est nécessaire d'utiliser deux répertoires temporaires pour les opérations d'archivage et de mise à jour sur cartouche.
  (*Modification de la table paramètre*).

- La définition du répertoire de la banque de données n'est plus dans un fichier mais dans une variable d'environnement.
  (*Modification de la table paramètre*).

Nouveau besoin de l'utilisateur :

- La banque de données BADORA devra pouvoir contenir des enregistrements qui ont déjà été archivés.

REMARQUE

Les modifications apparaîtront en grisé.

2. MISE A JOUR DES ENTITES

2.1. Fichier Catalogue Evénement

DESCRIPTION

C'est l'ensemble de tous les événements, qu'ils soient archivés ou non archivés.

NOM FICHIER

événement

CHAMPS

evddatde..... Date de début de l'événement. La date comporte le jour, le mois, l'année, l'heure, la minute et la seconde. Il s'agit notamment de la date de la première image de l'événement.

evddatfi ..... Date de fin de l'événement. La date comporte le jour, le mois, l'année, l'heure, la minute et la seconde. Il s'agit notamment de la date de la dernière image de l'événement.

evcsta ..... Indique si l'événement est sur disque dur, archivé ou ambivalent. La codification utilisée est :

- D : Disque Dur *(toutes les images de l'événement donné sont sur le disque dur et n'ont pas été archivées)*,
- A : Archivé *(toutes les images de l'événement donné ont été archivées sur une cartouche)*,
- M : Mixte *(toutes les images sont archivées sauf quelques unes qui sont sur le disque dur)*.

evncodsu..... Code du support où est stocké l'événement. Le code est 0 pour le disque dur et supérieur à 0 pour les cartouches.

evcqua..... Indique si l'événement est de bonne ou de mauvaise qualité en fonction de la qualité des images. Si une image appartenant à l'événement est de mauvaise qualité (0) l'événement sera donc de mauvaise qualité.

evtenrra..... Nombre d'enregistrements dans le fichier segment, concernant les images de l'événement.

evnnbrim..... Nombre d'images de l'événement.

CLES

evddatde..... Unique
evddatfi ..... Unique
evcsta
evncodsu

CONTROLE DE VALIDITE

- **evddatde** est inférieur ou égal à **evddatfi.**

- **evnnbrim** est supérieur ou égal à 1 *(un événement comprend au moins une image).*

- un événement correspondant à des images archivées pendant un intervalle de temps, pour deux événements différents, les deux intervalles [date de début, date de fin]$_{(1)}$ ne peuvent se chevaucher.

- si un événement comprend une seule image (**evnnbrim** = 1), les champs **evddatde** et **evddatfi** sont identiques.

REMARQUES

- /

LONGUEUR DE L'ENREGISTREMENT

15 octets

1 : **[evddatde, evddatfi]**.

2.2. Fichier Catalogue Image

DESCRIPTION

Description des images.

NOM FICHIER

image

CHAMPS

imddatde ..... Date de la première radiale de l'image.

imaazide ..... Azimut de départ *(Azimut de la première radiale de l'image)*.

imasecco ..... Secteur couvert.

imasit ..... Site de l'antenne radar lors de la saisie de l'image.

imcres ..... Résolution *(longueur des portes des radiales)*. La signification du code résolution est donnée dans la table radar.

imtenrra ..... Nombre d'enregistrements dans le fichier segment, concernant les radiales de l'événement.

imcarc ..... Indique si l'image est a été archivée :
- Oui : l'image a été archivée.
- Non : l'image n'a pas été archivée.

imcbdr ..... Indique si l'image est accessible directement :
- Oui : l'image est sur le disque dur.
- Non : l'image n'est pas sur le disque dur.

Si l'image n'est pas archivée (imcarc=Non), elle est accessible directement.

imctra ..... Indique si une copie de travail doit rester sur le disque dur, après l'archivage de l'image sur cartouche :
- Oui : Une copie doit être conservée sur le disque dur.
- Non : L'image devra être détruite sur le disque dur aprés son archivage.

Ce champs n'est significatif que si l'image n'est pas archivée (imcarc=Non).

imcqua..... Indique si l'image est de bonne ou de mauvaise qualité :
- 0 : mauvaise qualité,
- 1 : bonne qualité.

CLES

imddatde..... Unique
imcsta

2.3. Fichier Support

DESCRIPTION

Ce fichier décrit la place disponible et utilisée pour les supports de stockage *(ou d'archivage)* de la banque.

NOM FICHIER

support

CHAMPS

suncod ..... Numéro du support de stockage. Le numéro 0 est réservé au disque dur, les numéros supérieurs à 0 aux cartouches *(de 1 à 255)*.

sutvoloc ..... Volume occupé sur le support,

sutvolto ..... Volume total sur le support.

sudarc ..... Date d'archivage du support.
Pour le disque dur, la date d'archivage, est la date de création de la banque de donnée.

CLES

suncod ..... Unique

REMARQUES

- Une cartouche disposant de 40 Mo, une année complète représentant au maximum 270 Mo, l'application BADORA gérant au maximum 255 cartouches, on peut donc stocker 37 années dans BADORA.

3. MISE A JOUR DES TABLES

Ces tables représentent toutes les données définies comme constantes.

3.1. Passage des Tables aux Fichiers C-ISAM

Nous prenons la même codification que celle citée auparavant.

3.2. Fichier Radar

DESCRIPTION

Contient les caractéristiques du radar ainsi que les protocoles de mesure. Ces données sont considérées comme constantes pour un endroit donné, elles peuvent toutefois être modifiées.

NOM FICHIER

radar

CHAMPS

ralnom ..... Nom fonctionnel du radar.

raalat ..... Latitude en degré.

raalon ..... Longitude en degré.

raaouv ..... Ouverture en degré.

rasfreem ..... Fréquence d'émission en MHz.

raspuiem ..... Puissance d'émission en Kw.

rasdurip ..... Durée d'impulsion en seconde.

rasecaip ..... Ecart d'impulsion en Hz.

rasvolre ..... Volume de résolution en mètre.

ractyp ..... Type du radar :
- C : radar conventionnel,
- D : radar à effet Doppler,

rasres ..... Tableau contenant les résolutions possibles en mètre, d'occurrences 10.

CLES

- /

REMARQUES

- La liste des types de radar n'est pas exhaustive. La convention prise pour le code du type du radar est de prendre la première lettre du nom du type de radar.

- Les valeurs sont fixées pour un radar donné.

- Il y a qu'un seul enregistrement dans ce fichier.

3.3. Fichier Discontinuité

DESCRIPTION

Ce fichier rassemble les paramètres de regroupement et de synthèse des radiales en images et des images en événement.

NOM FICHIER

discontinuité

CHAMPS

diasit ..... Ecart maximum de site entre deux radiales de la même image. Cet écart est en degré.

diaazi ..... Ecart maximum d'azimut entre deux radiales successives de la même image. Cet écart est en degré.

diasec ..... Secteur couvert minimum pour qu'une image soit archivée. Ce secteur est en degré.

distem ..... Ecart maximum de temps entre deux images successives de l'événement. Cet écart est en minute.

diaech ..... Ecart qui différencie deux radiales. Cet écart est en degré *(échantillonnage de l'azimut)*.

CLES

- /

CONTROLE DE VALIDITE

- /

REMARQUES

- Attention ces paramètres peuvent évoluer.

- Il y a un seul enregistrement dans ce fichier.

3.4. Fichier Journal des Transferts

DESCRIPTION

Ce fichier contient le compte rendu de la création ou de la mise à jour de la banque de données. A chaque traitement d'un fichier **SANAGA** on écrit dans ce fichier le résultat de la transaction.

NOM FICHIER

transfert

CHAMPS

jtddat ..... Date de début de réception du fichier de données **SANAGA**. La date comprend l'année, le mois, le jour, l'heure, la minute et la seconde.

jtfnom ..... Nom du fichier transféré dans la banque

jttsan ..... Volume des données transférées dans la banque en Kilo Octets.

jtcrlt ..... Résultat du compte rendu de la transaction :
- 0 : la transaction s'est déroulée correctement,
- 1 : la transaction a été interrompue, il faut recommencer l'opération.

jtrsan ..... Répertoire du fichier transféré dans la banque.

CLES

jtddat ..... Unique

CONTROLE DE VALIDITE

- /

REMARQUES

- /

3.5. Fichier Journal des Archivages

DESCRIPTION

Ce fichier contient le compte rendu de l'archivage des événements. C'est à dire que l'on peut lire dans le journal le résultat de ces archivages.

NOM FICHIER

archivage

CHAMPS

jaddat ..... Date de début de réception du fichier de données **SANAGA**. La date comprend l'année, le mois, le jour, l'heure, la minute et la seconde.

jansup ..... Numéro de la cartouche où sont archivés les événements. Les numéros seront répartis comme suit :
- 0 : réservé pour le disque dur,
- 1 à 255 : pour les cartouches de streamer.

jactra ..... Code identifiant le type de traitement :
- A : archivage,
- M : mise à jour d'une cartouche.
- H : restauration d'archivage.
- I : restauration de mise à jour.

jacrlt ..... Résultat du compte rendu de la transaction d'archivage :
- 0 : la transaction s'est déroulée correctement,
- 1 : la transaction a été interrompue, il faut recommencer l'opération d'archivage.

CLES

jaddat ..... Unique

CONTROLE DE VALIDITE

- /

REMARQUES

- /

3.6. Fichier Paramètres

DESCRIPTION

Paramètrage de la configuration matérielle et logicielle de l'application.

NOM FICHIER

paramètre

CHAMPS

parsan..... Répertoire des fichiers SANAGA.

pararc..... Répertoire d'archivage *(désignation du périphérique streamer)*.

parprt..... Commande Unix d'impression d'un fichier.

partp1..... Répertoire temporaire sur disque dur.

partp2..... Répertoire temporaire sur disque dur.

patbad..... Volume disponible sur le disque dur pour la banque de données BADORA.

patsup..... Volume disponible sur les supports d'archivage

CLES

- /

CONTROLE DE VALIDITE

- /

REMARQUE

- Il y a un seul enregistrement dans ce fichier.
- Le champs **parbad**, défini par *"Répertoire de la banque de données BADORA"*, est remplacé par la variable d'environnement **BDRDR**.

CHAPITRE V

LES OUTILS DE DEVELOPPEMENT

1. INTRODUCTION

Le présent chapitre s'adresse à des personnes connaissant déjà UNIX et le langage C.

Nous présenterons les outils de développement utilisés pour l'application BADORA, au sein du Laboratoire d'Hydrologie du Centre ORSTOM de Montpellier$_{(2)}$.

BADORA est une application qui sera écrite en C, et devra être portable sur un autre système UNIX.

UNIX désigne UNIX BSD4.2 sous AEGIS

AEGIS désigne AEGIS SR 9.7

C-ISAM désigne C-ISAM version 3.10 pour UNIX

Turbo C désigne Turbo C version 1.5 pour PS/2 et compatible PC/XT, AT

MS/DOS désigne MS/DOS version 3.20

2 : Voir BADORA, Rapport Technique, Chap IV, L'environnement de l'application.

2. LE CYCLE DE TRAVAIL

L'analyse de l'application étant faite, le cycle de développement est :

- 1. Mise au point d'un programme sans erreur de syntaxe :
  - 1 saisie du source des programmes
  - 2 compilation
  - 3 correction des erreurs de syntaxe
  - 4 compilation (*retour en 3*, éventuellement)

- 2. Mise au point d'un programme sans erreur à l'exécution :
  - 5 Test
  - 6 Correction des erreurs à l'exécution
  - 7 Compilation
  - 8 Test (*retour en 6, éventuellement*)

2.1. Mise au point d'un programme sans erreur de syntaxe

Le compilateur C, sous DOMAIN/IX, ne disposant pas de la notion de prototype et l'utilitaire **lint** signalant trop d'erreurs, nous compilerons depuis le PC connecté au réseau Apollo/Domain, avec le compilateur Turbo-C, sous MS/DOS.

2.1.1. Connexion au réseau Apollo Domain

Avant d'accéder au réseau, il faut rajouter les extensions .c et .h dans le fichier **/sys/dpci/dpci_cvrt.dat**[3], afin d'accéder à ces fichiers normalement.

La connexion PC/Apollo s'effectue par le logiciel **DPCIRING** permettant les deux fonctions suivantes :

- Emulation d'un terminal VT100 connecté à l'Apollo (*commande TERM*),
- Accès aux ressources Apollo (disque, imprimante) depuis MS/DOS (*commande CONNEX*). La *home directory* est accessible par G:\ et l'imprimante Apple LaserWriter II par LPT2:.

3 : Voir annexe.

2.1.2. Utilisation de Turbo C.

Ce dernier effectue la plupart des tests de l'utilitaire lint.

Nous utiliserons alors la notion de prototype.

```
/*************************************************/
/*                                               */
/* Exemple de déclaration de fonction externe */
/*                                               */
/*************************************************/


#ifdef __MSDOS__

/*************************************************/
/*                                               */
/* Definition avec prototype pour Turbo C     */
/*                                               */
/* Le compilateur Turbo C vérifiera que lors  */
/* d'un appel de fonction1, le premier para-  */
/* mètre est un pointeur sur un caractère, et */
/* le deuxième paramètre un entier long.      */
/*                                               */
/*************************************************/

extern int fonction1( char *c, long l);

#else

/*************************************************/
/*                                               */
/* Definition pour le compilateur Domain C    */
/*                                               */
/* On déclare le nom de la fonction           */
/*                                               */
/*************************************************/

extern int fonction1();

#endif
```

Les options de compilations sont :

Options de compilations sous turbo C

Identifier length	8
Nested comments	Off
ANSI keywords only	On

A: Non-portable pointer conversion	On
B: Non-portable pointer assignment	On
C: Non-portable pointer comparison	On
D: Constant out of range in comparison	On
E: Constant is long	On
F: Conversion may lose significant digits	On
G: Mixing pointers to signed and unsigned char	Off

A: 'ident' not part of structure	On
B: Zero length structure	On
C: Void functions may not return a value	On
D: Both return and return of a value used	On
E: Suspicious pointer conversion	On
F: Undefined structure 'ident'	On
G: Redefinition of 'ident' is not identical	On

A: Function should return a value	On
B: Unreachable code	On
C: Code has no effect	On
D: Possible use of 'ident' before definition	On
E: 'ident' is assigned a value which is never used	On
F: Parameter 'ident' is never used	On
G: Possibly incorrect assignment	On

A: Superfluous & with function or array	On
B: 'ident' declared but never used	On
C: Ambiguous operators need parentheses	On
D: Structure passed by value	On
E: No declaration for function 'ident'	On
F: Call to function with no prototype	On

2.2. Mise au point d'un programme sans erreur à l'exécution

Cette phase du développement sera faite sur Apollo, sous AEGIS, sous UNIX Bsd4.2.

2.2.1. Le compilateur Domain C

Le compilateur Domain C sera utilisé pour générer des fichiers objets sur Apollo.

La syntaxe d'appel utilisée est :

```
cc -W0,-l,-std -g -Tsys5 -c nomfic.o nomfic.c
```

Avec :

nomfic.c, nom du fichier source à compiler,
nomfic.o, nom du fichier objet généré,
nomfic.lst, nom du fichier contenant le listing de compilation,
-W0,-l,-std envoie les paramètres -l et -std au compilateur AEGIS, où :
-l, demande la création du fichier nomfic.lst,
-std, compilera selon la syntaxe du C K&R.
-g, demande la création des informations pour le debugger.
-Tsys5, demande une compilation System V, cette option étant obligatoire pour utiliser C-ISAM.

Remarque : les fichiers include (.h), seront recherchés sur le répertoire de travail et sur le répertoire //CASTOR/sys5/usr/include.

2.2.2. L'éditeur de lien

L'éditeur de lien ld permettra d'obtenir des fichiers exécutables.

La syntaxe d'appel utilisé est :

```
ld -o nomfic.exe nomfic1.o nomfic2.o -lisam
```

Avec :

nomfic.exe, nom du fichier exécutable généré,
nomfic1.o, nom d'un fichier objet,
nomfic2.o, nom d'un fichier objet,
-lisam, paramètre demandant l'édition de lien avec la bibliothèque libisam.a, située dans le répertoire //CASTOR/sys5/usr/lib

2.2.3. L'utilitaire make

La compilation et l'édition de lien ne seront pas effectuées directement mais grâce à l'utilitaire **make**, qui prend en compte les dates de modification des fichiers source, objet et exécutable.

Ceci permet de mettre à jour l'ensemble d'une application, en ne lançant que les compilations et les éditions de liens nécessaires.

Le fichier **makefile**, situé dans le même répertoire que les fichiers source, décrit la dépendance entre les différents modules de l'application et les moyens de les générer.

L'exemple décrit une application utilisant C-ISAM et comprenant deux programmes exécutables bdrbuild.exe et testfile.exe.

Ces deux programmes utilisent le module objet bdrfile.o dont le source est dans le fichier bdrfile.c et la déclaration dans bdrfile.h.

Exemple de fichier makefile

```
#Ceci est un commentaire.
#Make BADORA

#Parametre de compilation
#   -Tsys5 : obligatoire avec C-ISAM
#   -g     : pour debugger
CFLAGS= -W0,-l,-std -g -Tsys5

#Parametre pour l'edition de lien (bib C-ISAM)
LIBISAM= -lisam

#Par défaut le but à atteindre est all.
#Lorsque la commande make all est lancée, make cherchera à générer all.
#C'est à dire bdrbuild.exe et testfile.exe
all: bdrbuild.exe testfile.exe


#  Pour générer bdrbuild.exe, make vérifie que la date de ce fichier est supérieure
#aux dates des fichiers bdrbuild.o et bdrfile.o
#  Si ce n'est pas le cas, la commande
#   ld -o bdrbuild.exe bdrbuild.o bdrfile.o -lisam
#est lancée.
bdrbuild.exe: bdrbuild.o bdrfile.o
      ld -o $@  bdrbuild.o bdrfile.o $(LIBISAM)

bdrbuild.o: bdrbuild.c bdrfile.h
      cc $(CFLAGS) -c $@ bdrbuild.c

#  Pour générer bdrbuild.o, make vérifie que la date de ce fichier est supérieure
#aux dates des fichiers bdrfile.c et bdrfile.h
#  Si ce n'est pas le cas, la commande
#   cc -W0,-l,-std -g -Tsys5 -c bdrfile.o bdrfile.c
#est lancée.
bdrfile.o: bdrfile.c bdrfile.h
      cc $(CFLAGS) -c $@ bdrfile.c


testfile.exe: testfile.o  bdrfile.o
      ld -o $@  testfile.o bdrfile.o $(LIBISAM)

testfile.o: testfile.c bdrfile.h
      cc $(CFLAGS) -c $@ testfile.c

#  clean efface tous les fichiers objet, tous les fichiers executable
#et tous les listings de compilation.
#  Donc aprés avoir lancé make clean, tous les fichiers executable, objet et listing
#auront été effacés.
#  Si on lance alors make all,
#toutes les compilations et toutes les éditions de liens seront faites.
clean:
      -rm *.o
      -rm *.exe
      -rm *.lst
```

REMARQUES :

- $@ désigne le nom du fichier à générer,
- Toutes les commandes UNIX sont précédées d'un caractère tabulation (Tab),
- Le signe moins (-), indique que make ne s'arrête pas lorsqu'un code d'erreur est renvoyé,
- La commande make sans paramètres équivaut à la commande make all.

3. LES OUTILS DE DOCUMENTATION

3.1. Index des fonctions

Unix propose la commande ctags qui permet de créer un index sur les fonctions définies dans un ensemble de fichier source.

Par exemple :

```
#ctags -x bdr*.c bdr*.h
bdra1            295 bdra1.c          int bdra1 ()
bdra2             43 bdra2.c          int     bdra2 ()
bdra4             41 bdra4.c          int     bdra4 ()
bdrerr            39 bdrt9.c          void    bdrerr(ptbderr0)
bdrm21            73 bdrm21.c         int     bdrm21 (pttaille1, ptpcentocc1, ptarch1)
bdrm231          388 bdrm231.c        int     bdrm231 (nomfic1)
bdrm24           163 bdrm24.c         int     bdrm24 ()
bdrm43           171 bdrm43.c         int     bdrm43 ()
```

La fonction bdrerr est donc définie à la ligne 39, du fichier bdrt9.c, et la première ligne de la définition est "void bdrerr...".

3.2. Relations entre les modules

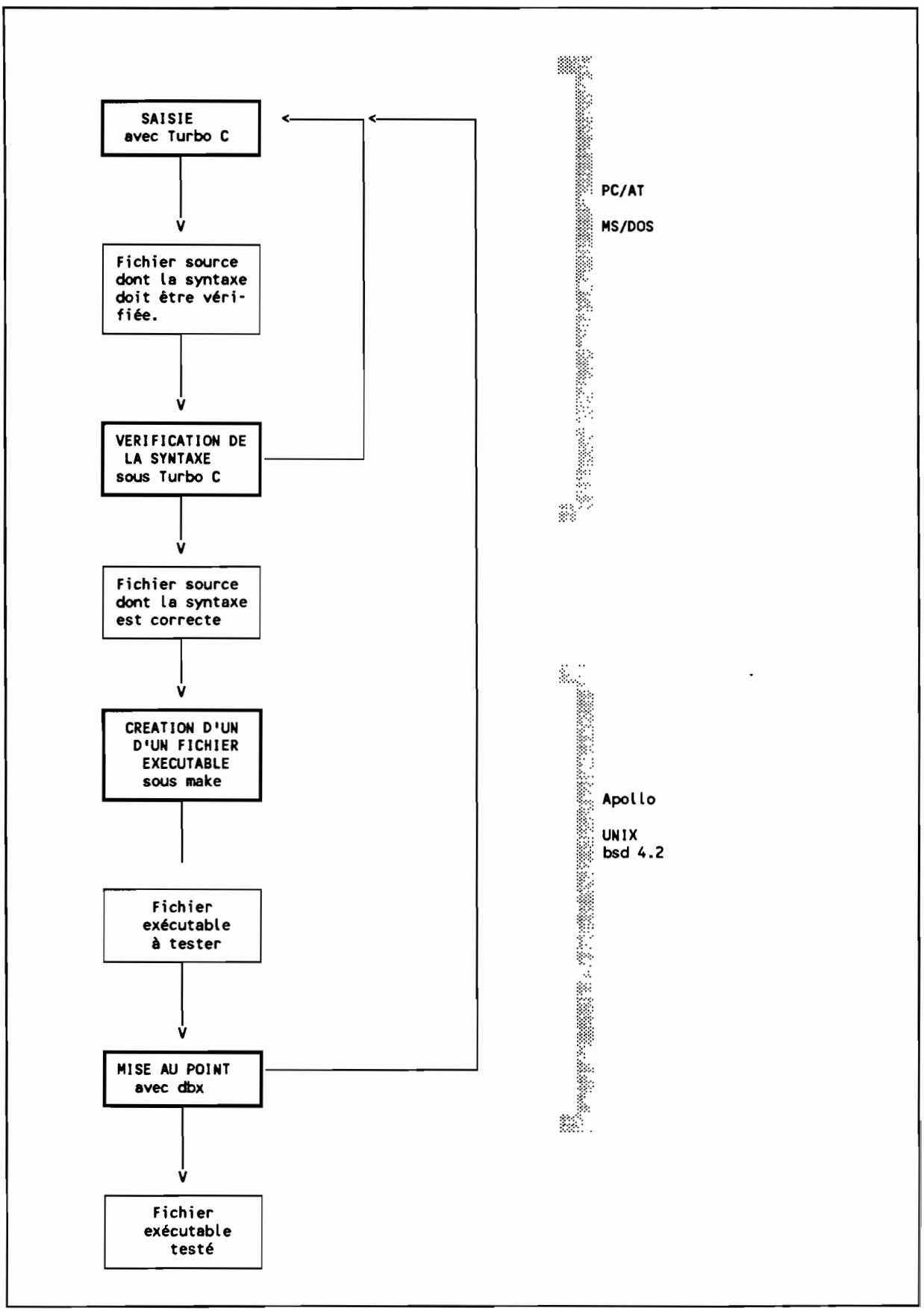
La commande lorder permet de définir les relations du type (module1, module2), signifiant que le module objet module1, utilise module2. Il est néanmoins nécessaire de trier le fichier créé par cette commande et de supprimer les lignes identiques.

Par exemple :

```
lorder | sort -u
bdra0.o bdra0.o
bdra0.o bdra1.o
bdra0.o bdra2.o
bdra0.o bdra4.o
bdra0.o bdrt1.o
bdra0.o bdrt4.o
bdra0.o bdrt5.o
bdra1.o bdrt9.o
```

Le module objet bdra0.o fait directement appel aux modules bdra1.o , bdra2.o,...

4. TABLEAU RECAPITULATIF



CHAPITRE VI

LES STANDARDS DE DEVELOPPEMENT

SOMMAIRE

LES STANDARDS DE DEVELOPPEMENT

1. INTRODUCTION

Nous allons présenter les choix relatifs aux standards de développement.

Dans un premier temps nous indiquerons dans quel langage C sera écrite l'application.

Dans un deuxième temps, les hypothèses à faire ou à ne pas faire, au sujet des types de données seront précisées.

Enfin, un mode de présentation des sources sera choisi.

Comme dans la plupart des phases de ce projet, la portabilité de l'application influera sur le choix des standards de développement.

Dans la suite de ce chapitre,

- Le langage C K&R, désigne le langage C défini dans *"The C Programming langage"* de D.Ritchie[1].
- Le langage C ANSI, désigne le langage C défini dans *"Draft Proposed American National Standard for Information Systems - Programming Langage C"*, du comité X3 de l'ISO[2].

Les choix effectués sont inspirés de *"Rationale for Draft Proposed American National Standard for Information Systems - Programming Langage C"*[3].

1 Voir bibliographie.

2 Voir bibliographie.

3 Voir bibliographie.

2. LA SYNTAXE DU LANGAGE C

Nous rapellons ici, que l'application BADORA fait l'objet d'un développement croisé. Les programmes sont d'abord compilés avec Turbo C, sous MS/DOS, avant que les fichiers exécutables soient créés avec le compilateur DOMAIN C, sous DOMAIN/IX.

2.1 Le compilateur Turbo C

Ce compilateur semble respecter la norme ANSI du langage C, mais néanmoins, accepte la syntaxe K&R. Les diagnostics du compilateur permettent de détecter la plupart des problèmes de portabilité.

2.2 Le compilateur DOMAIN C

Il respecte la syntaxe K&R, mais n'autorise pas la syntaxe ANSI. Les diagnostics à la compilations sont trop rares.

2.3 La synatxe choisie

La syntaxe K&R sera choisie.

Cependant, nous utiliserons les possibilités de compilation conditionnelle pour compiler selon le C ANSI sous MS/DOS. Plus précisément, sous MS/DOS :

- les fonctions seront déclarées à l'aide de prototype,
- le mot clef **`const`** précisera les paramètres non-modifiables.

La compilation sous MS/DOS permettra donc, un contrôle plus strict de la syntaxe que sous DOMAIN/IX.

3. LES DONNEES

3.1 Les entiers

- **`int`** : entier signé de -32768 à +32767, codé sur deux octets.
- **`long int`** : entier signé codé sur 4 octets.

On ne fera aucune hypothèse sur une précision supérieure et on n'accèdera pas directement aux octets. C'est-à-dire que des syntaxes du type :

```
int i;
int pf;

/* lecture du poids fort */
pf = *( (  (*char) &i  ) + 1 )
```

seront prohibées.

3.2 Le type char

Le type **char**, considéré comme un nombre, aura une étendue de -128 à +127. Lorsque l'on voudra des octets (non-signés), on précisera **unsigned char**.

Le type **char** en tant que caractère alphanumérique utilisera les codes ASCII de 0 à 127 (en fait, on se limitera à l'alphabet du langage C).

3.3 Les nombres en virgule flottante

Aucune hypothèse ne sera faite en ce qui concerne la précision ou le codage interne de ces nombres.

4. LES FONCTIONS

4.1 Les entrées sorties

Les fonctions d'entrées/sorties définies dans **<io.h>** ne seront pas utilisées. En effet, cette bibliothèque n'est pas normalisée et son comportement varie selon l'implémentation.

Nous utiliserons donc les fonctions définies dans **<stdio.h>** qui sont normalisées.

4.2 La gestion de l'heure

N'ayant aucune garantie quant à la conformité avec la norme ANSI de la bibliothèque **<time.h>** du compilateur DOMAIN C et vu l'importance de ce type de traitement dans l'application BADORA, nous écrirons les fonctions nécessaires (module BDRT8). Néanmoins nous utiliserons le type **struct tm**.

4.3 Les manipulations des chaînes de caractères

Les fonctions définies dans **<string.h>** seront appelées.

4.4 Les paramètres de l'implémentation

Les valeurs définies dans **<limits.h>** seront utilisées dans l'application BADORA, mais ne seront pas utilisées dans les fichiers.

4.5 Remarque

- Le compilateur DOMAIN C ne propose pas tous les fichiers en-têtes standards (**<stdlib.h>** par exemple).
- L'appel à des fonctions ne faisant pas partie du standard ANSI, sera dûment signalé et argumenté.

5. LA PRESENTATION

Le fichier **bdr.h** regroupera les primitives du macro-processeur concernant l'inclusion des fichiers standards.

Chaque module objet comprendra deux fichiers sources :

- Un fichier **.h**, comprenant la déclaration des fonctions et variables publiques,
- Un fichier **.c**, comprenant la définition des fonctions et procédures.

6. EXEMPLE

```
/*-------------------------------------------------------------*/
/* FICHIER bdrex.h                                             */
/*                                                             */
/* CONTENU : Exemple de module                                 */
/*                                                             */
/*                                                             */
/* PROTOTYPES : ./                                             */
/*                                                             */
/* VERSION : 1.0                                               */
/*                                                             */
/* DATE    : 16/03/90                                          */
*                                                             */
/*-------------------------------------------------------------*/

/*----------------- MAINTENANCE -------------------------------*/
/*                                                             */
/*  MODIFIE LE --/--/-- PAR ----------------------             */
/*                                                             */
/*  OBJET : ------------------------------------------------   */
/*          ------------------------------------------------   */
/*          ------------------------------------------------   */
/*                                                             */
/*-------------------------------------------------------------*/

#ifndef BDREX_INC
#define BDREX_INC

#ifdef __MSDOS__
extern int bdrex(n1 int );
#else
extern int bdrex();
#endif

#endif /* BDREX_INC */

/*-------------------------------------------------------------*/
/*   Fin du fichier bdrex.h                                    */
/*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*/

* FICHIER bdrex.c                                             */
/*                                                             */
/* CONTENU : Exemple de module                                 */
/*                                                             */
/*                                                             */
/* PROTOTYPES : ./                                             */
/*                                                             */
/* VERSION : 1.0                                               */
/*                                                             */
/* DATE    : 16/03/90                                          */
/*                                                             */
/*-------------------------------------------------------------*/


/*---------------- MAINTENANCE ------------------------------*/
/*                                                             */
/*  MODIFIE LE --/--/-- PAR ----------------------             */
/*                                                             */
/*  OBJET : ------------------------------------------------   */
/*          ------------------------------------------------   */
/*          ------------------------------------------------   */
/*                                                             */
/*-------------------------------------------------------------*/

#include "bdr.h"

int bdrex(n1)
int n1;
{
      printf("J'affiche le paramete %d \n",n1);
      return(0);
}


/*-------------------------------------------------------------*/
/*   Fin du fichier bdrex.c                                    */
/*-------------------------------------------------------------*/
```

CHAPITRE VII

C-ISAM

SOMMAIRE

C-ISAM

1. INTRODUCTION

Ce chapitre présente le produit C-ISAM (*Indexed Sequential Acess Method*), développé et commercialisé par la société INFORMIX$_{(1)}$. C-ISAM est un gestionnaire de fichier indexé pour les systèmes d'exploitation UNIX, MS/DOS et OS/2.

Dans un premier temps, les grandes fonctionnalités du produit seront abordées.

Puis, dans un deuxième temps, nous présenterons le produit, en nous attardant sur quelques points techniques, concernant le développeur.

1 : Voir Chap 4. Pour aller plus loin...

2. PRESENTATION DES FONCTIONNALITES

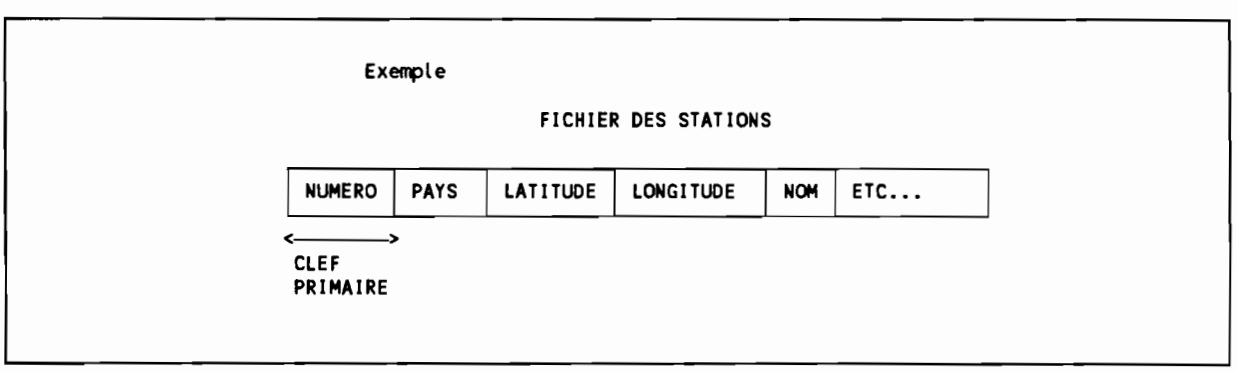
2.1. Gestion des fichiers indexés.

C-ISAM est principalement constitué par une bibliothèque de fonctions, que l'on peut appeler depuis des programmes écrits en C.

Ces fonctions permettent de manipuler des fichiers indexés, dont les enregistrements sont de taille fixe.

2.1.1 Clef primaire

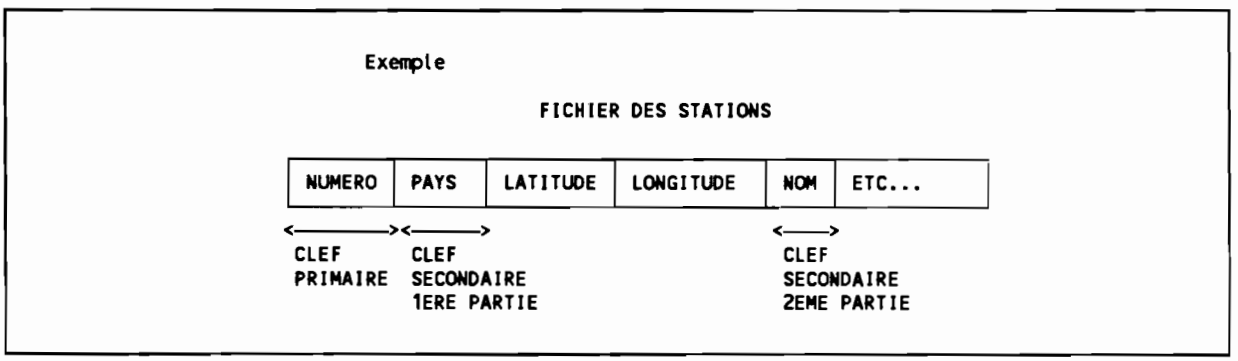
A ces enregistrements, on associe une **clef primaire** qui permettra d'accéder rapidement au fichier.



On pourra donc lire l'enregistrement du fichier des stations (*voir exemple*), dont le numéro de la station est 0003. De même, on pourra lire le fichier des stations, dans l'ordre des numéros de station.

2.1.2 Clefs secondaires

Des **clefs secondaires** peuvent être définies ou détruites.



Dans l'exemple, les enregistrements seront triés par pays et par nom de stations.

2.2. Gestion des accès concurrents

Dans le cas de système d'exploitation multi-tâches ou multi-utilisateurs, des problèmes peuvent apparaître si plusieurs programmes accèdent aux même fichiers.

C-ISAM offre deux solutions; le blocage des fichiers peut être géré par :

- le développeur,
- les transactions.

2.2.1 Gestion par le développeur du blocage des enregistrements.

Le programmeur a la faculté de :

- Verrouiller entièrement un fichier pour son usage exclusif,
- Verrouiller entièrement un enregistrement pour son usage exclusif.

2.2.2 Blocage automatique par transaction

Différents accès aux fichiers C-ISAM peuvent être regroupés dans une TRANSACTION. Tout se passe comme si deux transactions différentes étaient exécutées l'une après l'autre.

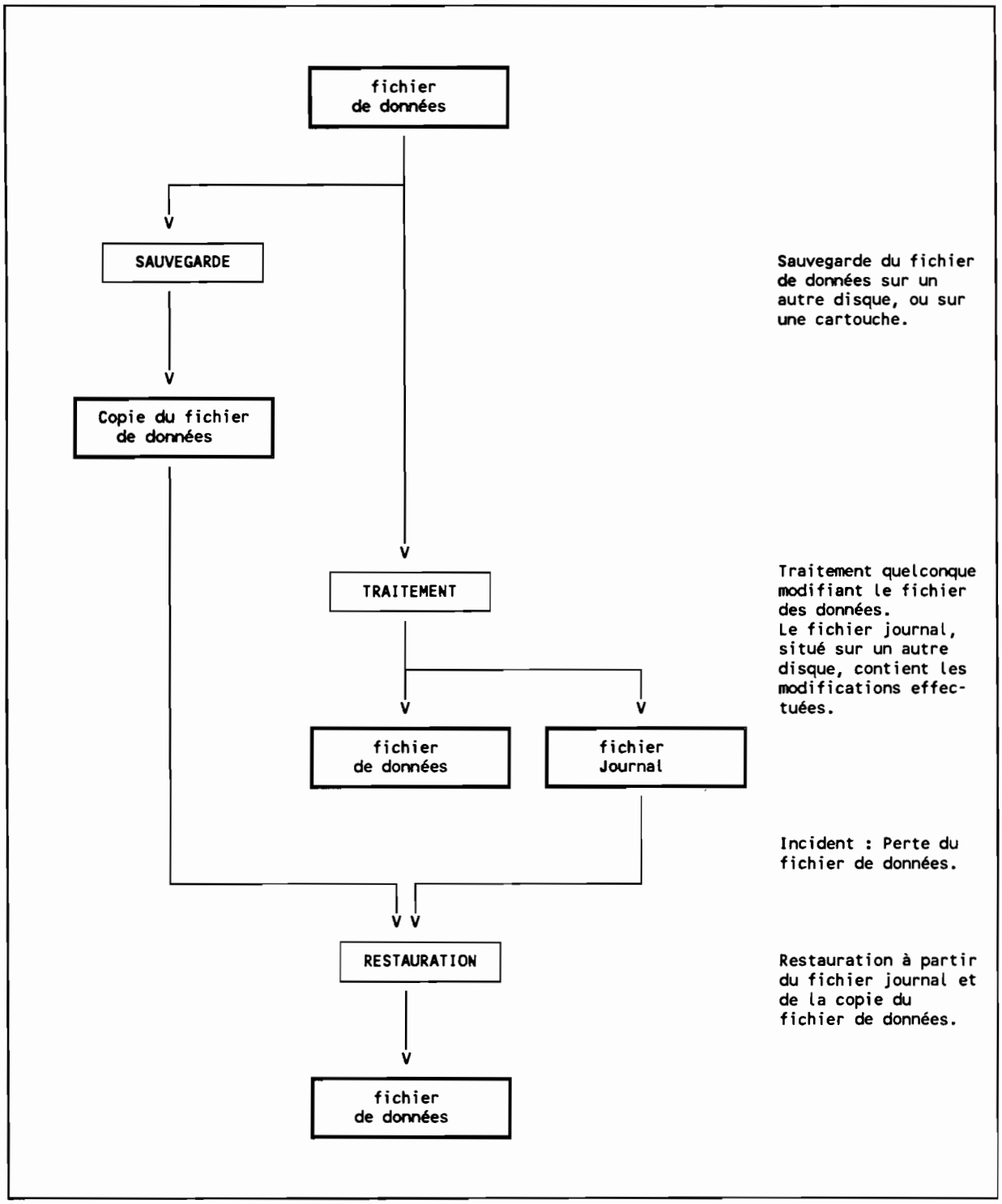
2.3. Intégrité des fichiers

2.3.1 Les Transactions

A l'intérieur d'une transaction, on peut annuler toutes les modifications effectuées depuis le début de cette transaction (*rollback*).

2.3.2 Le fichier Journal

Le système peut créer un fichier journal (*log*), qui contiendra toutes les modifications apportées aux fichiers. En cas de problème, à partir d'une copie, et du fichier journal, on peut restaurer l'état des données (*rollforward*).


fichier
de données
SAUVEGARDE
Copie du fichier
de données
TRAITEMENT
fichier
de données
fichier
Journal
RESTAURATION
fichier
de données
Sauvegarde du fichier de données sur un autre disque, ou sur une cartouche.
Traitement quelconque modifiant le fichier des données.
Le fichier journal, situé sur un autre disque, contient les modifications effectuées.
Incident : Perte du fichier de données.
Restauration à partir du fichier journal et de la copie du fichier de données.

2.4. Portabilité

C-ISAM existe pour les systèmes d'exploitation MS-DOS, OS/2 et UNIX.

2.4.1 Les programmes

L'accès à la bibliothèque C-ISAM est identique, quel que soit le système d'exploitation ou l'ordinateur choisi. Pour obtenir des programmes portables, il faut donc veiller à ce que la partie extérieure à C-ISAM le soit.

2.4.2 Les données

Les fichiers de données sont portables d'une machine à l'autre si l'on n'utilise pas les données de type **float** ou **double**[2]. Cependant, C-ISAM propose un type **decimal** indépendant de la machine.

2 : Voir Chap 3.1. Les types de données.

3. PRESENTATION TECHNIQUE

3.1. Type des données utilisées

C-ISAM propose plusieurs types de données pour les champs des enregistrements.

- **FLOAT** et **DOUBLE** : ces deux types sont identiques aux types float et double du C. Leur implémentation dépend donc de l'ordinateur utilisé.

Les autres types de données sont portables :

- **INT** : Entier signé sur deux octets,
- **LONG** : Entier signé sur quatre octets,
- **CHAR** : Caractère sur un octet,
- **DECIMAL** : Nombre en virgule flottante, ayant au plus 32 chiffres significatifs.

3.2. Structure des enregistrements.

Un enregistrement C-ISAM est défini comme un tableau de caractères. Différentes fonctions de conversion sont donc fournies pour transférer les données entre le tableau de caractères et des variables plus facilement accessibles en C.

Par exemple, supposons que l'on veuille manipuler un enregistrement composé de deux champs, la clef qui est un entier et un autre champ qui est une chaîne de caractères.

On définit l'enregistrement C-ISAM :

```
/* Enregistrement C_ISAM                              */
/* On reserve de la place pour :                      */
/*       un entier (INTSIZE),                         */
/*       huit caracteres (CHARSIZE*8)                 */

typedef char enreg_cisam[INTSIZE+CHARSIZE*8];


/* Variable facilement accessible dans laquelle on    */
*/ transfere enreg_cisam                              */
typedef struct
{
        int entier;
        char caracteres[8]
} var_lisible;
```

3.3. Fichiers manipulés

Lorsque l'on manipule, depuis C-ISAM, un fichier appelé "exemple", trois fichiers sont créés :

- **exemple.dat** : fichier contenant les données,
- **exemple.idx** : fichier contenant la description des clefs et les index correspondants,
- **exemple.lok** : fichier contenant les sémaphores gérant les accès concurrents.

4. POUR ALLER PLUS LOIN...

On pourra consulter :

R.S.TARE, Data processing in UNIX, Mc Graw-Hill, 1989.

C-ISAM, Indexed Sequential Access Method, Informix Software, Inc., 1988.

5. EXEMPLE

```
/*****************************************************************/
/*                                                               */
/*   Exemple C-ISAM                                              */
/*                                                               */
/*                                                               */
/*****************************************************************/


#if defined(__MSDOS__)
#include "process.h"
#include "stdlib.h"
#endif

#include "stdio.h"
#include "isam.h"
#include "decimal.h"


/* Variable recevant l'enregistrement C-ISAM (aprés conversion) */

typedef struct
{
    int     reint;
    double  redbl;
}               rec;


/* Enregistrement C-ISAM */

typedef char    isrec[INTSIZE + DECLEN (16, 1)];
/* INT, DECIMAL à 17 chiffres significatifs */

struct keydesc  key1;           /* Descripteur de clef      */
rec rec1;                       /* Enregistrement lisible   */
isrec isrec1;                   /* Enregistrement illisible */
int     fd,                     /* Descripteur du fichier   */
        cc,                     /* Code retour              */
        indchar;

dec_t tempdec;                  /* Var. decimal temporaire  */

main ()
{

 /*--------------------------------*/
 /* Definition de la clef          */
 /*--------------------------------*/

    key1.k_flags = ISNODUPS;            /* Clef unique                  */
    key1.k_nparts = 1;                  /* Une seule partie             */
    key1.k_part[0].kp_start = 0;        /* Commecant au début           */
    key1.k_part[0].kp_leng = INTSIZE;   /* La longueur est celle de INT */
    key1.k_part[0].kp_type = INTTYPE;   /* Le type est INT              */

 /*--------------------------------*/
 /* Construction du fichier        */
 /*--------------------------------*/

    cc = fd = isbuild ("ex", sizeof (isrec1), &key1, ISINOUT + ISEXCLLOCK);
    if (cc != 0)
    {
  printf ("isbuild error %d\n", iserrno);
  exit (1);
    }

    isclose (fd);
```

```
/* ouverture du fichier                          */


   cc = fd = isopen ("ex", ISAUTOLOCK + ISINOUT);
   if (cc != 0)
   {
printf ("isopen error %d\n", iserrno);
exit (1);
   }

/*--------------------------------*/
/* Ecriture du fichier            */
/*--------------------------------*/


   rec1.reint = 1;
   rec1.redbl = 12345678901234567.00;


/* Conversion rec1 -> isrec1                     */
   {
indchar = 0;

   /* Stockage de rec1.reint au début de isrec1 : */
stint (rec1.reint, &(isrec1[indchar]));
indchar += INTSIZE;
   /* Conversion de rec1.redbl en decimal */
deccvdbl (rec1.redbl, &tempdec);

   /* Stockage du decimal dans isrec1     */
stdecimal (&tempdec, isrec1 + indchar, DECLEN (16, 1));
indchar += DECLEN (16, 1);

   }

/* Ecriture rec1                                 */

   iswrite (fd, isrec1);
   if (cc != 0)
   {
printf ("iswrite error %d\n", iserrno);
exit (1);
   }


/*--------------------*/
/* Lecture du fichier  */
/*--------------------*/

/* Lecture rec1                                  */

   rec1.reint = 1;

   {
indchar = 0;

stint (rec1.reint, &(isrec1[indchar]));
indchar += INTSIZE;

deccvdbl (rec1.redbl, &tempdec);
stdecimal (&tempdec, isrec1 + indchar, DECLEN (16, 1));
indchar += DECLEN (16, 1);
   }
```

```
/* Lecture de l'enregistement dont la clef */
/* est égale à la valeur de rec1.reint (1) */

   cc = isread (fd, isrec1, ISEQUAL);

   if (cc != 0)
   {
 printf ("isread error %d\n", iserrno);
 exit (1);
   }

/* Conversion isrec1 -> rec1                    */
   {
 indchar = 0;

 rec1.reint = ldint (&isrec1[indchar]);
 indchar += INTSIZE;

 lddecimal (&isrec1[indchar], DECLEN (16, 1), &tempdec);
 indchar += DECLEN (16, 1);
 dectodbl (&tempdec, &(rec1.redbl));
   }

/* Affichage rec1                               */

   printf ("reint.............  %d\n", rec1.reint);
   printf ("redbl.............  %.17e\n", rec1.redbl);


   isclose (fd);

   iserase("ex");                    /* On efface le fichier */

   return (0);
}
```

CHAPITRE VIII

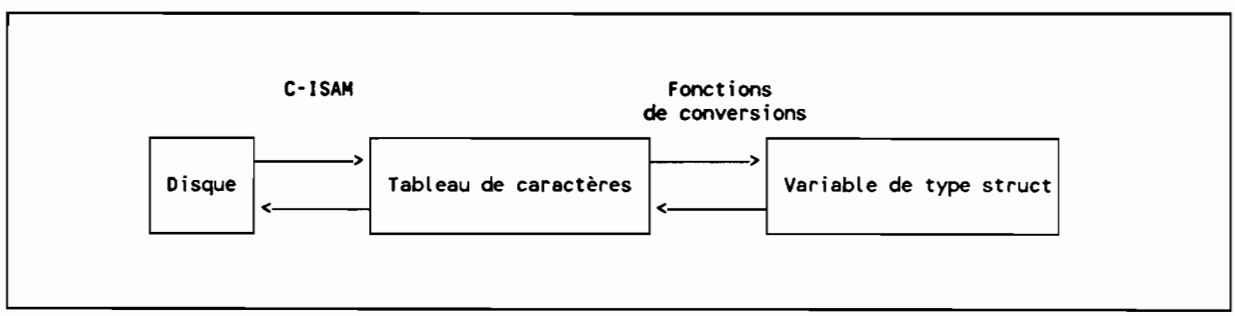
DESCRIPTION DES FICHIERS

TABLE DES MATIERES

1. INTRODUCTION

Dans ce chapitre, nous allons décrire les enregistrements des fichiers créés[1] par l'application C-ISAM, tels qu'ils sont vus par le programmeur.

C-ISAM ne manipulant que des enregistrements définis comme étant des tableaux de caractères, nous développerons des fonctions de conversion entre ces tableaux et des structures de données plus évoluées (*struct du langage C*).



Nous décrirons donc dans ce chapitre, pour chaque fichier :

- Le nom du fichier,
- L'enregistrement tel qu'il est vu par C-ISAM,
- La clef primaire,
- La (ou les) clefs secondaire(s), si nécessaire,
- L'enregistrement converti dans une structure.

1 Les fichiers SANAGA ont déjà été décrits dans BADORA - Rapport Technique - Chapitre VII - Spécifications détaillées.

2. TYPES DE DONNEES

TYPES DE DONNES

APPLICATION	C	C-ISAM	SIGNIFICATION
DATE	long int	LONGTYPE	Date comprenant l'année, le mois le jour, l'heure, les minutes et les secondes, exprimées en nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00.
CODE	unsigned char	CHARTYPE	Code alphanumérique.
ANGLE	short int	INTTYPE	Angle en dixièmes de degrés de 0 à 360 degrés (*la valeur 360 étant exclue*).
NUMERO	unsigned char	CHARTYPE	Numéro de 0 à 255.
REPERTOIRE	char[64]	CHARTYPE(64)	Nom d'un répertoire sur 63 caractères (*le 64ème étant réservé pour la marque de fin de chaîne /0*).
FICHIER	char[16]	CHARTYPE(64)	Nom d'un fichier sur 15 caractères (*le 16ème étant réservé pour la marque de fin de chaîne /0*).
LIBRE	A définir au cas par cas.		
SCIENTIFIQUE	double	DECIMAL(16)[2]	Nombre en virgule flottante avec 16 chiffres significatifs. Ce type de données occupe 10 octets[3].
TAILLE	long int	LONGTYPE	Taille en octets ou en nombre d'enregistrements.

[2] Type décimal avec 16 chiffres significatifs.

[3] On pourra vérifier dans la bibliothèque C-ISAM que 10 = DECLEN(16,1).

3. DESCRIPTIONS DES FICHIERS

3.1 Formalisme

NOM DU FICHIER..... *Nom du fichier C-ISAM sans extension*

DESCRIPTION

Description sommaire du fichier.

CHAMPS

NOM DU CHAMP *Signification*

CLEF PRIMAIRE

Description de la clef primaire en fonction des champs.

CLEF SECONDAIRE

Description de la clef secondaire en fonction des champs.

ENREGISTREMENT C-ISAM

NOM *Nom du type défini dans l'application*

DEFINITION

Définition en C

Nom du champ..... Début[4]..... *Fin..... Type..... Longueur*

CLEF PRIMAIRE

Description de la clef primaire (position, longueur, attribut).

CLEF SECONDAIRE

Description des clefs secondaires (position, longueur, attribut).

---

4 Le début et la fin de chaque zone désigne le numéro de l'octet dans l'enregistrement C-ISAM, le premier octet ayant le numéro 0.

VARIABLE STRUCTUREE

NOM *Nom du type défini dans l'application*

DEFINITION

Définition en C

3.2 Fichier support

NOM DU FICHIER..... su

DESCRIPTION

Chacun de ces enregistrements correspond à un support (*disque dur ou cartouche*).

CHAMPS

- suncod	NUMERO	Numéro du support; le numéro 0 est réservé pour le disque dur.
- sutvoloc	TAILLE	Nombre d'octets effectivement occupé sur le support.
- suvolto	TAILLE	Nombre d'octets utilisables (lorsque le support est vierge).
- sudarc	DATE	Date d'archivage ou de la dernière mise à jour du support.
- sufiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- suncod, unique.

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM issurec

DEFINITION

```
typedef char issurec[ CHARSIZE + LONGSIZE*4 ];
```

Nom du champ	Début	Fin	Type	Longueur
suncod	0	0	CHARTYPE	1
sutvoloc	1	4	LONGTYPE	4
sutvolto	5	8	LONGTYPE	4
sudarc	9	12	LONGTYPE	4
sufiller	13	14	LONGTYPE	4
		TOTAL		17

CLEF PRIMAIRE

suncod	0	0	CHARTYPE	1	ISNODUPS

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM surec

DEFINITION

```
typedef struct
{
    unsigned char   suncod;     /* Numéro              */
    long    sutvoloc;           /* Taille occupe       */
    long    sutvolto;           /* Taille totale       */
    long    sudarc;                     /* Date archivage     */
    long    sufiller;           /* Inutilise           */
}
                surec;
```

3.3 Fichier événement

NOM DU FICHIER..... ev

DESCRIPTION

Chacun de ces enregistrements correspond à un événement, c'est-à-dire à une succession d'images dans le temps.

CHAMPS

- evddatde	DATE	Date de la première image de l'événement.
- evddatfi	DATE	Date de la dernière image de l'événement.
- evcsta	CODE	Code statut de l'événement indiquant le statut de l'événement par rapport au traitement d'archivage : 'D' (disque) : Toutes les images de l'événement sont accessibles directement sur le disque dur. 'A' (archivé) : Toutes les images de l'événement ont été archivées. 'M' (mixte) : La plupart des images a été archivé.
- evncodsu	CODE	Numéro de code du support où est archivé l'événement. Si l'événement n'est pas archivé, le numéro de support est le numéro du disque dur (0).
- evcqua	CODE	Indique si l'événement est de bonne qualité : FALSE : Mauvaise qualité, TRUE : Bonne qualité, L'événement est de mauvaise qualité si l'une au moins de ses images est de mauvaise qualité.
- evtenrra	TAILLE	Nombre d'enregistrements radiales concernant toutes les images de l'événement.
- evtnbrim	TAILLE	Nombre d'images.
- evfiller	LONGTYPE	Réservée pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- evddatde, unique.

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM isevrec

DEFINITION

```
typedef char    isevrec[2 * LONGSIZE + 3 * CHARSIZE + 3 * LONGSIZE];
```

Nom du champ	Début	Fin	Type	Longueur
evddatde	0	3	LONGTYPE	4
evddatfi	4	7	LONGTYPE	4
evcsta	8	8	CHARSIZE	1
evncodsu	9	9	CHARSIZE	1
evcqua	10	10	CHARSIZE	1
evtenrra	11	14	LONGTYPE	4
evtnbrim	15	18	LONGTYPE	4
evfiller	19	22	LONGTYPE	4
		TOTAL		23

CLEF PRIMAIRE

evddatde	0	3	LONGTYPE	4	ISNODUPS

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM evrec

DEFINITION

```
typedef struct
{
    long    evddatde;           /* Date de debut      */
    long    evddatfi;           /* Date de Fin        */
    char    evcsta;             /* Code statut        */
    unsigned char evncodsu;     /* Numéro de support  */
    char    evcqua;             /* Code qualité       */
    long    evtenrra;           /* Nbr d'enr. dans le */
                                 /* fichier segment    */
    long    evtnbrim;           /* Nombre d'images    */
    long    evfiller;           /* Inutilise          */
}
                evrec;
```

3.4 Fichier image

NOM DU FICHIER..... im

DESCRIPTION

Chacun de ces enregistrements correspond à la description d'une image.

CHAMPS

- imddatde	DATE	Date de la première radiale de l'image.
- imaazide	ANGLE	Azimut de la première radiale de l'image.
- imasecco	ANGLE	Secteur couvert par l'image.
- imasit	ANGLE	Site de l'image
- imnres	NUMERO	Numéro de la résolution utilisée (*voir tableau des résolution dans le fichier radar*).
- imnenrra	TAILLE	Nombre d'enregistrements dans le fichier radiale (segment) correspondant à l'image.
- imcarc	CODE	Indique si l'image a été archivée : FALSE : L'image n'a pas été archivée, TRUE : L'image a été archivée.
- imcbdr	CODE	Indique si l'image est accessible sur disque dur : FALSE : L'image n'a pas été archivée. TRUE : L'image a été archivée.
- imctra	CODE	Indique si une copie de l'image doit rester sur le disque dur après le traitement d'archivage : FALSE : Une copie doit rester sur le disque dur, TRUE : Il ne doit rester sur le disque dur que la description de l'image, après le traitement d'archivage. Ce champ n'est significatif que si l'image n'est pas archivée.
- imcqua	CODE	Indique si l'image est de bonne ou mauvaise qualité : FALSE : L'image est de mauvaise qualité, TRUE : L'image est de bonne qualité.
- imfiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- imddatde, unique.

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM isimrec

DEFINITION

```
typedef char    isimrec[LONGSIZE + 3 * INTSIZE + CHARSIZE + LONGSIZE + 4 * CHARSIZE +
                        LONGSIZE];
```

Nom du champ	Début	Fin	Type	Longueur
imddtade	0	3	LONGTYPE	4
imaazide	4	5	INTTYPE	2
imascco	6	7	INTTYPE	2
imasit	8	9	INTTYPE	2
imnres	10	10	CHARTYPE	1
imtenrra	11	14	LONGTYPE	4
imcarc	15	15	CHARTYPE	1
imcbdr	16	16	CHARTYPE	1
imctra	17	17	CHARTYPE	1
imcqua	18	18	CHARTYPE	1
imfiller	19	22	LONGTYPE	4
		TOTAL		23

CLEF PRIMAIRE

imddatde	0	3	LONGTYPE	4	ISNODUPS

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM imrec

DEFINITION

```
typedef struct
{
    long    imddatde;          /* Date de debut      */
    int     imaazide;          /* Azimut de depart   */
    int     imasecco;          /* Secteur couvert    */
    int     imasit;            /* Site de l'image    */
    char    imcres;            /* Code résolution    */
    long    imnenrra;          /* Nbr d'enr. dans le */
                               /* fichier segment    */
    char    imcarc;            /* Code archivage     */
    char    imcbdr;            /* Code disque dur    */
    char    imctra;            /* Code copie travail */
    char    imcqua;            /* Code qualite       */
    long    imfiller;          /* Inutilise          */
}
                imrec;
```

3.5 Fichier segment

NOM DU FICHIER..... sg

DESCRIPTION

Ce fichier contient les radiales des images. Chaque enregistrement correspond à un segment[5].

CHAMPS

- sgddat	DATE	Date de l'image.
- sgnord	NUMERO	Numéro d'ordre du segment dans l'image.
- sglsto	LIBRE	Zone de stockage du segment défini par quatre tableaux char[128].
- sgfiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- sgddat, sgnord, unique.

CLEF SECONDAIRE

- /.

5 Pour plus de précision, on pourra se référer à BADORA - Rapport technique - Chapitre VII - Spécifications détaillées.

ENREGISTREMENT C-ISAM

NOM issgrec

DEFINITION

```
typedef char    issgrec[LONGSIZE + CHARSIZE + CHARSIZE * 512 + LONGSIZE];
```

Nom du champ	Début	Fin	Type	Longueur	
sgddat	0	3	LONGTYPE	4	
sgnord	4	4	CHARSIZE	1	
sglsto	5	516	CHARTYPE	512	
sgfiller	517	520	LONGTYPE	4	
		TOTAL		521	

CLEF PRIMAIRE

sgddat	0	3	LONGTYPE	4	
sgnord	4	4	CHARTYPE	1	ISNODUPS

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM sgrec

DEFINITION

```
typedef struct
{
    long    sgddat;             /* Date de debut      */
    unsigned char   sgnord;     /* Numero d'ordre     */
    char    sglsto[512];        /* Zone de stockage   */
    long    sgfiller;           /* Inutilise          */
}
                sgrec;

typedef char    issgrec[LONGSIZE + CHARSIZE + CHARSIZE * 512 + LONGSIZE];
```

3.6 Fichier radar

NOM DU FICHIER..... ra

DESCRIPTION

Ce fichier contient les caractéristiques du radar et du protocole de mesure. Ces données sont supposées constantes pour un endroit donné.

CHAMPS

- ralnom	LIBRE	Nom fonctionnel du radar défini par char[16] (chaîne de caractères de 15 caractères plus le caractère de fin de chaîne).
- raslat	SCIENTIFIQUE	Latitude en degrés (*positive pour une coordonnée Nord*).
- raslon	SCIENTIFIQUE	Longitude en degrés (*positive pour une coordonnée Est*).
- raaouv	ANGLE	Ouverture en degré.
- rasfreem	SCIENTIFIQUE	Fréquence d'émission en MHz.
- raspuiem	SCIENTIFIQUE	Puissance d'émission en kW.
- rasdurip	SCIENTIFIQUE	Durée d'impulsion en seconde.
- rasecaip	SCIENTIFIQUE	Ecart d'impulsion en seconde.
- rasvolre	SCIENTIFIQUE	Volume de résolution en mètre.
- ractyp	LIBRE	Type du radar défini par char[8];
- rasres[10]	SCIENTIFIQUE	Tableau des résolutions possibles en mètres. Par exemple, si la code résolution à la valeur 1 dans un fichier SANAGA, la résolution est donnée par la valeur de rasres[1].

CLEF PRIMAIRE

- /.

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM israrec

DEFINITION

```
typedef char    israrec[CHARSIZE *16 + DECLEN (16, 1) * 2 + INTSIZE
                        + DECLEN (16, 1) * 5 + CHARSIZE * 8 + DECLEN (16, 1) * 16
                        + LONGSIZE];
```

Nom du champ	Début	Fin	Type	Longueur	
ralnom	0	15	CHARTYPE	16	
raslat	16	25	DECIMAL	10	
raslon	26	35	DECIMAL	10	
raaouv	36	37	INTTYPE	2	
rasfreem	38	47	DECIMAL	10	
raspuiem	48	57	DECIMAL	10	
rasdurip	58	67	DECIMAL	10	
rasecaip	68	77	DECIMAL	10	
rasvolre	78	87	DECIMAL	10	
ractyp	88	95	CHARTYPE	8	
rasres[10]	96	195	DECIMAL	100	= 10 x 10
rafiller	196	199	LONGTYPE	4	
		TOTAL		200	

CLEF PRIMAIRE

- /.

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM rarec

DEFINITION

```
typedef struct
{
    char    ralnom[16];         /* Nom du radar          */
    double  raslat;             /* Latitude              */
    double  raslon;             /* Longitude             */
    int     raaouv;             /* Ouverture             */
    double  rasfreem;           /* Frequence emission */
    double  raspuiem;           /* Puissance emise       */
    double  rasdurip;           /* Duree impulsion       */
    double  rasecaip;           /* Ecart impulsion       */
    double  rasvolre;           /* Volume resolution     */
    char    ractyp[8];          /* Type du radar         */
    double  rasres[10];         /* Tableau resolution */
    long    rafiller;           /* Inutilise             */
}
                rarec;
```

3.7 Fichier discontinuité

NOM DU FICHIER..... di

DESCRIPTION

Ce fichier rassemble les paramètres de regroupement et de synthèse de radiales en images et d'images en événements.

CHAMPS

- diasit	ANGLE	Ecart maximum de site entre deux radiales de la même image
- diaazi	ANGLE	Ecart maximum d'azimut entre deux radiales successives de la même image.
- diasec	ANGLE	Secteur couvert minimum pour qu'une image soit archivée.
- didtem	DATE	Ecart maximum de temps entre deux images d'un même événement.
- diaech	ANGLE	Echantillonnage de l'azimut.
- difiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- /.

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM isdirec

DEFINITION

```
typedef char    isdirec[INTSIZE * 3 + LONGSIZE + INTSIZE + LONGSIZE];
```

Nom du champs	Début	Fin	Type	Longueur
diasit	0	1	INTTYPE	2
diaazi	2	3	INTTYPE	2
diasec	4	5	INTTYPE	2
didtem	6	9	LONGTYPE	4
diaech	10	11	INTTYPE	2
difiller	12	15	LONGTYPE	4
		TOTAL		16

CLEF PRIMAIRE

- /.

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM direc

DEFINITION

```
typedef struct
{
    int     diasit;             /* Ecart max site     */
    int     diaazi;             /* Ecart max azimut   */
    int     diasec;             /* Secteur couvert min */
    long    didtem;             /* Ecart max temps    */
    int     diaech;             /* Echantillonaage az. */
    long    difiller;           /* Inutilise          */
}

                direc;
```

3.8 Fichier paramètres

NOM DU FICHIER..... pa

DESCRIPTION

Ce fichier rassemble les paramètres matériels et logiciels de la configuration.

CHAMPS

- parsan	REPERTOIRE	Nom du répertoire où sont situés les fichiers SANAGA.
- pararc	REPERTOIRE	Nom du répertoire du périphérique d'archivage.
- parprt	REPERTOIRE	Commande Unix d'impression d'un fichier.
- partp1	REPERTOIRE	Nom du répertoire de stockage des fichiers temporaires n°1.
- partp2	REPERTOIRE	Nom du répertoire de stockage des fichiers temporaires n°2.
- patbad	TAILLE	Volume disponible sur le disque dur pour la banque de données BADORA.
- patsup	TAILLE	Volume disponible sur le périphérique d'archivage.
- pafiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- /.

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM isparec

DEFINITION

```
typedef char    isparec[CHARSIZE * 64 * 5 + LONGSIZE * 3];
```

Nom du champ	Début	Fin	Type	Longueur
parsan	0	63	CHARTYPE	64
pararc	64	127	CHARTYPE	64
parprt	128	191	CHARTYPE	64
partp1	192	255	CHARTYPE	64
partp2	256	319	CHARTYPE	64
patbad	320	323	LONGTYPE	4
patsup	324	327	LONGTYPE	4
pafiller	328	331	LONGTYPE	4
		TOTAL		332

CLEF PRIMAIRE

- /.

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM parec

DEFINITION

```
typedef struct
{
    char    parsan[64];         /* Repertoire SANAGA  */
    char    pararc[64];         /* Repertoire streamer*/
    char    parprt[64];         /* Repertoire impr.   */
    char    partp1[64];         /* Repertoire temp1   */
    char    partp2[64];         /* Repertoire temp2   */
    long    patbad;             /* Taille disque      */
    long    patsup;             /* Taille archivage   */
    long    pafiller;           /* Inutilise          */
}
                parec;

typedef char    isparec[CHARSIZE * 64 * 5 + LONGSIZE * 3];
```

3.9 Fichier journal des transferts

NOM DU FICHIER..... jt

DESCRIPTION

Ce fichier contient le compte-rendu du transfert des fichiers SANAGA vers la banque de données BADORA.

CHAMPS

- jtddat	DATE	Date du début du transfert
- jtfnom	FICHIER	Nom du fichier transféré
- jttsan	TAILLE	Taille du fichier
- jtcrlt	CODE	Compte-rendu du transfert : FALSE : La transaction a été interrompue, il faut recommencer l'opération. TRUE : Le transfert s'est déroulé normalement.
- jtrsan	REPERTOIRE	Repertoire du fichier.
- jtfiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- jtddat, unique

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM isjtrec

DEFINITION

```
typedef char    isjarec[LONGSIZE + CHARSIZE * 3 + LONGSIZE];
```

Nom du champ	Début	Fin	Type	Longueur
jtddat	0	3	LONGTYPE	4
jtfnom	4	19	CHARSIZE	16
jttsan	20	23	LONGTYPE	4
jtcrlt	24	24	CHARSIZE	1
jtrsan	25	88	CHATSIZE	64
jtfiller	89	92	LONGTYPE	4
		TOTAL		93

CLEF PRIMAIRE

- jtddat	04	3	LONGTYPE	4	ISNODUPS

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM jarec

DEFINITION

```
typedef struct
{
    long    jtddat;             /* Date                 */
    unsigned char   jansup;     /* Nom du fichier       */
    char    jtctra;             /* Code traitement      */
    char    jtcrlt;             /* Code resultat        */
    long    jtfiller;           /* Inutilise            */
}
                jtrec;
```

3.10 Fichier journal des archivages

NOM DU FICHIER..... ja

DESCRIPTION

Ce fichier contient le compte-rendu des opérations d'archivage.

CHAMPS

- jaddat	DATE	Date du début de l'opération.
- jansup	NUMERO	Numéro de la cartouche où sont archivés les événements.
- jactra	CODE	Code du traitement : 'A' : Archivage, 'M' : Mise à jour d'une cartouche, 'H' : Restauration après archivage, 'I' : Restauration après mise à jour.
- jacrlt	CODE	Compte-rendu du traitement : FALSE : Le traitement a été interrompu TRUE : Le traitement s'est déroulé normalement.
jafiller	LONGTYPE	Réservé pour un usage ultérieur. Actuellement initialisé à 0L.

CLEF PRIMAIRE

- jtddat, unique

CLEF SECONDAIRE

- /.

ENREGISTREMENT C-ISAM

NOM isjtrec

DEFINITION

```
typedef char    isjarec[LONGSIZE + CHARSIZE * 3 + LONGSIZE];
```

Nom du champ	Début	Fin	Type	Longueur
jaddat	0	3	LONGTYPE	4
jansup	4	4	CHARSIZE	1
jactra	5	5	CHARSIZE	1
jacrlt	6	6	CHARSIZE	1
jafiller	7	10	LONGSIZE	4
		TOTAL		11

CLEF PRIMAIRE

- jaddat	04	3	LONGTYPE	4	ISNODUPS

CLEF SECONDAIRE

- /.

VARIABLE STRUCTUREE

NOM jarec

DEFINITION

```
typedef struct
{
    long    jaddat;             /* Date                 */
    unsigned char   jansup;     /* Nom du fichier       */
    char    jactra;             /* Code traitement      */
    char    jacrlt;             /* Code resultat        */
    long    jafiller;           /* Inutilise            */
}
                jarec;
```

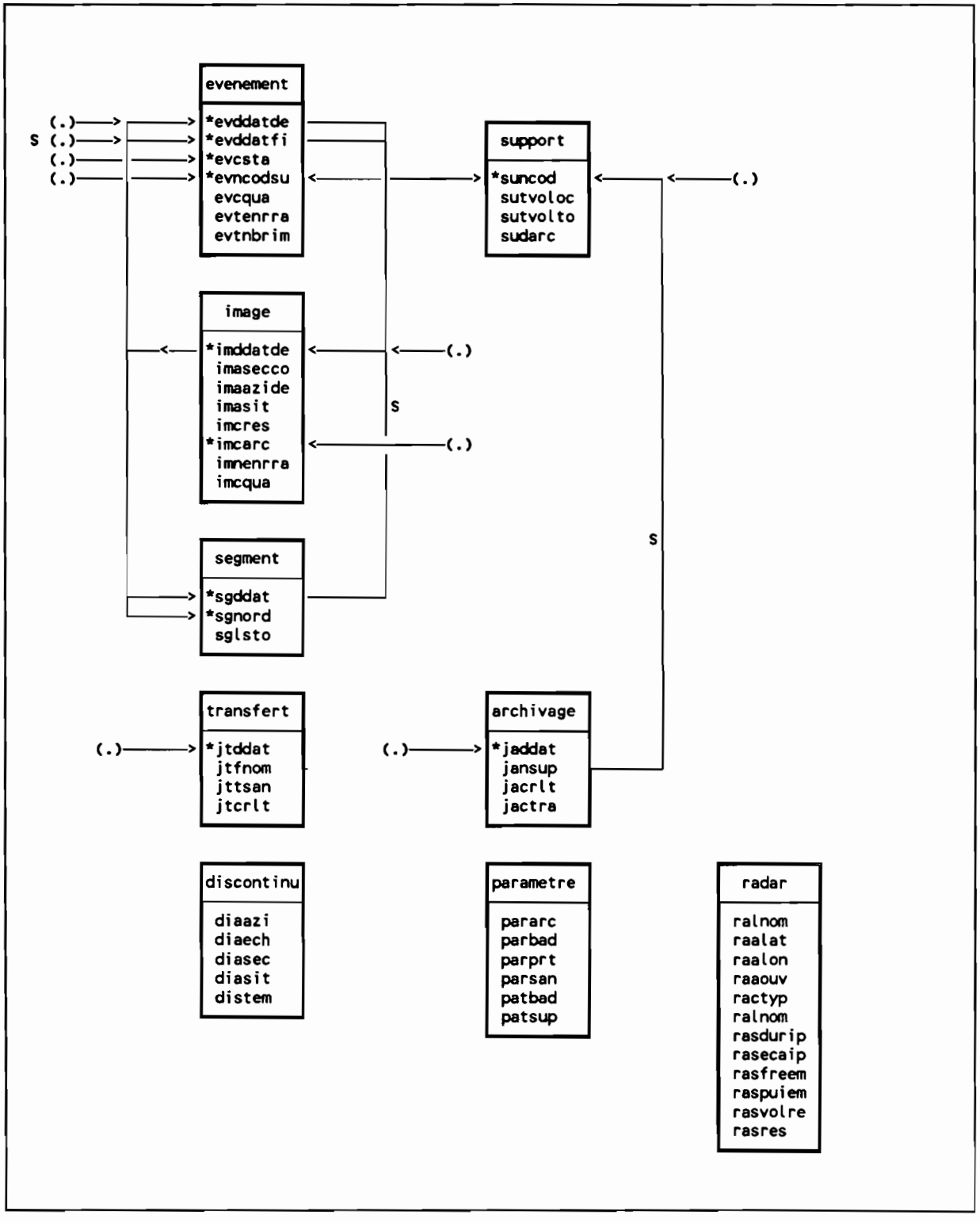
4. FICHIERS DISQUES/CARTOUCHES

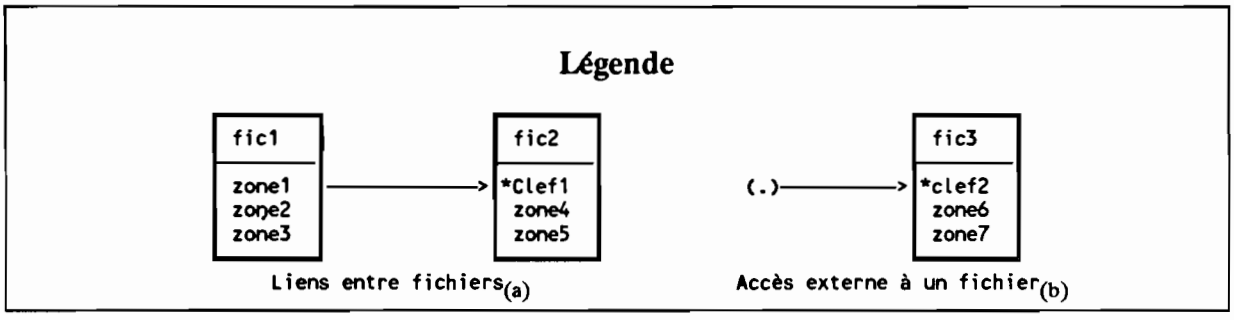
Le tableau suivant indique la signification des fichiers, selon le support utilisé *(cartouches ou disque dur)*.

Nom du fichier	Disque	Cartouche
Support	Totalité des supports	Le support correspondant à cette cartouche.
Evénement	Totalité des événements	Evénements stockés sur la cartouche.
Image	Totalité des images	Les images stockées sur la cartouche.
Segment (Radiale)	Les radiales des images non archivées et les radiales des images archivées mais dont il faut garder une copie de travail sur le disque dur[6].	Les radiales des images archivées sur la cartouche.
Journal Archivage	Totalité	/
Journal Transfert	Totalité	/
Discontinuité	Totalité	Totalité
Paramètres	Totalité	/
Radar	Totalité	Totalité

6 On garde donc les radiales des images telles que (imcarc = FALSE) ou ((imcarc = TRUE) et (imctra = TRUE))

5. SCHEMA RECAPITULATIF





(a) : A partir de zone1, on accède au fichier fic2, selon la clef clef1.

(b) : On accède au fichier fic3, selon clef2.

Remarque : Les traitements et la définition des fichiers ont été croisés. Certaines liaisons ne sont pas utilisées par BADORA. Cependant, SYNTHIA aura besoin de ces liaisons, indiquées sur le schéma par le signe (S).

CHAPITRE IX

STRUCTURE DE L'APPLICATION

SOMMAIRE

STRUCTURE DE L'APPLICATION

[1] le code source du module `INTRODUCTION` est dans le fichier `intro.c`, le prototype dans `intro.h`.

INTRODUCTION

Pour l'utilisateur, l'application BADORA se compose de quatre fichiers de commandes :

bdrinst : création de la banque de données,
bdrmain : utilisation interactive de BADORA,
bdrbatch : lecture en traitement par lot, des fichiers SANAGA d'un répertoire,
bdrsan : extraction de données de la banque au format SANAGA,
bdrraz : Dévérrouillage des fichiers après incident.

Ces fichiers de commandes utilisent des fichiers exécutables propres à l'application. Il s'agit de :

bdrinst.exe : création de la banque de données,
bdrmain.exe : utilisation interactive de BADORA,
bdrbatch.exe : lecture en traitement par lot, d'un fichier SANAGA.

Ces fichiers exécutables sont composés de modules objet.

Dans ce chapitre, nous présenterons l'architecture de l'application, et nous détaillerons chacun de ces modules.

0. LISTE GENERALE DES MODULES

Les noms des modules sont ceux définis dans l'étude précédente. Il faut rajouter le préfixe bdr pour le nom des fichiers.

- **Menu principal** **A0**
  - Restauration M314

- **Menu paramétrage** **A1**

- **Menu mise à jour** **A2**
  - Informations générales M21
    - Transfert fichier SANAGA M231
      - Manipulation image intermédiaire L2311
      - Filtrer radiale L2312
  - Consultation du journal des transferts M24

- **Menu archivage** **A3**

- **Menu édition** **A4**
  - Edition du journal des transferts M24
  - Edition du catalogue des événements M43

- **Interface utilisateur** **T1**

- **Imprimante** **T3**

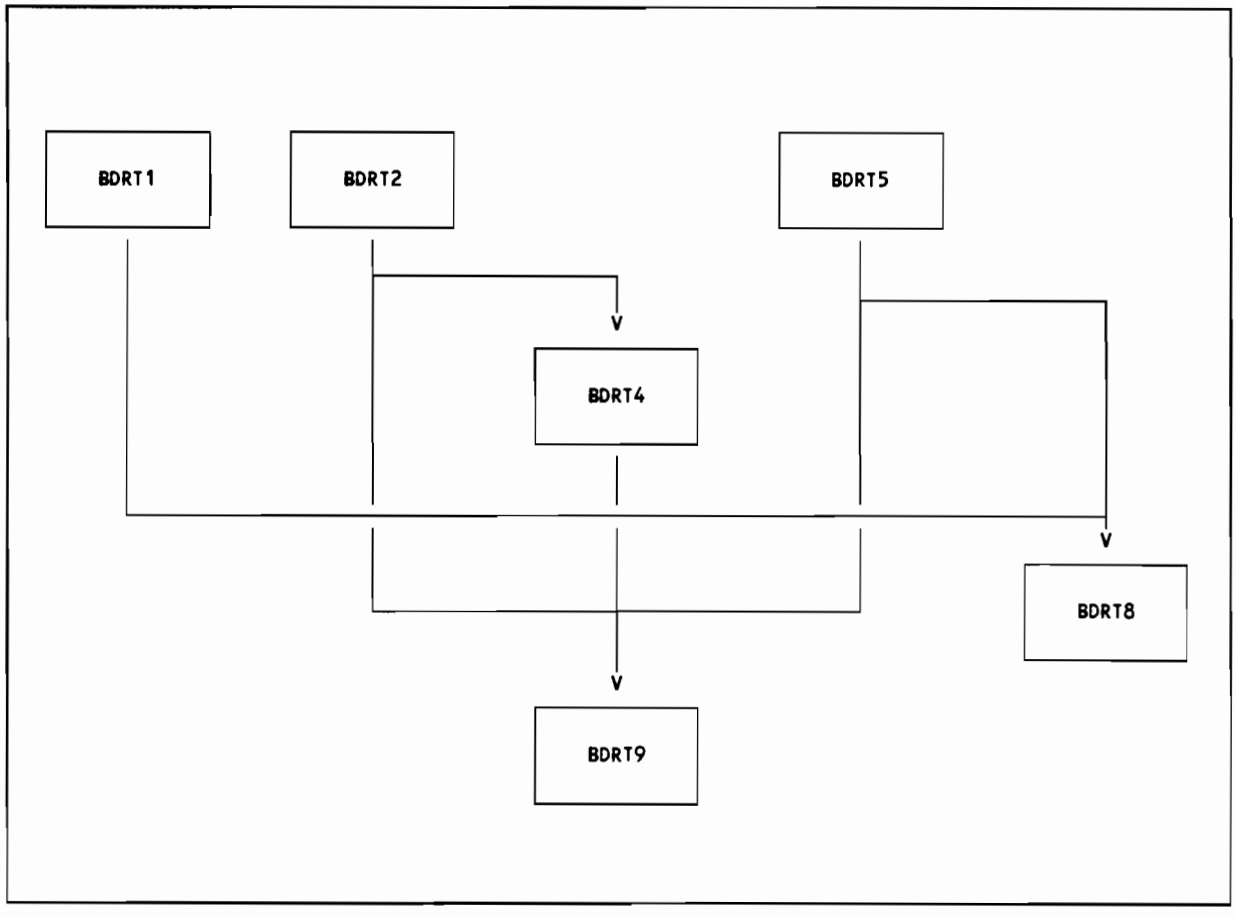
- **C-ISAM** **T4**

- **SANAGA** **T5**

- **DATES** **T8**

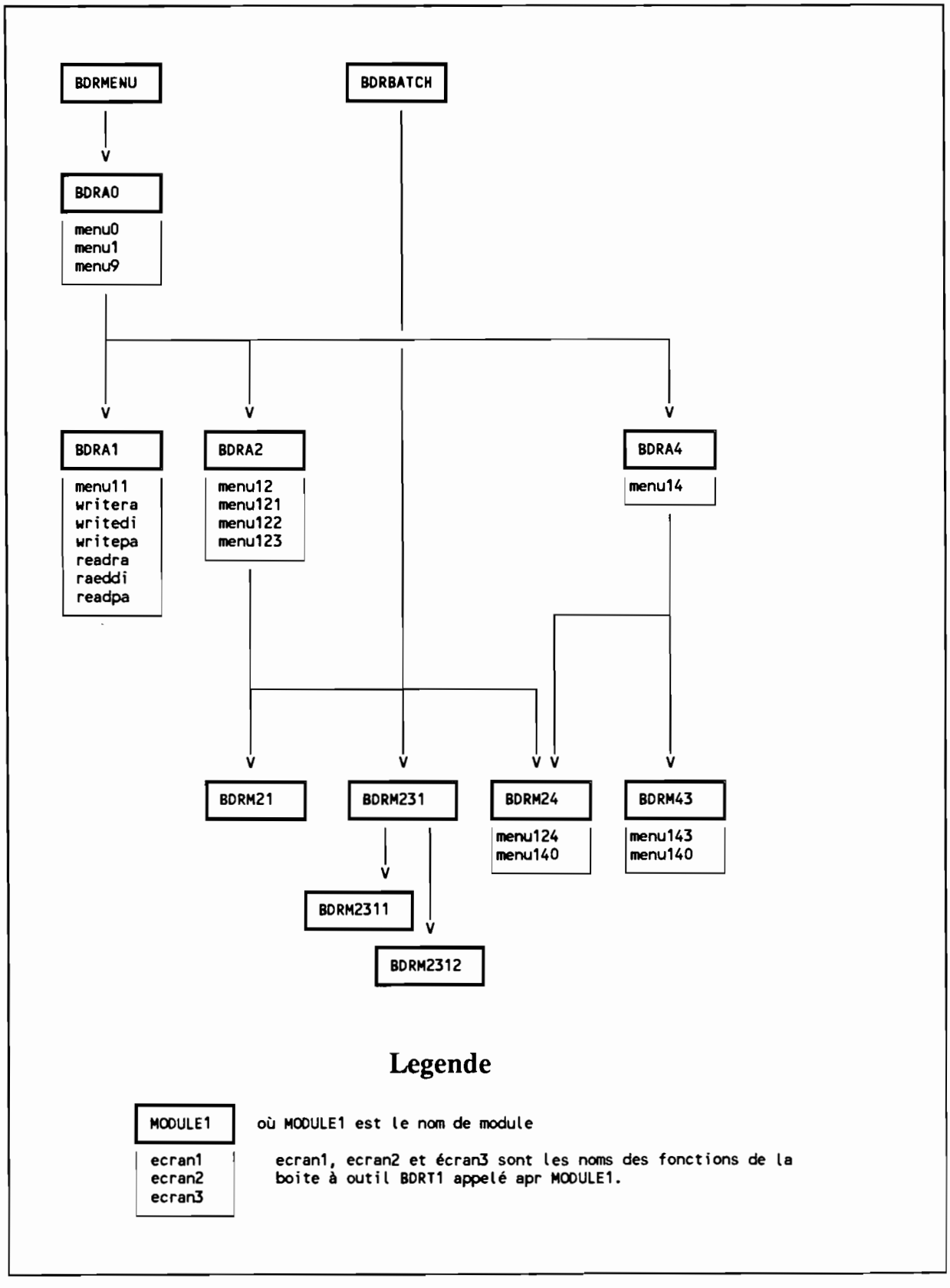
- **GESTION DES ERREURS** **T9**

2. DEPENDANCE ENTRE LES MODULES(1)



3. DEPENDANCE ENTRE LES MODULES(2)

Dans ce chapitre, nous indiquons les relations de dépendances, pour les modules ne faisant pas aprtie de la boîte à outil.



INTRO[2]. INTRODUCTION

OBJET

- Fonction générale remplie par le module : conventions utilisées pour la documentation des modules.

TYPES

- Liste des types définis dans le module.

VARIABLES GLOBALES

- Liste des variables globales définies dans le module.

USAGE

```
/*-------------------------------------------------*/
/*  Nom des fichiers inclus et entetes des fonctions  */
/*-------------------------------------------------*/
#include "intro.h"

void intro1(parametre)
long parametre;
```

DESCRIPTION

- **intro1** Description de intro1.

REMARQUES

On utilise les **caractères gras** pour les identificateurs et pour les codes sources.

CODES D'ERREUR

Codes d'erreurs renvoyés par les fonctions.

---

2 le code source du module INTRODUCTION est dans le fichier intro.c, le prototype dans intro.h.

STRUCTURE INTERNE

Explication sur le fonctionnement interne du module. Il n'est pas nécessaire de lire cette partie pour utiliser le module.

BDRT1. INTERFACE UTILISATEUR

OBJET

- Dialogue avec l'utilisateur.

TYPES

-/.

VARIABLES GLOBALES

-/.

USAGE

```
#include "bdrt1.h"

extern int  readpa ( parec *ptpa1);
extern void  writepa( CONST parec *ptpa1);
extern int  readdi ( direc *ptdi1);
extern void  writedi( CONST direc *ptdi1);
extern int  readra ( rarec *ptra1);
extern void  writera( CONST rarec *ptra1);
extern int menu0(void);
extern int menu1(void);
extern int menu9(void);
extern int menu99(CONST bderr *pterr0);
extern int menu11(void);
extern int menu12(void);
extern int menu121 (long taille1,float pcent1, int arch1);
extern int menu122(char rsan1[64],char lficsan1[20][16] );
extern int menu123(char nomfic1[16]);
extern int menu124(jtrec jtenr[10]);
extern int menu14(void);
extern int menu140(struct tm *pttmmin, long * ptsecmin,
 struct tm *pttmmax, long *ptsecmax );
extern int menu143(evrec evenr[10]);
extern int menu2(void);
extern int menu21(void);
extern void pause(void);
extern char confirm(void);
```

DESCRIPTION

- **menu0** Affichage du logo BADORA.
- **menu1** Menu principal de l'application BADORA.
- **menu11** Menu paramétrage.
- **menu12** Menu mise à jour.
- **menu121** Informations générales.
- **menu122** Choix des fichiers SANAGA.
- **menu123** Suivi mise à jour.
- **menu124** Consultation du journal des transferts.
- **menu14** Menu édition des catalogues et journaux.
- **menu140** Option impression.
- **menu143** Consultation du catalogue des événements.
- **menu9** Menu sortie.
- **menu2** Menu de confirmation de création de la banque.
- **menu21** Menu de fin de création de la banque.
- **readpa** Saisie d'un enregistrement paramètre.
- **writepa** Affichage d'un enregistrement paramètre.
- **readdi** Saisie d'un enregistrement discontinuité.
- **writedi** Affichage d'un enregistrement discontinuité.
- **readra** Saisie d'un enregistrement radar.
- **writera** Affichage d'un enregistrement radar.
- **cls** Effacement de l'écran.
- **pause** Attente de l'intervention de l'utilisateur.
- **confirm** Demande de confirmation.

REMARQUES

Fonctions `readra`, `readdi`, `readpa`, `menu122` , `menu140`.

Toutes ces fonctions de saisie affichent les anciennes valeurs, saisissent les nouvelles valeurs, puis demandent une confirmation de l'ensemble de la saisie. Par défaut, les nouvelles valeurs seront identiques aux anciennes.

La valeur nulle est renvoyée lorsque la saisie s'est déroulée normalement.

Fonctions `menu0`, `menu9`, menu21,`menu121`, menu 123.

Ces fonctions renvoient toujours la valeur nulle. En effet, elles affichent des informations, puis attendent que l'utilisateur soit prêt.

Fonction menu2.

La valeur nulle est renvoyée si la création de la banque de données n'est pas confirmée.

Fonctions `menu1`, `menu11`, `menu12`, `menu14`.

Ces fonctions gèrent des menus et renvoient le numéro de l'option choisie. Par convention, l'option 0 est réservée à l'option "retour au menu précédent".

Fonctions `menu124`, `menu143`

Ces fonctions affichent une partie d'un catalogue ou d'un journal. Les valeurs retournées sont : 0 pour terminer, 1 pour un défilement vers le bas, 2 pour un défilement vers le haut et 3 pour une demande d'impression.

Fonction menu140

Avant d'imprimer, cette fonction permet de saisir une date de début et une date de fin, qui seront converties au format `struct tm` et au format `long`[3]. Les valeurs par défaut seront uniquement converties au format `long` et prendront les valeurs `0L` et `LONG_MAX`.

---

3 Voir BDRT8.

Fonction Menu99

Son usage est réservé au module de gestion des erreurs.

STRUCTURE INTERNE

Ce module est le seul à dialoguer avec l'utilisateur.

L'interface est indépendant de la configuration matérielle puisqu'elle utilise uniquement les fonctions **`scanf`**, **`gets`** et **`printf`**.

Les fonctions de saisie, du type **`readxx`**, font appel aux fonctions d'affichages du type **`writexx`**.

Il n'y a pas d'accès aux fichiers SANAGA et C-ISAM dans ce module.

BDRT3. INTERFACE IMPRIMANTE

OBJET

- Impression

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrt3.h"

int prtopen(titre1,titr2)
char *titre1, *titre2;

int prtclose(void);

int prtline(ligne)
char *ligne;
```

DESCRIPTION

- **prtopen** Impression d'un en-tête, comprenant les deux lignes de titres (tronquées à 50 caractères).
- **prtclose** Fin de l'impression.
- **prtline** Impression d'une ligne.

REMARQUE

L'impression ne sera effective qu'après l'appel à **prtclose**. Il est déconseillé d'imprimer des lignes de plus de 80 caractères.

Le fichier paramètre doit être accessible.

CODES D'ERREUR

Si tout se passe correctement, la valeur nulle est retournée.

STRUCTURE INTERNE

Dès l'appel à **prtopen**, un fichier temporaire est créé grâce à **tmpnam**, défini dans **stdio.h**.

Lors de l'appel à **prtclose**, ce fichier est fermé. On lit alors la commande d'impression dans le fichier paramètre (champ **parprt**), puis on lance la commande grâce à la fonction **system**.

BDRT4. INTERFACE C-ISAM

OBJET

- Définition des fichiers (longueur des enregistrements, clefs, nom),
- Conversion entre les enregistrements C-ISAM et les types structurés associés,
- Gestion des transactions.

TYPES

- **isxxrec** Enregistrement C-ISAM du fichier **xx**$_{(4)}$.
- **xxrec** Type structuré associé au fichier **xx**.

VARIABLES GLOBALES

- **xxnomfic** : Nom du fichier **xx**. Plus exactement :
  - **xxnomfic[DIRBDR]** : Nom du fichier **xx** dans le répertoire BADORA,
  - **xxnomfic[DIRTP1]** : Nom du fichier **xx** dans le répertoire temporaire 1,
  - **xxnomfic[DIRTP2]** : Nom du fichier **xx** dans le répertoire temporaire 2,
- **xxkey[5]** : Tableau des clefs du fichier **xx**. **xxkey[0]** désigne la clef primaire.

USAGE

```
#include "bdrt4.h"

void initfile();

void ldxxrec(isxx1, ptxx1)
isxxrec isxx1;
xxrec *ptxx1;

void stxxrec(ptxx1, isxx1);
xxrec *ptxx1;
isxxrec isxx1

int transdeb();
int transfin();
```

4 (su)pport, (ev)enement, (im)age, (s)e(g)ment, (ra)dar, (di)scontinuite, (pa)rametres, (j)ournal (t)ransfert, (j)ournal (a)rchivage

DESCRIPTION

- **initfile**	Initialisation du module.
- **ldxxrec**	Conversion du type **isxxrec** vers le type **xxrec**.
- **stxxrec**	Conversion du type **xxrec** vers le type **isxxrec**.
- **transdeb()**	Début de transaction.
- **transfin()**	Fin de transaction.

REMARQUES

La fonction **initfile** nécessite un accès au fichier paramètres situé dans le répertoire BADORA.

Les variables globales ne sont disponibles que lorsque la fonction **initfile** a été appelée.

Les clefs sont représentées par la type **struct keydesc** défini dans **isam.h**

CODES D'ERREUR

- **initfile** retourne 0 lorsque l'initialisation s'est déroulée correctement. Si, la variable d'environnement BDRDIR n'existe pas, on provoque un arrêt du programme. Si le fichier paramètre n'est pas disponible, une valeur non-nulle est retournée. Néanmoins, les noms des fichiers dans le répertoire BADORA ainsi que les valeurs des clefs sont initialisés.
- **transdeb** et **transfin** renvoient la valeur nulle lorsque tout s'est passé correctement. Dans le cas contraire, le programme est interrompu.

STRUCTURE INTERNE

CISAM fournit deux fonctions **stchar** et **ldchar** pour transférer des chaînes de caractères entre un enregistrement CISAM et une variable. Ces fonctions tiennent compte du caractère de fin de chaîne. Pour transférer un tableau de caractères, nous avons créé les fonctions **stchar2** et **ldchar2** qui ignorent le caractère de fin de chaîne.

initkey initialise les éléments non-utilisés des tableaux **xxkey[]** comme étant les index "numéro de l'enregistrement".

initnom lit la variable d'environnement **BDRDIR**, situe et lit le fichier paramètre pour initialiser les noms **xxnomfic[DIRTP1]** et **xxnomfic[DIRTP2]**. C'est depuis cette fonction que l'on peut provoquer un arrêt du programme via **_bdrerr**[5].

initfile appelle **initkey** puis **initnom** pour chacun des fichiers.

---

5 Voir BDRT9.

BDRT5. INTERFACE SANAGA

OBJET

- Lecture des enregistrements d'un fichier SANAGA,

TYPES

- **sanrec** Type structuré associé à un enregistrement du fichier SANAGA

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrt5.h"

int sanopen(sannom1)
char *sannom1;

int sanclose(void);

int sanread(ptsanrec1)
sanrec *ptsanrec1;

int saneof(void);
```

DESCRIPTION

- **sanopen** Ouverture du fichier SANAGA dont le nom complet (avec le répertoire) est dans *sannom1*.
- **sanclose** Fermeture du fichier SANAGA.
- **sanread** Lecture d'une radiale.
- **saneof** Teste si l'on vient de lire la dernière radiale du fichier SANAGA.
- **sanlon** Donne la longueur du fichier en octets.

REMARQUE

Lors de la lecture d'une radiale (fonction **sanread**), le seul test effectué est la cohérence du champ donnant la longueur de l'enregistrement avec la position du drapeau fin de radiale (02_H).

CODES D'ERREUR

Si tout se passe correctement, la valeur nulle est retournée.

En cas de problème de lecture du fichier, on renvoie une valeur non-nulle. Cependant, un arrêt du programme est provoqué si l'on ne peut fermer ou ouvrir le fichier, sauf dans le cas où le fichier est inconnu (**sanread** renvoie une valeur non-nulle).

STRUCTURE INTERNE

sanom est une variable initilalisée par **sanopen** qui contient le nom du fichier SANAGA ouvert. **sanpos** initialisé à zéro par **sanopen**, contient le numéro[6] de l'octet du début de l'enregistrement SANAGA en cours de lecture, ou la valeur -1 si aucun fichier SANAGA n'est ouvert. Donc, si une erreur survient pendant ou après la lecture d'une radiale, **sanpos** désigne le début de cette radiale dans le fichier.

On veillera à préciser le type **unsigned char** pour les données lues afin d'éviter les interprétations (-1 et 255, par exemple).

6 Nous avons gardé la convention Unix qui prend la valeur nulle pour désigner le premier octet du fichier.

BDRT8. GESTION DES DATES

OBJET

- Conversion entre les types **struct tm** et le type **long** représentant le nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00.

TYPES

- ./

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrt8.h"

long tmlong(tm2)
struct tm *tm2;

void longtm(nbsec,pttm)
long nbsec;
struct tm *pttm);

int tmok(pttm)
struct tm *pttm;

int strtm(str1,pttm,ptsec)
char *str1;
struct tm *pttm;
long *ptsec;

int longstr(str1,nbsec)
char *str1;
long nbsec;
```

FONCTIONS

- **tmlong** Conversion d'une variable de type **struct tm** vers le type **long**.
- **longtm** Conversion d'une variable de type **long** vers une variable de type **struct tm**.
- **tmok** Teste si une date représentée dans une variable de type **struct tm** est valide.

- **strtm** Conversion d'une chaîne de caractères vers les types **struct tm** et **long**.
- **longstr** Conversion d'une variable de type **long** vers une chaîne de caractères.

REMARQUE

Nous désignons par date, l'année, le mois, le jour, l'heure, les minutes et les secondes.

Les années bissextiles sont prises en comptes.

On ignore les changements d'heures (heure d'hiver et heure d'été).

La structure **tm** est définie dans **time.h**.

Les conventions utilisées pour représenter les dates dans une structure **tm** sont identiques à celles données pour les fonctions déclarées dans **time.h**.

Le type **long** est utilisé pour représenter le nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00

Les dates représentées dans une chaîne de caractères ont le format "JJ/MM/AAAA HH:MM:SS". Cependant la fonction **strtm** accepte le format "JJ/MM/AAAA", auquel cas elle considère "JJ/MM/AAAA 00:00:00".

CODES D'ERREUR

- 0, si tout ce passe correctement,
- Une valeur non-nulle pour **strtm** et **tm_ok** si le paramètre en entrée ne représente pas une date valide.

STRUCTURE INTERNE

La fonction limMois retourne le nombre de jour dans un mois pour une année donnée, en tenant compte des années bissextiles. Par exemple limMois(2,1970) retourne 31, qui est le nombre de jours dans le mois de mars de l'année 1970.

Le comportement de ces fonctions a été comparé avec les fonctions définies dans **time.h**, sur un système Unix.

BDRT9. GESTION DES ERREURS

OBJET

- Impression des messages d'erreur dans le fichier standard (**stderr**) et arrêt approprié de la fonction en cours selon la gravité de l'erreur (pas d'interruption, sortie de la fonction en cours avec le code 1, arrêt du programme).

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrt9.h"

_bdrerr(no,gr,cl,tx)
unsigned char no;
int gr;
char *cl;
char *tx;

_bdrraz()
```

MACROS

- **_bdrerr** Signalisation d'une erreur, dont le numéro est **no**, la gravité **gr**, la classe **cl**, et le commentaire **tx**.
  Un message est alors écrit dans le fichier standard des messages d'erreur (**stderr**).
- **_bdrraz** Effacement de l'erreur après son traitement.

REMARQUE

_bdrerr et **_bdrraz** sont des macros.

Le paramètre **gr** peut prendre les valeurs :

- 0, erreur minime, l'exécution continue normalement,
- 9, erreur grave provoquant une annulation de la transaction en cours (**isrollback**) et un arrêt du programme (**abort**).
- autres valeurs, erreur importante, provoquant une annulation de la transaction en cours (**isrollback**) et une sortie de la fonction avec une valeur non-nulle.

Après une erreur grave, on peut retrouver les fonctions actives au moment de l'arrêt du programme grâce à la commande UNIX **tb** (trace back).

Lors de la signalisation d'une erreur C-ISAM (classe "IS", si le texte associé est la chaîne vide **"/0"**, la macro **_bdrerr** rajoute le numéro de l'erreur C-ISAM (variable **iserrno**) et le commentaire donné par C-ISAM (variable **is_errlist[iserrno-100]**).

Lors de la signalisation d'une erreur, si un fichier SANAGA est ouvert, un message supplémentaire est fourni, indiquant le nom du fichier (**sanom**) et le début de la dernière radiale lue(**sanpos**)[7].

Si l'erreur est grave, un écran est affiché avant l'arrêt du programme (**menu99**).

CODES D'ERREUR

- /.

STRUCTURE INTERNE

La macro **_bdrerr** appelle la fonction **bdrerr** qui, lors du premier appel, redirige le flux **stderr** vers le fichier **bdrerr.log**.

Nous avons utilisé une macro afin de lire le nom du fichier source et le numéro de la ligne grâce aux macros **__FILE__** et **__LINE__**.

Il est nécessaire de vider la mémoire tampon associée au flux (**fflush**), afin de ne pas perdre d'information en cas d'arrêt de l'application.

---

7 Voir BDRT5.

BDRA1. MENU PARAMETRAGE

OBJET

Paramétrage de l'application.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdra1.h"

int bdra1(void);
```

REMARQUE

CODES D'ERREUR

- /.

STRUCTURE INTERNE

La partie lecture et écriture dans les fichiers est incluse dans ce source (fonctions **lirera**, **liredi**, **lirepa**, **ecrirera**, **ecriredi**, **ecrirepa**).

La partie affichage et saisie est située dans le module **BDRT1** (fonctions **readra**, **readdi**, **readpa**, **writera**, **writera**, **writepa**).

Un début et une fin de transaction délimitent les accés aux fichiers (fonctions **transdeb** et **transfin**).

BDRA2. MENU MISE A JOUR

OBJET

Mise à jour de la banque à partir de fichiers SANAGA.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdra2.h"

int bdra2(void);
```

REMARQUE

Cette fonction affiche le menu "mise àjour" et agit en fonction de l'option choisie.

CODES D'ERREUR

- /.

STRUCTURE INTERNE

Dans un premier temps, nous lisons le répertoire SANAGA, dans le fichier paramètres (champ **parsan**).

L'option 1 (Information générales) fait successivement appel aux fonction **bdrm21** et **menu121**, pour le calcul et l'affichage de ces options.

L'option 2 fait appel à la fonction **menu122** pour la saisie du répertoire SANAGA et pour la saisie du nom des fichiers à lire. On peut remarquer que le fichier paramètre est mis à jour avec la nouvelle valeur du répertoire SANAGA.

L'option 3 fait appel à bdrm231 pour le transfert d'un fichier SANAGA. On transmet alors le nom complet du fichier avec répertoire

L'option 4 fait appel à **bdrm24** pour la consultation du journal.

Les noms des fichiers à traiter sont stockés dans une variable statique, et leurs valeurs sont conservées pendant toute la session de travail. Seul le nom des fichiers correctement traités est effacé.

BDRM21. INFORMATIONS GENERALES

OBJET

Calcul de la place occupée sur le disque dur.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrm21.h"

int bdrm21(pttaille1,ptpcentocc1,ptarch1)
long *pttaille1;
float *ptpcentocc1;
int *ptarch1;
```

REMARQUE

- **pttaille1** indique le nombre d'octets occupés par les fichiers C-ISAM,
- **ptpcentocc1** indique, en pourcentage, le rapport volume occupé/volume alloué, le volume alloué étant indiqué dans le fichier paramètre.
- **ptarch1** est égal à zéro si l'archivage n'est pas nécessaire.

CODES D'ERREUR

STRUCTURE INTERNE

La fonction **calc** détermine la place occupée par un fichiers C-ISAM en tenant compte des trois fichiers physiques (**.dat**, **.idx** et **.lok**). Cette fonction utilise **stat** défini dans **stat.h**.

L'archivage est nécessaire si la place disponible est inférieure à 1 Mo.

BDRM231. TRANSFERT D'UN FICHIER SANAGA

OBJET

Transfert d'un fichier SANAGA dans la banque de données.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrm231.h"

int bdrm231(nomfic);
char *nomfic;
```

REMARQUE

Cette fonction utilisant des transactions, la banque de données reste cohérente après un arrêt imprévu du programme. On peut aussi appeler plusieurs fois cette fonction pour le même fichier, la partie déjà traitée, étant ignorée.

CODES D'ERREUR

- La valeur nulle est renvoyée si tout c'est passé normalement.

STRUCTURE INTERNE

Le fichier **bdrm231h.h**, commun aux modules **bdrm231**, **bdrm2311**, et **bdrm2312**, décrit le type asssocié à une image en cours de constitution.

Les variables globales **di1** et **ra1**, contenant les valeurs des fichiers discontinuité et paramètres, à l'usage des modules **bdrm2311** et **bdrm2312**, sont initialisées dans ce module par la fonction **lectdira**.

jt_init écrit dans le journal des transferts un enregistrement signalant une erreur lors de la lecture du fichier. Si le transfert s'est déroulé correctement, ce même enregistrement sera modifié pour signaler un transfert normal. C'est l'objet de la fonction **jt_maj**.

rad_filtrer, comme son nom l'indique filtre les radiales. Une radiale n'est pas correcte si :

- La radiale n'est pas une radiale PPI,
- La date ne représente pas une date valide,
- Le site est négatif ou supérieur à 180 degrés,
- L'azimut est négatif,
- Il existe un code 254 ou 255 dans le champ données,
- La position du code 02, n'est pas cohérente avec les codes 00 et 01.
- Il y a plus de 1000 portes.

rad_appartient teste si une radiale peut appartenir à une image. Une radiale n'appartient pas à une image si

- la différence entre le site de la radiale, et le site minimum ou le site maximum de l'image, est strictement supérieure à la valeur indiquée dans le fichier discontinuité,

- La différence d'azimut entre la dernière radiale de l'image et la radiale à teser est strictement supérieure à la valeur indiqué dans le fichier discontinuité.

BDRM2311. RAJOUT D'UNE RADIALE DANS UNE IMAGE

OBJET

Rajouter une radiale dans une image.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdra2311.h"

int rad_rajouter(ptsanrec1, ptimgrec1)
const sanrec *ptsanrec1;
imgrec *ptimgrec;
```

REMARQUE

Cette fonction est appelée par le module **bdrm231**.

CODES D'ERREUR

- La valeur nulle est renvoyée si tout s'est passé normalement.

STRUCTURE INTERNE

rad_rajouter appelle les fonctions **raj_prem** et **raj_autre** qui mettent à jour l'image en cours, sans stocker les portes dans les segments. Ceci est fait au sein de la fonction **rad_rajouter**.

BDRM2312. MANIPULATION IMAGE INTREMEDIAIRE

OBJET

Manipulation d'une image intermédiaire.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdra2312.h"

int img_init(ptimgrec1)
imgrec *ptimgrec;

int img_existe(ptimgrec1)
const imgrec *ptimgrec1;

int img_filtrer(ptimgrec1)
imgrec *ptimgrec1;

int img_traiter(ptimgrec1)
imgrec *ptimgrec;
```

REMARQUE

img_init initialise une image de type imgrec.

img_existe teste l'existence de l'image dans le fichier des images.

img_filtrer indique si l'image doit être conservée. Une image ne sera pas conservée si le secteur couvert est inférieur strictement à la valeur indiquée dans le fichier discontinuité.

img_traiter effectue toutes les opérations de mise à jour des fichiers en vue de l'enregistrement d'une image.

CODES D'ERREUR

- **img_filtrer** renvoie zéro si l'image doit être conservée.

- img_existe renvoie zéro si l'image n'existe pas.STRUCTURE INTERNE

Toutes les mises à jours sont dans une même transaction.

L'écriture d'un événement peut correspondre à :

- La création d'un enregistrement du fichier événement,
- La mise à jour d'un enregistrement,
- La fusion de deux enregistrements.

C'est l'objet des fonctions **evt_rajouter**, **evt_fusionner**, et **evt_créer**.

BDRM24. AFFICHAGE DU JOURNAL DES TRANSFERTS

OBJET

Gestion de l'affichage et de l'impression du journal des transferts.

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrm24.h"

int bdrm24(void);
```

REMARQUE

- /.

CODES D'ERREUR

- /.

STRUCTURE INTERNE

Cette fonction gère la lecture et la mise en forme des données. L'affichage proprement dit est assuré par la fonction **`menu124`** du module BDRT1.

La fonction **`imprim`** est spécifiquement chargée de l'impression.

BDRA4. MENU EDITION DES CATALOGUES ET JOURNAUX

OBJET

Gestion du menu "Edition des catalogues et journaux".

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdra4.h"

int bdra4(void);
```

REMARQUE

- /.

CODES D'ERREUR

- /.

STRUCTURE INTERNE

- /.

BDRM43. AFFICHAGE DU CATALOGUE DES EVENEMENTS

OBJET

Gestion de l'affichage et du catalogue des événements

TYPES

- /.

VARIABLES GLOBALES

- /.

USAGE

```
#include "bdrm43.h"

int bdrm43(void);
```

REMARQUE

- /.

CODES D'ERREUR

- /.

STRUCTURE INTERNE

Cette fonction gére la lecture et la mise en forme des données. L'affichage proprement dit est assuré par la fonction **menu143** du module BDRT1.

La fonction **imprim** est spécifiquement chargée de l'impression.

CHAPITRE X

BILAN

BILAN

Sur le plan technique, cette expérience représente un atout considérable pour mon avenir. En effet, j'ai eu à mettre en oeuvre des outils de développement dans un environnement UNIX, et à me pencher sur les problèmes de portabilité des applications écrites en langage C.

Ce stage s'insère au sein d'un projet plus large, qui vise à mettre en place une chaîne de traitement des images radar. J'ai pu participer à la conception au cours du projet de troisième année. Le développement a été assuré au cours de ce stage. J'aurais à mettre en place l'application au centre ORSTOM de Niamey.

C'est donc un projet d'importance, le projet BADORA, que j'aurais pu suivre du début jusqu'à la fin.

En parallèle, au centre ORSTOM de Niamey, et en étroite collaboration avec Thierry LEBEL, mon collègue et ami Patrick MERDY, a terminé la phase de conception du projet SYNTHIA$_{(1)}$.

Avant la fin de la saison des pluies, les chercheurs disposeront donc d'un outil opérationnel constitué des applications SANAGA, BADORA et SYNTHIA.

1 : SYNTHèse d'Informations Atmosphériques.

CHAPITRE XI

ANNEXE

SOMMAIRE

ANNEXE

BIBLIOGRAPHIE

SANAGA

[RADARMETEOROLOGIE] H. SAUVAGEOT - 1982
Télédetection de l'atmosphère
EYROLLES

[SANAGA] H. SAUVAGOT - 1989
Radar météorologique de Niamey

BADORA

[BADORA] P.MERDY T.VALERO P.VIALETTO - Février 1990
Rapport technique
ISIM/ORSTOM

SYNTHIA

[SYNTHIA] P.MERDY - Juin 1990
Rapport technique
ISIM/ORSTOM

Le langage C

[LANGAGE C] M. DE CHAMPLAIN - 1986
Standards, styles et exercices en C
DUNOD
Ouvrage d'initiation au langage C, très orienté sur la portabilité et sur le style de programmation.

[THE C LANGUAGE] DENNIS M. RITCHIE - 1978
Reference Manual
Bell Laboratories
Ouvrage de référence assez court (40 pages)

[STANDARD C] ANSI - INFORMATION PROCESSING SYSTEMS - 1988
Draft Proposed American Standard for Information Systems
- Programming language C.
Ref X3J11/88-090
Projet de standardisation du langage C.

[STANDARD C] ANSI - INFORMATION PROCESSING SYSTEMS - 1988
Rationale for Draft Proposed American Standard for Information Systems
- Programming langage C.
Ref X3J11/88-151
Résumé des délibérations du comité X3J11 chargé de la normalisation du language C.

[DOMAIN] Apollo computer Inc - 1985
Domain C Language reference
Chapter 6 : C Program development

[TURBO C] Borland International France -
Guide de reference

[TURBO C] Borland International France -
Manuel d'utilisation

UNIX

[DOMAIN] Apollo computer Inc - 1985
Domain/IX user's guide
Section 2 : Shells
Chapter 1 : An Overview Of Shell Types
Chapter 3 : Using the C Shell
Section 4 : Support Tools
Chapter 3 : Lint - a C Program Checker
chapter 4 : Make - A Program for Maintaining Programms

[DOMAIN] Apollo computer Inc - 1987
Domain/IX Command Reference for BSD4.2
cc - a C compiler
csh - a shell (command interpreter) whith C-like syntax
ctags - create a tags file
dbx - debugger
indent - indent and format C program source
lint - a C program verifier
ld - link editor
lorder - find ordering relation for an objet library

Connexion PC/Apollo

[DOMAIN] Apollo computer Inc - 1987
Using Your Domain/PCI Connection
Chapter 4 : Using Domain Ressource with DOS.
Chapter 5 : Introducing to DTERM (the Domain Terminal Program)

C-ISAM

[C-ISAM] Informix Software Inc - June 1987
Programmer's manual

[C-ISAM] Informix Software Inc - November 1988
UNIX products
Installation & Release Notes

[DATA PROCESSING IN UNIX] R.S. TARE - 1989
Using Informix-SQL, ESQL/C, C-ISAM and Turbo
MC GRAW HILL
Une bonne présentation de la gestion des fichier avec les produits de la société Informix.

GLOSSAIRE

dBZ ..... : Mesure de la réflectivité du volume correspondant à une porte pour des ondes centimétriques. Cette mesure est exprimée en décibel. La formule mathématique correspondante est : $dB(X) = 10Log_{10}X$. On associe une valeur en dBZ à chaque porte.

Durée d'impulsion ..... : temps d'émission d'une onde radar (*2 s pour le radar de Niamey*).

Dx ..... : Longeur d'une porte *(deux valeurs possibles 250 ou 500 m)*.

Image ..... : Ensemble de radiales donnant l'état météorologique à un instant donné au cours d'un tour d'antenne du radar. Chaque image permet de visualiser l'état du champ de précipitation à un instant donné. L'acquisition de 360° dure environ une minute.

Ecart d'impulsion ..... : Temps qui sépare deux impulsions.

Effet Doppler ..... : Effet utilisé par les radars de suivi.

Evénement ..... : Regroupement d'images successives correspondant à un phénomène météorologique se déroulant sur une période déterminée.

Géodynamique ..... : Science qui a pour objet l'étude des propriétés dynamiques et mécaniques d'ensemble de la terre et de la lune, en tenant compte de l'interaction mutuelle des deux astres.

Hydrosphère ..... : Regroupe toutes les parties apparentées au milieu liquide de la terre.

Hydroclimatique ..... : Relatif aux comportements hydrologiques et pluvieux

Ouverture du radar : Angle caractérisant le volume d'émission du radar.

Pas de temps : Ecart temporel entre deux radiales.

Période d'acquisition : Période pendant laquelle le radar observe un événement pluvieux.

Pluviographe : Appareil de mesure de la pluie permettant un enregistrement continue des phénomènes météorologiques.

Pluviomètre : Appareil de mesure de la pluie permettant de totaliser des mesures sur une période donnée.

Porte : Volume d'échantillonage de la mesure du radar. Ce volume s'exprime en dBZ. Chaque radiale contient de 1 à 1000 portes au maximun.

Ppi : *(Plan position indicator)*. Observation, par le biais du radar, des événements pluvieux dans un plan horizontal.

Radiale : Regroupement de portes saisies à un instant donné, dans une direction donnée.

Réflectivité : La portion d'onde renvoyée par la cible.

Rhi : (*Radius High Indicator*). Observation, par le biais du radar, des événements pluvieux dans un plan vertical.

Site : Position dans laquelle le ppi reste constant.

Système précipitant : Ensemble de conditions physiques et météorologiques induisant des précipitations.

Transfert : Passage entre deux grandeurs physiques (*pluie mesurée/débit mesuré*)

LISTE DES PERSONNES CONTACTEES

M.HOEPFFNER, Laboratoire d'Hydrologie, responsable de l'Unité de Recherche 1B (Continent,Atmosphère; Séries climatiques) du Département Terre, Océan, Atmosphère (TOA).

T.LEBEL, Laboratoire d'Hydrologie, ingénieur de recherche, responsable du projet EPSAT-Niger.

F.DELCLAUX, Ingénieur de recherche, responsable informatique du laboratoire d'Hydrologie.

V.THAUVIN, Laboratoire d'Hydrologie, allocataire de recherche, thèsitive. Etude de la répartition spatio-temporelle des précipitations à l'aide de pluviographes.

F.CAZENAVE, Laboratoire d'Hydrologie, responsable de la maintenance du radar de Niamey.

Y.ARNAUD, Laboratoire d'Hydrologie, allocataire de recherche, thèsitif. Estimation des pluies par satellite.

A.GIODA, Laboratoire d'Hydrologie, ingénieur de recherche, responsable des relations publiques.

M.MICHAUX, Service Informatique, ingénieur d'étude, responsable informatique du centre ORSTOM de Montpellier.

J.GROS, Centre National Universitaire Sud de calcul, Support compilateur C.