MASTER EN SCIENCES INFORMATIQUES

EASY-MAIL : Une Interface simple et portable pour le courrier électronique

Loodts, Bruno

Award date:
1988

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année académique 1987-1988

EASY-MAIL :

Une Interface simple et portable pour le courrier électronique

Bruno LOODTS

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

AVANT-PROPOS

J'adresse mes remerciements très sincères,

A Madame Noirhomme, professeur à l'Université Notre Dame de la Paix qui a accepté de patronner ce travail,

A Monsieur Grosbol, Chef de département à l'ESO qui m'a accordé le stage et a veillé personnellement au bon déroulement de mon séjour,

A Monsieur Ounnas, manager à l'ESO qui a dirigé mon travail et m'a donné un appui particulièrement efficace,

A Monsieur Defert, responsable des télécommunications à l'ESO qui a été un guide précieux dans ce domaine et m'a encouragé par son amitié et son dévouement,

A Messieurs Ochsenbein et Richmond, ESO, pour leur grande disponibilité visant à me fournir une aide précieuse,

A Monsieur Geurts, assistant à l'Université Notre Dame de la Paix, qui m'a aidé a la rédaction de ce travail par ses conseils,

A Monsieur Ponz, ESO, pour ses encouragements et sa gentillesse,

Et à l'ensemble du personnel de l'ESO pour son accueil chaleureux et ses sentiments amicaux qu'il n'a pas cessé de me témoigner.

RESUME

Résumé

Ce mémoire fait suite au stage et travail réalisé à l'European Southern Observatory (ESO) à Munich durant le premier semestre de l'année académique 1987-1988 en vue de l'obtention du diplôme de "LICENCIE ET MAITRE EN INFORMATIQUE". Le travail a consisté en l'implémentation d'un software en vue d'équiper l'ESO d'une nouvelle interface de courrier électronique. Ce mémoire a pour but de montrer la nécessité de développer et d'installer un nouveau programme d'interface pour cette organisation internationale en mettant en évidence les problèmes à la base de ce projet. Une étude d'opportunité permet de dégager les principales caractéristiques que devra posséder la nouvelle interface. Parmis ces opportunités, on peut citer portabilité UNIX-VMS, répertoire général et individuel et une utilisation aisée par des utilisateurs non-informaticiens d'où le nom donné à la nouvelle interface de courrier électronique: EASY-MAIL. Après une étude comparative des réseaux, la spécification de chacune des commandes est présentée dans le guide de l'utilisateur de EASY-MAIL. En ce qui concerne l'implémentation proprement dite du software, la méthodologie utilisée a été d'élaborer une architecture logicielle en modules. La construction de l'interface a été facilitée par l'utilisation d'un système de gestion de dialogue. La programmation a été réalisée en langage C. Une évaluation de EASY-MAIL sur base d'une validation du produit logiciel ainsi qu'une étude sur les développements futurs à réaliser, complètent le travail.

Abstract

This dissertation completes the probationary period and work effected at the European Southern Observatory (ESO) of Munich. It was realized during the first semester of the academic year 1987-1988 for the grade of "bachelor and master in computer science". The work consisted in the implementation of a software with the purpose to equip ESO with a new user interface for electronic mail. The memory will show the necessity to develop and install a new interface program for this international organization by placing in a prominent position the problems which provocated the project. An expediency study disengages the principal caracteristics of a new interface for electronic mail. Among those, we can name : UNIX-VMS portable, general and individual catalogue and easy use by users without knowledge in computer science. From there the name of the new interface for electronic mail EASY-MAIL. After a comparative study of networks, the specification of each command is presented in the user's manual of EASY-MAIL. With regard to the implementation of the software, the used methodology is to take a software architecture composed of modules. The construct of the interface has been made easy by the use of a dialogue conduct system. The programs have been realized with the C language. An evaluation of EASY-MAIL based on the validation of the software and a study on futur developments complete the work.

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

Introduction générale

Cette introduction générale sur le travail réalisé vise à donner au lecteur un aperçu sur chacun des chapitres du livre et à lui permettre ainsi une lecture aisée. Il présente donc, dans une brève description, les différents chapitres et dégage ainsi la structure générale de l'ouvrage.

D'autre part, le mémoire a été rédigé de telle façon que chaque étape réalisée durant le stage se reflète par un chapitre qui lui est propre. On peut donc voir à travers ce document comment j'ai organisé et réparti le travail depuis mon arrivée dans l'organisation (ESO) avec laquelle j'ai dû d'abord me familiariser (Chapitre 1), jusqu'à la fin de mon stage où une évaluation du software développé et une analyse de ses développements futurs ont été présentées au responsable des télécommunications (Chapitre 6).

Le chapitre 1 est une présentation de l'ESO dans l'optique de ce projet ; la première section présentant l'ESO en tant qu'organisation internationale. Elle est suivie par une description de son réseau local. En troisième section, on y présente les réseaux internationaux qui lui sont accessibles en commençant par l'énumération de ces réseaux. L'installation des connexions existantes à ces réseaux est ensuite décrite et le projet de la nouvelle installation est présenté.

Le chapitre 2 concerne le courrier électronique de l'ESO. Il débute par une description des protocoles et se poursuit par le modèle OSI, ensuite par le lien qui existe entre les protocoles vis à vis de ce modèle et enfin par le lien entre les protocoles et les réseaux. La section suivante introduit le concept de courrier électronique. Le schéma relationnel entre le courrier électronique et les protocoles est présenté. Ensuite, le point de vue des utilisateurs est discuté en mettant en évidence un certain nombre d'inconvénients et de défauts dans l'utilisation actuelle du courrier électronique à l'ESO. La dernière section présente le nouveau programme d'interface, EASY-MAIL, pour pallier aux inconvénients énoncés précédemment.

Le chapitre 3 est une étude comparative des services offerts par les réseaux. On commencera par envisager la nécessité d'une telle comparaison et la distinction à faire entre services d'interface et services de protocole. Les services du standard X400 et ceux du standard RFC#822 sont alors présentés. Une comparaison des services des deux standards fait l'objet de la section suivante tandis que la dernière section offre une comparaison entre les services de RFC#822 et ceux de différents systèmes de messagerie électronique.

INTRODUCTION GENERALE

Le chapitre 4 présente la spécification de EASY-MAIL. Nous y verrons la démarche utilisée pour arriver à un interface possèdant les commandes souhaitées par les futurs utilisateurs du logiciel. Le guide de l'utilisateur de Easy-Mail est donné et spécifie les commandes à réaliser.

Le chapitre 5 décrit la démarche utilisée pour l'implémentation de EASY-MAIL. Après une introduction, la méthodologie concernant la construction des programmes est présentée. Les choix relatifs à l'architecture logicielle sont précisés tandis que la méthode employée pour la construction du logiciel est explicitée. Enfin, la dernière section concerne le choix des outils utilisés, dont notamment le langage de programmation, les logiciels de courrier électronique et un système de gestion du dialogue dont les principaux avantages sont explicités.

Enfin, le chapitre 6 est une conclusion du travail effectué. La première section contient une évaluation du logiciel construit avec, comme justificatif, la méthodologie de validation utilisée. En fin de chapitre, les développements futurs à y apporter sont présentés.

Chapitre 1 : Introduction à l'ESO

1.1 ESO, une organisation internationale.

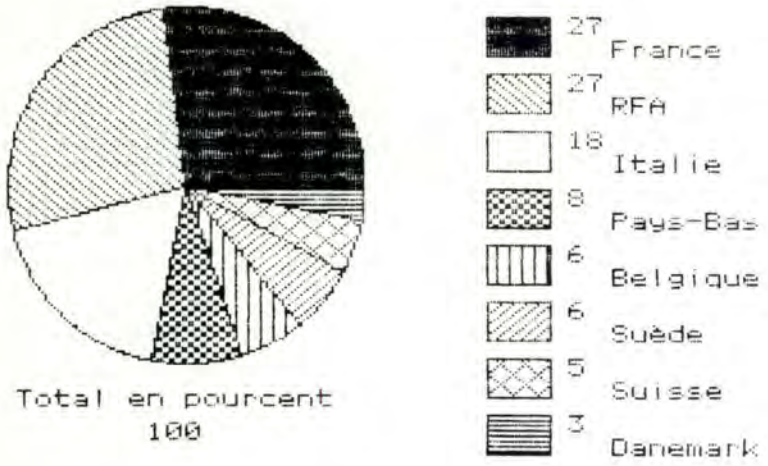
ESO est l'abréviation de European Southern Observatory. Il s'agit d'une organisation scientifique européenne regroupant huit pays membres: la République Fédérale d'Allemagne, la Belgique, le Danemark, la France, la Hollande, l'Italie et enfin la Suisse.

Le siège de l'organisation se trouve à Garching près de Munich et l'observatoire proprement dit est situé au sommet de la montagne appelée La Silla en bordure sud du desert d'Atacama et à environ six cents kilomètres au nord de Santiago du Chili. Cette région a été choisie pour ses nuits sans nuages et son ciel remarquablement transparent. Treize téléscopes optiques de grandeurs variables et d'usages spécifiques y sont actuellement en service tandis qu'un téléscope de Nouvelle Technologie (le NTT) de 3.5 métres ainsi qu'un radiotéléscope sont en construction.

L'ESO a été fondée en 1962 pour promouvoir la collaboration en astronomie et pour fournir aux scientifiques européens un observatoire destiné à l'étude du ciel austral, jusque-là relativement peu étudié. Les avant-projets pour les instruments sont élaborés à ESO et leur construction est réalisée sous contrats par l'industrie européenne. On ne peut manquer de citer comme exemple le VLT (Very Large Telescope) dont l'Europe vient de décider la construction. Le VLT sera constitué d'un réseau de quatre téléscopes de huit métres chacun pouvant fonctionner ensemble ou séparément selon les besoins du moment. Ils atteindront leur puissance maximum une fois pointés simultanement sur le même point de l'espace donnant alors l'equivalence d'un téléscope de seize métres et dotant du même coup l'Europe du plus grand téléscope du monde.

L'ESO est dirigée selon la politique établie par le conseil qui se compose des délégués des divers pays membres et est financée par les contributions de ces pays. La figure 1 illustre la répartition du financement de l'ESO par les pays membres.

Figure 1 : Contributions des pays membres



1.2 Le réseau local de l'ESO

Les réseaux locaux constituent une catégorie à part dans l'univers des réseaux télématiques. Créés par les entreprises pour relier entre eux l'ensemble de leurs équipements informatiques, ils sont installés dans un site privé et échappent donc aux réglementations en vigueur pour les réseaux publics. Les principaux constructeurs d'ordinateurs ont développé leur propre réseau local, adapté à leur production et exploitant au mieux les performances de celle-ci. Le choix d'une marque d'ordinateurs subordonne dès lors le plus souvent l'adoption du système du réseau local de cette même marque. Parmis les constructeurs les plus connus, citons WANG pour le réseau local WANGNET, XEROX pour ETHERNET et IBM pour TOKEN RING. De par le support de transmission choisi et les courtes distances à parcourir, les réseaux locaux permettent des transmission à grande vitesse (plusieurs mégabits par seconde).

Les ordinateurs de l'ESO sont reliés ensemble par un réseau local ETHERNET. Celui-ci à une topologie physique et logique en "bus" et utilise la technique CSMA/CD (Carrier Sense Multiple Access/Collision Detection) définie par la norme IEEE 802.3 (Institute of Electrical and Electronics Engineers). Le support physique est constitué de cables coaxiaux. ETHERNET permet jusqu'à la connexion de 1024 stations. L'ensemble du réseau ainsi que la configuration des ordinateurs ont été clairement dessinés et schématisés par le Manager du système. Ce dessin est à la disposition des utilisateurs et est régulièrement mis à jour. Il est repris par la figure 2. Nous allons en faire la description.

Les deux Vax 8600 travaillant sous VMS sont appelés plus communement ESOMC0 et ESOMC1. Ils supportent actuellement la plus grande partie de la charge de travail. ESOMC1, en plus des utilisateurs connectés, est notamment chargé des télécommunications vers l'exterieur comme nous le verrons au chapitre suivant. Ces deux machines ont accès à 9 disques de 456 mégabytes par l'intermédiaire d'un ordinateur chargé d'optimiser les temps d'accès (le HSC50). En plus de différents types de terminaux et imprimantes, ESOMC1 a accès à quatre armoires à bandes magnétiques (MTA0,MTA1 et MTA2) et ESOMC0 à trois (MTA0,MTB0 et MTB1). Outre l'utilisation de ces armoires à bande magnétique pour la sauvegarde d'anciennes versions de fichiers, celles-ci ont un rôle important puisqu'elles servent aussi à la lecture des données recueillies par l'observatoire situé au Chili. Il n'existe en effet à ce jour aucune liaison satellite entre l'observatoire et le siège de l'organisation. Les données sont donc acheminées par bandes magnétiques ce qui explique ce nombre relativement élevé d'armoires à bandes. L'ordinateur ESOMC0 possède deux disques supplémentaires de 551 mégabytes gérés par un controlleur (DCP). Dans le cadre du traitement d'images obtenues avec les téléscopes par le système MIDAS, il faut noter les terminaux avec processeur local (EPA0, EPA1, EPA2, EPB0, EPB1) et une machine (DISCOMED) permettant de faire des dias positives à partir de ces images.

CHAPITRE 1 : INTRODUCTION A L'ESO

Un micro-Vax de seize mégabytes travaillant également sous VMS et disposant de deux disques (respectivement 456 mégabytes et un gigabyte) et d'un lecteur de bande magnétique est raccordé au réseau local. Il est plus facilement dénommé sous le pseudonyme de ESOMC2.

Deux ordinateurs sous UNIX font également partie intégrante du réseau. D'abord WS0, il s'agit d'un micro-Vax GPX de 11 mégabytes. Il travaille sous UNIX BSD. Le second, ESOMC3, est un BULL SPS7 travaillant sous UNIX SYSTEM 5 avec extension BSD.

Pour terminer notre approche du réseau local, il nous faut encore signaler le NBI 64 et NBI 5000, deux ordinateurs plus spécialement conçus pour le traitements de textes et le TI EXPLORER conçu lui pour l'intelligence artificielle et travaillant avec LISP comme langage natif. Des terminaux supplémentaires peuvent être directement raccordés au réseau grâce à un PBX (Private Branch Exchange).

Enfin, il est à noter que l'ESO possède un réseau local TOKEN RING. Celui-ci sera utilisé ultérieurement pour le contrôle d'instruments tel que des contrôleurs de téléscopes par des appareils de mesures automatiques. Ce réseau utilise les lignes torsadées. Il a une topologie physique en "star" et une logique en "ring". Sa vitesse est de 4 ou 16 mbps tandis que le nombre de stations peut aller jusqu'a 72 ou 260. Sa principale caractéristique est d'utiliser comme méthode de partage "l'anneau en jeton" qui a été définie par la norme IEEE 802.5.

Figure 2 : Configuration des ordinateurs de l'ESO

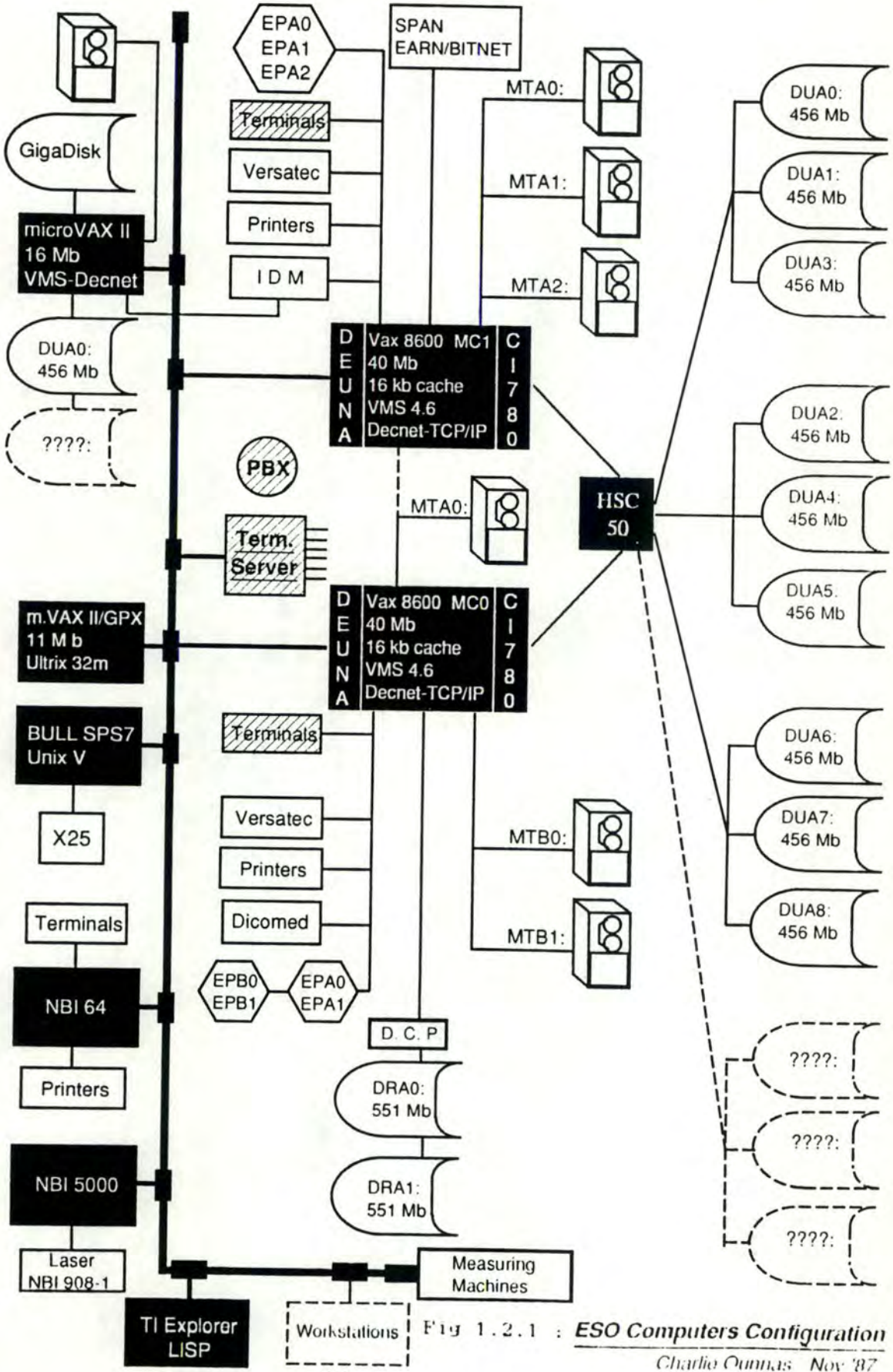


Fig 1.2.1 : *ESO Computers Configuration*
Charlie Cumnas Nov '87

1.3 Les réseaux internationaux accessibles de ESO

1.3.1 Introduction aux réseaux

Il nous est apparu nécessaire de faire une petite introduction aux réseaux informatiques internationaux qui seront traités et discutés dans ce travail. Cependant, il nous faut d'ors et déjà faire remarquer qu'on ne peut mettre sur le même pied tous ces réseaux. Afin d'expliquer ce fait, précisons d'abord quelques point de terminologie:

réseau :
moyen partagé de communication utilisant la notion d'adresse.

réseau physique :
ensemble des moyens matériels et des ressources logicielles mis en oeuvre pour permettre les communications (Ex: câbles, modems, procédure, etc...).

réseau logique:
système de coopération entre systèmes informatiques offrant aux utilisateurs un ensemble de services de haut niveau (Ex: messagerie électronique, transfert de fichiers, etc...). Cette coopération s'établit autour d'un réseau physique. Le réseau logique est aussi parfois appelé réseau à valeur ajoutée.

De ces quelques définitions, on peut distinguer 3 niveaux correspondant aux 3 types de réseaux cités. Le premier niveau correspond aux réseaux physiques, le deuxième aux réseaux logiques et enfin le troisième qui correspond lui aux réseaux qui possèdent chacun un réseau physique et logique mais dont il n'est pas possible de les distinguer étroitement l'un de l'autre, les deux étant liés ensembles.

1) Les réseaux physiques

DATEX-P est un réseau de la R.F.A. qui fait partie des réseaux PSDN (Packet Switching Data Network) car le message est envoyé et traverse les réseaux selon une technique appelée "Packet Switching". C'est sur ce type de réseau physique que s'appuie le réseau logique SPAN. Nous avons en Belgique un réseau équivalent à DATEX-P connu sous le nom de DCS (Data Communication Service).

2) Les réseaux logiques

EUNET (European Unix Network) est un réseau ouvert à toute machine utilisant UNIX, quelle que soit la raison sociale de l'utilisateur. EUNET est bâtis autour de UNIX et plus particulèrement du logiciel UUCP. Ce réseau transfert ses informations par des réseaux physiques PSDN et des lignes louées.

INTERNET (INTERconnecting heterogeneous computer NETworks) a été développé suite au projet d'interconnexion de réseaux de types différents. Ce projet était subventioné par DARPA et s'appelait "internet system". Il a donné naissance au réseau INTERNET qui est le fruit des travaux entrepris par des équipes de chercheurs de l'industrie et d'universités. INTERNET est en fait la réunion d'une douzaine de réseaux interconnectés ensemble par des passerelles. Parmi ces réseaux, les principaux sont ARPANET, SATNET et WIDEBANDNET. INTERNET s'étend actuellement sur les Etats-Unis et sur l'Europe. On peut le considérer comme un réseau à valeur ajoutée car, pour offrir ses services, il s'appuye sur le fonctionnement d'autres réseaux.

SPAN (Space Physics Analysis Network) est le réseau pour l'analyse de la physique spatiale. Son but était d'utiliser le Mail électronique afin de faciliter l'analyse de données de l'espace,l'étude de documents, l'accès aux bases de données, le transfert de données numériques et graphiques. Il est devenu opérationnel en 1981 avec 3 noeuds principaux aux Etats-Unis et s'est développé très rapidement. Actuellement, il en compte environ 900 reliant ensemble les Etats-Unis, l'Europe et le Canada. Les diciplines supportant le réseau sont en extension pour y inclure les sciences pour l'étude du climat, de l'océan, de l'atmosphère, etc...Span utilise pour la transmission des informations les réseaux physiques PSDN, des lignes louées et des liaisons satellites.

3) Les réseaux physiques-logiques

ARPANET est la création de DARPA (Defense Advanced Research Projects Agency) des Etats-Unis. Ce réseau informatique est le résultat de recherches de simulation de réseaux d'ordinateurs effectués avec des universités et des organismes privés à la fin des années 1960. Il est devenu opérationnel en décembre 1969 et n'a cessé de grandir depuis lors. Actuellement, il couvre principalement l'entièreté des Etats-Unis. Ce projet a été à la base de la plupart des connaissances sur la gestion des réseaux. ARPANET fait maintenant partie intégrante du réseau INTERNET.

EARN (European Academic and Research Network) est un réseau international, ouvert aux universités, aux grandes écoles et aux centres de recherche. Mis en oeuvre en 1984 par une dizaine de pays d'Europe, il couvre aujourd'hui la presque totalité des pays européens, depuis la Finlande jusqu'à Israël. L'initiative du projet revient à IBM. Ce réseau est dérivé des réseaux nord-américain BITNET (Because It's There NETwork) et NETNORTH auquels il est connecté et avec lesquels il totalise environ 3000 machines. Outre ces connexions vers ses frères d'outre-Atlantique, EARN offre diverses passerelles vers d'autres réseaux: citons par exemple EUNET, INTERNET.

1.3.2 Installation existante

Actuellement, c'est l'ordinateur ESOMC1 qui est en liaison avec les réseaux internationaux comme l'illustre la figure 3. Les seuls réseaux accessibles directement de ESO sont SPAN, EARN et DATEX-P. Pour envoyer un message sur un autre réseau, l'utilisation d'un noeud de EARN faisant office de passerelle s'avère être nécessaire. On utilise le noeud CERNVAX pour accèder à EUNET ou CERN (le réseau local du CERN), WISCVM pour INTERNET et UKACRL pour JANET. Bien sûr, un grand nombre d'autres réseaux sont ainsi accessibles depuis ESO mais les énumérer tous ne rentre pas dans le cadre de cette étude.

L'ordinateur ESOMC1 en liaison avec SPAN, EARN et DATEX-P ne veut bien sûr pas dire que seuls les utilisateurs branchés sur cette machine peuvent envoyer et recevoir des messages de l'extérieur. Un système de routage à travers le réseau local de l'ESO permet de bénéficier des réseaux internationaux quelle que soit la machine sur laquelle on travaille. Prenons l'exemple d'un utilisateur travaillant sur ESOMC0 et désirant envoyer un message à un utilisateur travaillant sur une machine connectée à SPAN. Après la rédaction du message et avoir donné l'adresse correcte du destinataire, ESOMC0 redirige le tout vers un processus d'un ordinateur capable de réenvoyer le message sur le réseau spécifié c'est-à-dire en l'occurence vers le processus MAILER de ESOMC1. MAILER est donc chargé de réenvoyer les messages reçus vers leurs destinataires respectifs. Bien sûr l'inverse fonctionne également: les messages reçus par ESOMC1 venant de l'extérieur et s'adressant à un utilisateur d'une autre machine de ESO sont également redirigés vers celle-ci par le réseau local.

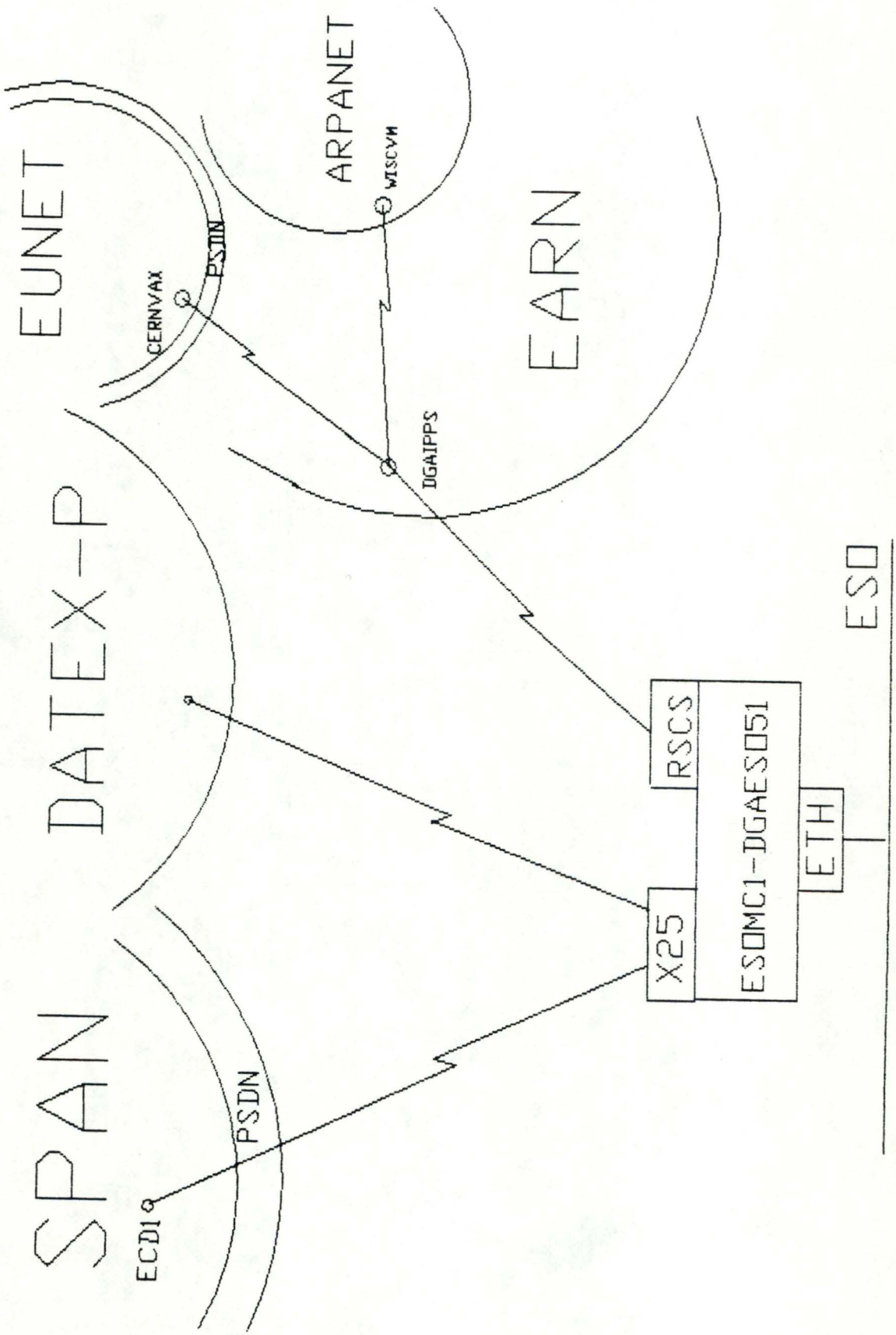
La situation existante possède 2 inconvénients majeurs:

- -Le premier est que la fonction d'envoi, de reception et de routage des messages par ESOMC1 augmente considérablement sa charge de travail aux dépens des utilisateurs.

- -Le deuxième inconvénient est beaucoup plus grave et nécessite une intervention urgente. Le protocole Decnet de DIGITAL utilise une boîte aux lettres commune à tous les utilisateurs. Il est apparu que celle-ci constitue le talon d'Achille de la machine pour d'éventuels agresseurs plus souvent appelés dans le jargon informatique "Hackers" ou pirates.

Une évolution de l'installation actuelle s'impose donc et fait l'objet de la rubrique suivante.

Figure 3 : Installation existante



1.3.3 Le projet

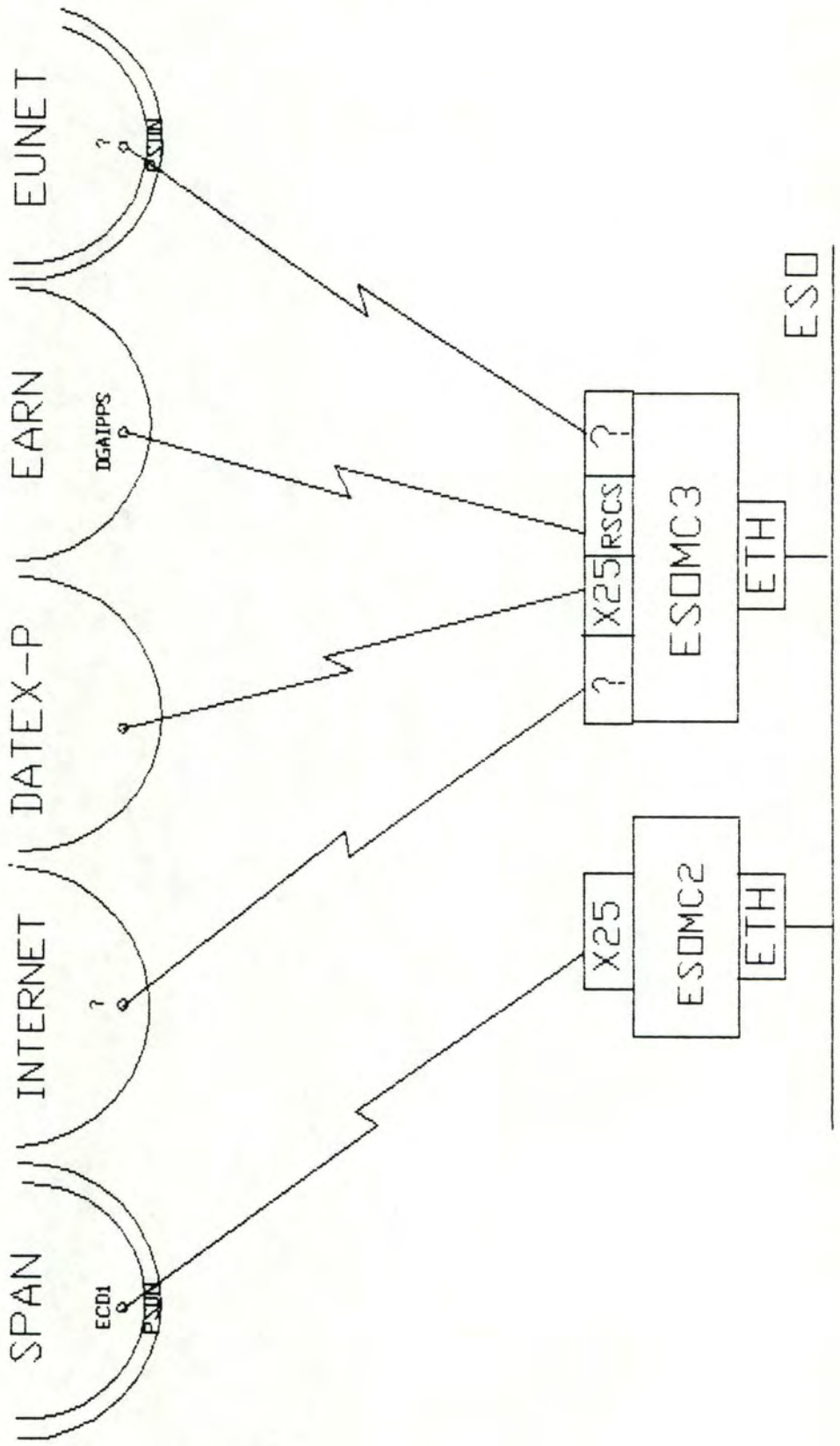
La solution envisagée actuellement pour faire face aux problèmes discutés précédemment est celle-ci:

-Afin d'alléger la charge de travail de ESOMC1, celui-ci sera complètement déchargé des communications avec les réseaux externes. Une nouvelle liaison au réseau INTERNET ainsi qu'à EUNET est prévue. Ce travail sera pris en charge par ESOMC3 qui se limitera à cette activité. La manière d'accéder à ces deux réseaux, que ce soit par ligne louée ou via un réseau PSDN dépendra du software disponible sur ESOMC3 et/ou des noeuds EUNET et INTERNET accessibles.

-En ce qui concerne la menace d'intrusion d'éventuels pirates pouvant détruire des disques entiers d'informations et paralyser ainsi les ordinateurs pendant plusieurs jours, celle-ci sera fortement réduite en installant la liaison DECNET sur le micro-Vax ESOMC2.

Le projet est clairement illustré par la figure 4.

Figure 4 : Le projet



Chapitre 2 : Le courrier électronique de l'ESO

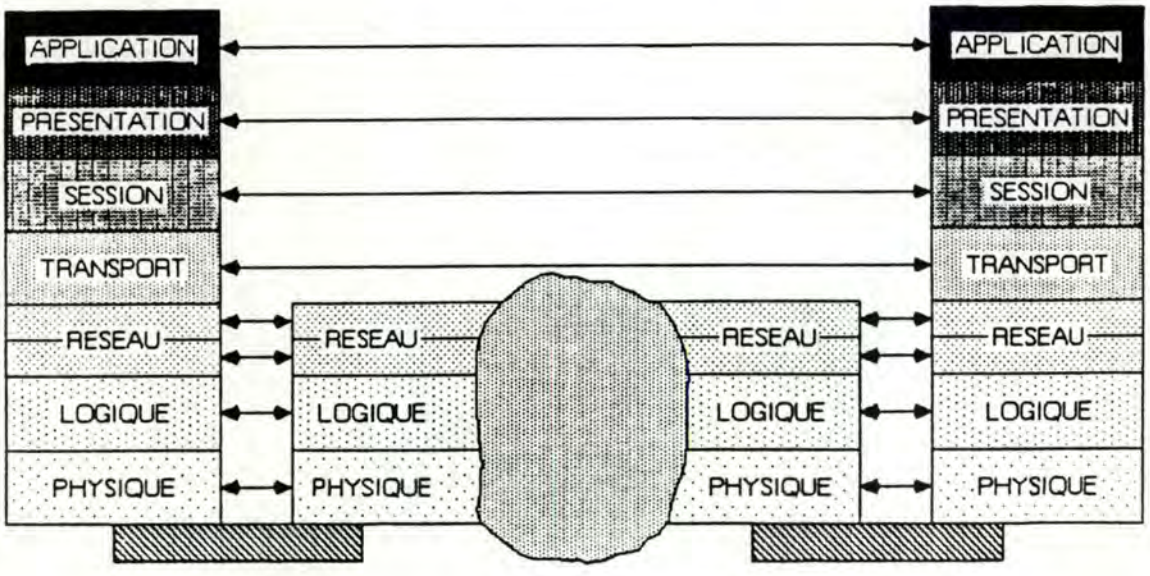
2.1 Système d'exploitation et protocoles de télécommunication

2.1.1 Le modèle OSI

On assiste actuellement au développement d'un ensemble de standards internationalement reconnus définissant les modalités d'échange entre systèmes informatiques de constructeurs différents. C'est l'institut international de normalisation appelé plus couramment l'ISO (International Standards Organisation) qui a pour tâche majeure de déterminer cet ensemble de standards connus sous l'appelation d'OSI (Open Systems Interconnection). L'objectif poursuivi est donc d'améliorer les possibilités d'interconnexion à la disposition des utilisateurs de réseaux informatiques, en établissant les standards d'un système unique et internationalement acccepté pour la communication entre réseaux. Lorsque ces normes seront totalement définies et supportées par une majorité de réseaux informatiques, les obstacles à la communication de données seront levés : il n'y aura plus alors qu'un seul réseau.

Les spécifications d'OSI définissent un système comportant sept couches. Toutes les couches sont concernées lors d'une communication. Elles sont organisées de manière strictement hiérarchique c'est-à-dire que chaque couche fournit des services de communication à la couche immédiatement supérieure et utilise, pour ce faire, les services offerts par la couche immédiatement inférieure. La figure 5 illustre la découpe en couches de la norme ISO. Les quatre premières couches sont généralement considérées comme formant un tout responsable des fonctions de communication tandis que les suivantes assurent plutôt des fonctions de traitements. La découpe en couches est celle-ci:

Figure 5 : La découpe en couche du modèle OSI.



La couche physique (1) fournit l'interface entre la machine communicante et le moyen de communication physique appelé aussi circuit de données. Elle travaille au niveau des bits en les transmettant sur le circuit de données. Une des difficultés à vaincre par la couche physique est s'assurer la transmission : si elle envoie un bit à 1, il faut que ce bit soit reçu de l'autre côté du circuit de données comme un bit à 1 et non à zéro. D'autres difficultés sont par exemple, de savoir combien de volts doivent être utilisé pour représenter un bit à un et combien pour un bit à zéro ou combien de micro-secondes doivent durer un bit. Les problèmes à résoudre à ce niveau sont largement fonction des aspects mécaniques et électriques du circuit de données.

La couche de liaison de données (2) ou couche logique a pour but d'aménager la transmission "brute" en assurant la détection et la correction des erreurs. Elle accomplit cette tâche en groupant les bits en trames, en les transmettant et traitant celles reçues. Le standard le plus utilisé à ce niveau est le protocole HDLC (High level Data Link Control) proposé par l'ISO. Alors que la couche physique accepte et transmet purement et simplement un ensemble de bits sans distinguer de structure, la couche liaison de données crée et reconnait les trames. Cela se fait en ajoutant en début et fin de trame une séquence bien connue de bits. Cette séquence de bits peut se trouver accidentellement dans les données de sorte que des mesures spéciales doivent être prises pour éviter la confusion. Cette couche doit résoudre les problèmes de trames perdues ou endommagées pour que la couche supérieure puisse être sûr de travailler avec une ligne sans erreurs. Une autre difficulté qui surgit à ce niveau (et à la plupart des couches supérieures) est de savoir augmenter ou diminuer le flot des informations envoyées sur la ligne afin de ne pas "noyer" un récepteur plus lent.

La couche réseau (3) ajoute les fonctions de routage pour permettre à des paquets de données d'être transmis vers un destinataire bien précis. Cela requiert donc l'inclusion d'une adresse dans les paquets. Cette couche peut aussi fournir un contrôle d'erreur, un contrôle de flux et un multiplexage. Le standard le plus connu est la recommandation X25 pour les réseaux publics à commutation par paquets du CCITT (Comité Consultatif International pour la Téléphonie et la Télégraphie) dont s'est largement inspiré l'ISO pour construire les couches 1, 2 et 3 du modèle OSI. Une difficulté relative à cette couche est de savoir comment le routage d'un paquet est déterminé.

La couche de transport (4) assure la communication entre les processus tournant sur des machines différentes. Elle fournit les fonctions nécessaires pour combler le trou entre les services offerts par la couche réseau et ceux requis par la couche session. Elle offre un contrôle de flux et d'erreurs de bout en bout sur les mouvements de données entre les processus. Elle peut opèrer une concaténation ou segmentation sur les données. Si nécessaire, la couche transport peut faire éclater une connexion transport sur plusieurs connexions réseaux d'une façon transparente à la couche session. Réciproquement, elle peut également multiplexer plusieurs connexions transport sur une seule connexion réseau.

La couche de session (5) prend les services de communication offerts par la couche transport et y ajoute des fonctions orientées "application". Elle est responsable de la gestion du dialogue entre processus tournant sur les machines. Elle règle par exemple les problèmes de synchronisation dans leur dialogue. En ignorant la couche présentation qui exécute purement et simplement certaines transformations sur les données, la couche session joue le rôle d'interface utilisateur. C'est avec cette couche que l'utilisateur doit négocier pour établir une connexion avec un processus sur une autre machine. Une fois la connection établie, la couche session peut alors engager le dialogue. Une telle connexion entre utilisateurs est d'habitude appelée une session. Une session pourrait être utilisée pour permettre à un utilisateur de se "loger" sur une autre machine ou pour transférer un fichier entre deux machines. Pour établir une session, l'utilisateur doit donner l'adresse de la machine avec laquelle il veut correspondre. Une adresse session est destinée à l'utilisateur tandis qu'une adresse transport est, elle, destinée a être employée par les stations "transport". La couche session doit donc être capable de convertir une adresse session en son adresse transport afin qu'une connexion transport puisse être ouverte. Ouvrir une session est une opération difficile. D'abord, il peut être nécessaire d'identifier chacune des extrémités de la session afin de s'assurer qu'elle a le droit d'engager une session. Ensuite, les deux extrémités doivent se mettre d'accord sur des options qui peuvent ou non être prises en considération pendant la session comme par exemple si la communication est half-duplex ou full-duplex.

La couche de présentation (6) est chargée de l'interprétation des données échangées lors d'un dialogue. Elle gère donc le codage des caractères et les règles d'encryptage. Elle assure également le compactage des données, l'adaptation des terminaux et le transfert de fichiers. Elle offre donc des fonctions pour résoudre des problèmes auquels les utilisateurs sont affrontés suffisamment souvent pour ne pas les laisser les résoudre, seul chacun de leur côté. Ces fonctions peuvent être exécutées par des routines d'une librairie disponible aux utilisateurs.

La couche application (7) gère les fonctions spécifiques aux utilisateurs. Citons comme exemple le courrier électronique et le télétex. C'est aussi à ce niveau qu'on trouvera des protocoles relatifs aux bases de données réparties tels que la réservation aérienne ou la gestion d'un système bancaire.

2.1.2 Les protocoles de télécommunication

2.1.2.1 Introduction

ESO qui a accès à plusieurs réseaux doit posséder les protocoles nécessaires à l'utilisation de ceux-ci. Cette introduction a pour but de vous les présenter.

Mais d'abord, il est important de bien voir que les protocoles dont nous allons parler se situent à des niveaux différents les uns par rapports aux autres. Certains protocoles sont orientés "Communication de données" et sont donc des protocoles de bas niveau tandis que d'autres se situent beaucoup plus proches de fonctions spécifiques aux utilisateurs. Afin de ne pas laisser de côté cette notion de niveau des protocoles et pour bien savoir les situer les uns par rapport aux autres, nous allons les présenter dans un ordre qui correspond à celui des sept couches du modèle OSI. Nous partirons donc avec les protocoles orientés "Communication de données" et avancerons progressivement pour arriver finalement aux protocoles proches de la couche application. Cependant, il est essentiel de préciser qu'il n'est pas toujours possible de dire avec exactitude à quelle(s) couche(s) du modèle OSI appartient tel ou tel protocole. Ceci a d'ailleurs déjà fait l'objet de contreverses dans la littérature. En effet, certains protocoles ont été implémentés avant ou pendant la rédaction même de la norme. De ce fait, l'architecture et l'encapsulage des protocoles les uns aux autres se présenteront d'une manière différente selon les constructeurs. La figure 6 qui compare l'architecture du modèle OSI à celle du réseau Arpanet, à SNA qui est l'architecture du constructeur IBM et à DECNET pour la Digital Equipment corporation permet d'illustrer les différences qui peuvent exister entre plusieurs architectures et montre de ce fait qu'il n'est pas toujours possible de relier un protocole à une ou plusieurs couches de la norme OSI.

Figure 6 : Comparaison approximative des architectures.

<table>
<tr><th>Layer</th><th>ISO</th><th>ARPANET</th><th>SNA</th><th>DECNET</th></tr>
<tr><td>7</td><td>Application</td><td>User</td><td>End user</td><td rowspan="2">Application</td></tr>
<tr><td>6</td><td>Presentation</td><td>Telnet, FTP</td><td>NAU services</td></tr>
<tr><td>5</td><td>Session</td><td rowspan="2">(None)</td><td>Data flow control</td><td rowspan="2">(None)</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2">Transport</td><td rowspan="2">Transmission control</td></tr>
<tr><td>Host-host</td><td rowspan="2">Network services</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">Network</td><td rowspan="2">Source to destination IMP</td><td rowspan="2">Path control</td></tr>
<tr><td rowspan="2">IMP-IMP</td><td rowspan="2">Transport</td></tr>
<tr><td>2</td><td>Data link</td><td>Data link control</td><td>Data link control</td></tr>
<tr><td>1</td><td>Physical</td><td>Physical</td><td>Physical</td><td>Physical</td></tr>
</table>

En conclusion, l'association des protocoles aux sept couches du modèle OSI que nous faisons ci-après ne doit être considérée que comme une approximation permettant de mieux situer les protocoles les uns par rapport aux autres et non comme une finalité en soi.

X25 est la norme pour accéder à un réseau PSDN (Packet Switching Data Network) et échanger des données. La norme a été rédigée par le CCITT (Comité Consultatif International pour la Téléphonie et la Télégraphie). X25 a trois niveaux de protocoles qui correspondent aux couches un, deux et trois de la norme OSI.

PSI (Packenet System Interface) est un produit de la Digital Equipment Corporation et permet à une de ses machines de communiquer par l'intermédiaire d'un réseau PSDN avec une autre machine (Digital ou non) en utilisant un lien momentané X25. PSI est une implémentation de la norme X25.

ETHERNET est le protocole qui implémente le réseau local du même nom. Il a été construit par Xerox. Les caractéristiques principales de ce réseau peuvent être résumées comme étant : longueur limitée à un kilomètre, topologie physique et logique en "bus", vitesse de transmition jusqu'à dix Mbps, utilisant le cable coaxial et la technique CSMA/CD (Carrier Sense Multiple Access / Collision Detection). Celle-ci est définie par la norme IEEE 802.3. ETHERNET a trois niveaux de protocoles qu'on peut rapprocher des niveaux un, deux et trois de OSI.

TOKEN RING est le réseau local d'IBM. Ses caractéristiques principales peuvent être résumées comme utilisant la ligne torsadée, la topologie physique en "star" et la logique en "ring", une vitesse de 4 ou 16 mbps et un nombre de stations allant jusqu'à 72 ou 260. Il utilise comme méthode de partage "l'anneau en jeton" définie par la norme IEEE 802.5. Le protocole TOKEN RING peut également être rapproché aux niveaux un, deux et trois de OSI.

TCP/IP (Transfer Control Protocol/Internet Protocol) est un protocole pour la communication et l'interconnexion de différents réseaux. Il a été initialement construit par le département de la défense des USA pour connecter ARPANET avec d'autres réseaux. Rappelons d'ailleurs que c'est cette interconnexion de réseaux qui a donné naissance au réseau INTERNET. TCP garantit la transmission correcte des données, indépendamment des moyens physiques utilisés pour transmettre ces données. Il offre également un service de contrôle de flux. IP est responsable du multiplexage des données entre le réseau et la couche quatre. Il peut fonctionner aujourd'hui au dessus de ETHERNET, TOKEN RING et X25. TCP se situe au niveau Transport de OSI tandis que IP se situe au niveau réseau de OSI.

UUCP (Unix to Unix CoPy) est un programme de copie de fichiers entre machines UNIX qui reçoit en paramètres les spécifications des fichiers sources et de destination. Il est aussi utilisé pour le courrier électronique et les "News" Unix. SENDMAIL est le programme qui permet d'assurer le routage des fichiers à travers le réseau EUNET. UUCP peut être rapproché des couches 4 et 5 de OSI.

DECNET est un ensemble de programmes et protocoles produit par la Digital pour être
utilisé sur ses machines. L 'architecture de DECNET est appelée DNA (Digital Network Architecture). DECNET a été conçu pour permettre aux clients de Digital de créer leurs propres réseaux privés. On trouvera donc à travers le monde plus d'un réseau utilisant DECNET. On le trouve au-dessus de ETHERNET, X25 ou de certains bus bien spécifiques. Lorsque DECNET utilise le protocole X25, on parlera alors de PSI-DECNET. Le protocole PSI-DECNET est d'ailleurs utilisé par l'ESO pour envoyer des messages sur le réseau SPAN dont nous avons parlé à la section 1.3.1. DECNET peut être rapproché des couches 4 et 6 de OSI, la couche session étant absente. Parmi les programmes de la Digital Equipment Corporation on peut encore citer PSI MAIL qui permet d'envoyer et de recevoir directement des messages d'une autre machine VAX/VMS par l'intermédiaire d'un réseau PSDN.

TELNET (Virtual Terminal Protocol) donne à un utilisateur la possibilité de se connecter sur un autre noeud du réseau comme si son terminal était un terminal de ce noeud. Ce protocole utilise TCP/IP. Il se situe dans la couche présentation de l'architecture ARPANET qu'on peut mettre au même niveau que celle de l'architecture OSI.

RLOGIN (Remote LOGIN) donne à un utilisateur la possibilité de se connecter sur un autre noeud du réseau comme si son terminal était un terminal de ce noeud. Ce protocole utilise TCP/IP. Il se situe dans la couche présentation de l'architecture OSI.

FTP (File Transfert Protocol) a été initialement conçu pour le réseau ARPANET et utilise TCP/IP. Il permet à un utilisateur de communiquer avec un autre noeud du réseau dans le but de manipuler des fichiers. Grâce à lui, on peut donc afficher le répertoire d'un disque, transférer ou exécuter des opérations sur un fichier d'un autre noeud du réseau. FTP se situe aussi dans la couche présentation de l'architecture ARPANET qu'on peut mettre au même niveau que celle de l'architecture OSI.

SMTP (Simple Mail Transfert Protocol) est utilisé pour le courrier électronique. Il permet l'envoi, le relai et le retour automatique de messages non-délivrés. Il utilise TCP/IP. SMTP se situe au niveau présentation de OSI.

NFS (Network File System) rend à partir d'un noeud du réseau le disque d'un autre noeud accessible. Il utilise TCP/IP. NFS se situe au niveau présentation de OSI.

RSHELL (Remote Shell) permet à un utilisateur d'exécuter une commande sur une autre machine du réseau tout comme si il se trouvait sur cette autre machine. RSHELL se situe au niveau présentation de OSI.

RSCS (Remote Spooling Communications Subsystem Networking) est un programme qui supporte la transmission et la réception de messages, commandes, jobs sur un réseau. RSCS peut donc être rapproché des couches 1 à 6 de OSI.

2.1.2.2 Liens entre les protocoles

Il nous est apparu intéressant de faire apparaître par un schéma comment ces différents protocoles s'articulent les uns aux autres. Nous avons fait la distinction entre les protocoles tournant sous système VMS illustré par la figure 7 et ceux tournant sous système UNIX illustré par la figure 8.

Figure 7 : Système d'exploitation et protocoles en VMS

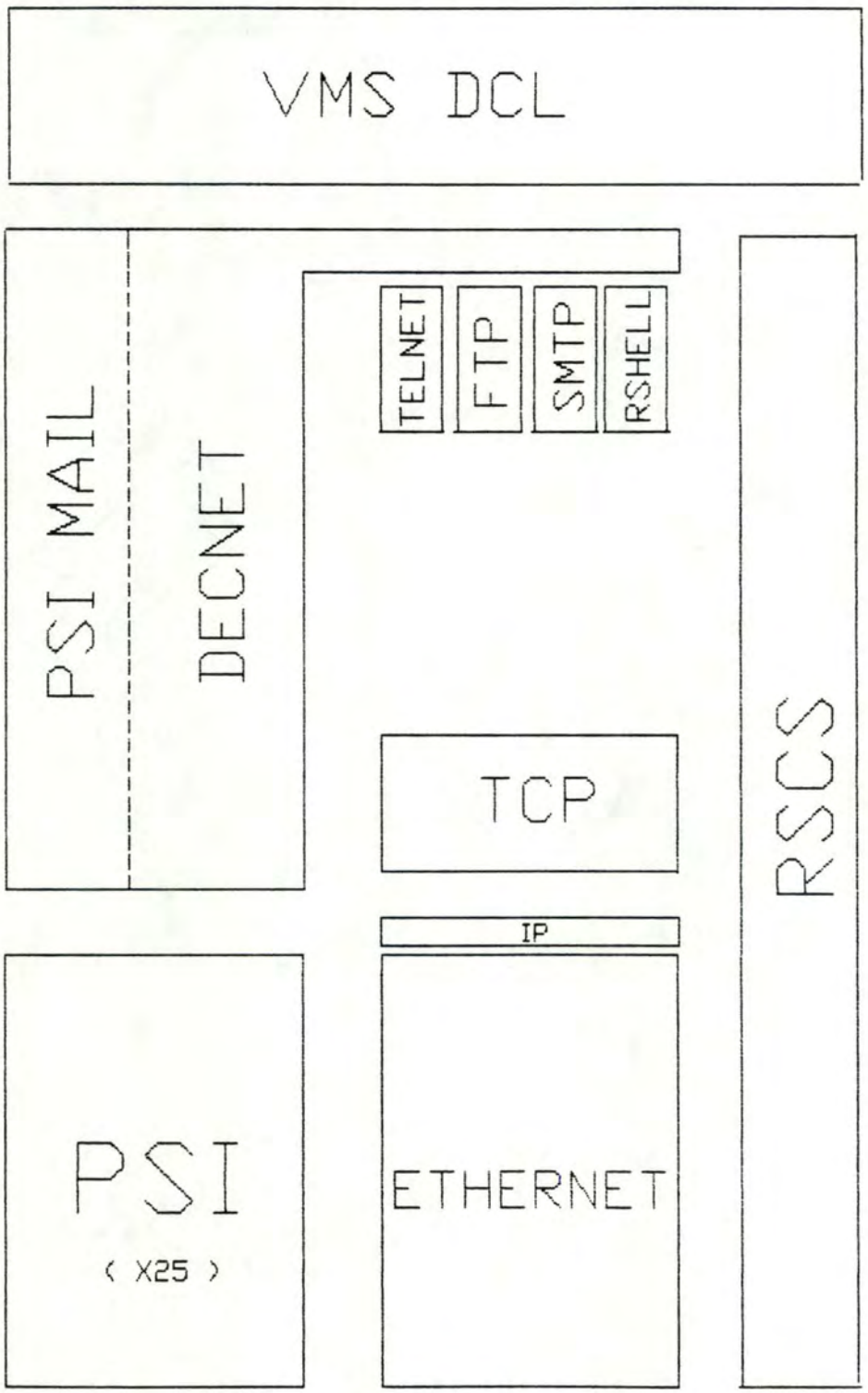
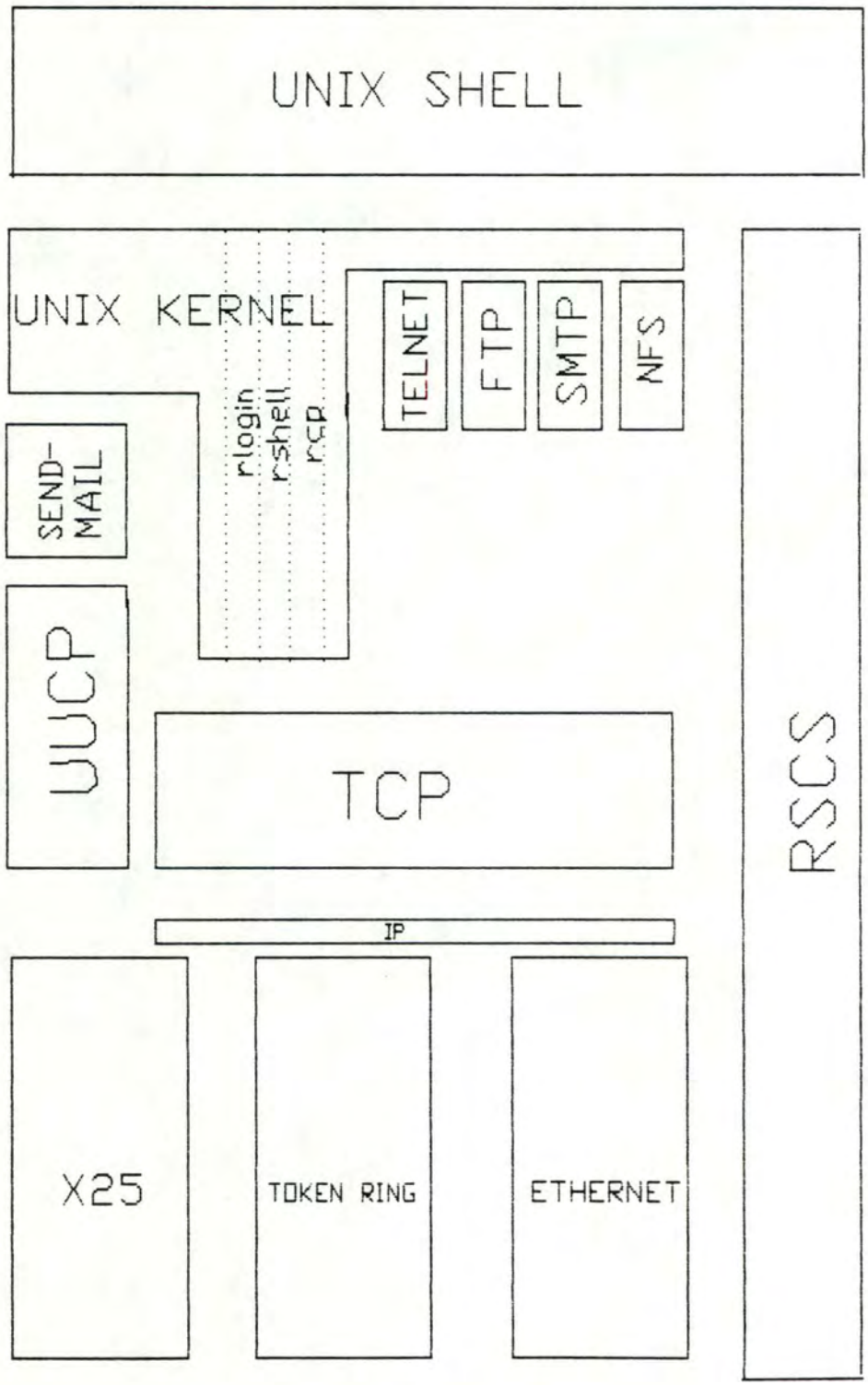


Figure 8 : Système d'exploitation et protocoles en UNIX



2.2 Courrier électronique et protocoles de télécommunication

2.2.1 Introduction

Le courrier électronique ou messagerie électronique couvre le domaine des communications de textes non formels, messages courts le plus souvent et transmis sans souci de présentation. Les expériences du VIDEOTEX (système d'information produits par des équipements informatiques et permettant la visualisation sur écran de pages d'information) ont fait découvrir aux professionnels comme au grand public l'interêt de la messagerie électronique. Celle-ci est disponible à travers des réseaux publics ou privés.

Le service de messagerie électronique est d'une grande utilité dans l'organisation du travail de bureau. Il remplace idéalement les communications par coups de téléphone dont 70 % des appels, en moyenne, sont inutiles, le correspondant étant absent ou occupé. Avec le courrier électronique, le destinataire prend connaissance des appels lorsqu'il le souhaite, il peut sauvegarder également leur contenu en mémoire et l'imprimer. La messagerie supprime également les échanges de correspondance par lettres aux délais beaucoup trop longs ou les déplacements à l'intérieur mais surtout à l'extérieur de l'entreprise.

2.2.2 schéma relationnel

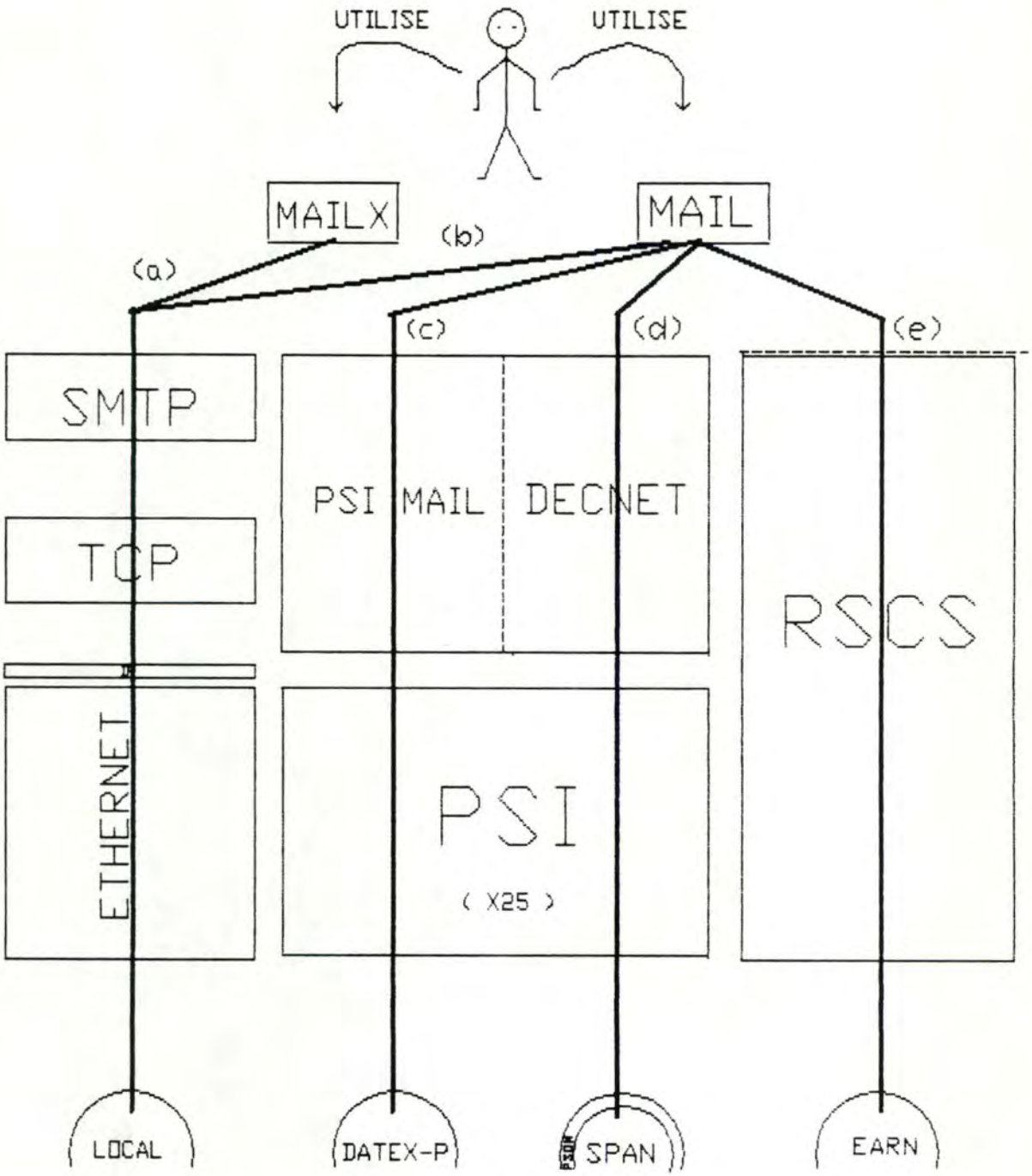
Lorsqu'un utilisateur désire envoyer un message, il utilise pour ce faire un programme de courrier électronique. Celui-ci offre à l'utilisateur de nombreux services. Parmi ceux-ci on peut citer : composer un message, offrir un service de stockage des messages, prendre connaissance d' un message reçu et bien sûr envoyer un message. Pour effectuer ce dernier service, le programme de courrier électronique va faire appel à un programme de protocole. Celui-ci est choisi en fonction du réseau spécifié dans l'adresse du destinataire. Rappelons en effet que pour utiliser un réseau avec succès, il est nécessaire de respecter un certain nombre de normes comme par exemple la structure du message. Ces normes varient d'un réseau à l'autre et sont appliquées par les programmes de protocoles.

L'ESO possède trois interfaces de courrier électronique: le MAIL de Digital Equipment Corporation qui fonctionne sur les ordinateurs VAX/VMS (essentiellement ESOMC0 et ESOMC1), le MAILX de UNIX qui tourne sur la machine BULL SPS7 sous UNIX SYSTEM 5 (ESOMC3) et le MAIL également de UNIX qui lui fonctionne sur le micro-vax GPX sous UNIX BSD (WS0). Les deux interfaces de courriers électroniques MAIL et MAILX de UNIX ne diffèrent essentiellement que par le système d'exploitation sur lequel ils tournent (BSD4.2 ou SYSTEM 5). Un utilisateur désireux d'envoyer un message doit utiliser l'interface de courrier électronique se rapportant à la machine sur laquelle il travaille. Nous ne parlerons pas ici de l'interface MAIL de UNIX, le micro-vax WS0 n'étant pas en effet disponible pour le grand public.

CHAPITRE 2 : LE COURRIER ELECTRONIQUE DE L'ESO

Il m'a semblé important de montrer par un schéma la relation qui existe entre l'utilisateur, les deux interfaces de courrier électronique MAIL de Digital et MAILX de UNIX, les protocoles de communication et enfin les réseaux. La figure 9 illustre cette relation. Le cheminement des messages à travers les couches de la norme OSI est également mis en évidence.

Figure 9 : Schéma relationnel



Cheminement du message:
(a) = envoi au réseau local de l'ESO à partir du MAILX.
(b) = envoi au réseau local de l'ESO à partir du MAIL.
(c) = envoi au réseau DATEX-P à partir du MAIL.
(d) = envoi au réseau SPAN à partir du MAIL.
(e) = envoi au réseau EARN à partir du MAIL.

2.2.3 Le point de vue des utilisateurs

Nous avons vu précédemment dans les rubriques de ce chapitre la notion de courrier électronique et la relation qui existe entre l'utilisateur, les deux interfaces de courrier électronique, les protocoles de communication et enfin les réseaux. Nous allons maintenant continuer notre démarche d'étude à propos du courrier électronique de l'ESO en nous mettant à la place de l'utilisateur. Il faut souligner que la plupart des utilisateurs de la messagerie électronique de l'ESO sont des scientifiques n'ayant pas ou peu de formation en informatique. Nous allons ainsi pouvoir mettre en évidence les difficultés que ceux-ci rencontrent dans l'utilisation des deux interfaces de courrier électronique MAIL et MAILX. Nous illustrerons les difficultés par des exemples que nous commenterons.

A. Appel au courrier électronique

La première difficulté que l'utilisateur va rencontrer est très certainement l'opération d'appel à l'interface de courrier électronique. Cet appel se fera en frappant "MAIL" ou "MAILX" sur le clavier. Le choix de la commande variera en fonction de la machine sur lequel l'utilisateur travaille.

B. Syntaxe des commandes

Une autre difficulté est la différence de syntaxe qui existe entre deux commandes similaires des deux interfaces MAIL et MAILX. Prenons par exemple la commande permettant de changer de folder. Rapellons pour le lecteur non initié à la messagerie électronique que les folders sont des classeurs ou fardes dans lesquels l'utilisateur peut ranger ses messages.

La syntaxe de cette commande pour MAILX est :

FOLDER (filename)
'filename' est le nom du nouveau folder.

Il existe des conventions pouvant être utilisées:
-si 'filename'=# alors on utilise le folder précédent.
-si 'filename'=% alors on utilise le folder MAILBOX
-si 'filename'=%name alors on utilise le folder MAILBOX de l'utilisateur ayant le nom 'name'.
-si 'filename'=& alors on utilise le folder MBOX.
-par défaut, on utilise le folder MAILBOX
MAILBOX est le folder dans lequel arrive les nouveaux messages. Lorsqu'un nouveau message a été lu il est transféré vers le folder MBOX.

La syntaxe de cette même commande mais pour MAIL est :

SELECT (foldername)
ou bien encore:
SET FOLDER (foldername)
'foldername' est le nom du nouveau folder.

Il existe des paramètres pouvant être ajoutés à la commande:
/BEFORE=date
Ce paramètre indique que les messages à prendre en considération sont les messages datés d'avant la date précisée dans le paramètre.
/NEW
Ce paramètre signifie que seuls les nouveaux messages c'est-à-dire les messages qui n'ont pas encore été lu sont à prendre en considération.
/SINCE=date
Enfin, ce dernier paramètre signifie que les messages à prendre en considération sont les messages datés après la date précisée dans le paramètre.

C. Syntaxe des adresses

Il faut aussi souligner la difficulté que l'utilisateur va rencontrer dans la syntaxe de l'adresse du destinataire qu'il devra donner pour l'envoi de son message. Précisons les termes employés pour l'adressage d'un destinataire:

-"username" : Il s'agit du nom de "login" d'un utilisateur connecté sur un ordinateur.
-"nodename" : C'est le nom d'une unité de traitement qui est directement connectée à un réseau et qui est adressable. Il peut s'agir d'un ordinateur, d'une station de travail ou d'un appareil périphérique.
-"network" : Il s'agit du nom du réseau.
-"dte-address": C'est l'adresse d'identification d'un DTE (Data Terminal Equipment) sur un réseau PSDN.

Nous pouvons faire maintenant la séparation entre les deux interfaces MAIL (VMS) et MAILX (UNIX).

C.1 MAIL

L'utilisateur se trouve connecté à un ordinateur VAX/VMS en l'occurence ESOMC0 ou ESOMC1. Il doit donc pour envoyer son message utiliser l'interface MAIL. L'adresse du destinataire doit être spécifiée après le "TO :" demandé par l'interface. Nous pouvons distinguer plusieurs cas selon le réseau auquel est connecté le destinataire souhaité:

le destinataire se trouve sur le même ordinateur
La syntaxe de l'adresse sera alors:
TO : username

Exemple: envoi d'un message à l'utilisateur RICHMOND qui travaille sur le même ordinateur que moi et dont le username est "RICHMOND" :
TO : RICHMOND

le destinataire se trouve connecté à SPAN
La syntaxe de l'adresse sera alors:
TO : nodename::username

Exemple: envoi d'un message à l'université de STANFORD dont le nom de noeud sur le réseau SPAN est "STAR" pour l'utilisateur Dupont ayant le username "DUPONT" :
TO : STAR::DUPONT

le destinataire se trouve connecté à EARN
La syntaxe de l'adresse sera alors:
TO : EARN::"username@nodename"

Exemple: envoi d'un message au CERN dont le nom de noeud sur le réseau EARN est "CERNVAX" pour l'utilisateur Defert ayant le username "DEFERT" :
TO : EARN::"DEFERT@CERNVAX"

le destinataire se trouve connecté à DATEX-P (utilisation de PSI MAIL)
La syntaxe de l'adresse sera alors:
TO : PSI%dte-address::username

Exemple: envoi d'un message à l'institut d'informatique de Namur dont le numéro de dte-address est 02062816000 pour l'utilisateur Leclercq dont le username est "JPLECLERCQ" :
TO : PSI%02062816000::JPLECLERCQ

le destinataire se trouve connecté au réseau local de l'ESO
La syntaxe de l'adresse sera alors:
TO : EXOS%"username@nodename"

Exemple: envoi d'un message à l'ordinateur ESOMC0 à partir de l'ordinateur ESOMC1 pour l'utilisateur Ghigo et dont le username est "GHIGO" :
TO : EXOS%"GHIGO@ESOMC0"

Le destinataire se trouve connecté à un autre réseau

Certains réseaux qui n'ont pas une liaison avec l'ordinateur ESOMC1 sont cependant accessibles via une passerelle ou "Gateway" sur EARN. Citons comme exemple JANET, EUNET ou encore INTERNET. Bien qu'il soit donc possible d'envoyer un message sur ces réseaux, on constate que la syntaxe de l'adresse est alors inextricable. En effet, voyons cela à l'aide d'exemples:

Envoi d'un message au CERN dont le nom de noeud sur le réseau EUNET est "CERNVAX" pour l'utilisateur Durant ayant le username "DURANT" :
TO : EARN::"DURANT@CERNVAX.UUCP"

envoi d'un message au noeud "UCBSSL" du réseau INTERNET pour l'utilisateur Gay :
TO : EARN::"GAY@UCBSSL.ARPA"

C.2 MAILX

Dans ce deuxième cas, l'utilisateur travaille sur le BULL SPS7 (ESOMC3). L'interface de courrier électronique implémentée sur cette machine est le MAILX. Ici, l'adresse doit être indiquée à la suite de la commande d'envoi d'un message. Conformément à l'installation existante, nous pouvons distinguer deux cas:

le destinataire se trouve sur le même ordinateur

La syntaxe de l'adresse sera alors:
MAIL username

Exemple: envoi d'un message à l'utilisateur RICHMOND qui travaille sur le même ordinateur que moi et dont le nom username est "RICHMOND" :
TO : RICHMOND

le destinataire se trouve connecté au réseau local de l'ESO

La syntaxe de l'adresse sera alors:
MAIL username@hostname

Exemple: envoi d'un message à l'ordinateur ESOMC0 à partir de l'ordinateur ESOMC3 pour l'utilisateur Ghigo et dont le username est "GHIGO" :
MAIL GHIGO@ESOMC0

Enfin, ajoutons deux derniers exemples qui s'ajouteront aux précédents lorsque ESOMC3 fera partie du réseau EUNET et EARN:

le destinataire se trouve connecté au réseau EUNET
La syntaxe de l'adresse sera alors:
MAIL hostname!username
ou bien
MAIL username@hostname.UUCP

Exemple: envoi d'un message au CERN dont le nom de noeud sur le réseau EUNET est "CERNVAX" pour l'utilisateur Dupont ayant le username "DUPONT":
MAIL CERNVAX!DUPONT
ou bien
MAIL DUPONT@CERNVAX.UUCP

le destinataire se trouve connecté au réseau EARN
La syntaxe de l'adresse sera alors:
MAIL username@hostname.BITNET
ou bien
MAIL username@hostname.EARN

Exemple: envoi d'un message au CERN dont le nom de noeud sur le réseau EARN est "CERNVAX" pour l'utilisateur Lambert ayant le username "LAMBERT":
MAIL LAMBERT@CERNVAX.BITNET
ou bien
MAIL LAMBERT@CERNVAX.EARN

D. Absence de répertoire

Pour terminer, nous pouvons encore attirer l'attention sur le problème de la gestion des adresses. Il n'existe actuellement à l'ESO aucun répertoire ou bottin consultable par les utilisateurs pour trouver l'adresse d'un destinataire. Pour pallier à cet inconvénient, la plupart des utilisateurs gardent dans des folders les messages reçus dans le but de conserver les adresses de leurs correspondants.

Au CERN, un tel répertoire existe et peut se consulter de tout point du réseau local. Un projet a débuté pour intégrer ce service dans l'interface de courrier électronique EAN.

2.2.4 Conclusion

En guise de conclusion, nous pouvons dire que le courrier électronique est un outil puissant et intensément utilisé par le personnel de l'ESO. L'interface de l'utilisateur n'est cependant pas bien adaptée à l'environnement et est à la base de difficultés que rencontrent quotidiennement les utilisateurs. Celles-ci peuvent être résumées en quatre points:

1. L'interface de l'utilisateur est fonction de la machine sur laquelle on travaille.

2. La différence de syntaxe des commandes dans les deux interfaces entraîne une grande confusion et rend l'apprentissage du courrier électronique difficile.

3. Le format d'adressage d'un destinataire est fonction du réseau par lequel va transiter le message. Cette difficulté peut provoquer de nombreux essais avant l'envoi correct du message et donne un aspect négatif au courrier électronique.

   Remarque:
   En ce qui concerne le format d'adressage, on peut encore faire remarquer que répondre à un message ayant transité par plusieurs réseaux avant de parvenir à sa destination finale n'est pas chose aisée. En effet, le format d'adressage correct à donner dans le message de réponse n'est pas le même que celui précisé dans la clause "From:" du message reçu. De ce fait, la commande REPLY du MAIL VMS qui reprend exactement, comme adresse du message à envoyer, l'adresse contenue dans la clause "From:" du message reçu ne fonctionne pas lorsque ce dernier a dû passer par une ou plusieurs passerelles ou "GATEWAYS". Ce problème n'est d'ailleurs pas soluble actuellement car les passerelles n'indiquent pas toutes leurs adresses dans les messages qui transitent par elles et donc l'implémentation d'une commande de réponse fonctionnant aussi avec des passerelles n'est pas possible.

4. L'absence de répertoire des adresses oblige les utilisateurs à un stockage intempestif de leurs messages en vue de garder trace de leurs correspondants.

L'évolution de l'installation existante vers la nouvelle configuration a donc permis de mettre en évidence les problèmes rencontrés par les utilisateurs. Résoudre ces différents problèmes s'est alors ajouté aux spécifications de départ.

2.3 EASY-MAIL

2.3.1 Introduction

C'est dans le cadre de la nouvelle configuration du réseau local de l'ESO que le projet de munir les ordinateurs d'une nouvelle interface de courrier électronique a vu le jour. Ce dernier a pour but de faciliter l'emploi de la messagerie électronique. La nouvelle interface de courrier électronique a été nommé "EASY-MAIL", nom qui sans conteste met en évidence la qualité principale de la messagerie électronique de l'ESO.

2.3.2 Etude d'opportunité

Les spécifications de la nouvelle interface d'utilisation du courrier électronique ont été conçues en vue de résoudre les difficultés rencontrées par les utilisateurs et de permettre ainsi une utilisation aisée de la messagerie électronique par l'ensemble du personnel de l'ESO. Ces difficultés ont fait l'objet de la rubrique 2.2.3 où elles ont été mises en évidence. Pour les résoudre, EASY-MAIL devra avoir les caractéristiques suivantes:

1. Portabilté UNIX-VMS

L'interface devra être identique sur les différentes machines de l'ESO. L'utilisateur sera ainsi libéré des problèmes de comportement différent des interfaces actuelles : MAIL (VMS) et MAILX (UNIX).

2. Répertoire général et individuel

Chaque utilisateur doit pouvoir disposer d'un répertoire individuel qui contiendra les adresses de ses correspondants. Ce répertoire pourra être mis à jour à l'aide de EASY-MAIL ou bien d'un éditeur. L'ESO disposera également d'un répertoire qui sera le répertoire officiel de l'organisation. Celui-ci contiendra toutes les adresses validées de l'organisation c'est-à-dire celles considérées comme correctes car opérationnelles. Ce répertoire sera régulièrement mis à jour en lisant les adresses validées des répertoires individuels, que les utilisateurs demandent d'inclure.

3. Transparence du réseau utilisé

Rappelons que la plupart des utilisateurs de l'ESO sont des scientifiques ayant peu ou pas du tout de formation en informatique et particulièrement en télé-informatique. Il est donc bon de cacher au maximum les informations qui ne sont pas strictement nécessaires à l'utilisateur pour le bon fonctionnement de EASY-MAIL. Il s'agit essentiellement de l'adresse électronique à utiliser pour atteindre le correspondant. Nous avons vu à la rubrique 2.2.3 que le format exact de celle-ci varie en fonction du réseau par lequel va transiter le message. Rendre transparent à l'utilisateur les adresses électroniques permettra du même coup de cacher le réseau sur lequel sera envoyé le message. Ceci, en effet, n'intéresse pas l'utilisateur car il ne donne de l'importance qu'au fait que son message arrive sans encombre au destinataire spécifié.

En plus de ces caractéristiques permettant de résoudre les difficultés rencontrées, nous avons également voulu que les commandes de EASY-MAIL soient définies et construites d'une manière à apparaître claires dans leurs actions et simples à utiliser. Dans ce but, il nous est apparu essentiel de leurs donner les qualités suivantes:

4. Nombre limité de commandes

Un défaut des interfaces traditionnelles de courrier électronique pour un utilisateur débutant ou non-informaticien est le grand choix de commandes possibles. Nous en avons dénombré une cinquantaine pour MAILX et une quarantaine pour MAIL. Ce grand nombre de commandes rend l'apprentissage et l'utilisation d'un courrier électronique difficile, l'utilisateur ayant beaucoup trop de commandes à mémoriser. Une quinzaine de commandes nous est apparu comme étant un bon compromis entre un nombre raisonnable de commandes et un outil devant quand même offrir des services variés.

5. Simplicité des noms des commandes

Le nom d'une commande doit bien illustrer le service que celle-ci offre à l'utilisateur. Cette qualité permet une mémorisation plus facile des noms et un affichage de menus plus clairs.

6. Nombre limité de paramètres à une commande

Les commandes auquelles peuvent s'appliquer un grand nombre de paramètres nous sont apparues comme étant contraires à un emploi facile de l'outil. Nous avons donc pris comme option de limiter à quatre le nombre maximum de paramètres relatifs à une commande.

Enfin, il nous reste à souligner une dernière caractéristique à laquelle une bonne interface de courrier électronique ne peut échapper:

7. Diversité des services

Notre nouvelle interface de courrier électronique qui doit, rappelons le, être simple et facile à l'emploi, ne doit pas pour autant être un outil moins compétent en offrant beaucoup moins de services. Il est important de garder certaines notions des interfaces traditionnelles comme par exemple la gestion de folders, l'écriture d'un message dans un fichier ou bien encore son impression sur papier.

Chapitre 3 : Etude comparative des réseaux

3.1 Nécessité d'une comparaison

Une seule et simple interface pour l'envoi et la réception des messages, dans laquelle les réseaux utilisés sont transparents pour l'utilisateur, prive celui-ci de particularités et services intéressants et propres à chacun des réseaux. En effet, l'interface unique réduit les possibilités des réseaux à un commun dénominateur. Une comparaison des différents protocoles utilisés s'est donc avérée être intéressante à faire.

3.2 Services de protocoles et d'interfaces

Une distinction importante entre les services de courrier électronique qui sont offerts à un utilisateur s'impose. Celui-ci, pour utiliser le réseau informatique mis à sa disposition, va dialoguer avec la machine par l'intermédiaire d'un logiciel appelé interface de l'utilisateur. Il est appelé ainsi parce qu'il gère la présentation des écrans et joue le rôle d'intermédiaire entre l'homme et la machine. C'est ce programme qui permettra à l'utilisateur par exemple de rédiger un message et de demander son envoi à un destinataire précisé. Le message sera alors envoyé sur le réseau par la machine à l'aide du protocole adéquat. Il nous faut bien différencier les services offerts par le protocole et ceux offerts par l'interface même si cette distinction peut rester invisible à l'utilisateur. Un exemple typique est le service de notification de livraison et celui de stockage et recherche d'un message dans un folder, le premier étant un service de protocole et le second un service d'interface.

Le tableau 1 nous montre le lien qui existe entre les interfaces et les protocoles et cela pour le système VMS et UNIX. Les noms des softwares implémentant les protocoles sont également précisés.

Tableau 1 : lien entre protocoles et interfaces.

PROTOCOLES	UNIX		VMS	
	INTERFACE	SOFTWARE IMPLEMENTANT LE PROTOCOLE	INTERFACE	SOFTWARE IMPLEMENTANT LE PROTOCOLE
DECNET	_	-	MAIL	DECNET
X400	EAN (BSD4.2)	EAN	MAIL	inclus dans VMS
RSCS	MAILX	UREP (Penn state university)	MAIL NJE emulator	NJE emulator + JNET
SMTP	MAILX	SMTP	MAIL	EXOS 8040 (Excelan)
PSI-MAIL	-	-	MAIL	PSI-MAIL
UUCP	MAILX	STANDARD BSD4.2	-	-

3.3 X400

3.3.1 Introduction

Il y a actuellement beaucoup de systèmes qui fournissent des services de messagerie électronique. Tous ces systèmes ont été conçus et développés indépendament les uns des autres et sans aucune collaboration. Il est devenu nécessaire de concevoir des standards dans le but d'intégrer tous ces systèmes dans un système unique de messagerie. Parmi ces standards, on peut citer X400 qui a été rédigé par le CCITT (Comité Consultatif International pour la Téléphonie et la Télégraphie). Le but du protocole X400 est de permettre l'échange de messages entre utilisateurs. Bien que ce travail n'est pas une étude appronfondie de X400, il faut quand même étudier le fonctionnement de X400 afin de mieux comprendre ses services.

3.3.2 Le modèle fonctionnel

3.3.2.1 Le MHS (Message Handling System)

L'ensemble des entités matérielles et logicielles respectant les protocoles spécifiés et reliées entre elles par le réseau forment le système de messagerie ou MHS (Message Handling System). Il y a deux entités dans le MHS: le "user agent" (UA) et le "message transfert agent" (MTA). Ces deux entités peuvent être séparées physiquement ou regroupées au sein d'un même système. Cependant, toutes les fonctions d'une entité sont toujours implémentées sur un même système. Une vue fonctionnelle du modèle MHS est présentée à la figure 10.

Un MTA en collaboration avec d'autres MTA fournit aux utilisateurs les services de transfert de messages. L'ensemble des MTAs est appelé service de transfert de messages ou MTS (Message Transfer Services).

Un UA est un processus qui interagit avec le MTS. Les UAs sont groupés en classe suivant le type de contenu du message qu'ils peuvent traiter. Les UAs d'une classe donnée sont appelés UAs coopérants puisqu'ils coopèrent les uns avec les autres pour améliorer la communication entre leurs utilisateurs respectifs.

Enfin, l'utilisateur est une personne ou un processus d'application. Il échange des messages avec d'autres utilisateurs.

Figure 10 : Vue fonctionnelle du modèle MHS

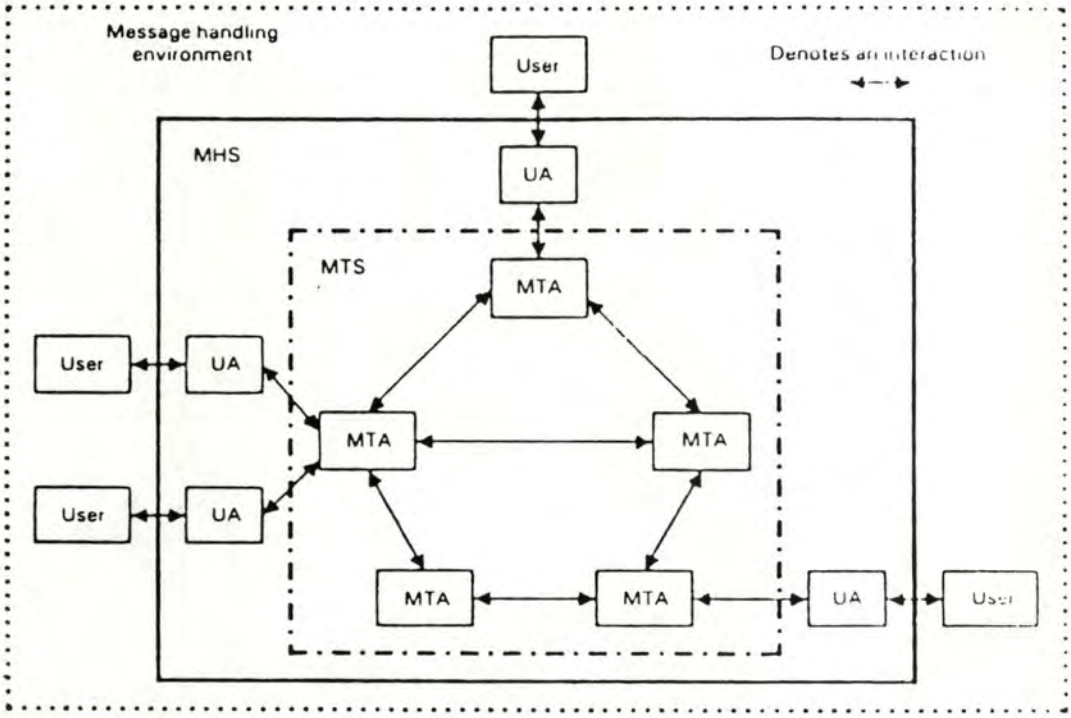
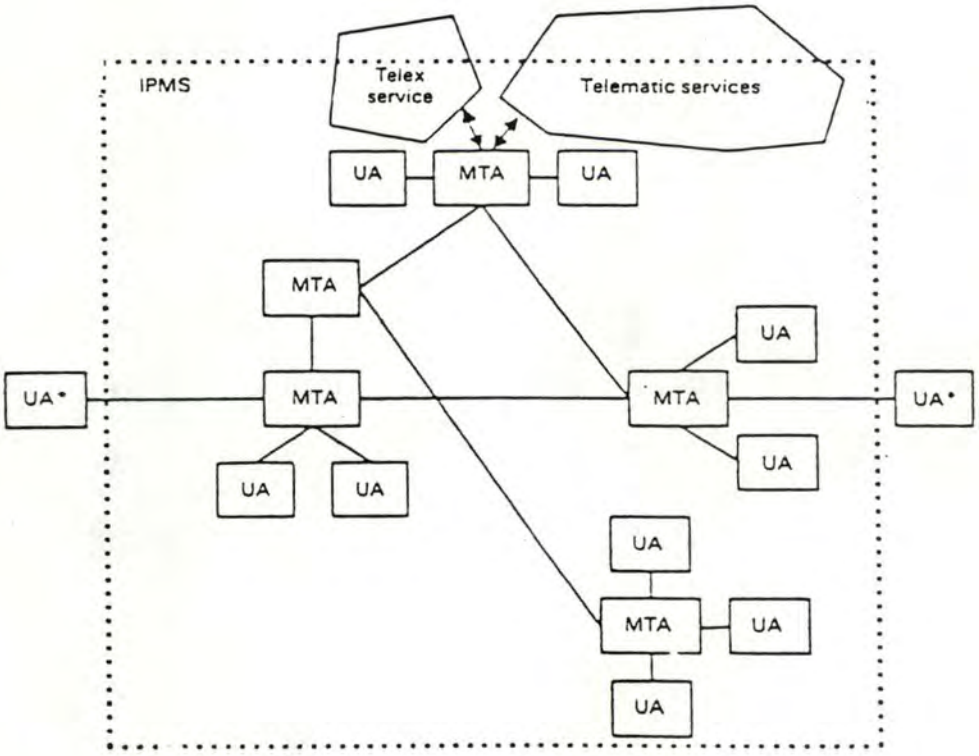


Figure 11 : l'IPMS



3.3.2.2 L'IPMS

L'IPMS (InterPersonnal Messaging System) est l'ensemble des UAs et MTAs coopérants pour l'échange de messages interpersonnels. Il offre à l'utilisateur des services pour l'assister dans sa communication avec un autre utilisateur. Ces services sont cités à la rubrique 3.3.6.5. L'IPMS comprend le MTS, une classe spécifique de UAs coopérants appelés les IPM UAs et l'accès au télex et à d'autres services télématiques. Les utilisateurs de l'IPMS sont typiquement des personnes. L'IPMS est schématisé par la figure 11.

3.3.2.3 Aspects physiques

La disposition des UAs et MTAs peut être diverse:

UA et MTA co-résidents

Le UA et MTA sont implantés comme un ensemble de processus d'application coopérant sur un même système. Il n'y a ici aucun protocole défini et standardisé puisque les interactions sont situées sur un même système. Cette configuration physique est celle utilisée par la plupart des messageries basées sur des ordinateurs et disponible le plus souvent actuellement.

UA seul (Stand-alone UA)

Dans ce cas, les deux entités UA et MTA sont implémentées sur deux systèmes séparés avec peut-être des caractéristiques différentes. Dans le but d'être capable de fournir des services de soumission et de livraison des messages, un protocole spécifique a été défini. Ce protocole s'intitule "Submission and delivery protocol" et l'entité qui l'implémente se nomme SDE (Submission Delivery Entity).

3.3.2.4 Aspects organisationnels

Les UAs et MTAs peuvent être la propriété d'organisations privées ou d'administrations publiques:

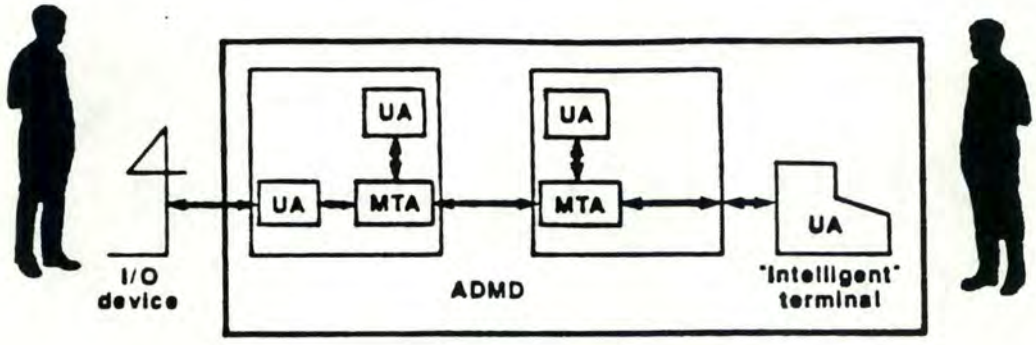
Un MD (Management Domain) est un ensemble consistant d'au moins un MTA et de zéro ou plusieurs UAs. Il appartient à une organisation ou à une administration. Le MD dirigé par une organisation est appelé un PRMD (PRivate Management Domain). Le MD dirigé par une administration est appelé un ADMD (ADministration Management Domain).

L'accès au ADMD peut être réalisé par ses utilisateurs d'une des manières suivantes:

Utilisateur et un UA fourni par une administration

Deux cas sont possibles. Soit le souscripteur possède seulement un terminal I/O et interagit avec l'UA fourni par l'administration via un dialogue homme-machine. Soit l'administration peut fournir un terminal "stand-alone" contenant les fonctions de l'UA. Ce cas est illustré par la figure 12.

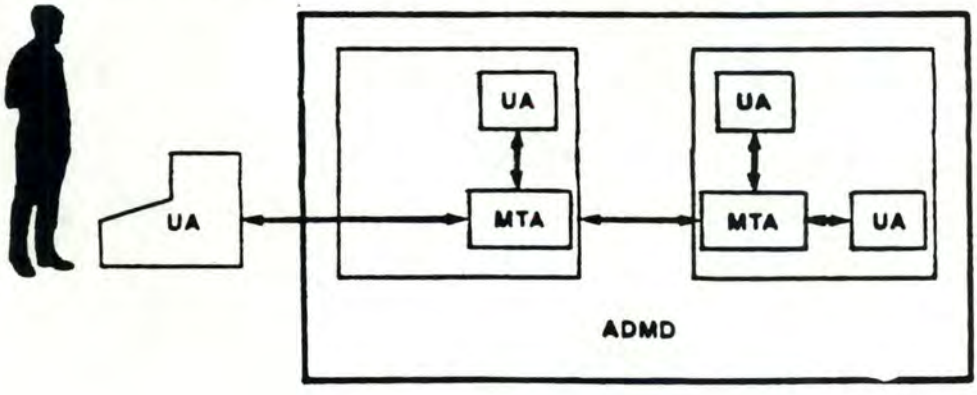
Figure 12 : utilisateur et un UA fourni par une administration



UA privé à MTA d'une administration

Dans ce cas, le souscripteur possède un terminal UA stand-alone. L'UA interagit avec le MTA fourni par l'administration en utilisant les procédures établies pour obtenir le service de transfert de messages. Cet UA ne fait donc pas partie du MD mais est associé à celui-ci. Ce deuxième cas est illustré par la figure 13.

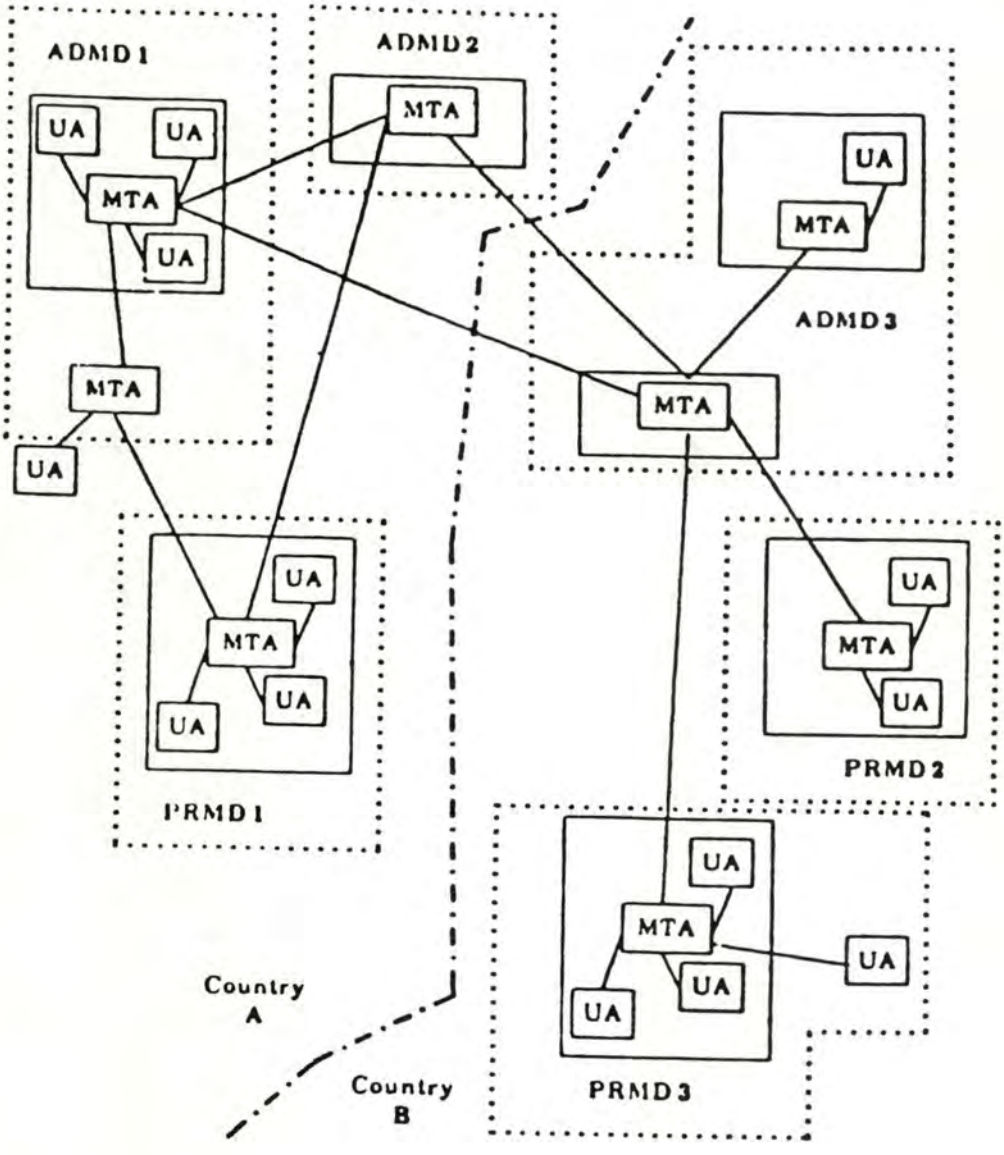
Figure 13 : UA privé à MTA d'une administration



MTA privé à une MTA d'une administration

Dans ce cas, le MTA souscripteur et le UA forment un PRMD qui peut interagir avec le ADMD sur une base MD à MD (MTA à MTA). Un PRMD ne peut exister qu'à l'intérieur d'un et un seul pays. A l'intérieur d'un pays, un PRMD peut avoir accès à un ou plusieurs ADMD. Un ADMD peut agir comme relai entre deux PRMD tandis qu'un PRMD ne peut agir comme relai entre deux ADMD. Un exemple est donné par la figure 14.

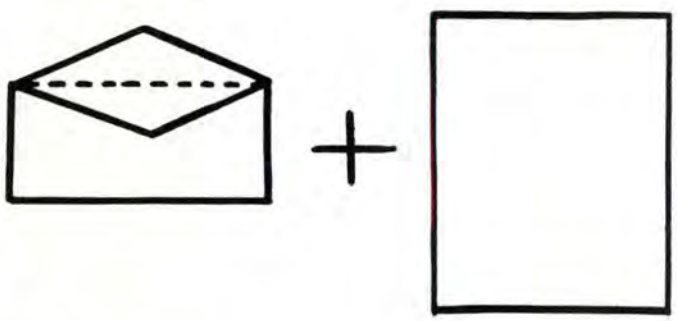
Figure 14 : MTA privé à un MTA d'une administration.



3.3.3 Modèle du message

Un message est divisé en deux parties logiques: l'enveloppe et le contenu. L'enveloppe contient des informations utilisées lors du transfert du message. Le contenu est la pièce d'information que l'UA expéditeur désire envoyer à un ou plusieurs UAs destinataires. La figure 15 illustre cette notion. On peut remarquer qu'il y a correspondance entre le modèle du message et le modèle du MHS: la partie enveloppe est utilisée par le MTS tandis que la partie contenu est crée et utilisée uniquement par les utilisateurs et les UAs.

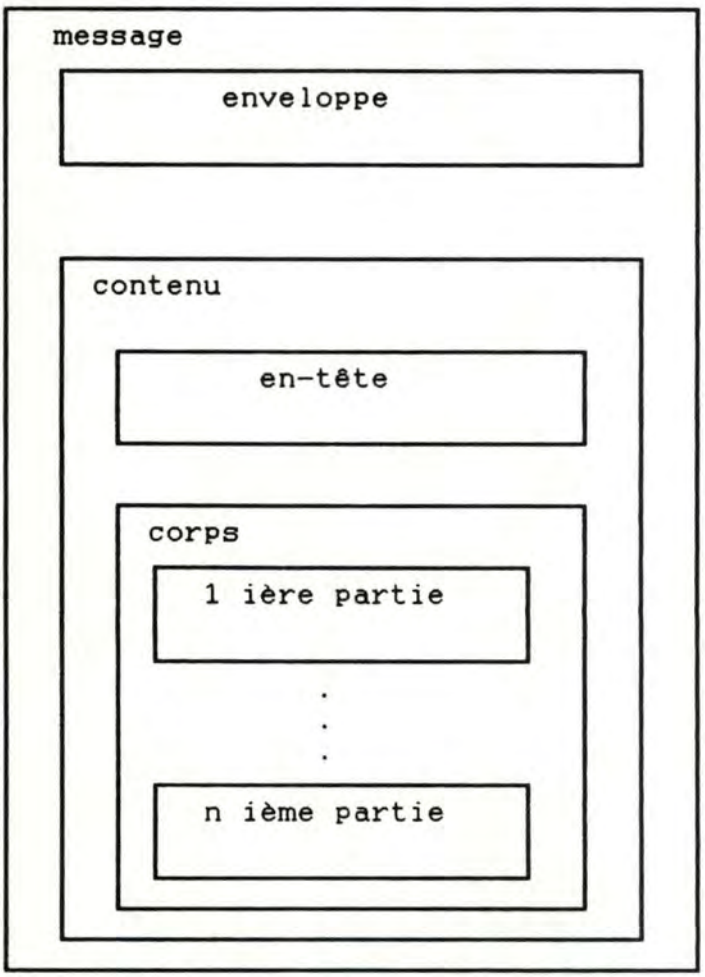
Figure 15 : l'enveloppe et le contenu du message



Enveloppe Contenu

Lorsqu'un UA de la classe des IPM UAs crée un message, son contenu est construit selon le modèle présenté à la figure 16. Le contenu du message possède en effet un en-tête et un corps. Dans ce cas on dira que le message est un IP-message. L'en-tête construit par l'IPM UA contient des informations données par l'utilisateur comme par exemple les destinataires primaires et secondaires ou la date d'expiration du message. Ces informations contenues dans l'en-tête du message permettent d'offrir un sous-ensemble des services de X400 qu'on intitule en juste cause les services de l'IPMS. Ils seront décrits à la section 3.3.6.5. Le corps contient l'information que l'utilisateur désire communiquer à son correspondant. Il peut être subdivisé en plusieurs parties, chaque partie reprenant des informations d'un type particulier (texte, graphique, etc...)..

Figure 16 : Structure d'un IP-message



3.3.4 Identification et adressage

3.3.4.1 Définitions préliminaires

Les noms primitifs

Un nom primitif est donné à une entité spécifique par une autorité d'un MD désignée à cet effet. Cette autorité constitue en fait une réserve de noms qu'elle attribue. Elle ne peut attribuer deux fois le même nom. Cependant il faut bien voir que deux autorités différentes peuvent attribuer le même nom primitif. De ce fait, un seul nom primitif n'est pas suffisant pour assurer une identification d'un utilisateur qui pourrait se trouver dans n'importe quel MD.

Les noms descriptifs

Un nom descriptif représente exactement un et un seul UA dans le MHS.

Attributs et liste d'attributs

On a vu qu'un nom descriptif identifie un UA. Il spécifie pour cela un ou plusieurs attributs. Une liste d'attributs sera représentée par l'ensemble des attributs et de leurs valeurs associées à ce nom descriptif.

Ensemble d'attributs de base

Un ensemble d'attributs de base est un ensemble minimum d'attributs dont les valeurs identifient sans ambiguité un MD particulier.

3.3.4.2 Les noms de l'émetteur et du récepteur

Les utilisateurs du MHS appelés émetteurs et récepteurs sont les principales entités qui nécessitent un mécanisme d'identification. En fait, les utilisateurs étant à l'extérieur du système MHS, le nom descriptif n'identifie pas un utilisateur particulier mais bien l'UA associé à cette personne. Ce nom est appelé "O/R name" (Originator/Recipient name) et est constitué d'une liste d'attributs. Pour assurer l'envoi d'un message, l'UA émetteur doit fournir au MTS le nom de l'UA relatif à chaque récepteur auquel le message est destiné. Chaque O/R name possédera deux parties dans sa liste d'attributs : d'une part un ensemble d'attributs de base identifiant son MD et d'autre part une liste d'attributs identifiant un UA à l'intérieur du MD spécifié.

3.3.4.3 Les adresses de l'émetteur et du récepteur

Généralement, une adresse identifie un utilisateur en spécifiant où il se situe géographiquement. On définit une O/R adresse (Originator/Recipient address) comme étant un nom descriptif d'un UA possédant certaines caractéristiques qui aident le MTS à localiser l'endroit où se situe cet UA. D'où toute O/R adresse est un O/R name mais pas le contraire. L'utilisateur émetteur peut donc se servir de l'O/R adresse pour identifier le récepteur auquel il veut envoyer son message. L'avantage de l'O/R adresse est qu'elle évite le recours à un répertoire réduisant de ce fait le coût et le délai de livraison du message. Cependant, les O/R adresses ont une forme plus difficile à retenir pour l'utilisateur émetteur et sont plus suceptibles de changer que les O/R names.

3.3.4.4 Mécanisme de routage

Le principe du routage est que chaque MD situé sur le chemin du message détermine le MTA dans le MD suivant auquel le message devra être envoyé. Le routage est donc progressif: il y a passage d'un MTA à l'autre dans des MDs différents. Ce principe n'est valable que pour le routage entre MDs. On ne considère pas le routage à l'intérieur d'un même MD. Le routage est mené à bien grâce à l'ensemble d'attributs de base figurant dans l'O/R adresse du message. Le MD chargé du routage d'un message examine l'ensemble des attributs de base afin de relayer le message. Le processus continue ainsi jusqu'au moment où on arrive au MD du récepteur. A ce moment, le reste des attributs se trouvant dans l'O/R adresse est interprété afin de permettre le routage jusqu'à l'UA destinataire.

3.3.5 Insertion du MHS dans le modèle OSI

D'un point de vue de standardisation du MHS, la perspective la plus importante est, sans conteste, la communication entre les différentes entités qui ont été décrites ci-dessus. En effet, ce qui doit être modélisée dans un système tel que le MHS, est la communication entre entités plutôt que les comportements internes de ces entités. Les entités actives du MHS (UA, MTA, SDE) sont vues comme occupant la couche application du modèle OSI. De cette façon, le MHS peut utiliser les couches inférieures du modèle. Afin de respecter la nomenclature OSI, nous donnerons respectivement les noms UAE et MTAE aux entités fonctionnelles UA et MTA.

La totalité de l'application MHS se divise en deux sous-couches correspondant à deux fonctions différentes:

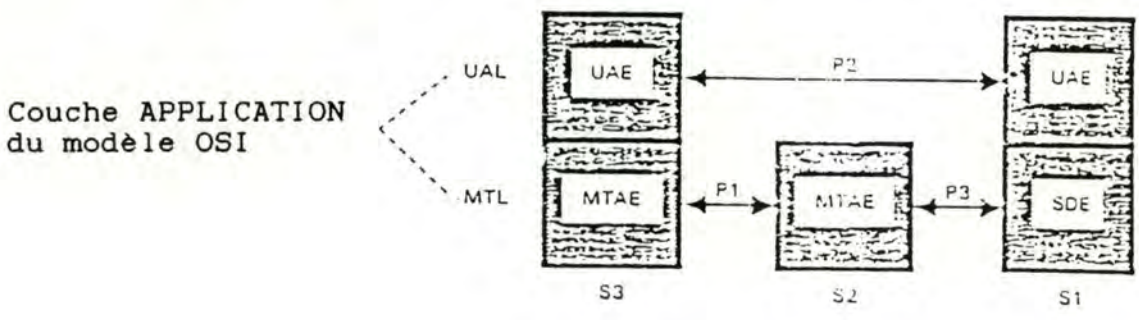
-La sous-couche UAL (User Agent Layer) qui contient les fonctions UA associées à la définition des contenus des messages.

-La sous-couche MTL (Message Transfer Layer) qui contient les fonctions MTA et fournit donc les service de transport des messages.

Comme l'illustre la figure 17, on peut distinguer 3 types de systèmes:

S1: Système ne comportant que les fonctions UA.
S2: Système ne comportant que les fonctions MTA.
S1: Système comportant les fonctions UA et MTA.

Figure 17 : la description en couches du MHS



Puisque ces deux sous-couches sont insérées dans le modèle OSI, elles assurent les services qui leur ont été assignés de la même manière que toute couche de ce modèle c'est-à-dire en utilisant les primitives de services offertes par les couches inférieures et en utilisant des protocoles de dialogue entre deux entités d'une même couche. A ce propos, la figure 17 nous montre les trois protocoles existants: P1, P2 et P3.

Le protocole P1

Le protocole P1 est le protocole de transfert des messages entre les MTAEs. Il définit le relai des messages ainsi que d'autres interactions nécessaires entre deux MTAEs pour fournir les services MTL. L'exécution de P1 entre paires successives de MTAs assure donc le transfert "store and forward" des messages au travers du MTL. Les MTAEs transfèrent les messages et fournissent d'autres services MTL par l'échange de MPDUs (Message Protocol Data Units). Il existe deux types de MPDU:

-Le UMPDU (User MPDU) qui contient un message en provenance d'un UAE (via une primitive de soumission) et destiné à un autre UAE (via une primitive de distribution)

-Le SMPDU (Service MPDU) qui contient une sonde ou un message de notification et est généré par le MTL. Il ne provient donc pas d'un UAE.

Le protocole Pc

Le protocole Pc est le protocole de dialogue entre deux UAEs. Lorsque deux utilisateurs font appel aux services de l'IPMS, c'est une version bien spécifique de Pc qui est utilisée: le protocole P2 (Interpersonal Messaging Protocol). Ce protocole définit la syntaxe et la sémantique des messages que deux UAEs s'échangent. Lorsqu'un message est complètement assemblé par un UAE, il constitue avant soumission au MTL un UM-UAPDU (User Message/User Agent Protocol Data Unit) du protocole P2. Un tel message comprend deux parties: un en-tête et un corps. L'en-tête du message contient des paramètres de soumission et d'autres informations de contrôle. Le corps du message contient le message proprement dit que deux UAEs veulent échanger. Les notifications de réception ou de non-réception d'un message échangé entre deux UAEs sont par contre contenues dans le SR-UAPDU (Status Report/Agent Protocol Data Unit). Après soumission d'un UM-UAPDU ou d'un SR-UAPDU au MTA, ce message devient le contenu d'un message MTL et après addition de l'enveloppe par le MTAE, on obtient le UMPDU du protocole P1

Le protocole P3

Le protocole P3 permet à un UAE éloigné de son MTAE (cas où son système ne comprend que les fonctions de l'UA) d'obtenir les services du MTL et ce via un SDE. Il s'agit donc d'un protocole permettant la transmission à distance d'invocations de primitives, de confirmations, d'indications et de réponses entre un UAE et son MTAE éloigné. L'interface entre un UAE et un SDE doit être identique à celle entre un UAE et un MTAE car on ne peut pas imposer à un UAE de savoir ce qu'il a en-dessous de lui.

Le protocole P3 est un protocole d'opération à distance c'est-à-dire un protocole permettant de définir la même interface que celle existant entre un UAE et un MTAE non éloigné (invocations de primitives, confirmations, indications et réponses). Pour permettre cela, le protocole d'opération à distance définit des OPDUs (Operation Protocol Data Units) contenant des informations telles que le code de l'opération (ou code d'erreur) et les arguments de l'opération.

3.3.6 Description fonctionnelle des services

Nous allons voir maintenant les services offerts par X400. Ils sont décomposés en cinq grandes classes. Chaque service est numéroté. Cette numérotation permettra de donner les références de ces services dans la section 3.5.

3.3.6.1 Les services de base

(X1) Modalité d'accès

L'ouverture d'une connexion de dialogue entre UA et MTA est nécessaire lorsqu'un UA veut utiliser les services de son MTA. Notons cependant que cette connexion n'intervient pas lorsque le UA et MTA sont co-résidents c'est-à-dire implémentés comme un ensemble de processus d'application coopérants sur le même système. Ce service permet à l'UA et son MTA de s'identifier. Il permet également à l'UA de spécifier son "O/R name" et de maintenir une sécurité d'accès à l'aide de mots de passe qui peuvent être régulièrement mis à jour.

(X2) Type de contenu d'un message

Il est nécessaire que l'UA d'origine précise le type de formation du contenu du message bien qu'il n'y ait actuellement qu'une seule convention de formation possible. Ce service a été conçu pour le jour où on pourra utiliser un autre protocole que le protocole P2. Rappelons que ce protocole définit la syntaxe et la sémantique des messages que deux UAEs s'échangent.

(X3) Indication de conversion

Lorsqu'une conversion est intervenue par le MTS sur les types de l'information contenue dans le message, celles-ci ainsi que les nouveaux types sont sont signalés à l'UA destinataire.

(X4) Date et heure de livraison

La date et heure de livraison du message à l'UA destinataire lui sont signalées par le MTS.

(X5) Identification d'un message

Un identificateur unique est attribué à chaque message. Cet identificateur sert au MTS et UAs pour référencer un message dans des services tel que, par exemple, une notification de livraison ou de non livraison.

(X6) Notification de non-livraison

Lorsqu'un message n'a pas pu être livré aux UA(s) destinataire(s), l'UA d'origine en est averti par une notification de non-livraison. Celle-ci contiendra un justificatif. Ce service est automatique mais un utilisateur a toujours la possibilité de demander à ne pas recevoir ces notifications. Dans le cas d'un message envoyé vers plusieurs destinataires, la notification de non-livraison référencera tous les UA destinataires auquels le message n'a pu être livré.

(X7) Types d'origine

Les types d'information initialement utilisés par l'UA d'origine dans un message sont signalés au MTS. Lorsqu'un message est livré, le MTS indique à l'UA destinataire quels étaient les types d'information d'origine.

(X8) Limitation des types d'information acceptable

Un UA a la possibilité en informant le MTS de restreindre les types d'information qui peuvent lui être envoyés.

(X9) Date et heure de soumission

La date et heure auquelles le message a été soumis au MTS peuvent être signalées à l'UA d'origine et l'UA destinataire.

3.3.6.2 Services de soumission et de livraison

(X10) Permission d'un destinataire alternatif

Un UA d'origine peut spécifier qu'un message soumis au MTS soit livré à un UA destinataire alternatif par défaut dans le cas où les attributs donnés correspondent bien à un MD mais à aucun UA de ce MD. Dans le cas où le MD supporte ce service, le message sera alors livré à un UA spécifique du MD chargé de recevoir de tels messages et qui les traitera d'une façon manuelle.

(X11) Livraison différée

Un UA d'origine peut demander au MTS le retardement d'une livraison d'un message jusqu'à une date et heure spécifiées.

(X12) Interdire une notification de non-livraison

L'UA d'origine peut informer le MTS de ne pas lui retourner une notification de non-livraison dans le cas où le message soumis au MTS ne sait pas être livré.

(X13) Annulation de livraison différée

Il s'agit de la suppression du service X11. Un UA informe le MTS de supprimer une demande de livraison différée qui a été acceptée. Notons que ce service n'est pas toujours possible notamment dans le cas où la date et heure de livraison qui ont été spécifiées sont expirées ou que le message a déjà été expédié dans le MTS.

(X14) Notification de livraison

Un UA d'origine peut demander à recevoir une notification de livraison pour confirmer que le message qu'il envoie a bien été livré aux UA(s) destinataire(s). En cas d'un message envoyé vers plusieurs destinataires, la notification de livraison référencera tous les UAs destinataires auquels le message a pu être livré.

(X15) Divulgation des autres destinataires

Lors d'un envoi d'un message à plusieurs UAs destinataires, l'UA d'origine peut donner au MTS l'autorisation de mentionner à chaque UA destinataire l'identité des autres UAs destinataires. Dans ce cas, chacun de ceux-ci devront être informés de la liste des "O/R names" de tous les UAs auquels est adressé le message.

(X16) Degré de livraison

Un UA d'origine peut préciser un degré d'urgence pour son message soumis au MTS. Les valeurs possibles sont "urgent", "normal" ou "non-urgent". Le temps de livraison du message dépendra du degré d'urgence donné au message.

(X17) Livraison à plusieurs destinataires

Un UA d'origine peut préciser que le message soumis au MTS doit être livré à plusieurs UA destinataires. Le nombre d'UA destinataires est illimité tandis qu'une livraison simultanée à tous les UA destinataires n'est pas garantie.

(X18) Retour du message

En cas de non-livraison d'un message, celui-ci peut être retourné à l'UA d'origine dans une notification de non-livraison. Ce service n'est pas possible si des conversions ont été effectuées sur le contenu du message pendant son parcours.

(X19) Interdiction de conversion

L'UA d'origine informe le MTS que des conversions sur le type d'information d'un message particulier ne pourront se faire.

(X20) Conversion implicite

Ce service permet à un UA de bénéficier d'une conversion sur le type d'information des messages par le MTS vers un type plus approprié. Si plusieurs conversions sont possibles, le MTS choisira celle la plus appropriée. L'UA destinataire sera informé des types originals de l'information du message ainsi que des types actuels.

(X21) Conversion explicite

Un UA d'origine peut demander explicitement une conversion sur le type d'information du message comme c'est requis pour la collaboration de différents services télématiques. L'UA destinataire sera informé des types originals de l'information du message ainsi que des types actuels.

3.3.6.3 Service de test

(X22) Sonde

Ce service permet de tester la possibilité d'envoyer un message à un ou plusieurs UAs destinataires en soumettant cette demande au MTS. L'UA demandeur est informé du résultat du test par une notification de livraison ou de non-livraison. Le service de la sonde inclut la possibilité de vérifier si la taille du message ou le type des informations du message le rendrait non livrable. Le résultat de la sonde dépend du fait que l'UA destinataire a informé ou non le MTS des types d'information et de la taille maximum des messages qu'il peut accepter. Le degré de livraison de la sonde est "urgent".

3.3.6.4 Services d'état et d'information

(X23) Création d'un destinataire alternatif

Ce service permet à un UA responsable d'un MD de recevoir les messages dont le "O/R name" n'a pas été correctement spécifié. Ce service permet le traitement des messages dont l'UA destinataire a mal été spécifié. Lors de la livraison d'un message à l'UA alternatif, une notification de livraison à cet UA est généré si l'UA émetteur en a fait la demande.

(X24) Garde des messages

L'UA peut demander à son MTA de garder les messages et les notifications qui lui sont destinés pendant une période de temps.

3.3.6.5 Services de l'IPMS

(X25) Identificateur du message

Un numéro d'identification est généré pour chaque IP-message. Cet identificateur peut être utilisé par l'utilisateur pour référencer un ancien message reçu ou envoyé.

(X26) Type des informations

Les types des informations contenues dans l'IP-message sont spécifiés.

(X27) Destinataire(s) caché(s) ("blind copy")

Ce service permet à l'utilisateur de donner une liste de destinataires supplémentaires pour l'envoi de l'IP-message. Ces destinataires ne seront pas connus par les autres c'est-à-dire par les destinataires primaires et secondaires. Savoir si chacun de ces destinataires cachés auront ou n'auront pas connaissance des autres destinataires cachés est une mesure locale.

(X28) Notification de non-réception

A la demande de l'utilisateur d'origine, lorsque l'IP-message n'a pu être réceptionné par l'utilisateur destinataire, une notification de non-réception lui est retournée.

(X29) Notification de réception

L'utilisateur d'origine peut demander à être informé de la réception de l'IP-message par l'utilisateur destinataire à l'aide d'une notification de réception.

(X30) Indication de détournement ("auto-forward")

Ce service permet à un utilisateur de savoir si un IP-message reçu a été détourné c'est-à-dire que l'IP-message était initiallement destiné à une autre personne. Le détournement d'un IP-message donne lieu à une notification de non-réception si l'utilisateur émetteur en a fait la demande.

(X31) Indication de l'origine

L'identité de l'utilisateur d'origine est indiqué au destinataire. Il ne s'agit pas de l' "O/R name" de son UA mais simplement d'une identification non formelle de l'utilisateur.

(X32) Indication des utilisateurs autorisant l'envoi (authorizing users indication)

L'utilisateur d'origine peut indiquer dans l' IP-message le nom des personnes ayant autorisé son envoi.

(X33) Destinataires primaires et secondaires (copy)

Dans la liste des destinataires de l'IP-message à envoyer, l'utilisateur d'origine peut faire la distinction entre les primaires et les secondaires. Les premiers étant les vrais destinataires auquels l'IP-message s'adresse, les deuxièmes ne recevant l'IP-message qu'à titre documentaire.

(X34) Date d'expiration de l'IP-message

Ce service permet à l'utilisateur d'origine de spécifier la date à partir de laquelle l'IP-message qu'il envoie n'est plus valide.

(X35) Référenciation à d'autres IP-messages

L'utilisateur désireux d'envoyer un message peut donner une liste d'identificateurs d'autres IP-messages à titre de référence.

(X36) Indication de l'importance de l'IP-message

L'utilisateur peut donner un degré d'importance à son IP-message. Les valeurs possibles sont "bas", "normal" ou "haut". Cette indication n'influence cependant pas le comportement du système.

(X37) Indication de validité de IP-messages antérieurs (obsoleting indication)

L'utilisateur d'origine peut préciser les IP-messages rendus non-valables par l'envoi d'un nouvel IP-message. L'action résultant de la réception de cet IP-message est une mesure locale.

(X38) Indication de confidentialité

L'utilisateur d'origine peut donner un degré de confidentialité à son IP-message. Trois degrés sont prévus: "personnel" (=normal), "privé" (=confidentiel) ou "confidentiel" (=ultra-confidentiel). Ce degré de confidentialité pourrait par exemple:

- -obliger l'utilisateur à prouver son identité
- -interdire l'impression du IP-message sur papier
- -interdire à l'IPM UA de permettre à l'utilisateur de renvoyer son IP-message.
- -interdire que l'IP-message soit détourné ("auto-forward")

(X39) Sujet de l'IP-message

L'utilisateur d'origine peut indiquer au destinataire le sujet de l'IP-message lors de l'envoi de celui-ci.

(X40) Indication de réponse à un autre IP-message

L'utilisateur qui répond à un IP-message peut donner la référence de l'IP-message auquel il répond.

(X41) Demande de réponse

L'utilisateur peut préciser dans son IP-message qu'il demande une réponse. Il peut également préciser la date à laquelle la réponse devra être envoyée et les "O/R names" des destinataires qui devront également la recevoir. Cependant, le destinataire reste libre de ne pas le faire.

(X42) Message réenvoyé ("forward")

Ce service permet de réenvoyer un IP-message reçu en placant son contenu dans le corps d'un nouvel IP-message. Lorsque le corps de celui-ci est subdivisé en parties, on peut y inclure d'autres informations comme par exemple les informations de livraison de l'IP-message à réenvoyer. Ces informations sont les informations données à l'IPM UA par le MTS lorsque celui-ci livre l'IP-message. Il s'agit par exemple de la date et heure de livraison, une indication de conversion, etc...

(X43) Indication de cryptographie

Signale qu'une partie du corps est codé. On peut préciser la nature du codage.

(X44) Corps subdivisé en parties

Il est possible de subdiviser le corps de l'IP-message en plusieurs parties distinctes. Lorsqu'on doit envoyer dans un même message des informations de plusieurs types différents, on les disposera dans des parties distinctes. Le type de l'information contenue dans chacune de ces parties sera précisé par le service X26.

3.4 Standard RFC#822

3.4.1 Introduction

Jusqu'à 1977, le réseau INTERNET utilisait plusieurs standards non formels pour l'envoi de ses messages. De cette expérience est né le standard RFC#733 qui reprenait les caractéristiques des services offerts jusque là. Ce standard a été remplacé par le standard RFC#822 dans le but d'offrir les services d'un réseau plus grand et plus complexe. Certaines caractéristiques de l'ancien standard ont été abandonnées pour simplification du software. Un nouveau modèle d'adressage a été conçu afin de prévoir une messagerie inter-réseaux tandis que le concept de retransmission a vu le jour. Notons encore que par rapport à X400, RFC#822 décrit seulement le format et la sémantique du contenu du message. Il ne fait d'ailleurs pas de distinction comme le fait X400 entre l'enveloppe et le contenu du message. Ce nouveau standard est indépendant de l'environnement du réseau INTERNET pour lequel il a vu le jour car il ne définit pas le système de transport des messages. Il peut donc être utilisé pour d'autres réseaux. On peut d'ailleurs considérer les systèmes traditionnels de messagerie électronique comme offrant chacun un sous-ensemble des services de RFC#822 tout en gardant leurs propres systèmes de transport. Cette caractéristique fait d'ailleurs l'objet de la section 3.6.

3.4.2 Description fonctionnelle des services

Le contenu d'un message a été décrit dans le langage BNF (Backus Naur Form). On peut déduire de cette description les services offerts. Chaque service a été numéroté pour savoir s'y référer dans les sections 3.5 et 3.6.

(R1) Date et heure de création

La date et l'heure de création du message sont précisées dans le message lui-même.

(R2) Routage du message

L'utilisateur peut s'il le désire, donner la liste des noeuds par lesquels le message devra obligatoirement transiter.

(R3) Adresse de l'émetteur ("sender")

L'adresse de l'émetteur peut être indiquée dans le cas où l'émetteur n'est pas la même personne que celle responsable de l'envoi du message. Il s'agit donc ici par exemple de l'adresse de la secrétaire ayant envoyé le message.

(R4) Adresses des responsables de l'envoi ("from")

Les adresses des personnes qui décident l'envoi du message doivent être spécifiées pour permettre une réponse à ceux-ci. Cette clause est obligatoire. Il se peut que le responsable de l'envoi du message soit aussi l'émetteur du message. Lorsque le message est envoyé par une personne qui n'est pas à proprement parlé responsable de son envoi (par exemple une secrétaire), l'adresse de celle-ci sera alors spécifiée par le service R3.

(R5) Adresse pour la réponse

Ce service permet de préciser dans un message les adresses où la réponse devra être envoyée lorsque celle-ci n'est pas destinée aux auteurs du message.

(R6) Destinataires primaires et secondaires

Dans la liste des destinataires du message à envoyer, on peut faire la distinction entre les primaires et les secondaires. Les premiers étant les vrais destinataires auquels le message s'adresse, les deuxièmes ne recevant le message qu'à titre documentaire.

(R7) Destinataires cachés

Ce service permet à l'utilisateur de donner une liste de destinataires supplémentaires pour l'envoi du message. Ces destinataires ne seront pas connus par les autres. Savoir si chacun de ces destinataires cachés auront ou pas connaissance des autres destinataires cachés est une mesure locale.

(R8) Identificateur unique du message

Un identificateur unique est attribué à chaque message. Référencer un message est possible grace à cet identificateur.

(R9) Indication de réponse à un autre message

L'utilisateur qui répond à un message peut donner la référence du message auquel il répond.

(R10) Indication d'autres références

L'utilisateur désireux d'envoyer un message peut donner une liste d'identificateurs d'autres messages à titre de référence.

(R11) Indication de mots clés ou de phrases

Il est possible de donner une liste de mots clés et/ou de phrases séparés par des virgules.

(R12) Sujet du message

Indiquer le sujet du message est possible.

(R13) Commentaires sur le message

On peut ajouter des commentaires au message. Ce service permet de ne pas modifier le contenu du message en lui-même.

(R14) Indication de cryptographie

Signale qu'une partie du corps est codé. On peut préciser la nature du codage.

(R15) Champs pour usages individuels

L'utilisateur est libre de définir et d'utiliser des champs supplémentaires.

(R16) Information de trace

Le chemin parcouru par le message peut être consulté.

(R17) Date et heure de réception

La date et heure de réception du message à chaque noeud du réseau (aussi appelé unité de transport) sont précisées dans le message. Remarquons que la terminologie n'est pas identique à celle de X400 puisqu'ici réception du message ne veut pas dire prise en charge du message par l'utilisateur.

(R18) Livraison à plusieurs destinataires

Un message soumis au système de transport peut être envoyé à plusieurs destinataires.

(R19) Message réenvoyé ("forward")

La norme permet de réenvoyer un message reçu vers un nouvel destinataire. Il est en effet possible d'ajouter dans le contenu du message l'adresse d'un nouvel émetteur et destinataire.

(R20) Divulgation des autres destinataires

Lorsqu'un message est envoyé vers plusieurs destinataires, chacun de ceux-ci aura connaissance de la liste de tous les autres destinataires auquels est adressé le message.

3.5 Comparaison des services de X400 et RFC#822

3.5.1 Introduction

Après avoir vu X400 et RFC#822, il nous a semblé qu'une comparaison de ces deux standards permettrait de faire la synthèse des services étudiés.

La façon de procéder a été celle-ci: inspiré directement des services offerts par X400 et RFC#822 nous avons construit une liste de services qui allient les avantages des deux standards. Cette liste nous a alors permis de mettre en évidence les différences entre les deux standards. Il est très difficile de pouvoir affirmer qu' un service de X400 équivaut exactement à un service de RFC#822. En effet, la syntaxe mais surtout la sémantique même d'un service peut varier d'un standard à l'autre. C'est pourquoi il nous est apparu judicieux de faire suivre parfois une comparaison par un commentaire explicitant certaines divergences intervenant entre les deux standards.

C'est ainsi qu'on peut déjà faire la remarque suivante: puisque RFC#822 ne décrit , à première vue, que le format et la sémantique du contenu d'un message il est normal et correct de ne comparer ses services qu'avec ceux de l'IPMS de X400 car les autres services de X400 n'utilisent, en effet, que des informations de l'enveloppe du message. Cependant, nous nous sommes efforcés de comparer les services de RFC#822 à l'ensemble complet des services de X400 car certains services de RFC#822 se retrouvent chez X400 comme des services utilisant l'enveloppe. Nous pensons notamment ici à "la date et heure de livraison" (service X4 de X400) qui peut être rapproché de "date et heure de réception" (service R17 de RFC#822). Remarquons encore que le service R16 de RFC#822 "information de trace" est typiquement un service d'enveloppe. De plus, le système de transport utilisant le format de message défini par RFC#822 peut également offrir des services plus ou moins similaire à X400.

3.5.2 Comparaison

(C1) Indication de conversion

X400: X3 RFC#822: néant

(C2) Date et heure de livraison

X400: X4 RFC#822: R17

Pour X400, la date et heure de livraison du message à l'UA destinataire lui sont signalées par le MTS. Pour ce faire, ces informations sont écrites sur l'enveloppe du message avant la livraison à l'UA. Pour RFC#822, on peut considérer dans une certaine mesure que la date et heure de livraison du message de X400 correspond à la date et heure de réception du message à la dernière unité de transport intervenant dans le parcours du message c'est-à-dire le noeud destinataire.

(C3) Identification d'un message

X400: X5 RFC#822: néant

Chez RFC#822, il existe bien, en effet, un identificateur de message mais nous avons jugé plus aproprié de le comparer au service X25 plutôt qu'au service X5 de X400. Cela fait d'ailleurs l'objet de la comparaison C23. En effet le service X5 utilise l'enveloppe du message tandis que le service X25 utilise, lui, le contenu du message.

(C4) Notification de non-livraison

X400: X6 RFC#822: néant

RFC#822 ne parle pas des notifications de non-livraison. Cependant, parfois, un utilisateur travaillant avec un système de courrier électronique utilisant un réseau particulier, sait savoir si son message a été livré; notamment, avec les réseaux SPAN (PSI-DECNET), DATEX-P (PSI MAIL), EUNET (UUCP+X25) et le réseau local de l'ESO (SMTP+ETHERNET), on peut mettre en évidence la situation suivante:

Avec le réseau SPAN

Après la soumission du message, on peut distinguer les cas suivants:

1) affichage immédiat du message "remote node unknow" Dans ce cas le nom du noeud donné pour l'adresse du destinataire est incorrect. On ne sait rien dire sur la validité du username donné.

2) affichage immédiat du message "no such user". On peut ici affirmer que le nom du noeud donné dans l'adresse est correct mais le username est incorrect.

3) affichage immédiat du message "not currently reachabel". On ne peut dans ce cas rien affirmer. Il faut attendre et réessayer l'envoi du message plus tard.

4) aucun affichage immédiat de message. Dans ce dernier cas, on peut affirmer que l'adresse du destinataire a été correctement spécifiée et que le message a bien été livré.

Avec le réseau DATEX-P (PSI MAIL)

Après la soumission du message, on peut distinguer les cas suivants:

1) affichage immédiat du message "no such user". On peut ici affirmer que le username est incorrect.

2) affichage immédiat du message "remote node unknow" Dans ce cas le numéro de dte-address donné est incorrect. On ne sait rien dire sur la validité du username donné.

3) affichage immédiat du message "error creating network link". On ne peut dans ce cas rien affirmer; il faut attendre et réessayer l'envoi du message plus tard. remarquons qu'il existe d'autres messages faisant référence à ce troisième cas.

4) aucun affichage immédiat de message.
Dans ce dernier cas, on peut affirmer que l'adresse du destinataire a été correctement spécifiée et que le message a bien été livré.

Avec le réseau local de l'ESO

Après la soumission du message, si le nodename est incorrect, on aura un affichage immédiat du message "remote node unknow". De plus, le message envoyé est retourné à l'émetteur en cas de nodename ou de username incorrect.

Avec le réseau EUNET

Après la soumission du message, on ne peut affirmer que le message sera bien livré au destinataire demandé. Seule la réception d'une notification de non-livraison du message dans un délai de temps non précisé permettra de dire que l'adresse spécifiée n'était pas correcte et que le message n'a donc pas été livré. Le message est en effet retourné dans une notification de non-livraison lorsque le nodename ou le username est incorrect.

Avec le réseau EARN

Après la soumission du message, on ne peut affirmer que le message sera bien livré au destinataire demandé. Seule la réception d'une notification de non-livraison du message dans un délai de temps non précisé permettra de dire que l'adresse spécifiée n'était pas correcte et que le message n'a donc pas été livré. Le message est en effet retourné dans une notification de non-livraison lorsque le nodename est incorrect. De plus, si l'émetteur n'a pas reçu une notification de non-livraison du message plusieurs heures après l'envoi, cela ne signifie pas que le message a été livré car si le username a mal été spécifié, il peut y avoir livraison du message à un autre destinataire chargé de recevoir de tel message.

(C5) Types d'origine

X400: X7 RFC#822: néant

(C6) Limitation des types acceptables d'information

X400: X8 RFC#822: néant

(C7) Date et heure de soumission/création

X400: X9 RFC#822: R1

En ce qui concerne X400, la date et l'heure de soumission du message au MTS sont inscrites sur l'enveloppe du message ce qui permettra à l'UA d'en avoir connaissance. Pour RFC#822, la date et heure de création du message font partie de son contenu. Il ne s'agit pas de la même implémentation. L'information elle-même traitée par les deux normes n'est pas tout à fait la même puisque l'information date et l'heure de création du message ne veut pas dire que c'est à ce moment qu'il a été soumis au système de transport. On peut considérer dans une certaine mesure que la date et l'heure de soumission du message de X400 correspondent à la date et l'heure de réception du message à la première unité de transport intervenant dans le parcours du message c'est-à-dire le noeud de départ.

(C8) Permission d'un destinataire alternatif

X400: X10 RFC#822: néant

Bien que RFC#822 ne précise rien à ce sujet, le protocole RSCS dans sa fonction de transport du message offre une certaine similitude avec le service X10 de X400. En effet sur EARN, lorsqu'un message arrive bien au noeud destinataire, mais a son username mal spécifié, le message est livré chez un utilisateur bien précisé qui est chargé de recevoir de tel message et de les traiter d'une manière manuelle. Généralement, il s'agira du manager du système.

(C9) Livraison différée

X400: X11 RFC#822: néant

(C10) Interdire une notification de non-livraison

X400: X12 RFC#822: néant

(C11) Annulation de livraison différée

X400: X13 RFC#822: néant

(C12) Notification de livraison

X400: X14 RFC#822: néant

(C13) Divulgation des autres destinataires

X400: X15 RFC#822: R20

Alors que dans X400 la divulgation des autres destinataires peut ou ne peut pas se faire, celle-ci ne peut que se faire dans RFC#822. D'autre part, le service X15 utilise l'enveloppe du message tandis que RFC#822 le contenu.

(C14) Degré de livraison

X400: X16 RFC#822: néant

(C15) Livraison à plusieurs destinataires

X400: X17 RFC#822: R18

Il s'agit dans ce cas du même service. Notons cependant un différence essentielle : les destinataires sont inscrits sur l'enveloppe pour X400 tandis qu'ils sont inscrits dans le contenu du message pour RFC#822.

(C16) Retour du message

X400: X18 RFC#822: néant

Pour RFC#822, rien n'a été prévu à cet effet. Cela dépend essentiellement de la fonction de transport utilisé pour transmettre le message. Ainsi, dans le réseau EUNET, une notification de non-livraison (réception au sens RFC#822) retournée à l'utilisateur émetteur contient le message envoyé précédemment et qui est à la base de la notification.

(C17) Interdiction de conversion

X400: X19 RFC#822: néant

(C18) Conversion implicite

X400: X20 RFC#822: néant

(C19) Conversion explicite

X400: X21 RFC#822: néant

(C20) Test d'adressage

X400: X22 RFC#822: néant

Il n'est pas possible pour RFC#822 d'utiliser une sonde comme c'est le cas pour X400. RFC#822 ne précise d'ailleurs rien à ce sujet. Notons cependant que dans ce dernier cas, l'utilisateur peut toujours envoyer un message bidon et interpréter les résultats de l'envoi comme cela a été décrit dans la comparaison C13.

(C21) Création d'un destinataire alternatif

X400: X23 RFC#822: néant

RFC#822 ne précise rien à ce sujet. Rappelons cependant comme on l'a déjà dit dans la comparaison C8 que, sur EARN, les messages arrivant au noeud destinataire avec un username incorrect, sont dirigés vers un utilisateur spécifique de ce noeud que sera chargé de les traiter.

(C22) Garde des messages

X400: X24 RFC#822: néant

(C23) Identificateur du message

X400: X25 RFC#822: R8

Il y a dans les deux services correspondant aux deux normes une grande similitude. Ils exécutent en effet exactement la même fonction quelque soit la norme et l'information nécessaire à leurs fonctionnements se trouve toujours dans le contenu du message.

(C24) Type des informations

X400: X26 RFC#822: néant

(C25) Destinataire(s) caché(s)

X400: X27 RFC#822: R7

Il s'agit dans ce cas du même service. Notons cependant un différence essentielle : les destinataires cachés sont inscrits sur l'enveloppe pour X400 tandis qu'ils sont inscrits dans le contenu du message pour RFC#822.

(C26) Notification de non-réception

X400: X28 RFC#822: néant

Pour RFC#822, rien n'a été prévu à cet effet. Cela dépend essentiellement de la fonction de transport utilisé pour transmettre le message. En ce qui concerne les protocoles DECNET, PSI-MAIL, SMTP, RSCS et UUCP, ils ne permettent pas d'envoyer une notification de non-réception telle qu'elle est définie pour X400.

(C27) Notification de réception

X400: X29 RFC#822: néant

(C28) Indication de détournement

X400: X30 RFC#822: néant

(C29) Indication non formelle de l'origine

X400: X31 RFC#822: néant

(C30) Indication des responsables de l'envoi

X400: X32 RFC#822: R4

Bien que ce soit ici deux services utilisant le contenu du message, ils sont différents. Pour X400, il ne s'agit que d'une indication non formelle des utilisateurs autorisant l'envoi du message. Cette indication ne permet pas de leur envoyer une réponse. En ce qui concerne RFC#822, l'indication concerne les adresses des responsables de l'envoi du message. Ici, une réponse est possible.

(C31) Destinataires primaires et secondaires

X400: X33 RFC#822: R6

Il y a dans les deux services correspondant aux deux normes une grande similitude. Ils exécutent en effet exactement la même fonction quelque soit la norme et l'information nécessaire à leurs fonctionnements se trouve toujours dans le contenu du message.

(C32) Date d'expiration du message

X400: X34 RFC#822: néant

(C33) Référenciation à d'autres messages

X400: X35 RFC#822: R10

Il y a dans les deux services correspondant aux deux normes une grande similitude. Ils exécutent en effet exactement la même fonction quelque soit la norme et l'information nécessaire à leurs fonctionnements se trouve toujours dans le contenu du message.

(C34) Indication de l'importance du message

X400: X36 RFC#822: néant

(C35) Indication de validité de messages antérieurs

X400: X37 RFC#822: néant

(C36) Indication de confidentialité

X400: X38 RFC#822: néant

(C37) Sujet du message

X400: X39 RFC#822: R12

Il y a dans les deux services correspondant aux deux normes une grande similitude. Ils exécutent en effet exactement la même fonction quelque soit la norme et l'information nécessaire à leurs fonctionnements se trouve toujours dans le contenu du message.

(C38) Indication de réponse à un autre message

X400: X40 RFC#822: R9

Il y a dans les deux services correspondant aux deux normes une grande similitude. Ils exécutent en effet exactement la même fonction quelque soit la norme et l'information nécessaire à leurs fonctionnements se trouve toujours dans le contenu du message.

(C39) Demande de réponse

X400: X41 RFC#822: néant

Tandis que X400 permet d'indiquer explicitement dans le contenu du message une demande de réponse, RFC#822 ne le permet pas bien qu'il donne la possibilité de donner les adresses des destinataires à qui un message de réponse éventuel devra être adressé.

(C40) Message réenvoyé ("forward")

X400: X42 RFC#822: R19

Bien que le service en lui-même soit identique pour les deux normes, l'implémentation en est différente. Avec X400, le contenu du message à réenvoyer est placé dans le corps d'un nouveau message. Avec RFC#822, on ajoute dans le contenu du message l'adresse du nouvel émetteur et destinataire avant de le réexpédier.

(C41) Indication de cryptographie

X400: X43 RFC#822: R14

Il y a dans les deux services correspondant aux deux normes une grande similitude. Ils exécutent en effet exactement la même fonction quelque soit la norme et l'information nécessaire à leurs fonctionnements se trouve toujours dans le contenu du message.

(C42) Corps subdivisé en parties

X400: X44 RFC#822: néant

(C43) Routage du message

X400: néant RFC#822: R2

(C44) Adresse de l'émetteur

X400: X31 RFC#822: R3

En ce qui concerne X400, il ne s'agit ici que d'une identification non formelle de l'utilisateur émetteur du message. Pour RFC#822 par contre, le service R3 permet de préciser l'adresse de l'émetteur ainsi qu'un identificateur non formel d'un utilisateur ou d'un programme.

(C45) Adresses pour la réponse

X400: X41 RFC#822: R4 etR5

Il est possible de préciser dans le message les adresses des personnes à qui une réponse pourrait être envoyée. Ces adresses se trouveront dans le contenu du message dans les deux normes.

(C46) Indication de mots clé ou de phrases

X400: néant RFC#822: R11

(C47) Commentaires sur le message

X400: néant RFC#822: R13

(C48) Champs pour usage individuel

X400: néant RFC#822: R15

(C49) Information de trace

X400: néant RFC#822: R16

3.6 Lien entre RFC#822 et les réseaux

3.6.1 Introduction

Nous avons tenté dans cette section 3.6 de comparer les services offerts par RFC#822 à ceux offerts par différents réseaux : SPAN, EARN, DATEX-P, EUNET. Nous avons pris pour cela la même organisation de courrier électronique que celle de l'ESO c'est-à-dire:

pour SPAN : MAIL VMS + DECNET
pour EARN : MAIL VMS + RSCS
pour DATEX-P : MAIL VMS + PSI MAIL
pour EUNET : MAILX UNIX + UUCP

La comparaison que nous allons effectuer a deux limites:

1) RFC#822 ne décrivant que le contenu du message et non l'enveloppe, nous devons nous limiter à ne prendre en considération dans les services des réseaux que les services liés au contenu du message. Ainsi un service de notification de livraison comme il existe par exemple sur EARN ne pourra pas intervenir dans la comparaison.

2) L'interface de messagerie électronique restreint souvent le nombre de services disponibles. C'est ainsi que, bien qu'EUNET offre la possibilté du service "forward", il n'est pas possible à un utilisateur travaillant avec MAILX de bénéficier de ce service, l'interface ne le permettant pas.

3.6.2 Comparaison de protocoles au standard RFC#822

Pour manifester une grande clarté dans la comparaison, nous avons utilisé le tableau 2. La liste des services offerts vient directement de la rubrique 3.4.2. Pour une description plus détaillée d'un service, veuillez donc vous référer à cette rubrique.

Tableau 2 : Comparaison des protocoles au standard RFC#822
(Légende : O = Oui, N = Non)

SERVICES OFFERTS PAR LE STANDARD RFC#822	MAIL+DECNET	MAIL+-RSCS	PSI+MAIL	MAILX+UUCP
(R1) Date et heure de création	O	O	O	N
(R2) Routage du message	N	N	N	O
(R3) Adresse de l'émetteur	O	O	O	O
(R4) Adresses des responsable de l'envoi	N	N	N	O
(R5) Adresse pour la réponse	N	N	N	O
(R6) Destinataires primaires & secondaires	N	N	N	O
(R7) Destinataires cachés	N	N	N	O
(R8) Identificateur unique du message	N	N	N	O
(R9) Indication de réponse à un autre message	N	N	N	N
(R10) Indication d'autres références	N	N	N	N
(R11) Indication de mots clés ou phrases	N	N	N	N
(R12) Sujet du message	O	O	O	O
(R13) Commentaires sur le message	N	N	N	N
(R14) Indication de cryptographie	N	N	N	N
(R15) Champs pour usages individuels	N	N	N	N
(R16) Information de trace	N	N	N	N
(R17) Date et heure de réception	N	N	N	N
(R18) Livraison à plusieurs destinataires	O	O	O	O
(R19) Message réenvoyé ("forward")	O	O	O	N
(R20) Divulgation des autres destinataires	O	O	O	O

3.6.3 Conclusions

Après examen du tableau 2, on constate que SPAN, EARN et DATEX-P offrent les mêmes services. Cela est tout à fait normal puisque nous avons utilisé pour les trois la même interface de courrier électronique (le MAIL VMS) et que nous avons omis les services d'enveloppe. Le MAIL VMS offre d'ailleurs vis-à-vis de RFC#822 un nombre très limité de services puisque ceux-ci sont au nombre de six. En ce qui concerne EUNET, on remarque que le MAILX UNIX offre déja un nombre plus varié de services. En conclusion, on peut affirmer que les systèmes de messagerie électronique utilisant le MAIL VMS et le MAILX UNIX sur les réseaux SPAN (DECNET), EARN (RSCS), DATEX-P (PSI MAIL) et EUNET (UUCP) offrent chacun un nombre assez limité de services équivalant à ceux de RFC#822.

Chapitre 4 : Spécification de EASY-MAIL

4.1 Démarche utilisée

4.1.1 construction du guide de l'utilisateur

La démarche utilisée pour écrire les spécifications de EASY-MAIL est la suivante :
Nous avons pensé que le guide d'utilisation de EASY-MAIL pourrait servir également de spécification pour le logiciel à construire. En effet, le guide d'utilisation se doit de décrire d'une façon précise la spécification de chaque commande offerte par l'interface de courrier électronique. La meilleure façon d'obtenir un outil adapté à des utilisateurs non-informaticiens était certainement de rédiger le guide avec leur collaboration. De cette façon, nous avons pu construire un outil vraiment adapté à leurs besoins. Nous avons donc écrit, dans un premier jet, une première version du guide dans lequel apparaissaient les concepts mis en évidence dans l'analyse d'opportunité (rubrique 2.3.2). Pour rappel, il s'agissait des notions de :

-portabilité UNIX-VMS
-répertoire général et individuel
-transparence du réseau utilisé
-nombre limité de commandes
-simplicité des noms des commandes
-nombre limité de paramètres à une commande
-diversité des services offerts.

Par trois fois, le guide a été soumis aux critiques des utilisateurs et modifié en fonction des remarques qui nous ont été retournées. Cette manière de travailler nous a permis de voir :

-que l'idée de rendre transparentes aux utilisateurs les adresses électroniques et les réseaux apparaissait comme très attrayante.

-qu'il était souhaitable:

  -d'expliquer d'une façon plus précise l'usage des commandes
  -de choisir des noms de commandes mieux appropriés
  -de compléter les spécifications de commandes jugées trop élémentaires.
  -de diversifier davantage les services offerts par notre interface
  -de mieux préciser l'idée de répertoire

4.1.2 Prototypage

Pour en arriver à un outil parfaitement adapté à des utilisateurs non-informaticiens il est apparu comme nécessaire de compléter les spécifications concernant la sémantique du logiciel par une présentation du comportement de l'interface. Cette idée a donné lieu à la construction d'un prototype. Afin de faciliter cette construction, nous avons choisi d'utiliser un gestionnaire de fenêtre développé à l'ESO : Termwindow. Cependant, notre choix s'est vite porté sur un autre outil d'interface plus puissant dans la mesure où il nous offrirait d'autres services que la gestion de fenêtres: Protéus qui fait partie de l'ensemble des SGDs (Système de Gestion de Dialogue)

4.2 Guide de l'utilisateur de EASY-MAIL

Nous vous présentons maintenant dans les pages suivantes la dernière version du guide de l'utilisateur de EASY-MAIL.

Contents

1 Introduction : First Screen

WELCOME
TO
EASY_MAIL
Version 0.7

The main purpose of the development of **EASY_MAIL** is to get a single user interface for all mailing purposes and protocols.

The different networks that are covered by the first version of this program are as follow:

- **Local Ethernet**: with TCP/IP protocol
- **EARN/BITNET**: with RJE protocol
- **SPAN**: under DECNET protocol
- **VAX-PSI world**: under PSI_MAIL
- **Gatewaying**: as currently exists with EARN only.

Type **Help** to list the commands.

2 Prompts : Question - and - Answer Dialogue

The program puts prompts on the line which is 2nd from the bottom on the screen. If the program detects an error, e.g. the response to a prompt was invalid, then an error message (*not* magic words or numbers) is displayed beneath the prompt, which is repeated; another response may be given.

Some prompts have *default values;* pressing **CReturn** causes this value to be taken as the response. Default values are indicated in the prompt by underlining:

Topic : commands

— if the response to this prompt is **CReturn** then the available commands are listed.

When it is known what the sequence of prompts will be, and the replies to be given, they can be typed-ahead on one input line, terminated by **CReturn**, as the response to the first prompt in the sequence. The replies may be separated by one or more spaces (which count as one). The subsequent prompts will not appear, unless an error is detected; in this case, the corresponding prompt is displayed, the rest of the input line is discarded, and you may enter a valid response.

A command selects a particular function required of the program. For example, the command **Help** causes the program to provide information on how to use the program. A command is selected by answering the prompt

Command : Help

with its name, or an *abbreviation;* that is, only enough first characters of the command name are needed, to distinguish it from other commands in the same program. Also, you may enter parameters on the same line as the command, instead of being prompted for them.

3 Commands : A List of Commands

Command name	Purpose
AddressValidate	Validate address.
AppendMailbook	Append a new recipient in the Mailbook.
DeleteMessage	Delete message.
DirMessage	Display the list of the received messages.
Exit	Exit this program.
FileWrite	Write message without mail header.
ForwardMessage	Forward message to another user.
Help	Display detailled help.
Menu	Display menu of commands.
Msend	Send to multiple recipients.
PrintMessage	Print a message to a device.
ReadMessage	Read message.
ReplyMessage	Reply to message.
Send	Send message.
Spawn	Pass control to DCL or Shell.
WriteMessage	Write message to a file.
$*command*	Immediate execution of *command*

You may enter one of the above command names, or an abbreviation. If you know the prompts that follow a command, then you may also enter your responses on the same input line.

3.1 AddressValidate : Validate an address

Syntax: **Addressvalidate** *[recipient [.network]]*

- *Recipient* may be the surname of the person whose address you want to validate, or a "#" followed by an institute name.
- *Network* is the network name of which you want validate the E_Mail address. It must be preceded by a ".". The network name may be replaced by a "+" character, which means the faster address or by a "?" in which case the user will be given the choice among the different Electronic Mail (E_mail) addresses found. If the specification of the network is omitted, the default value is "+" i.e. the faster E_mail address is choosen.

This command only changes the status of the flags of an E_mail address saying that this E mail address is correct.

If no parameters are given, the user is prompted for the professional address. If more than one recipient corresponding to the professional address is found, they are displayed on the screen and the user is prompted for a choice. If the recipient possesses more than one E_mail address with the specified network, they are all displayed on the screen and the user is prompted for a choice.

A procedure regularly collects all the personnal "mailbooks" of users and updates the ESO's central "mailbook" with the new validated addresses.

3.2 AppendMailbook : append a new recipient in the Mailbook

Syntax: **AppendMailbook**

When this command is called, the user is prompted for a new professional address and its E_Mail address(es). Theses new data are written in the user's "mailbook".

3.3 DeleteMessage : Delete a received message

Syntax: **DeleteMessage** *[number]*

- *number* is the number of the message that you want to delete.

This command deletes a received message from the "old" folder. If no number is given, the user is prompted for it.

3.4 DirMessage : List of received messages

Syntax: **DirMessage** *[folder]*

- *folder* is the name of a folder. The two folder names that you can use are "old" and "new". The first is for the messages that were already readen. When you receive new mail message, it automatically enter in the "new" folder. The default value is "new".

This command lists the received messages of a folder. If no folder name is given then the "new" folder is used. If the used folder contains no mails, then the message "The folder is empty" is displayed on the screen.

3.5 Exit : Exit Easy_Mail

Syntax: **Exit** or **CTRL/z**

This command terminates an Easy_Mail session, clearing the screen except for the 'title' line, and returning you to the host operating system user command level (e.g. DCL).

CTRL/Z (Control key + z at the same time: ^z) is equivalent to EXIT.

3.6 FileWrite : Write a received message without header to a file

Syntax: **FileWrite** *[number] [@filename]*

- *number* is the number of the message that you want to write. Its default value is the last new message.
- *filename* is the file name in wich you want to write the message. It must be preceded by a "@".

This command allows writing a received message from the "old" folder to a file, without the mail header. If no file name is given, the user is prompted for it.

3.7 ForwardMessage : Forward a received message to another recipient

Syntax: **ForwardMessage** *[number] [list]*

- *number* is the number of the message that you want to forward. Its default value is the new read message.
- *list* is a set of [recipient [.network]] and/or [@filename].
  The separation character is the space.
  - *Recipient* may be the surname of the person you want to reach, or a "#" followed by an institute name.
  - *Network* is the network name on which you want the message to be sent. It must be preceded by a ".". The network name may be replaced by a "+" character, which means the faster address or by a "?" in which case the user will be given the choice among the different Electronic Mail (E mail) addresses found. If the specification of the network is omitted, the default value is "+" i.e. the faster E_mail address is choosen.
  - *filename* is the name of a "distribution list" file.

This command allows the forwarding of a received message from the old folder to another(s) recipient(s).

If no *list* is given, the user is prompted for it. If more than one recipient corresponding to the professional address is found, they are displayed on the screen and the user is prompted for a choice. If the recipient possesses more than one E_mail address with the specified network, they are all displayed on the screen and the user is prompted for a choice.

3.8 Menu : Display menu of commands

Syntax: **Menu**

Type **Menu** to display the list of commands;press the key for the command code (after "..."), or use arrow keys to move through the menu and press **CReturn** to invoke underlined command.

3.9 Msend : Send a message to multiple recipients

Syntax: **Msend** *[list]*

- *list* is a set of [recipient [.network]] and/or [@filename].
  The separation character is the space.
  - *Recipient* may be the surname of the person you want to reach, or a "#" followed by an institute name.
  - *Network* is the network name on which you want the message to be sent. It must be preceded by a ".". The network name may be replaced by a "+" character, which means the faster address or by a "?" in which case the user will be given the choice among the different Electronic Mail (E_mail) addresses found. If the specification of the network is omitted, the default value is "+" i.e. the faster E_mail address is choosen.

– *filename* is the name of a "distribution list" file.

The same message is sent to a set of people. The set is determined by the list and the "distribution list" file(s). If no *list* is given, the user is prompted for it. The message composition is done as a single send. If a used E mail address is not yet validated, a comment is displayed on the screen to say that the correct transmission of the message is not guaranteed.

3.10 PrintMessage : Print a message to a device

Syntax: **PrintMessage** *[number] [#device]*

- *number* is the number of the message that you want to print. Its default value is the last new message.
- *device* is the number of the device that you want to use to print the message. Its default value is the number "1". The Centronix printer has the device number "1", the laser QMS printer "2", the Diablo printer "3" and the laser Digital printer "4". This parameter must be preceded by a "#".

This sends a message from the "old" folder to the specified *device* defined in the local system. If no *number* is given, the last new message is printed. If no *device* is given, the Centronix printer is used.

3.11 ReadMessage : Read a received message

Syntax: **ReadMessage** *[folder] [number]*

- *folder* is the name of a folder. The two folder names that you can use are "old" and "new". The first is for the messages that were already readen. When you receive new mail message, it automatically enter in the "new" folder. The default value is "new".
- *number* is the number of the message that you want to read. Its default value is the first new message.

This command displays a received message of a specified folder. If no folder name is given the "new" folder is used. If no number is specified the last new message is displayed. After you read a message in the "new" folder, it automatically moves into the "old" folder.

3.12 ReplyMessage : Reply to a received message

Syntax: **Reply** *[number]*

- *number* is the number of the message that you want to reply. Its default value is the last new message.

This command allows replying to a received message from the "old" folder. If no number is given, the last new message is used.

3.13 Send : Send a message

Syntax: **Send** *[recipient [.network] [@filename] [subject]]*

- *Recipient* may be the surname of the person you want to reach, or a "#" followed by an institute name.
- *Network* is the network name on which you want the message to be sent. It must be preceded by a ".". The network name may be replaced by a "+" character, which means the faster address or by a "?" in which case the user will be given the choice among the different Electronic Mail (E_mail) addresses found. If the specification of the network is omitted, the default value is "+" i.e. the faster E_mail address is choosen.
- *filename* is the name of the file that you will send. It must be preceded by a "@".
- *subject* is the subject of your message.

The message is reliably sent with the right protocol. The user is prompted for the professional address if he did not provide *recipient* before. His personal mailbook is queried to find the E_mail address(es) of the recipient. If more than one recipient corresponding to the professional address is found, they are displayed on the screen and the user is prompted for a choice. If the recipient possesses more than one E_mail address with the specified network, they are all displayed on the screen and the user is prompted for a choice.

If the chosen E_mail address is not yet validated, a comment is displayed on the screen to say that the correct transmission of the message is not guaranteed.

If a query for a specified *recipient* is not successful, the user has to introduce the complete E_mail address to send his mail.

If no *filename* is specified the message can be composed online with or without the editor. The user may choose the "mailer editor". The user can abort the command before the send too.

3.14 Spawn : To send commands to operating system

Syntax: **Spawn**

This command creates a new user process, in which DCL (Digital Command Language) or shell commands may be given; it must be terminated with the LOG or EXIT command.

3.15 WriteMessage : Write a received message to a file

Syntax: **WriteMessage** *[number] [@filename]*

- *number* is the number of the message that you want to write. Its default value is the last read message.
- *filename* is the file name in wich you want to write the message. It must be preceded by a "@".

This command allows the writing of a received message from the "old" folder to a file.

If no number is given, the last new message is written. If no file name is given, the user is prompted for it.

Chapitre 5 : Implémentation de EASY-MAIL

5.1 Introduction

Ce chapitre explique comment a été réalisé EASY-MAIL en mettant en évidence les choix qui ont été faits lors de son implémentation.

La section 5.2 explique la méthodologie employée pour réaliser les programmes. On y verra la politique choisie d'une part pour le choix de l'architecture logicielle et d'autre part pour la méthode de construction d'un module.

La dernière section explicite les outils logiciels qui ont été utilisés pour la réalisation du projet. Elle présente Protéus, le Système de Gestion de Dialogue (SGD) avec lequel EASY-MAIL a été construit. Le lecteur désireux d'avoir d'abord une présentation théorique de ce qu'est un SGD et en quoi il peut être avantageux trouvera cette description dans l'annexe I.

5.2 Méthodologie

5.2.1 Choix de l'architecture logicielle

Le logiciel a été développé sous forme de structuration modulaire. Rappelons que un module peut être vu comme une unité de compilation séparée. Le choix d'une telle architecture logicielle a été étroitement lié à la décision d'utiliser un SGD (Système de Gestion de Dialogue) pour faciliter la programmation de la gestion des écrans. Le SGD Protéus impose en effet de découper le logiciel en fonctions avec le critère de découpe suivant : à une commande offerte à l'utilisateur doit correspondre une fonction dont l'execution satisfait la sémantique de la commande. En donnant à chaque fonction un module particulier, on obtient donc une structuration modulaire pour laquelle on peut donner aux modules les qualités suivantes:

informations cachées

Chaque module possède une forte capacité de cacher de l'information. Cette information va du mode de représentation ou de mémorisation d'une donnée jusqu'à la stratégie employée pour construire l'algorithme du module. Cette caractéristique permet qu'une modification dans un module ne provoque pas de changements dans les autres.

faible degré de couplage

Le degré de couplage mesure la complexité de l'interface entre modules. Chaque module n'ayant pas de passage de paramètres, nous pouvons donc dire que le degré de couplage est faible.

forte cohésion

Chaque module possède une forte cohésion c'est-à-dire qu'il réalise le moins possible de fonctions différentes. Dans notre cas, chaque module réalisera une et une seule commande de EASY-MAIL.

5.2.2 Méthode de construction d'un module

Chaque module de EASY-MAIL a été construit par une conception descendante c'est à dire à l'aide du principe de la décomposition en sous-problèmes. Ainsi, un module est constitué d'un ensemble de fonctions, chaque fonction ayant un travail bien précis à réaliser, un sous-problème bien limité à résoudre.

5.3 Choix des outils

5.3.1 Le langage C

EASY-MAIL a été implémenté dans le langage C. Ce langage a été choisi pour son "bas niveau", sa portabilité UNIX - VMS et en fonction des couches inférieures dont les fonctions ont été écrites et définies dans ce même langage.

5.3.2 Le Mail (VMS) et le Mailx (UNIX)

Afin que EASY-MAIL soit terminé dans le délai imparti, il a été décidé qu'il s'appuierait, pour offrir les commandes qui ont été spécifiées, sur un logiciel de courrier électronique. Celui-ci sera fonction de l'environnement sur lequel tournera EASY-MAIL. En ce qui concerne l'environnement VMS, nous avons choisi de prendre le logiciel VAX/VMS Mail de Digital tandis que pour UNIX nous avons sélectionné le logiciel Mailx. Ce choix a été fonction des logiciels de courrier électronique dont disposait l'ESO.

5.3.3 Protéus

5.3.3.1 Introduction

Protéus est le nom d'un software développé à l'ESO dans le cadre du projet STARCAT. Ce projet consiste essentiellement au développement d'une base de données contenant des informations astronomiques. Des milliers d'étoiles y seront cataloguées. Protéus fait partie de la famille des SGDs (Système de Gestion de Dialogue). Il s'agit d'un logiciel récent qui n'avait jusqu'à ce jour encore été utilisé que par son concepteur. L'emploi que j'en ai fait et les entretiens que j'ai eu avec lui ont été très bénéfiques. Ils ont, en effet, permis au concepteur de Protéus d'avoir une autre vision de son produit et cela a abouti à la mise en évidence de changements ou améliorations à apporter. Le choix d'utiliser Protéus pour implémenter EASY-MAIL se justifie par la section suivante. Elle présente en effet les avantages offerts par un tel SGD d'une part à moi, concepteur du programme et d'autre part aux les futurs utilisateurs de EASY-MAIL.

5.3.3.2 Facilités offertes par Protéus

Les facilités offertes par Protéus peuvent être partitionnées en deux classes. D'une part celles dont bénéficiera l'utilisateur et d'autre part celles offertes au concepteur du programme:

1) Principales facilités pour l'utilisateur

Les prompts

Un "prompt" est une chaine de caractères qui pose une question auquel doit répondre l'utilisateur. La réponse sera le prochain champ en entrée donné par celui-ci. Ce mécanisme de prompt permet l'utilisation d'abréviation dans les noms des commandes rentrées au clavier, de bénéficier d'un rapport d'erreurs ou de rediriger l'entrée vers un fichier contenant une liste de commandes à executer, ce qui soulage l'utilisateur de devoir toujours répéter la même séquence de commandes.

Les menus

Les commandes possibles offertes à l'utilisateur peuvent être listées dans un menu. L'utilisateur indique alors son choix en positionant le curseur avec les touches fléchées du clavier sur la commande désirée. La commande ainsi sélectionnée est alors mise en évidence en étant soulignée. Ce système de menus permet de faire le lien entre le nom d'une commande sélectionnée et la fonction associée qui sera executée. Il existe une autre façon de sélectionner une commande. qui consiste à rentrer au clavier le nom de la commande ou son abréviation.

Les formulaires

Un formulaire consiste en un ensemble de champs affichés sur l'écran et qui sont à remplir par l'utilisateur à l'aide du clavier. Chacun de ces champs possède un nom, un type et un commentaire pouvant aider l'utilisateur. A chaque champ peut également être ajouté un écran d'aide auquel l'utilisateur pourrait se référer en cas de besoin.

Le système d'aide

L'utilisateur peut disposer d'un certain nombre d'écrans d'aide. Ceux-ci sont rédigés dans un fichier particulier auquel le programme a accès. En plus des commentaires constituant les écrans d'aide, on y trouvera également des commandes LaTeX. Ces commandes incorporées dans le texte d'aide lui-même permettent une meilleur présentation des informations en offrant le souligné, les caractères gras ou encore une numérotation automatique des différentes sections. Le fichier peut également être imprimé sur papier à l'aide de l'interpréteur LaTeX et constituer ainsi le guide d'utilisation du logiciel. C'est d'ailleurs de cette façon que le guide d'EASY-MAIL a été construit. Pour rappel, celui-ci a été présenté au lecteur à la section 5.2.

2) Principales facilités pour le concepteur

séparation entre l'application et l'interface

La séparation entre le programme d'application et l'interface est un principe fondamental dans le domaine du génie logiciel. Cette séparation permet d'alléger la charge du programme d'application qui ne doit alors plus s'occuper que des tâches propres à la sémantique du logiciel. La gestion des prompts, menus ou formulaires, l'affichage des écrans d'aide ou encore la validation des informations rentrées au clavier sont totalement prises en charge par le programme d'interface.

Modularité

Protéus propose d'associer à chaque commande offerte à l'utilisateur une fonction qui remplirait la sémantique de cette commande. Après l'entrée d'une commande par l'utilisateur, celle-ci est validée par l'interface. Si la commande est correcte, l'exécution de la fonction qui lui est associée est lancée. Cette façon de procéder permet au concepteur de l'application de construire son architecture logicielle sous forme de structuration modulaire en associant à chaque fonction un module.

Prototypage

La séparation entre le programme d'application et l'interface utilisateur donne également la possibilité de faire du prototypage. Le prototypage permet de montrer rapidement à l'utilisateur comment va se comporter le logiciel en terme essentiellement de présentation des écrans et cela même avant que soit développés les différents modules constituant l'application.

Adaptation aisée des anciens programmes

Les programmes plus anciens peuvent bénéficier du nouveau style offert par le mécanisme de prompt de cet outil. L'installation de ce mécanisme dans ces programmes ne nécessite pas une réorganisation de la structure du code existant.

Portabilité

L'interface offerte par Protéus est portable. Il tourne sous l'environnement VMS et UNIX. Utilisant TernWindow, un système de gestion de fenêtres, il est également indépendant vis-à-vis du type de terminaux utilisés.

Chapitre 6 : Conclusions

6.1 Evaluation de EASY-MAIL

6.1.1 Démarche utilisée

L'évaluation de EASY-MAIL a été basée sur la validation du produit logiciel. Cette validation a été constituée de tests en vue de détecter la présence d'erreurs. La méthodologie suivie a été la suivante:

6.1.1.1 Principes de base

Il est d'abord bon de rappeler quelques principes de base utilisés pour la validation :

- -Les tests doivent être construits non pas pour se convaincre que le logiciel est totalement correct mais dans l'optique de trouver un maximum d'erreurs.
- -Chaque module doit être testé individuellement dès la fin de sa programmation.
- -Un module doit être testé par une personne autre que son réalisateur. Notons que bien que ce principe n'a pas été respecté ici puisqu'une seule personne a travaillé sur la réalisation des modules du logiciel, ceux-ci ont quand même également été soumis à la vérification de personnes responsables à l'ESO lors de démonstrations.

6.1.1.2 Critère de couverture utilisé

Le critère de couverture d'une validation d'un logiciel par test est le critère auquel l'ensemble des jeux de tests satisfait. Nous avons choisi de prendre comme critère de couverture le fait de couvrir l'ensemble de toutes les combinaisons possibles des paramètres des commandes et cela pour l'ensemble des commandes de EASY-MAIL.

6.1.1.3 Comparaison des résultats

Avant chaque test, le résultat attendu est explicité et cela en se basant sur les spécifications des commandes de EASY-MAIL réalisées en début de travail et présentées à la section 4.2. Une fois ces résultats explicités, ils sont utilisés dans une confrontation avec les résultats obtenus lors du test. Si des différences apparaissent dans cette comparaison, elles montrent que le module ne réalise pas sa spécification. Ce dernier doit donc être corrigé. Cette procédure peut être répétitive dans la mesure où un module doit être testé après chaque correction intervenue et cela jusqu'au moment où il réalisera enfin correctement sa spécification.

6.1.2 Conclusions

Toutes les commandes spécifiées dans le guide d'utilisation de EASY-MAIL ont été réalisées et terminées à temps conformément au calendrier prévu. Chaque commande a été testée ce qui a permis d'obtenir la fiabilité nécessaire à l'utilisation du logiciel par le grand public.

6.2 Développements futurs

6.2.1 EASY-MAIL et MAILGATE

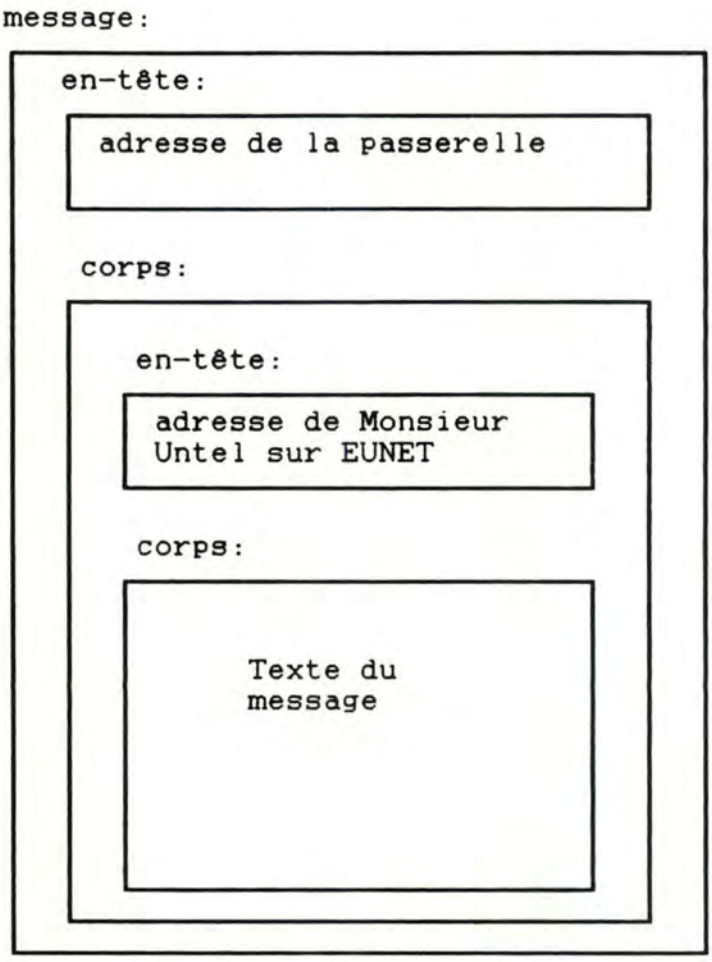
MAILGATE est un programme tournant dans l'environnement VMS. Il s'agit d'une version modifiée du programme SENDGATE qui fonctionne avec le VAX VMS NJE Emulator et le MAIL VMS. La première version de SENDGATE a été écrite par Paul Kunz du centre d'accélérateur linéaire de Standford en Décembre 1983. Le but de MAILGATE est de permettre au MAIL VMS de pouvoir envoyer un message sur un noeud d'un réseau non directement accessible de l'ESO. Le message est initialement envoyé sur EARN et atteindra le réseau souhaité en transitant par une ou plusieurs passerelles.

Nous allons brièvement expliquer la technique utilisée en employant pour ce faire un exemple. Supposons donc un utilisateur, Monsieur Dupont, se trouvant sur ESOMC1 et désireux d'envoyer un message à Monsieur Untel. Celui-ci possède comme username "UNTEL" et se trouve sur le noeud "MCVAX" appartenant au réseau EUNET. Afin d'envoyer son message, Monsieur Dupont utilise donc le MAIL VMS et précise comme adresse électronique:

to: EARN::"UNTEL@MCVAX.UUCP"

Lorsque le message a été rédigé, le MAIL VMS détecte par la chaîne de caractères "EARN::" dans l'adresse électronique qu'il doit passer le message au software RSCS. La chaîne de caractères ".UUCP" indiquant qu'il ne s'agit pas d'un réseau directement accessible, l'adresse ainsi que le GATE. MAILGATE va alors chercher dans ses tables quelle est l'adresse électronique de la passerelle qui correspond à la chaîne de caractères ".UUCP" spécifiée. Il s'agira bien sûr d'une passerelle permettant d'atteindre le réseau EUNET. Celle-ci a comme username "MAILER" et comme nom de noeud "CERNVAX" sur le réseau "EARN". Le message construit alors par MAILGATE est illustré par la figure 18.

Figure 18 : Structure d'un message construit par MAILGATE



Lorsque le message est reçu par la passerelle, celle-ci n'a qu'à enlever l'en-tête pour obtenir le nouveau message qu'elle doit réexpédier sur EUNET. Ce nouveau message sera bien conforme puisqu'il possédera un en-tête correct car possèdant l'adresse du destinataire sur EUNET et un corps contenant le texte du message rédigé par Monsieur Dupont.

Il serait intéressant de munir EASY-MAIL de cette possibilité de formatage des messages en fonction de l'utilisation ou non d'une passerelle. C'est dans ce but que nous avons muni le MAILBOOK de la possibilité d'insérer les adresses électroniques des passerelles. Cette démarche permettrait de rendre EASY-MAIL plus compact.

6.2.2 EASY-MAIL et UNIX

La version UNIX de EASY-MAIL doit être implémentée et installée sur ESOMC3, le BULL SPS7 travaillant sur UNIX SYSTEM 5. Cette démarche devrait se faire en quatre étapes:

1) Installation du software Protéus sur ESOMC3.

2) Implémentation de la commande PutUNIX spécifiée mais non encore existante dans Protéus et nécessaire à EASY-MAIL pour exécuter une commande sur le logiciel MAILX.

3) Installation de EASY-MAIL sur ESOMC3 en complètant ses modules sources. Il s'agit simplement d'ajouter aux macro-instructions "#DEFINE" déjà prévues et écrites pour la version UNIX les chaînes de caractères devant correspondre aux commandes du logiciel MAILX. Cette étape ne devrait pas prendre plus que quelques jours.

4) Tests de EASY-MAIL version UNIX.

BIBLIOGRAPHIE

Bibliographie

CCITT, X400 RECOMMENDATION, 1984

BULL, COMMUNICATION TUTORIAL, France, novembre 1986

J. COUTAZ, ABSTRACTION POUR INTERFACES INTERACTIVES, Technique en science informatique, volume 5 numéro 4, 1986, 239-249.

J. COUTAZ, LA CONSTRUCTION D'INTERFACES HOMME-MACHINE, laboratoire de génie informatique (IMAG), St-Martin-d'Hères (France), novembre 1986.

J. COUTAZ, THE CONSTRUCTION OF USER INTERFACES, centre de recherche Bull - laboratoire de génie informatique (IMAG), St-Martin-d'Hères (France), février 1986.

J. COUTAZ, THE CONSTRUCTION OF USER INTERFACES, First European engineering conference, Strasbourg (France), septembre 1987.

D. H. CROCKER, STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES, University of Delaware, Newark (U.S.A.), août 1982.

DATA SYSTEMS USERS WORKING GROUP, INTRODUCTION TO THE SPACE ANALYSIS NETWORK, novembre 1986

Ph. DEFERT, INTRODUCTION TO TCP/IP AND RELATED COMMUNICATION SOFTWARE, European Southern Observatory, Garching (RFA), Juin 1987.

DIGITAL EQUIPMENT CORPORATION, VAX PSI INTRODUCTION, Media Publishing and Design Services department in Reading, U.K., Mars 1986.

DIGITAL EQUIPMENT CORPORATION, VAX/VMS MAIL UTILITY REFERENCE MANUAL, Massachusset (U.S.A.), septembre 1984.

D. HEAGERTY, A USER GUIDE TO ELECTRONIC MAIL SERVICES AT CERN, CERN, France, janvier 1987

IBM, REMOTE SPOOLING COMMUNICATION SUBSYSTEM NETWORKING, U.S.A., octobre 1986.

F. OCHSENBEIN, TERMWINDOWS, Southern Observatory, Garching (RFA), juillet 1987.

J. B. POSTEL, SIMPLE MAIL TRANSFERT PROTOCOL, Information Sciences Institute University of Southern California, Californie (U.S.A.), août 1982.

M. POULLET, LA TELEMATIQUE, MARABOUT, VERVIER, Février 1985.

RICHARD C. RAFFENETTI, PARTICIPATION OF VAX VMS COMPUTERS IN IBM FILE TRANSFER NETWORKS, National Energy Software Center,U.S.A., septembre 1983.

BIBLIOGRAPHIE

A. RICHMOND, SOFTWARE DESIGN FOR USER INTERFACES, European Southern Observatory, Garching (RFA), 1987.

A. RICHMOND, THE PROTEUS SOFTWARE HANDBOOK, European Southern Observatory, Garching (RFA), 1987.

K. SHOENS, MAIL REFERENCE MANUAL, juillet 1983

A. S. TANENBAUM, COMPUTER NETWORKS, Prentice-hall inc., Amsterdam, 1981.

THE WOLLONGONG GROUP INC., THE TCP/IP PRIMER, The Wollongong Group Inc., Californie (U.S.A.), mai 1986.

Ph. VAN BASTELAER, COURS DE TELE-INFORMATIQUE, FNDP, Namur, 1986-1987.

Ph. VAN BASTELAER, COURS DE TELE-INFORMATIQUE ET RESEAUX (matière appronfondie), FNDP, Namur, 1986-1987.

Ph. VAN BASTELAER, INTRODUCTION AUX RESEAUX LOCAUX, FNDP, Namur, novembre 1985.

A. VAN LAMSWEERDE, COURS DE METHODOLOGIE DE DEVELOPPEMENT DE LOGICIELS, FNDP, Namur, 1986-1987.

A. VAN LAMSWEERDE, COURS DE METHODOLOGIE DE DEVELOPPEMENT DE LOGICIELS (matière appronfondie), FNDP, Namur, 1987-1988.

ANNEXE I : LES SYSTEMES DE GESTION DE DIALOGUE

ANNEXE I : Les Systèmes de Gestion de Dialogue

Annexe I : Les Systèmes de Gestion de Dialogue

I.1 Introduction aux SGDs

Une application peut être décomposée en deux parties: d'une part l'algorithmique et d'autre part l'interface de dialogue. Alors que l'algorithmique a bénéficié de très nombreux travaux ayant abouti à des outils, méthodes et langages de programmation, l'intérêt porté à l'interface est beaucoup plus récent. L'interface peut être vue sous deux perspectives différentes: celui de l'utilisateur ou celui du réalisateur de l'application interactive.

Vu sous l'angle de l'utilisateur, on peut sans peine affirmer que la communication "homme-machine" est un domaine très complexe. Elle a, en effet, pour objet de comprendre comment l'être humain interagit avec l'ordinateur. Certaines notions d'ergonomie et de psychologie sont des concepts déterminants pour un interface de qualité mais il n'existe encore actuellement aucune théorie montrant comment intégrer ces notions au logiciel. A l'heure actuelle où l'utilisateur n'est pas le plus souvent un informaticien, la qualité de l'interface conditionne le succès du logiciel. Pour l'utilisateur, l'interface sera la représentation du dialogue "homme-machine" et de la tâche associée.

En ce qui concerne le réalisateur de l'application interactive, l'interface peut être vue comme les composants softwares et hardwares rendant le dialogue "homme-machine" ainsi que la tâche possibles. C'est sous cette optique que nous allons parler des SGDs.

ANNEXE I : LES SYSTEMES DE GESTION DE DIALOGUE

Les outils pour la construction d'interfaces "homme-machine" se présentent sous deux formes: les boîtes à outils et les systèmes génériques. Tandis qu'une boîte à outils n'est qu'une librairie de procédures pour l'écriture d'interfaces "homme-machine", un système générique fournit un squelette standard sur lequel viennent se greffer les composants spécifiques aux applications. On peut également distinguer deux types de système générique. D'une part les Applications Exensibles et d'autre part les Systèmes de Gestion de Dialogue aussi appelés SGDs. Les applications extensibles regroupent le code qui implémente les fonctions usuelles de l'interface "homme-machine" sous la forme d'un squelette réutilisable et extensible. La tâche du programmeur consiste alors à remplir les trous et à redéfinir les parties standard non adaptées à l'application. Une application extensible est donc une extension de la boîte à outils qui structure les traitements communs à toutes les applications interactives. Cependant, le niveau d'abstraction de l'interface de programmation est encore trop bas: l'implémenteur est contraint de rechercher dans la boîte à outils les fonctions nécessaires à l'extension ou à la modification du squelette de base. Les SGDs vont relever le niveau d'abstraction de l'interface de programmation en permettant la spécification du dialogue dans un langage de haut niveau.

I.2 Le modèle SGD

I.2.1 Mécanismes

Le SGD va faire le lien entre la sémantique et les éléments lexico-syntaxiques du dialogue.

Les éléments lexicaux consistent en des opérations d'entrée/sortie contruites à partir des opérations hardwares tel que par exemple un clic souris pour l'entrée et la désignation d'une couleur pour la sortie.

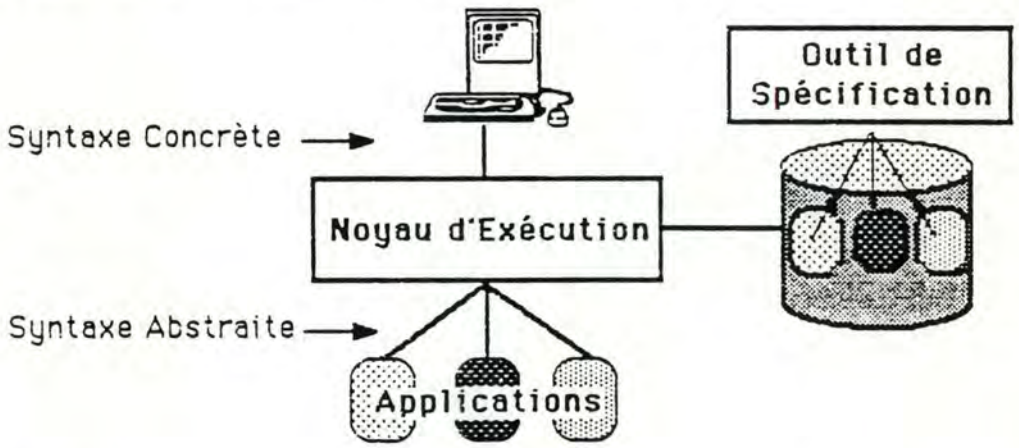
La syntaxe de l'entrée, elle, se réfère aux séquences permises des opérations d'entrées (par exemple sélectioner une icône) pour former les actions (par exemple supprimer un objet). La syntaxe de sortie définit les lois de composition pour les actions en sortie (par exemple présentation de message, menus ou icônes).

Enfin, la sémantique est exprimée par les fonctions que l'application exécute en réponse aux commandes de l'utilisateur.

ANNEXE I : LES SYSTEMES DE GESTION DE DIALOGUE

Comme montré à la figure A-1 , le SGD maintient pour chaque application interactive une base de données des spécifications. Ces spécifications définissent la syntaxe concrète du dialogue et la correspondance entre syntaxe concrète et syntaxe abstraite. Tandis que la syntaxe concrète définit le mode d'emploi de l'application tel que le perçoit l'utilisateur, la syntaxe abstraite définit le format des informations échangées lors de l'exécution entre le noyau d'execution et l'application: le SGD décompose les détails interactifs syntaxiques des commandes données par l'utilisateur et les délivrent alors, à l'application, dans une forme directement traitable. Inversément, le SGD peut aussi recevoir des réactions de l'application et insérer des ornements syntaxiques et lexicaux avant de les délivrer à l'utilisateur.

Figure A-1 : Schéma d'un Système de Gestion de Dialogue



I.2.2 Analogie avec les Systèmes de Gestion de Base de Données

Le modèle SGD résulte des principes qui ont plus tôt motivé le développement des systèmes de gestion de base de données.

Comme pour un SGBD (Système de Gestion de Base de Données), un SGD définit un modèle abstrait et y ajoute un formalisme pour l'expression d'une vue logique du système imposée par le modèle abstrait. Ce formalisme permet une séparation franche entre l'interface et le fonctionnement de l'application. Similairement, les SGBD permettent une division entre le langage d'interrogation et les techniques de gestion du stockage des informations dans la base de données.

Le SGD peut inclure des facilités comme une aide locale ou une communication de données entre applications. De la même façon, un SGBD offre des capacités supplémentaires aux manipulations physiques de stockage comme le recouvrement, le partage de données ou la sécurité.

I.2.3 Le contrôle du dialogue

Le contrôle du dialogue peut être interne, externe ou mixte. Le contrôle est interne quand il réside dans l'application. Il est externe lorsqu'il est maintenu par le SGD. Il est mixte quand le contrôle est alternativement traité par l'application et le SGD.

Avec le contrôle interne, le SGD est implémenté comme une librairie de procédures que l'application appele pour exécuter des tâches d'entrées/sorties. Le contrôle interne ne permet pas une séparation franche entre les fonctions de l'application et les tâches de dialogue.

Inversément, avec le contrôle externe, le SGD maintient le contrôle, il est responsable pour invoquer les procédures de l'application en réponse aux actions spécifiques de l'utilisateur. L'application est vue alors comme une collection de procédures qui implémentent les actions sémantiques du dialogue. Cette façon de faire entraîne la séparation de la sémantique avec la syntaxe. Elle ouvre la porte à des programmes plus modulaires bien qu'elle puisse imposer un style de programmation particulier. Le contrôle externe mène à moins de contraintes arbitraires imposées par le programmeur à l'utilisateur. Par opposition avec le contrôle interne, le contrôle externe opère dans un style plus proche de l'utilisateur: un SGD avec contrôle externe offre à l'utilisateur les sous-routines de l'application pouvant être utilisées et appelées lorsqu'il est approprié de le faire.

Entre ces deux extrêmes, le contrôle mixte permet au réalisateur de l'application de changer librement entre les modes de contrôle interne ou externe. Cette flexibilité est utile pour les systèmes qui ont besoin de notifier au SGD des évènements asynchrones (tel que par exemple la sonnerie d'une alarme) ou pour les commandes interactives qui requièrent dynamiquement à l'utilisateur de spécifier les paramètres avant que le traitement ne puisse continuer.

I.2.4 Les critère de classification

Les SGDs peuvent être classés selon différents critères. Parmi ceux-ci, on retiendra les principaux : le niveau d'abstraction, la localisation du contrôle et l'ordonnancement des évènements.

Le niveau d'abstraction

Le niveau d'abstraction définit l'unité d'échange entre l'application et le SGD. Cette unité peut correspondre à un évènement de bas niveau comme la frappe d'une touche du clavier ou bien être une entité de haut niveau comme une commande complète et syntaxiquement correcte. Il faut faire remarquer que le niveau d'abstraction influe directement sur la qualité du logiciel et sur la difficulté de la tâche du programmeur. En effet, lorsque le niveau d'abstraction est bas, les traitements nécessaires à la mise à niveau des unités d'échange avec les entités propres à l'application risquent d'être dilués dans le code de l'application et détournent l'implémenteur du rôle qui devrait être le sien: la mise en oeuvre des fonctions sémantiques.

Localisation du contrôle

Comme nous l'avons déjà mentionné, le contrôle du dialogue peut être interne, externe ou mixte. La localisation du contrôle est un critère de classification des SGDs dans la mesure où le contrôle interne offre certains inconvénients par rapport aux deux autres modes de contrôle.

Ordonnancement des évènements

Le dernier critère pour classer les outils de dialogue concerne l'ordre d'occurence des évènements utilisateurs. Il peut être spécifié explicitement, implicitement ou pas du tout.

I.3 Avantages des SGDs

Le modèle SGD ne facilite pas seulement la tâche du programmeur mais aussi ouvre la porte à la construction de meilleures interfaces pour les utilisateurs.

Il force à une franche séparation entre la sémantique et la syntaxe. De cette propriété résulte trois avantages pour l'utilisateur:

-la syntaxe concrète peut être désignée par un expert en ce domaine.

-la syntaxe concrète peut être raffinée sans modifier le système fondamental

-plusieurs systèmes peuvent se partager la même syntaxe concrète, donnant à l'utilisateur une vue uniforme de la station de travail.

De plus, en prenant en charge les actions de l'utilisateur qui sont sémantiquement sans rapport avec l'application, le SGD permet au programmeur de se concentrer sur la logique du système à mettre en oeuvre. Par exemple, un SGD est capable d'exécuter des vérifications syntaxiques tel que la complétude d'une commande ou la vérification du type de paramètres. De cette façon un système de gestion de fichiers construit avec un SGD n'est pas concerné sur la manière dont sera invoqué l'ordre "supprimer tel fichier" par l'utilisateur (par exemple en frappant l'ordre au clavier ou par sélection dans un menu). Il ne sera pas non plus concerné par les actions intermédiaires de l'utilisateur qui mènent à la commande finale (exemple d'action intermédiaire: une requête pour une aide locale).

ANNEXE II : Documentation des programmes

Annexe II : Documentation des programmes

II.1 Introduction à la documentation

Cette annexe a pour but de documenter l'ensemble du software construit.

La section II.2 concerne le programme EASY-MAIL. Pour une utilisation aisée de cette documentation, il faut savoir que les textes sources des fonctions qui implémentent une commande particulière de EASY-MAIL sont regroupés dans un même fichier pour former un module. A un module correspond donc généralement une commande de EASY-MAIL. Il faut cependant faire remarquer qu'on a préféré isoler dans des modules à part certaines fonctions particulièrement complexes. On trouvera donc aussi des modules auxquels ne correspondent aucune commande de EASY-MAIL mais offrant un service nécessaire à l'implémentation d'une ou plusieurs d'entre elles. Donnons comme exemple le module LECTURE qui regroupe les fonctions pour accéder au Mailbook selon une clé d'accès. Inversément, à certaines commandes de EASY-MAIL ne correspond aucun module. En effet, les commandes 'Exit', 'Spawn', et 'Menu' sont des commandes déja implémentées par Protéus. Elles ne sont donc pas décrites ici. La documentation de EASY-MAIL a été rédigée par module de la façon suivante: pour chaque module désigné par son nom, son utilité est explicitée et l'algorithme utilisé est décrit. De plus, chacune de ses fonctions est également spécifiée: sa déclaration avec ses arguments dans le langage employé, une description de l'action exécutée et enfin les valeurs qu'elle peut retourner avec leurs conditions associées sont décrites.

La section II.3 présente deux fichiers référencés par les différents modules : les fichiers 'longueur.h' et 'record.h'

L'avant dernière section donne la spécification de deux autres programmes implémentés avec EASY-MAIL.

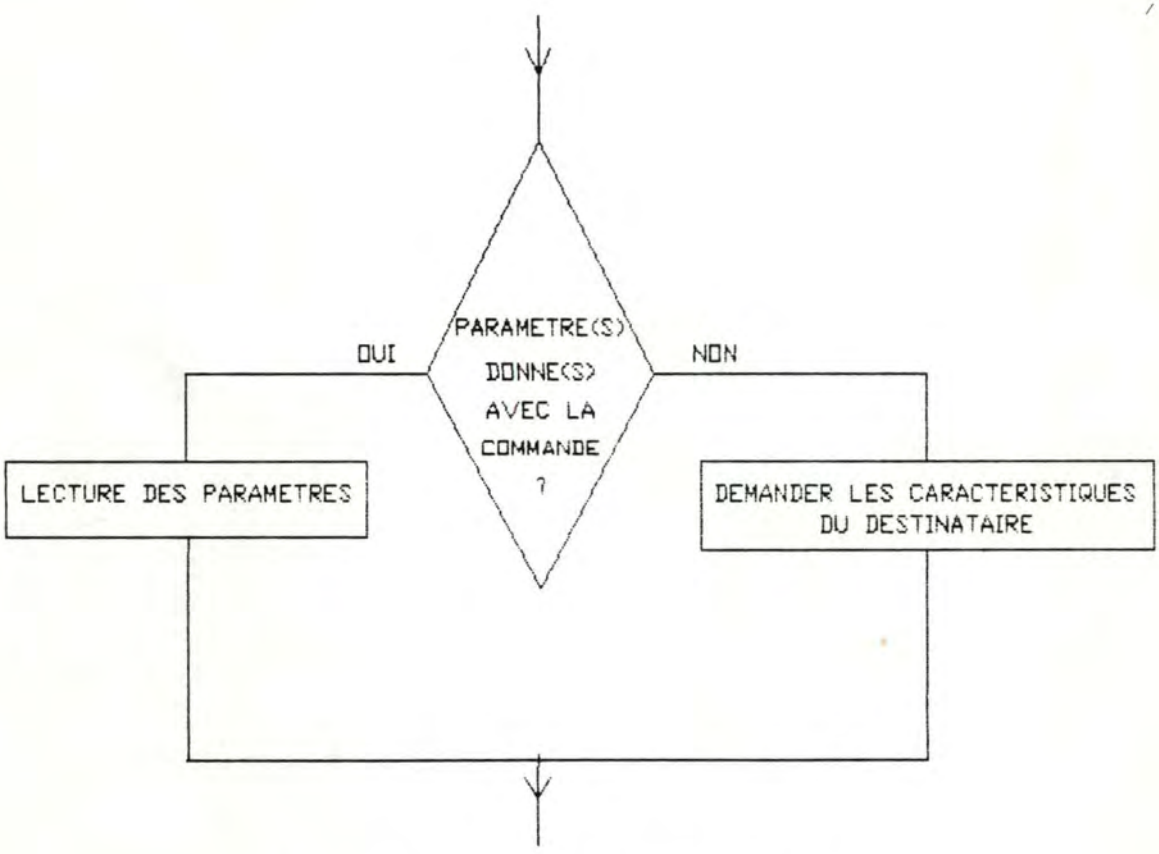
L'annexe II se termine enfin avec l'organisation du fichier Mailbook. Cette dernière section peut être consultée par l'utilisateur désireux de modifier son Mailbook à l'aide d'un éditeur.

II.2 Description de EASY-MAIL

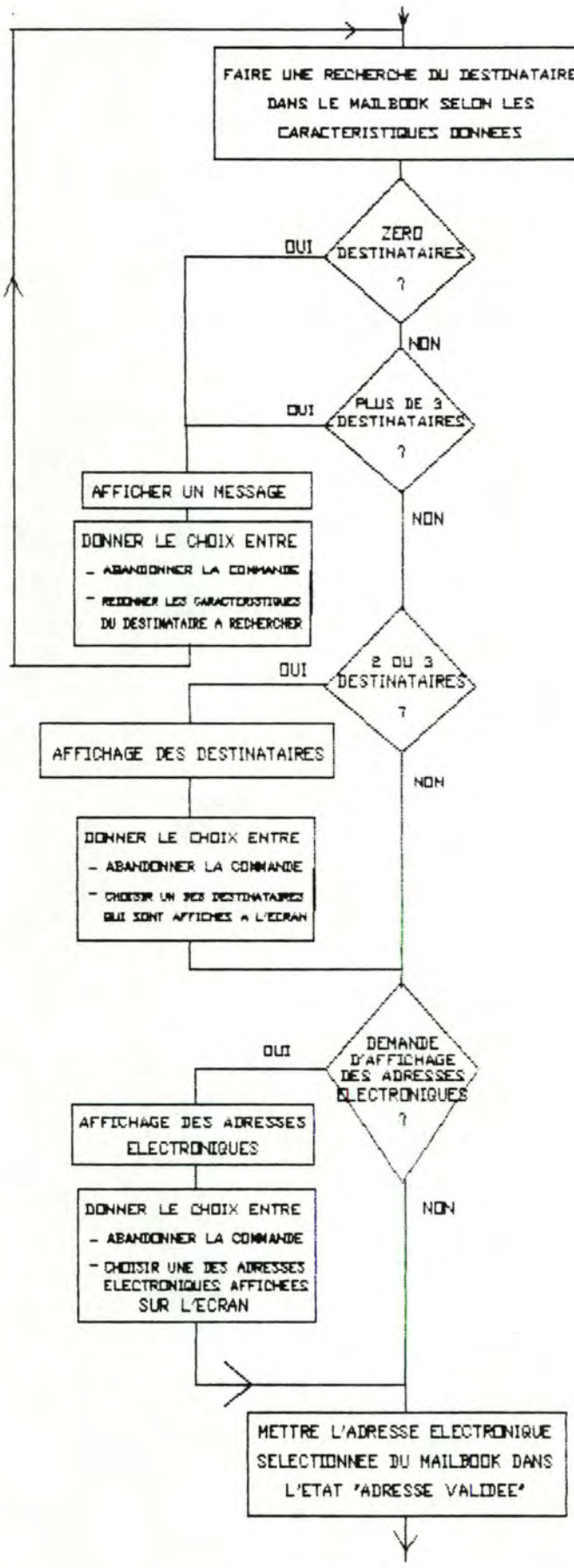
II.2.1 Module ADDRVALID

Utilité

Ce module permet de changer les flags d'une adresse électronique du 'Mailbook' afin de la mettre dans l'état de 'adresse validée'.



(suite sur la page suivante)


FAIRE UNE RECHERCHE DU DESTINATAIRE DANS LE MAILBOOK SELON LES CARACTERISTIQUES DONNEES
ZERO DESTINATAIRES ?
OUI
NON
PLUS DE 3 DESTINATAIRES ?
OUI
NON
AFFICHER UN MESSAGE
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- REDONNER LES CARACTERISTIQUES DU DESTINATAIRE A RECHERCHER
2 OU 3 DESTINATAIRES ?
OUI
NON
AFFICHAGE DES DESTINATAIRES
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- CHOISIR UN DES DESTINATAIRES QUI SONT AFFICHES A L'ECRAN
DEMANDE D'AFFICHAGE DES ADRESSES ELECTRONIQUES ?
OUI
NON
AFFICHAGE DES ADRESSES ELECTRONIQUES
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- CHOISIR UNE DES ADRESSES ELECTRONIQUES AFFICHEES SUR L'ECRAN
METTRE L'ADRESSE ELECTRONIQUE SELECTIONNEE DU MAILBOOK DANS L'ETAT 'ADRESSE VALIDEE'

II.2.1.3 Description des fonctions définies dans ce module

F_Val_adr ()

Spécification

Cette fonction est la fonction principale implémentant la commande 'AppendMailbook' du Mailer.

Valeur retournée

'OK'

```
F_Maj_Flag (result,prof_choisi,mail_choisi)
struct trouve *result;
int prof_choisi,mail_choisi;
```

Spécification

Cette fonction met les flags d'une adresse électronique d'une personne répertoriée dans le Mailbook dans l'état 'adresse validée'. L'enregistrement à modifier se trouve dans la structure 'result' la personne étant précisée par 'prof_choisi' tandis que son adresse à modifier par 'mail_choisi'.

Valeur retournée

'OK'

```
F_Recopiage (fil_des1,fil_des2,code)
FILE *fil_des1,*fil_des2;
char *code;
```

Spécification

Cette fonction recopie un enregistrement du Mailbook (référencé par 'fil_des1) dans un autre fichier (référencé par 'fil_des2').

Valeur retournée

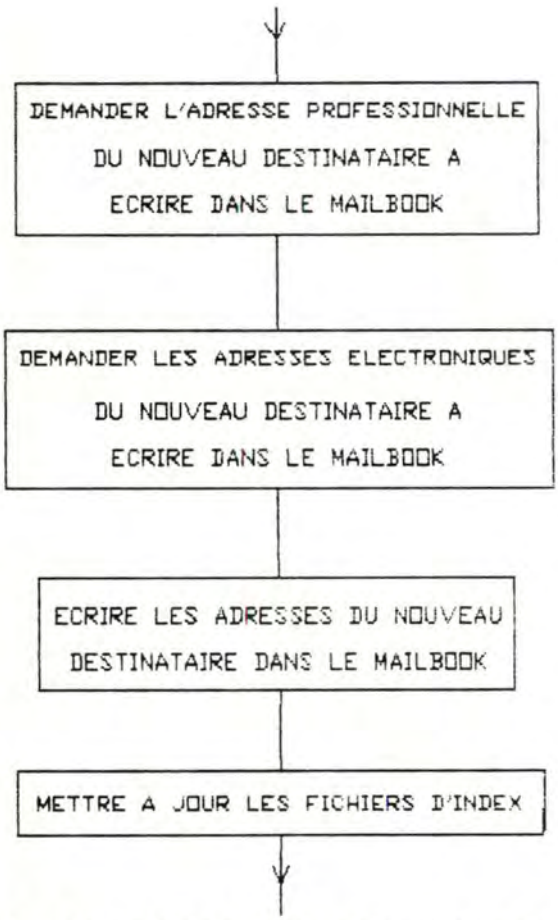
'OK'

II.2.2 Module APPEND

Utilité

Ce module permet à un utilisateur de rentrer un nouveau destinataire dans son Mailbook. Il s'agit donc de l'implémentation de la commande 'AppendMailbook' de EASY-MAIL.

Algorithme



Description des fonctions définies dans ce module

F_AppendMailbook ()

Spécification

Cette fonction est la fonction principale implémentant la commande 'AppendMailbook' de Easy-Mail.

Valeur retournée

'OK'

```
F_EnterProf (car)
struct un_trouve *car;
```

Spécification

Cette fonction permet à l'utilisateur de donner l'adresse professionnelle du nouveau destinataire à inscrire dans son Mailbook.Celle-ci est ensuite inscrite dans la structure 'car'.

Valeur retournée

'OK'

```
F_EnterMail (car)
struct un_trouve *car;
```

Spécification

Cette fonction permet à l'utilisateur de donner les adresses électroniques du nouveau destinataire à inscrire dans son Mailbook. Celles-ci sont ensuite inscrites dans la structure 'car'.

Valeur retournée

'OK'

```
F_Change (car)
struct un_trouve *car;
```

Spécification

Cette fonction écrit dans le Mailbook un nouveau destinataire dont les informations se trouvent dans la structure 'car'.

Valeur retournée

'OK'

```
F_Ecriture (ptr,file_des)
struct un_trouve *ptr;
FILE *file_des;
```

Spécification

Cette fonction écrit dans un fichier indiqué par 'file_des' un nouveau destinataire dont les informations se trouvent dans la structure 'car' après s'être positionné à l'endroit précisé par la variable 'location' de la structure.

Valeur retournée

'OK'

```
F_Index (ptr)
struct un_trouve *ptr;
```

Spécification

Cette fonction met à jour les fichiers d'index du Mailbook en fonction d'un nouveau destinataire ajouté et se trouvant dans la structure 'ptr'.

Valeur retournée

'OK'

```
F_Maj (file,key,location)
char *file,*key;
int location;
```

Spécification

Cette fonction met à jour un fichier d'index dénommé par la variable 'file' en y ajoutant un nouvel enregistrement constitué de la clé 'key' et de l'adresse 'location'.

Valeur retournée

'OK'

```
F_Write (file_des,ptr)
FILE *file_des;
struct index_record *ptr;
```

Spécification

Cette fonction ecrit l'enregistrement constitué de la structure 'ptr' dans le fichier précisé par 'file_des'.

Valeur retournée

'OK'

```
F_Ajoute (chaine,longueur)
char *chaine;
int longueur;
```

Spécification

Cette fonction rajoute des blancs à la fin de la chaîne de caractères contenue dans la variable 'chaine' sur une longueur donnée par la variable 'longueur' soustraite de un. On soustrait de un car le dernier caractère blanc est suivi par le caractère de fin de chaîne de caractères.

Valeur retournée

'OK'

```
F_Blanc (chaine,longueur)
char *chaine;
int longueur;
```

Spécification

Cette fonction remplit de blancs la variable 'chaine' sur une longueur donnée par la variable 'longueur'.

Valeur retournée

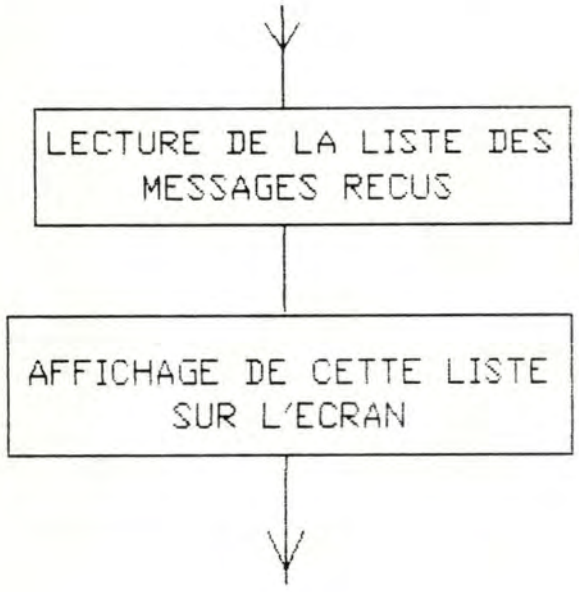
'OK'

II.2.3 Module DIR

Utilité

Afficher sur l'écran la liste des messages reçus sous la forme : adresse de l'envoyeur - date - sujet du message. Il s'agit de l'implémentation de la commande 'DirMessage'.

Algorithme



Description des fonctions définies dans ce module

F_Dir ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'DirMessage'.

Valeur retournée

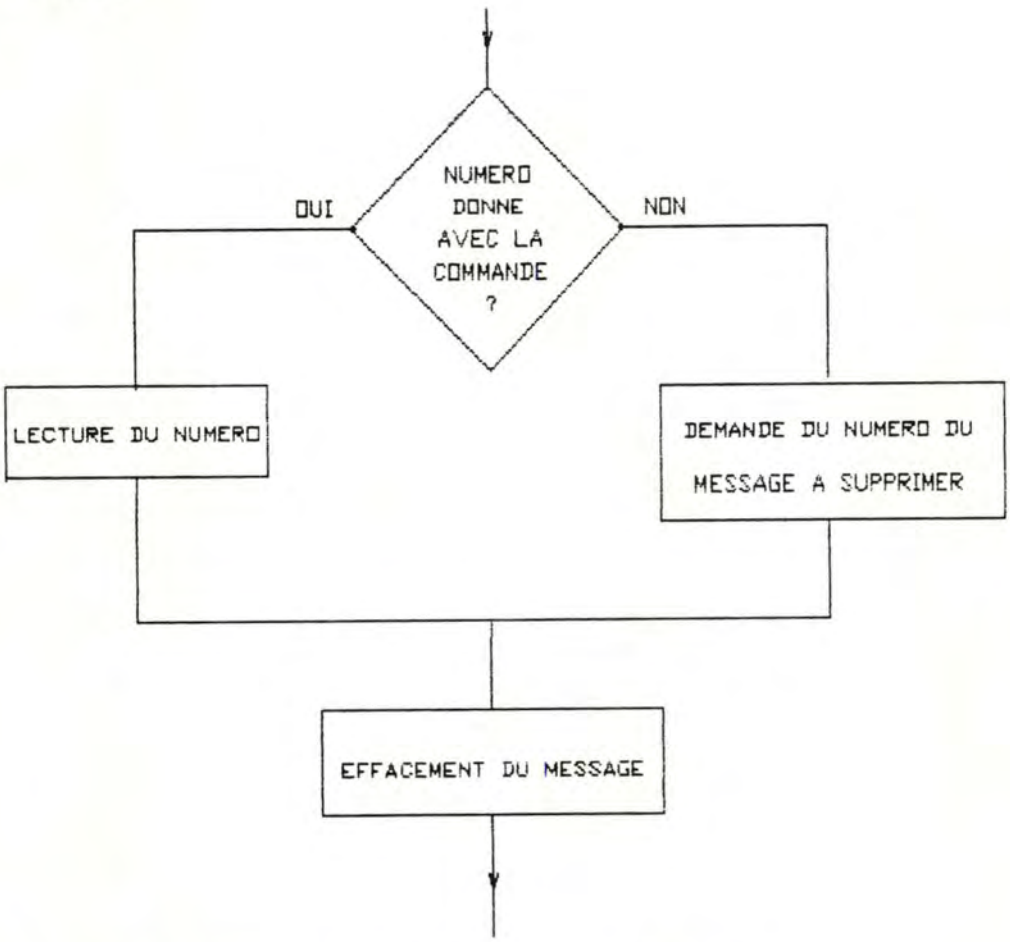
'OK'

II.2.4 Module DMES

Utilité

Ce module permet d'effacer un message reçu. Il implémente la commande 'DeleteMessage' de EASY-MAIL.

Algorithme



Description des fonctions définies dans ce module

F_Dmes ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'DeleteMessage'.

Valeur retournée

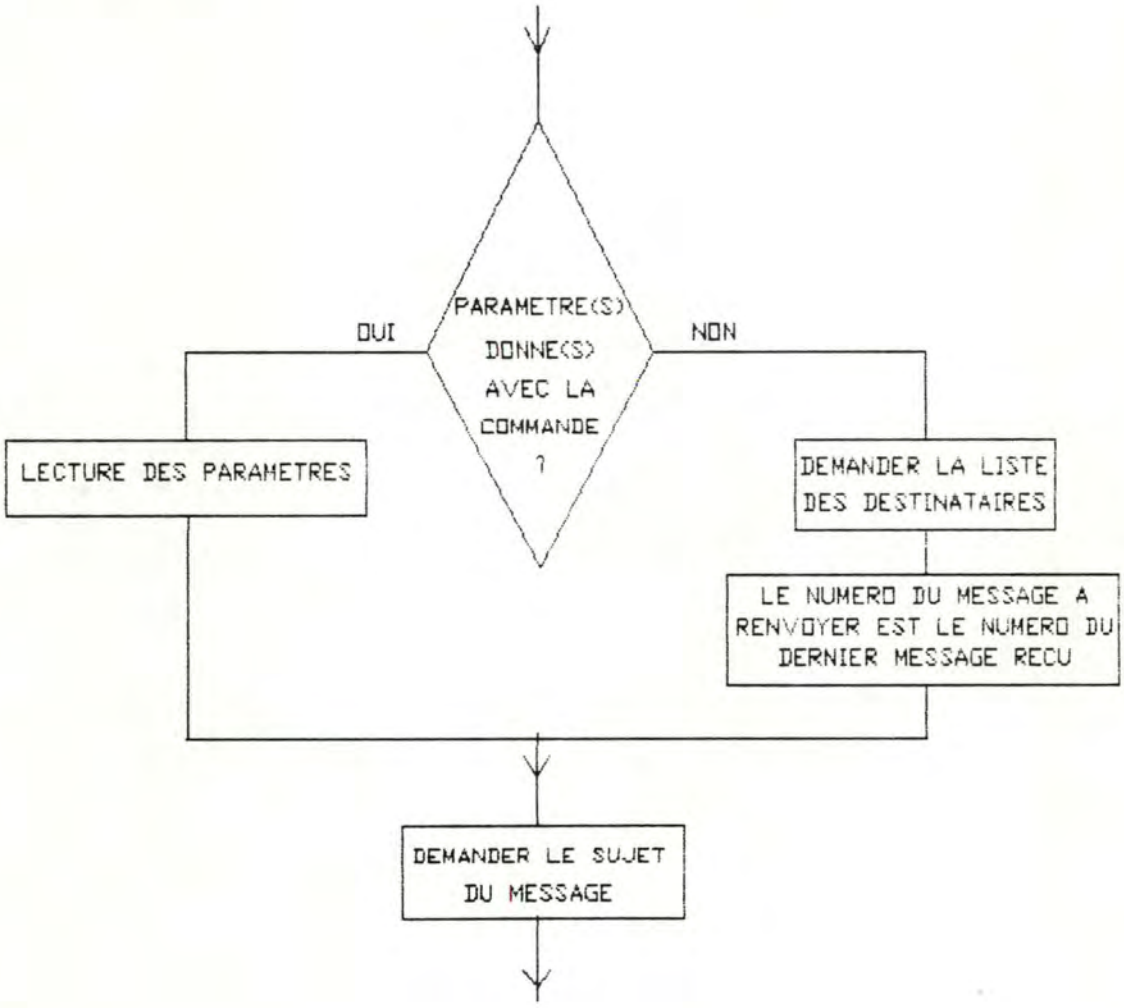
'OK'

II.2.5 Module FORWARD

Utilité

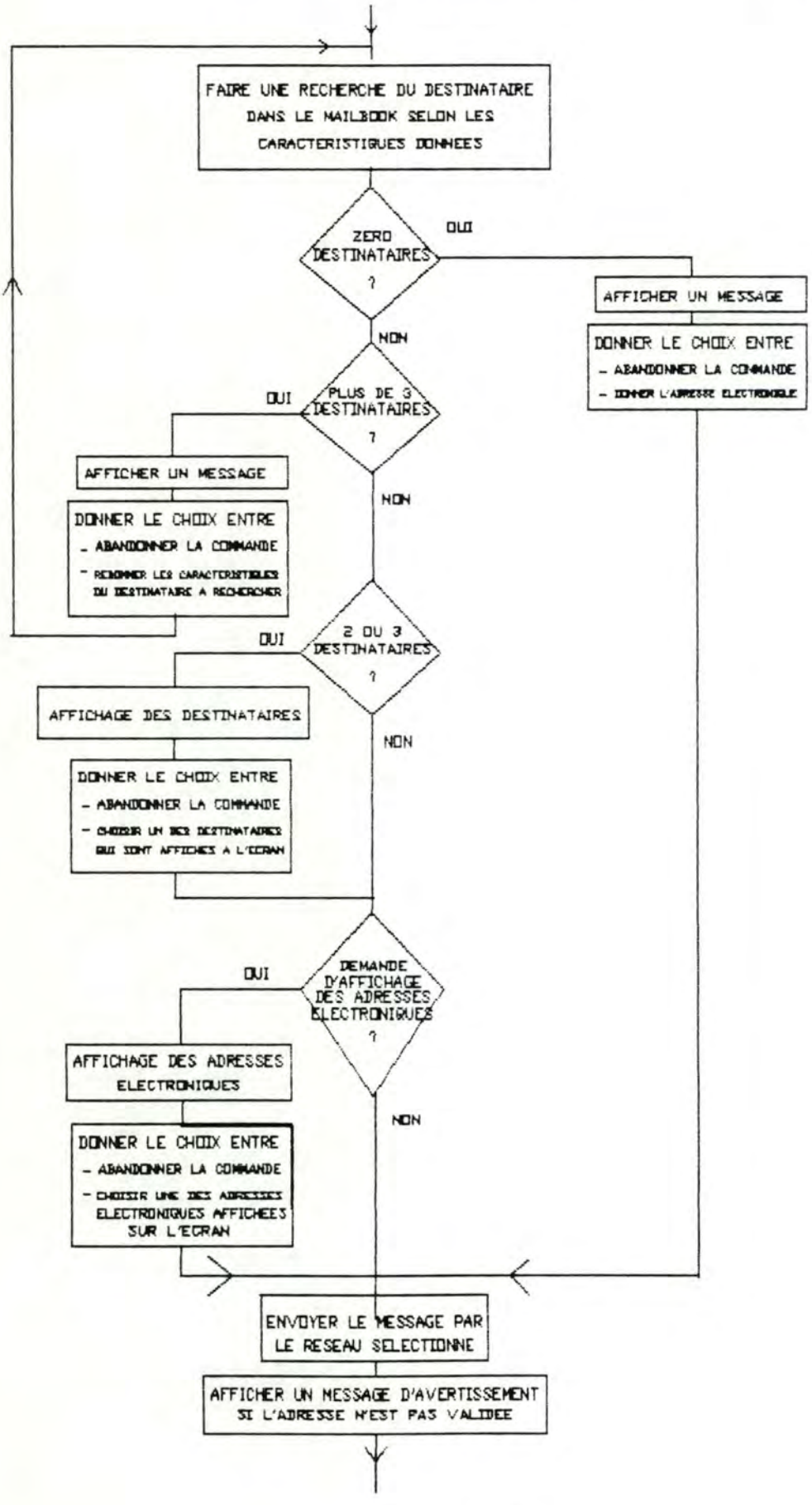
Ce module permet d'envoyer un message reçu vers un ou plusieurs destinataires. Il s'agit du module implémentant la commande la commande 'ForwardMessage' de EASY-MAIL.

Algorithme



(suite sur la page suivante)

POUR CHAQUE DESTINATAIRE DE LA LISTE



Description des fonctions définies dans ce module

F_Fmes ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'ForwardMessage'.

Valeur retournée

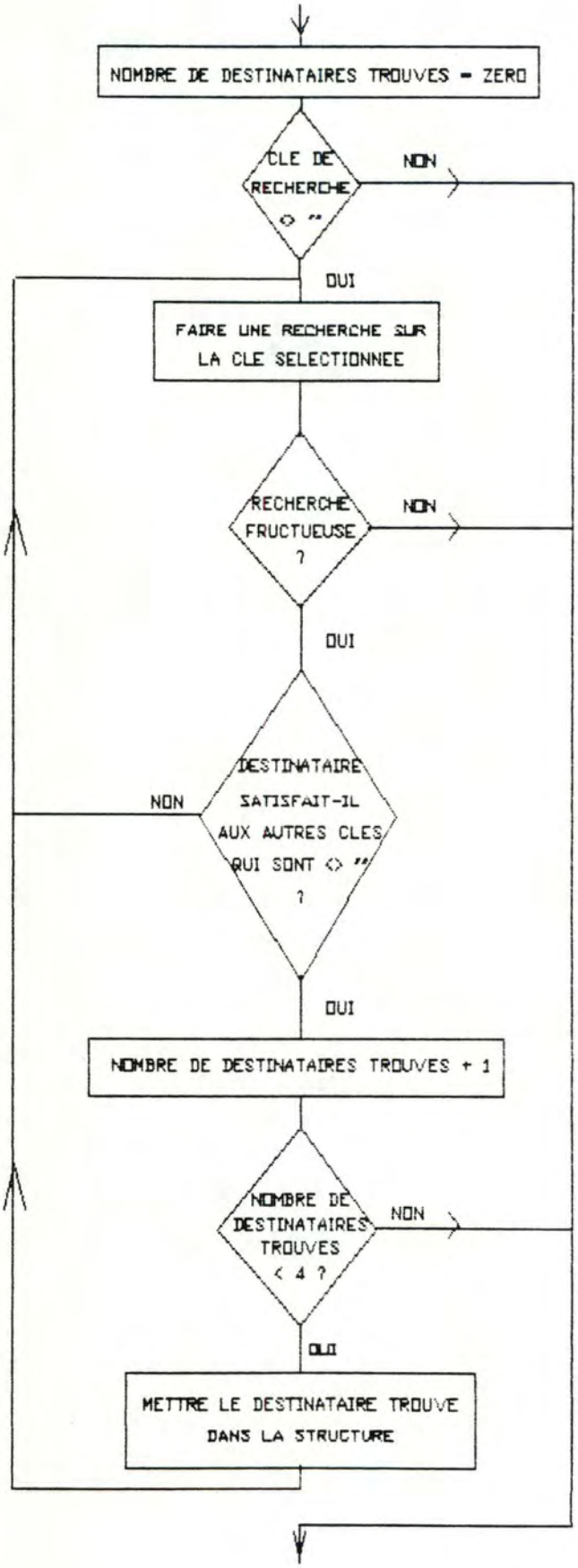
'OK'

II.2.6 Module LECTURE

Utilité

Ce module lit dans le Mailbook les adresses professionnelles et les adresses électroniques des personnes qui correspondent aux clés de recherche données par l'utilisateur.

Algorithme


NOMBRE DE DESTINATAIRES TROUVES = ZERO
CLE DE RECHERCHE <> ''
NON
OUI
FAIRE UNE RECHERCHE SUR LA CLE SELECTIONNEE
RECHERCHE FRUCTUEUSE ?
NON
OUI
DESTINATAIRE SATISFAIT-IL AUX AUTRES CLES QUI SONT <> '' ?
NON
OUI
NOMBRE DE DESTINATAIRES TROUVES + 1
NOMBRE DE DESTINATAIRES TROUVES < 4 ?
NON
OUI
METTRE LE DESTINATAIRE TROUVE DANS LA STRUCTURE

Description des fonctions définies dans ce module

```
F_Recherche (file,cle,car_ptr,nombre_trouve,
             trouve_ptr)
char *file,*cle;
struct caracteristique *car_ptr;
struct trouve trouve_ptr;
```

Spécification

Cette fonction fait une recherche de personnes dans le Mailbook selon une clé et un fichier d'index précisé (les variables 'cle' et 'file'). Elle teste si la personne trouvée correspond aux clés données par l'utilisateur. Celles-ci se trouvent dans la structure 'car_ptr'. Si oui, alors l'adresse professionnelle de cette personne ainsi que ses adresses électroniques sont recopiées dans la structure 'trouve_ptr'. La variable 'nombre_trouve' contient le nombre de personnes déja trouvées et inscrites dans la structure. La recherche s'arrête lorsque on ne trouve plus de personnes satisfaisants aux conditions demandées ou bien lorsque on en a trouvée plus que trois.

Valeur retournée

'OK'

```
F_Lecture(car_ptr,nombre_trouve,trouve_ptr)
int *nombre_trouve;
struct caracteristique *car_ptr;
struct trouve *trouve_ptr;
```

Spécification

Cette fonction recherche dans le Mailbook les personnes correspondant aux clés données par l'utilisateur et se trouvant dans la structure 'car_ptr'. Elle place les adresses professionnelles des personnes trouvées ainsi que leurs adresses électroniques dans la structure 'trouve_ptr'. La variable 'nombre_trouve' contient le nombre de personnes trouvées. La recherche s'arrête lorsque on ne trouve plus de personne satisfaisant aux conditions demandées ou lorsqu'on en a trouvée plus que trois.

Valeur retournée

'OK'

```
F_Mailbook (location,buf)
struct un_trouve *buf;
int location;
```

Spécification

Cette fonction lit dans le Mailbook l'adresse professionnelle et les adresses électroniques de la personne dont l'enregistrement débute à la position donnée par la variable 'location' et les place dans la structure 'buf'.

Valeur retournée

'OK'

```
F_Compare (buf,car)
struct un_trouve *buf;
struct caracteristique *car;
```

Spécification

Cette fonction compare si la personne se trouvant dans la structure 'buf' vérifie les caractéristiques se trouvant dans la structure 'car'.

Valeur retournée

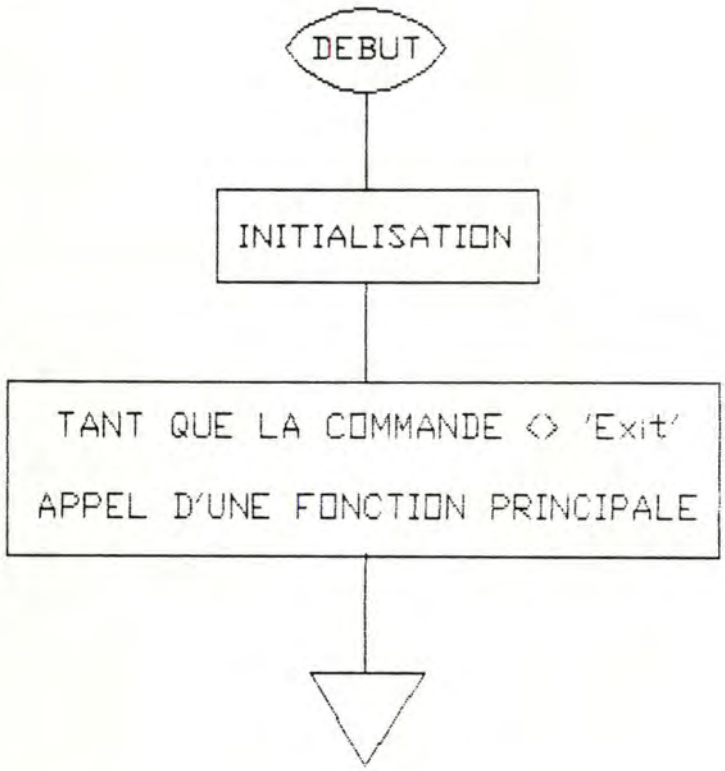
'OK' si les caracteristiques données sont vérifiées
'NOK' sinon.

II.2.7 Module MAILER

Utilité

Initialiser EASY-MAIL.

Algorithme



Lorsque l'utilisateur demande à executer une commande en tapant sur le clavier le nom de celle-ci, une fonction est appelée. La correspondance entre commande, fonction appelée et module contenant cette fonction est la suivante :

Commande	Fonction	Module
AddressValidate	F_Val_Adr()	ADDRVALID
AppendMailbook	F_AppendMailbook()	APPENDMAILBOOK
DirMessage	F_Dir()	DIR
DeleteMessage	F_Dmes()	DMES
ForwardMessage	F_Fmes()	FORWARD
Help	Help()	cfr Protéus
Menu	SetupMode()	cfr Protéus
Msend	F_Msend()	MSEND
PrintMessage	F_Print()	PRINT
Reply	F_Reply()	REPLY
ReadMessage	F_Rmes()	RMES
Send	F_Send()	SEND
Spawn	HostSpawn()	cfr Protéus
WriteFile	F_Wfile()	WFILE
WriteMessage	F_Wmes()	WMES

ANNEXE II : DOCUMENTATION DES PROGRAMMES

Description des fonctions définies dans ce module

PGM (mailer)

Spécification

C'est le programme principal de EASY-MAIL. Il fait les initialisations necessaires a la bonne marche de Proteus. Il appele ensuite la fonction Mailcom.

Valeur retournée

néant

Mailcom ()

Spécification

Cette fonction lit la commande frappée au clavier et appele la fonction qui y est associée. La fonction se termine lorsque la commande appelée est 'Exit'.

Valeur retournée

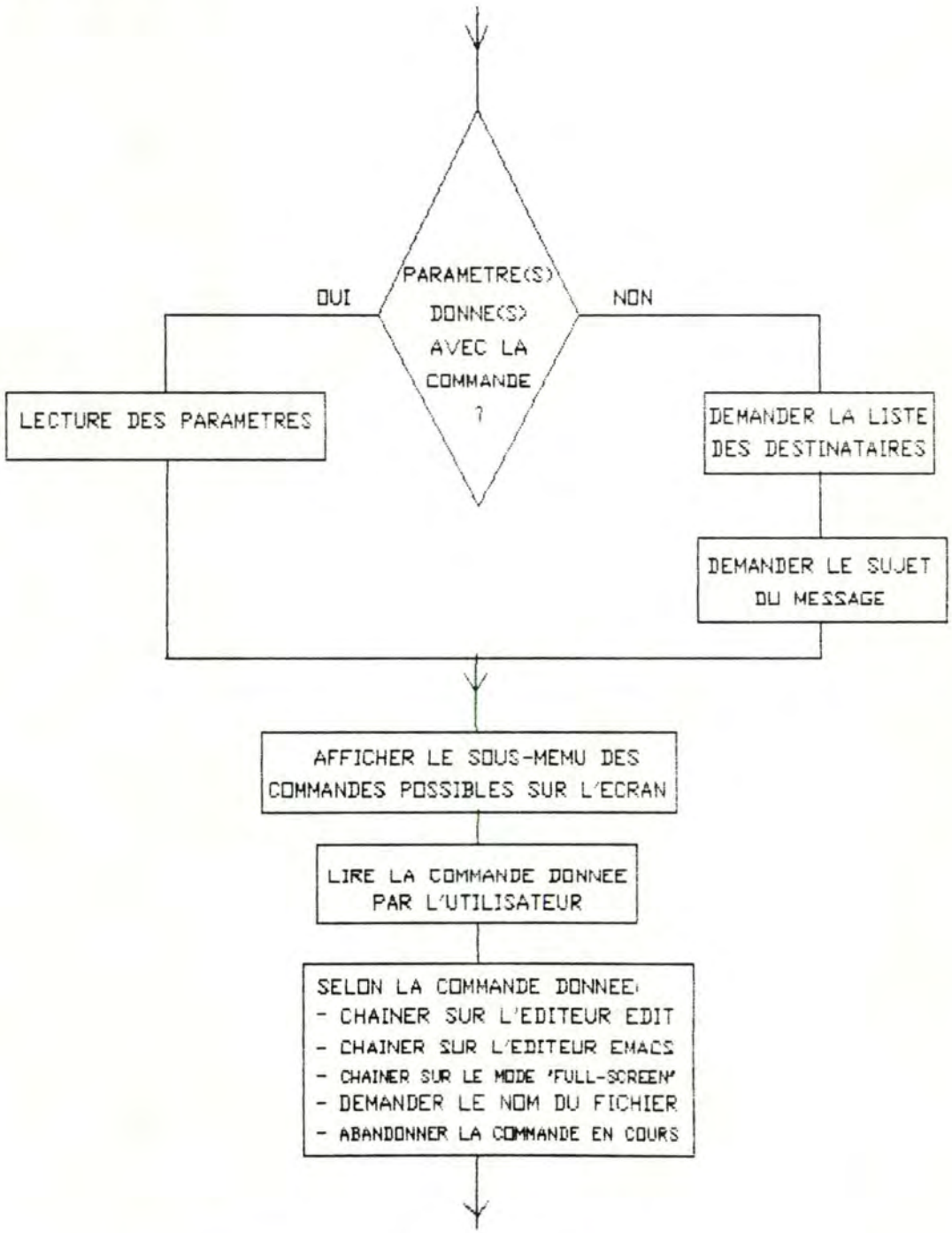
'OK'

II.2.8 Module MSEND

Utilité

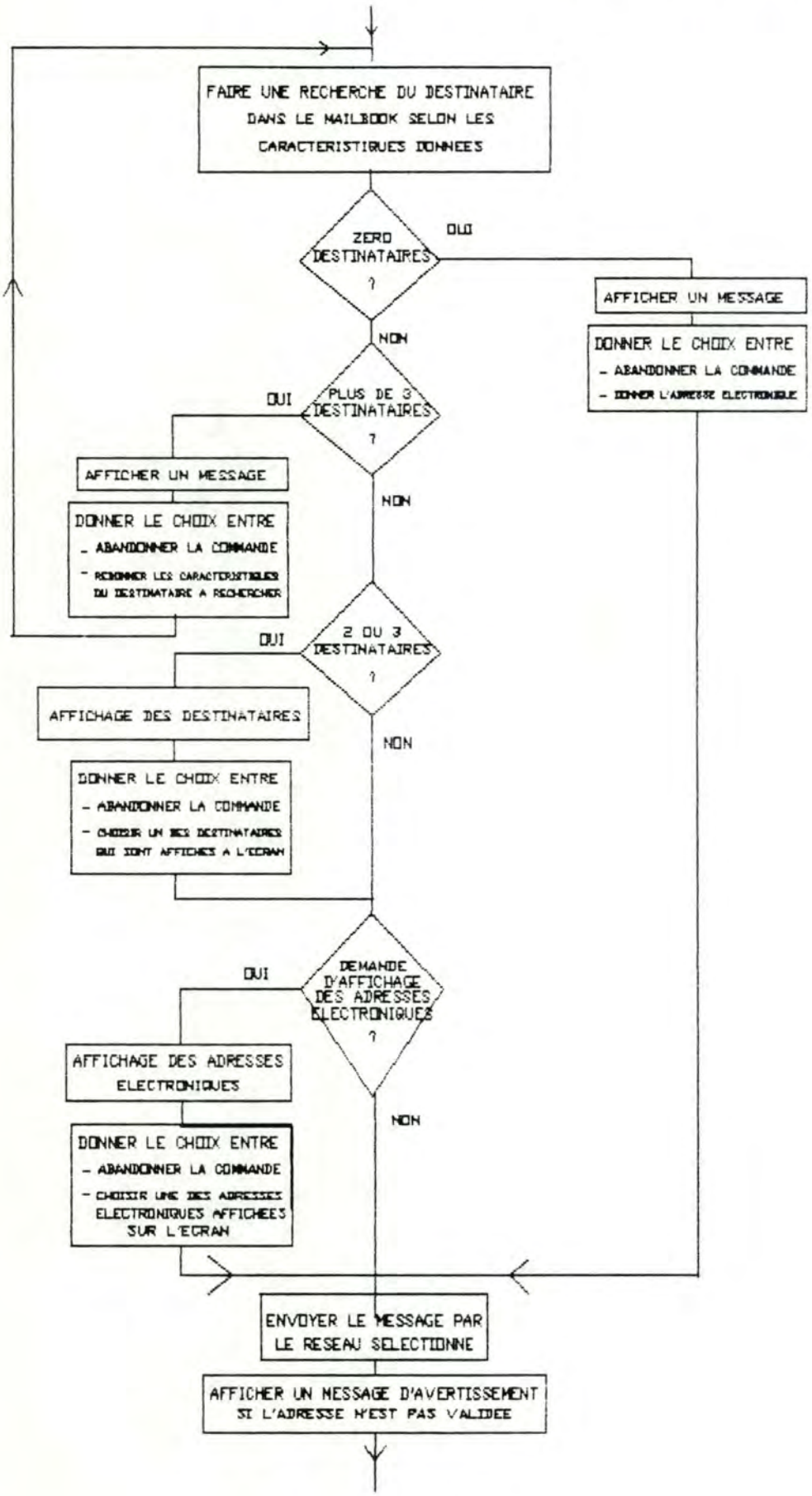
Ce module permet d'envoyer un message à plusieurs destinataires. Il s'agit de l'implémentation de la commande 'Msend' de EASY-MAIL.

Algorithme



(suite sur la page suivante)

POUR CHAQUE DESTINATAIRE DE LA LISTE



Description des fonctions définies dans ce module

F_Msend ()

Spécification

Cette fonction est la fonction principale implémentant la commande 'Msend' de EASY-MAIL.

Valeur retournée

néant

```
F_List (buffer)
char *buffer;
```

Spécification

Cette fonction permet à l'utilisateur de donner la liste des destinataires et la place dans la variable 'buffer'.

Valeur retournée

'OK'

```
F_Extraction (nb,buffer,car)
char *buffer;
struct caracteristique car  ;
int *nb;
```

Spécification

Cette fonction fait l'analyse syntaxique de la variable 'buffer' et place les destinataires trouvés dans la table de structures 'car'. Elle incrémente 'nb' de un à chaque destinataire trouvé.

Valeur retournée

'OK'

```
F_Analyse (variable,car,nb)
char variable  ;
struct caracteristique car  ;
int *nb;
```

Spécification

Cette fonction fait l'analyse syntaxique du destinataire contenu dans la variable 'variable' et en fonction de cette analyse, elle place le destinataire dans la table de structures 'car'. Elle incrémente 'nb' de un à chaque nouveau destinataire placé dans la structure.

Valeur retournée

'OK'

```
F_FileAnalyse (variable,car,nb)
char variable  ;
struct caracteristique *car  ;
int *nb;
```

Spécification

Cette fonction fait l'analyse syntaxique du fichier dont le nom est contenu dans la variable 'variable' et, en fonction de cette analyse elle place les destinataires trouvés dans la table de structures 'car'. Elle incrémente 'nb' de un à chaque nouveau destinataire placé dans la structure.

Valeur retournée

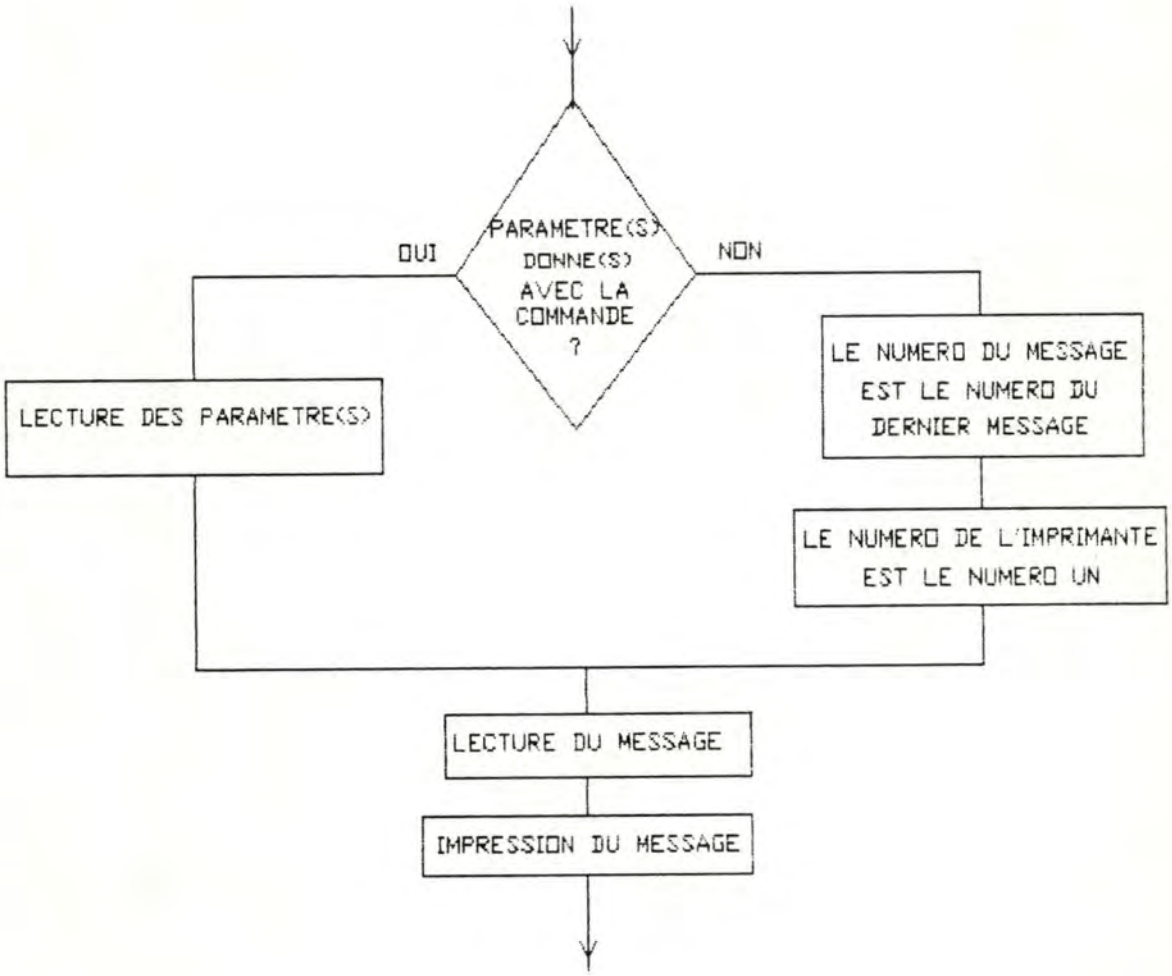
'OK'

II.2.9 Module PRINT

Utilité

Ce module permet d'imprimer un message sur une imprimante spécifiée. Il implémente donc la commande 'PrintMessage' de EASY-MAIL.

Algorithme



Description des fonctions définies dans ce module

F_Print ()

Spécification

Cette fonction est la fonction principale implémentant la commande 'PrintMessage' de EASY-MAIL.

Valeur retournée

'OK' si le numero donné par l'utilisateur correspond à une imprimante connue
'NOK' sinon.

```
F_Para2 (number,device)
char *number,device;
```

Spécification

Cette fonction permet de lire les paramètres number et device rentrés avec le nom de la commande conformement à la syntaxe définie dans le guide de l'utilisateur de EASY-MAIL. Elle place number dans 'number' (avec "1" comme valeur par défaut) et device dans 'device' (avec comme valeur par defaut le dernier message reçu).

Valeur retournée

'OK'

```
F_File_Name ()
```

Spécification

Cette fonction permet de donner un nom de fichier à la variable 'file_name'. Afin d'avoir un nom reconnaissable et personnalisé (il y a peu de chances que l'utilisateur ait déja dans sa directory un fichier ayant ce nom), il est composé de la maniere suivante :
JJMMMhhmmss.MAI où JJ est le jour en chiffres,
MMM sont les trois premieres lettres du mois,
hh est l'heure en chiffres,
mm est la minute en chiffres,
ss est la seconde en chiffres.

Valeur retournée

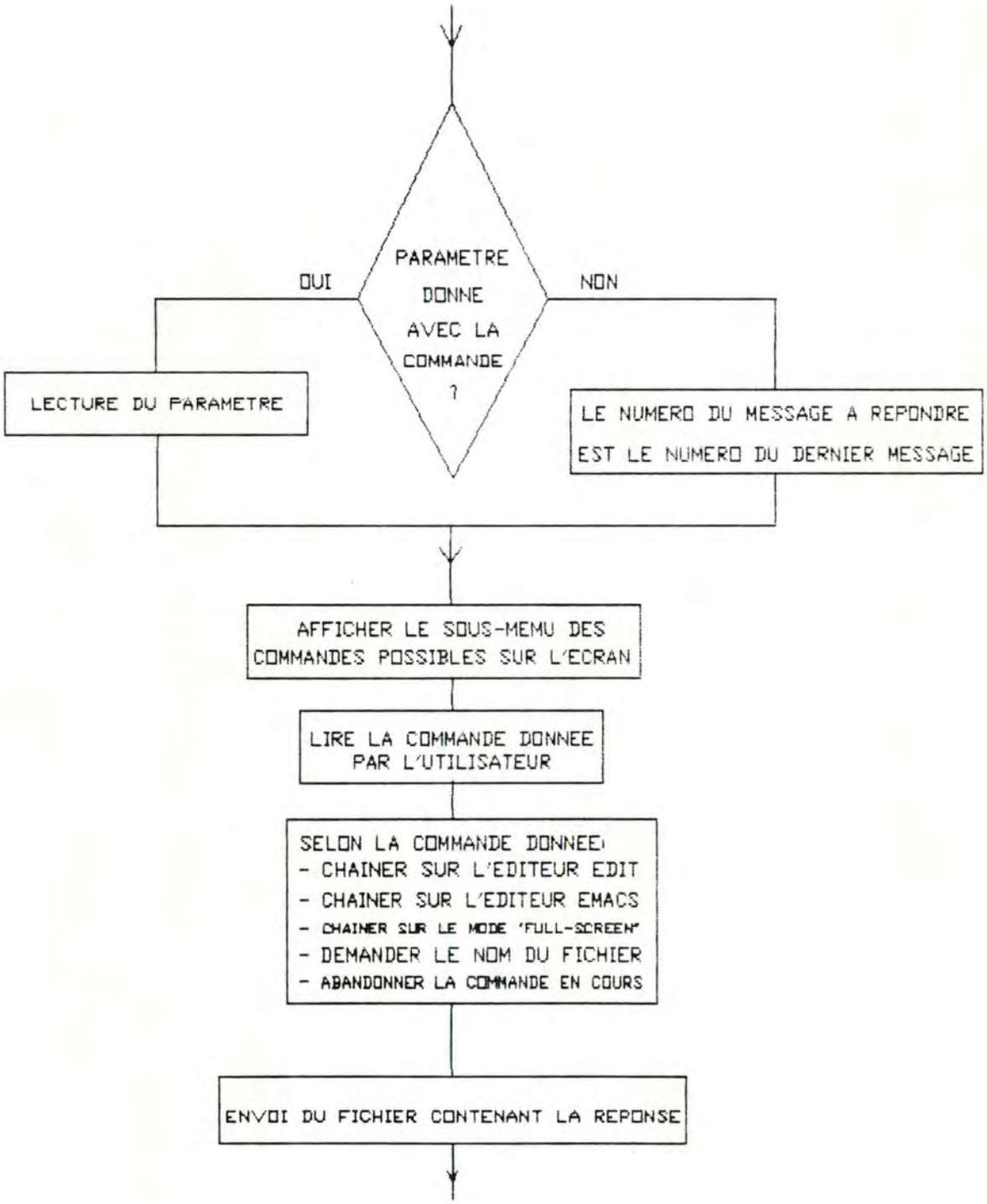
'OK'

II.2.10 Module REPLY

Utilité

Ce module permet de répondre à un message reçu. Il implémente donc la commande 'ReplyMessage'.

Algorithme



Description des fonctions définies dans ce module

F_Reply ()

Spécification

Cette fonction permet de répondre à un message reçu. Il s'agit donc de la fonction principale implémentant la commande 'ReplyMessage'.

Valeur retournée

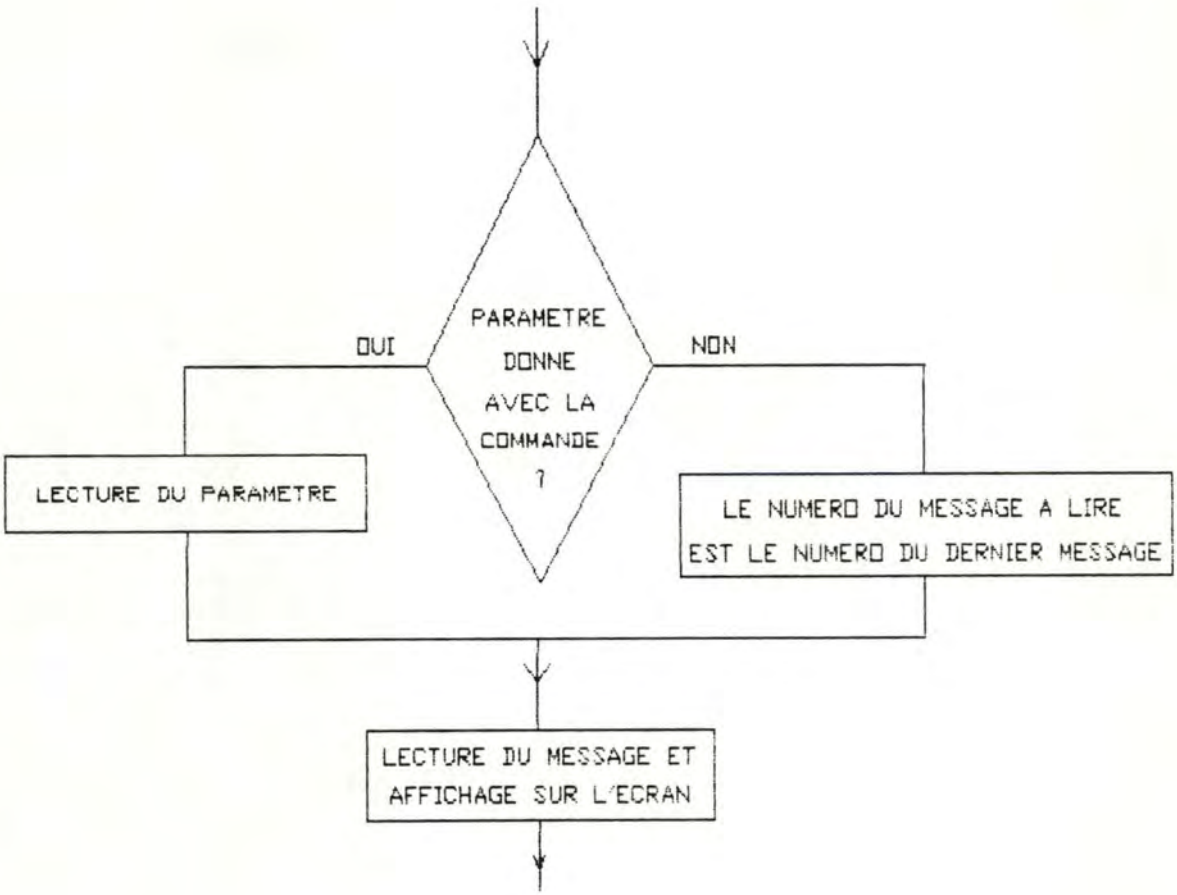
'OK'

II.2.11 Module RMES

Utilité

Permettre la lecture d'un message en imprimant celui-ci sur l'écran. Il s'agit de l'implémentation de la commande 'ReadMessage'.

Algorithme



Description des fonctions définies dans ce module

F_Rmes ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'ReadMessage'.

Valeur retournée

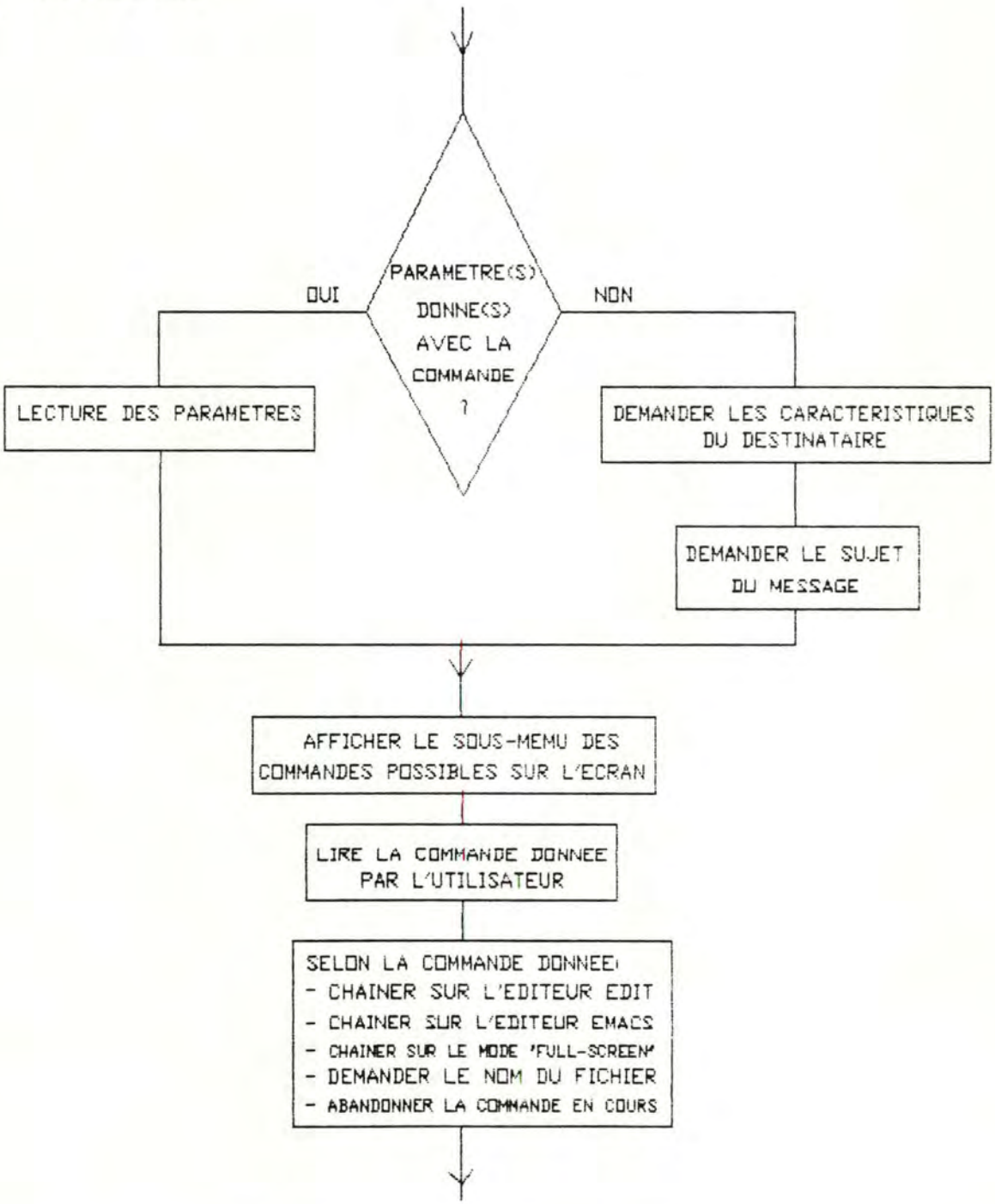
'OK'

II.2.12 Module SEND

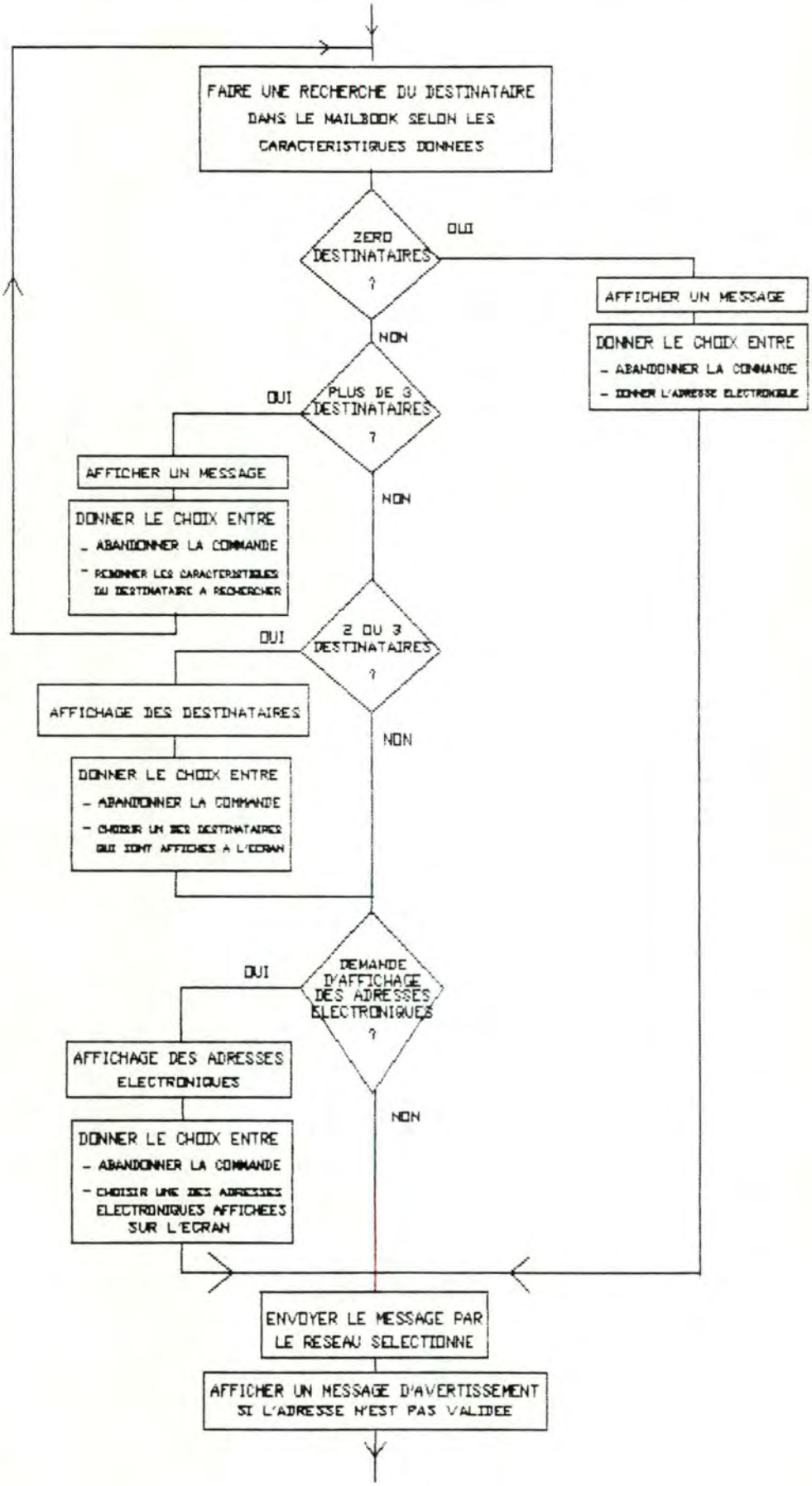
Utilité

Permettre l'envoi d'un message vers un destinataire. Il s'agit de l'implémentation de la commande 'Send' de EASY-MAIL.

Algorithme



(suite sur la page suivante)


FAIRE UNE RECHERCHE DU DESTINATAIRE DANS LE MAILBOOK SELON LES CARACTERISTIQUES DONNEES
ZERO DESTINATAIRES ?
OUI
AFFICHER UN MESSAGE
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- DONNER L'ADRESSE ELECTRONIQUE
NON
PLUS DE 3 DESTINATAIRES ?
OUI
AFFICHER UN MESSAGE
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- REDONNER LES CARACTERISTIQUES DU DESTINATAIRE A RECHERCHER
NON
2 OU 3 DESTINATAIRES ?
OUI
AFFICHAGE DES DESTINATAIRES
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- CHOISIR UN DES DESTINATAIRES QUI SONT AFFICHES A L'ECRAN
NON
DEMANDE D'AFFICHAGE DES ADRESSES ELECTRONIQUES ?
OUI
AFFICHAGE DES ADRESSES ELECTRONIQUES
DONNER LE CHOIX ENTRE
- ABANDONNER LA COMMANDE
- CHOISIR UNE DES ADRESSES ELECTRONIQUES AFFICHEES SUR L'ECRAN
NON
ENVOYER LE MESSAGE PAR LE RESEAU SELECTIONNE
AFFICHER UN MESSAGE D'AVERTISSEMENT SI L'ADRESSE N'EST PAS VALIDEE

Description des fonctions définies dans ce module

F_Send ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'Send'.

Valeur retournée

'OK'

F_Envoi (car,sw)
struct caracteristique *car;
int *sw;

Spécification

Cette fonction est utilisée pour envoyer un message. 'car' référence une structure contenant les caractéristiques du destinataire à rechercher dans le Mailbook tandis que la variable 'sw' permet d'exprimer que l'envoi du message a pu ou n'a pas pu se faire. Des lors sw égale zéro ou un.

Valeur retournée

'OK'

F_Charast (car)
struct caracteristique *car;

Spécification

Cette fonction permet à l'utilisateur de donner les caractéristiques du destinataire dont l'adresse doit être recherchée dans le Mailbook. Les informations ainsi données sont inscrites dans la structure 'car'.

Valeur retournée

'OK'

Help_Form ()

Spécification

Cette fonction imprime l'écran d'aide correspondant à la commande 'Send'. Elle est employée lorsque l'utilisateur qui doit donner les caractéristiques du destinataire désire avoir de l'aide.

Valeur retournée

'OK'

```
F_MailAddress (result,prof_choisi,mail_choisi,choix)
struct trouve *result;
int prof_choisi,*mail_choisi;
char *choix;
```

Spécification

Cette fonction imprime à l'écran les adresses électroniques du destinataire et permet à l'utilisateur d'en sélectionner une de son choix pour l'envoi de son message si le contenu de la variable 'choix' est "?". Si, par contre, son contenu est "+" (valeur par défaut), alors c'est la première adresse électronique qui est automatiquement sélectionnée c'est-à-dire celle par laquelle le message arrivera le plus vite à son destinataire. 'result' est une structure contenant le résultat de la recherche du destinataire dans le Mailbook. 'mail_choisi' est un indice qui sera affecté selon l'adresse électronique sélectionnée tandis 'prof_choisi' est un indice permettant de retrouver dans la structure le destinataire sélectionné auparavant.

Valeur retournée

'OK'

```
F_Choix (n,code)
int *code,n;
```

Spécification

Cette fonction pose la question "your choice from 1 to n ('a' to abort)" à l'utilisateur ou n est le nombre maximum accepté et retourne la valeur choisie soustraite de un dans 'code'. La valeur choisie est soustraite de un car elle est utilisée comme indice dans un tableau. Si l'utilisateur choisit d'abandonner en tapant sur 'a' c'est la valeur -1 qui est retournée dans 'code'.

Valeur retournée

'OK'

```
F_Message (message,code)
```

```
char *message;
int *code;
```

Spécification

Cette fonction imprime la chaîne de caractères contenue dans la variable 'message' sur l'écran et permet à l'utilisateur, s'il le désire, d'abandonner la commande en cours. La valeur -1 est retournée dans 'code ' si oui tandis que la valeur zero sinon.

Valeur retournée

'OK'

```
F_Prof_adress(result,nb_trouve,nb_choisi)
struct trouve *result;
int nb_trouve,*nb_choisi;
```

Spécification

Cette fonction imprime à l'écran les adresses professionnelles des destinataires trouvés dans le Mailbook et permet à l'utilisateur de sélectionner parmi celles-ci le destinataire à qui il veut envoyer son message. Les destinataires se trouvent dans la structure 'result'. Le nombre de destinataires trouvés est indiqué par la variable 'nb_trouve' tandis que la variable 'nb_choisi' est affectée selon le choix fait par l'utilisateur.

Valeur retournée

'OK'

```
F_Parametre (result)
struct trouve *result;
```

Spécification

Cette fonction permet à l'utilisateur de donner le nom du destinataire, du noeud et enfin du reseau à utiliser lorsque le destinataire ne se trouve pas dans le Mailbook et qu'il désire quand même envoyer son message. L'adresse électronique ainsi donnée est inscrite dans la structure 'result'.

Valeur retournée

'OK'

```
F_Network (car)
struct caracteristique *car;
```

Spécification

Cette fonction initialise l'élément 'choix ' de la structure 'car'. Celle-ci contient les caractéristiques du destinataire à rechercher dans le Mailbook. Si l'utilisateur a donné dans les caractéristiques de la recherche à faire la valeur "?" à 'reseau' alors "?" est affectée à 'choix' tandis que le contenu de 'reseau' devient "". Si, par contre, la valeur "" ou "+" a été donnée à la variable 'reseau' par l'utilisateur, "+" est affecté à 'choix' tandis que 'reseau' devient "".

Valeur retournée

'OK'

```
F_SendMail(result,prof_choisi,mail_choisi)
struct trouve *result;
int prof_choisi,mail_choisi;
```

Spécification

Cette fonction envoie le message contenu dans le fichier dénommé par la variable 'file_name' à un des destinataires contenus dans la structure 'result'. La variable 'prof_choisi' est un indice permettant de retrouver dans la structure le destinataire du message envoyé tandis que 'mail_choisi' est un indice spécifiant l'adresse électronique de ce destinataire qui va être utilisée.

Valeur retournée

'OK'

```
F_MailLocal (user,node)
char *user,*node;
```

Spécification

Cette fonction envoie le message contenu dans le fichier dénommé par la variable 'file_name' sur le réseau local d'ESO. L'adresse du destinataire est spécifiée par 'user' et 'node'.

Valeur retournée

'OK'

```
F_MailEarn (user,node)
char *user,*node;
```

Spécification

Cette fonction envoie le message contenu dans le fichier dénommé par la variable 'file_name' sur le réseau EARN. L'adresse du destinataire est spécifiée par 'user' et 'node'.

Valeur retournée

'OK'

```
F_MailPsi (user,node)
char *user,*node;
```

Spécification

Cette fonction envoie le message contenu dans le fichier dénommé par la variable 'file_name' sur le réseau DATEX-P. L'adresse du destinataire est spécifiée par 'user' et 'node'.

Valeur retournée

'OK'

```
F_MailGate (user,node,network)
char *user,*node,*network;
```

Spécification

Cette fonction envoie le message contenu dans le fichier dénommé par la variable 'file_name' sur le réseau précisé par la variable 'network' et ceci par l'intermédiaire d'une passerelle située sur EARN. L'adresse du destinataire est spécifiée par 'user', 'node' et 'network' tandis que la passerelle est choisie en fonction du réseau du destinataire par un programme appelé "Mailgate".

Valeur retournée

'OK'

```
F_MailSpan (user,node)
char *user,*node;
```

Spécification

Cette fonction envoie le message contenu dans le fichier dénommé par la variable 'file_name' sur le réseau SPAN. L'adresse du destinataire est spécifiée par 'user' et 'node'.

Valeur retournée

'OK'

```
F_Avertissement (user,node,network)
char *user,*node,*network;
```

Spécification

Cette fonction affiche un message d'avertissement sur l'écran pour dire que l'adresse formée de 'user','node' et 'network' n'a pas été validée et que en conséquence, l'envoi correct du message ne peut être garanti.

Valeur retournée

'OK'

```
F_Retire (chaine,longueur)
char chaine  ;
int longueur;
```

Spécification

Cette fonction retire les blancs finaux de la variable 'chaine' et de longueur 'longueur'.

Valeur retournée

'OK'

```
F_Sujet ()
```

Spécification

Cette fonction permet à l'utilisateur de donner le sujet de son message. Celui-ci est placé dans la variable 'sujet'. Cette fonction n'a pas d'arguments car 'sujet' a été déclaré "extern".

Valeur retournée

'OK'

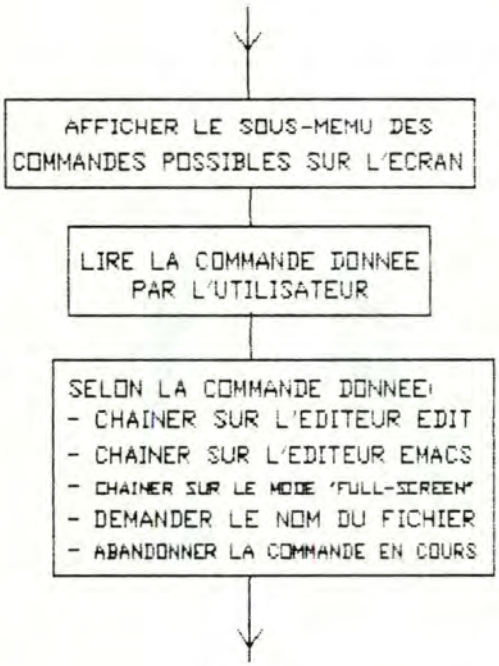
II.2.13 Module SUBMENU

Utilité

Permettre à l'utilisateur de rédiger un message. Il affiche un menu qui offre le choix entre :

- utiliser l'éditeur EDIT
- utiliser l'éditeur EMACS
- utiliser le mode "full-screen"
- spécifier le nom du fichier contenant le message déja rédigé.
- abandonner la commande en cours.

Algorithme



Description des fonctions définies dans ce module

Submenu ()

Spécification

Permettre à l'utilisateur de rédiger un message. Il affiche un menu qui offre le choix entre

- utiliser l'éditeur EDIT
- utiliser l'éditeur EMACS
- utiliser le mode "full-screen"
- spécifier le nom du fichier contenant le message déja rédigé.
- abandonner la commande en cours.

Valeur retournée
'OK' si l'utilisateur a rédigé son message
'NOK' si l'utilisateur a choisi d'abandonner la commande en cours.

F_Compose ()

Spécification

Cette fonction permet à l'utilisateur de rédiger son message en mode "full-screen". Celui-ci est ensuite inscrit dans un fichier dénommé par la variable 'file_name'.

Valeur retournée

'OK'

F_Edit ()

Spécification

Cette fonction permet à l'utilisateur de rédiger son message en employant l'éditeur EDIT. Le nom du fichier ainsi créé se trouve dans la variable 'file_name'.

Valeur retournée

'OK'

F_Emacs ()

Spécification

Cette fonction permet à l'utilisateur de rédiger son message en employant l'éditeur EMACS. Le nom du fichier ainsi créé se trouve dans la variable 'file_name'.

Valeur retournée

'OK'

F_Exit ()

Spécification

Cette fonction permet de sortir du sous-menu sans rédiger de message et ainsi abandonner la commande en cours de EASY-MAIL.

Valeur retournée

'OK'

F_File ()

Spécification

Cette fonction permet à l'utilisateur de donner le nom du fichier dans lequel se trouve le message déja rédigé auparavant.

Valeur retournée

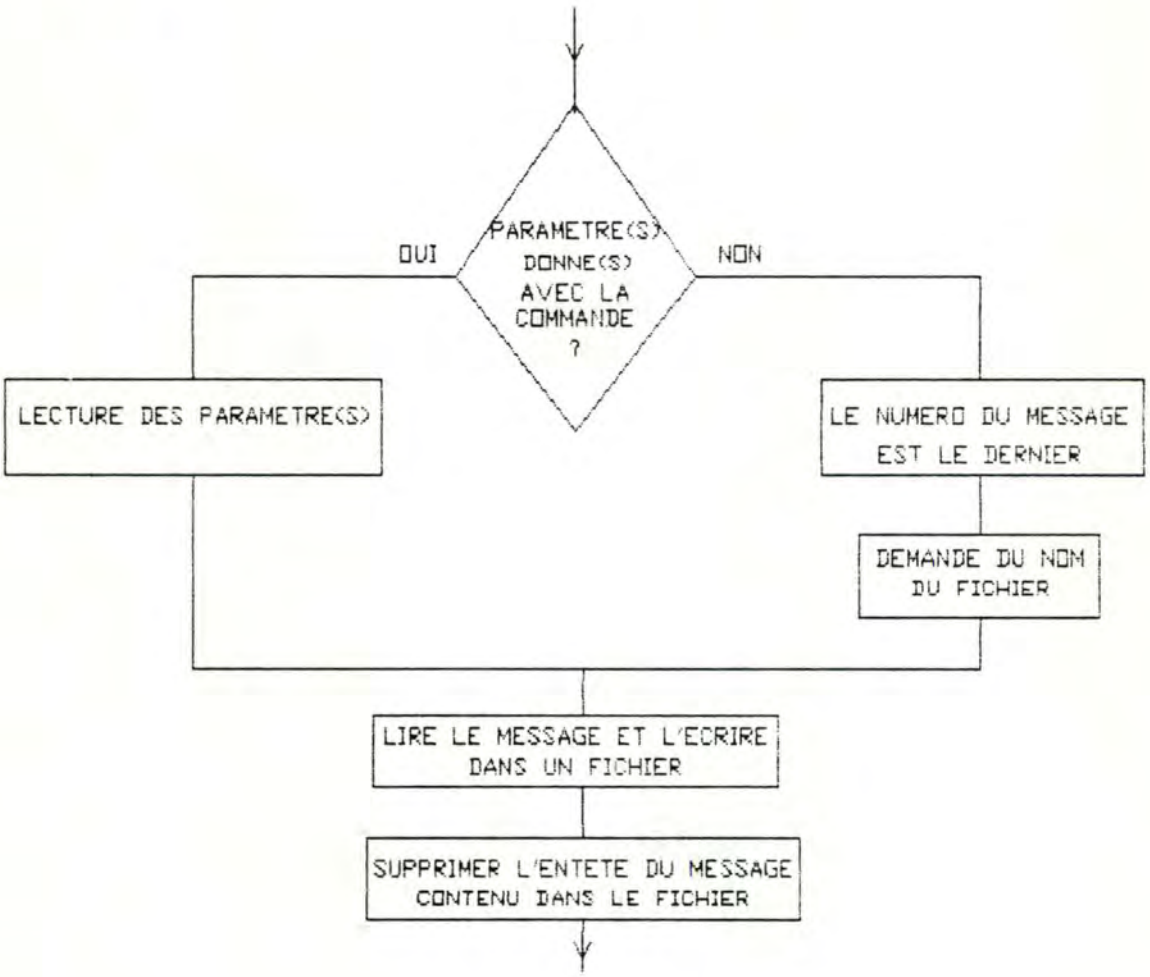
'OK'

II.2.14 Module WFILE

Utilité

Permettre d'écrire un message sans son header dans un fichier. Il s'agit de l'implémentation de la commande 'WriteFile' de EASY-MAIL.

Algorithme



Description des fonctions définies dans ce module

F_Wfile ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'WriteFile' de EASY-MAIL.

Valeur retournée

'OK'

```
F_ExtractMessage (number,nom_fichier)
char *number,*nom_fichier;
```

Spécification

Cette fonction extrait le message dont le numéro est contenu dans la variable 'number' et l'écrit dans un fichier dont le nom est contenu dans la variable 'nom_fichier'.

Valeur retournée

'OK'

```
F_Para (number)
char *number;
```

Spécification

Cette fonction permet de lire les éventuels paramètres [number] et [@file-name] rentrés avec le nom de la commande. Elle place [number] dans la variable 'number' et [@file-name] dans la variable 'file_name'. Si la close [number] n'est pas donnée, alors ce sera le dernier message qui sera utilisé (valeur par défaut) tandis que si la close [@file-name] n'est pas donnée, l'utilisateur sera interrogé sur le nom du fichier.

Valeur retournée

'OK'

```
F_InFile ()
```

Spécification

Cette fonction permet à l'utilisateur de donner un nom de fichier. Celui-ci sera placé dans la variable 'file_name'.

Valeur retournée

'OK'

```
F_Header (nom)
char *nom;
```

Spécification

Cette fonction permet de supprimer le header d'un message contenu dans le fichier dénommé par la variable 'nom' et elle renomme ensuite ce fichier avec le nom situé dans la variable 'file_name'.

Valeur retournée

'OK'

```
F_DelHeader (nom,n)
char *nom;
int n;
```

Spécification

Cette fonction recopie le fichier dénommé par la variable 'nom' dans le fichier dénommé par la variable 'file_name' sans les 'n' premières lignes.

Valeur retournée

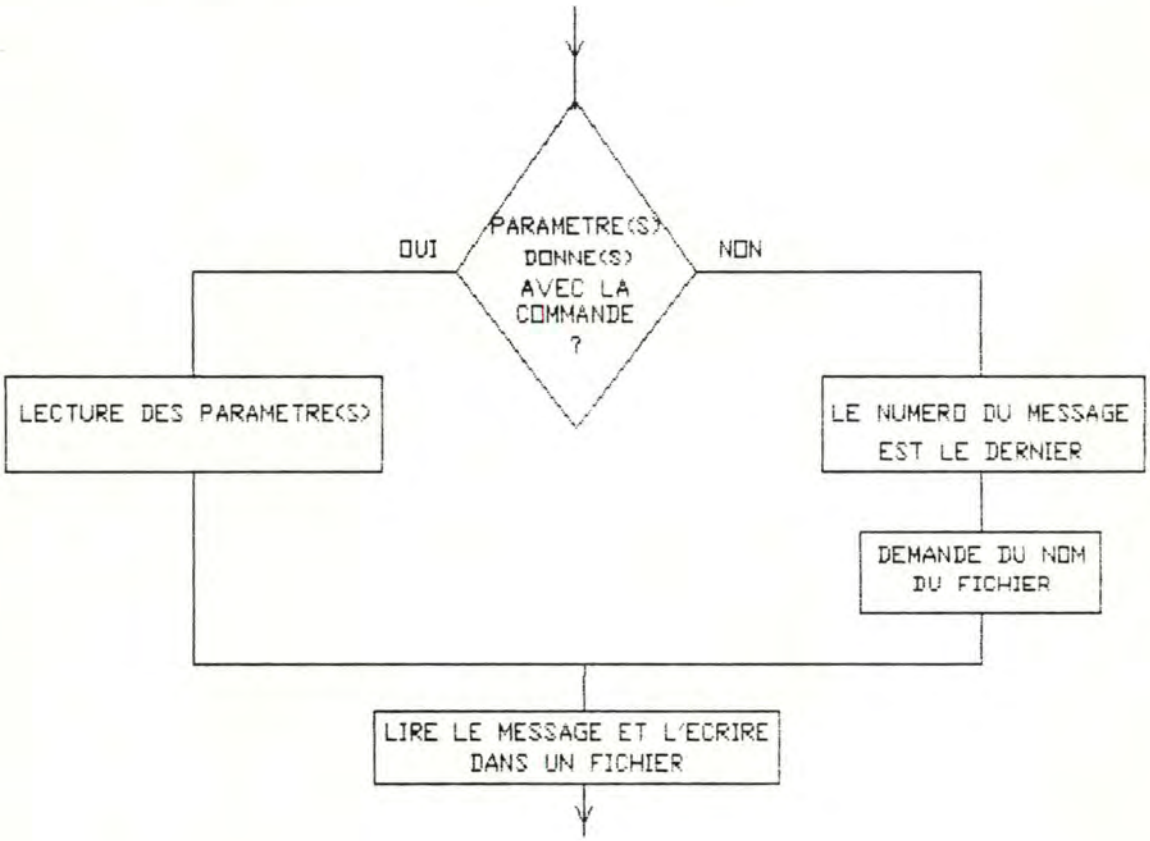
'OK'

II.2.15 Module WMES

Utilité

Permettre d'écrire un message dans un fichier. Il s'agit de l'implémentation de la commande 'WriteMessage' de EASY-MAIL.

Algorithme



Description des fonctions définies dans ce module

F_Wmes ()

Spécification

Il s'agit de la fonction principale implémentant la commande 'WriteMessage' de EASY-MAIL.

Valeur retournée

'OK'

II.3 Description des fichiers

II.3.1 Le fichier longueur.h

Le fichier intitulé longueur.h contient la définition des longueurs des variables et champs principaux de Easy-Mail. Un changement de longueur n'entraînera donc pas d'autre modification que celle de ce seul fichier.

II.3.2 Le fichier record.h

Le fichier intitulé record.h contient la définition des structures utilisées par EASY-MAIL.

II.4 Descriptiondes autres programmes

II.4.1 Le programme InitMailbook

Le programme intitulé InitMailbook permet d'initialiser le mailbook ainsi que les fichiers d'index d'un nouveau utilisateur en y inscrivant un premier enregistrement. L'utilisateur est interrogé et doit donner les différents champs composant l'enregistrement.

II.4.2 Le programme BuildIndex

Le programme BuildIndex met à jour les fichiers d'index lorsque le Mailbook a été modifié manuellement. Il doit être exécuté après chaque correction du Mailbook par un éditeur.

II.5 Description du Mailbook

II.5.1 le Mailbook

Le Mailbook est le fichier contenant l'adresse professionnelle ainsi que les adresses électroniques des éventuels destinataires. Il est composé d'enregistrements, chaque enregistrement décrivant un destinataire possible. Un enregistrement est composé de plusieurs lignes. Il commence par la ligne "/BEG" (BEG pour begin) et se termine par la ligne "/END". La deuxième ligne de l'enregistrement donne le nom et le prénom du destinataire. La troisième ligne spécifie l'institut,la ville et le pays. Ce dernier est précisé en utilisant des abréviations. Ensuite viennent les adresses électroniques du destinataire. Une adresse électronique commence toujours par la ligne "/MAI" (MAI pour mail). Sur la ligne suivante se trouve le nom d'utilisateur, le nom du noeud et le nom du réseau. Si une des adresses nécessite une passerelle, l'adresse de cette dernière est donnée à la suite et de la même façon, à part qu'elle ne commence plus par la ligne "/MAI" mais par "/GAT" (GAT pour gateway).

On aurait pu éviter la répétitivité des adresses des passerelles en les mettant dans un fichier séparé au lieu d'associer, à chaque adresse électronique nécessitant une passerelle, l'adresse de cette passerelle. La non-répétitivité des passerelles aurait simplifié une éventuelle correction de l'une d'elles. Cependant, l'utilisation d'un éditeur permet de résoudre ce problème. En effet, un éditeur permet de changer une portion de texte par une autre partout dans le fichier.

Une description détaillée des différents champs d'un enregistrement est donnée ci-après:

Nom du champs	N° colonne	long. max.	N° ligne
Nom	1	20	2
Prénom	22	12	2
Institut	1	30	3
Ville	32	20	3
Pays	53	3	3
Nom d'utilisateur	1	16	X
Nom du noeud	18	35	X
Nom du reseau	54	61	X

II.5.2 Description des fichiers d'index

Il existe plusieurs clés d'accès pour accéder à un enregistrement. On peut accèder à un enregistrement en précisant le nom du destinataire, le prénom, l'institut, la ville, ou même le pays. A chacune de ces clés d'accès correspond un fichier appelé respectivement 'surname.ind', 'first_name.ind', institute.ind', 'city.ind' et 'country.ind'. Chaque ligne de ces fichiers est constituée d'une valeur de la clé et du numero du byte où débute l'enregistrement correspondant à cette valeur de clé.

Une description détaillée des différents champs d'un enregistrement est également donnée ci-dessous:

Nom du champs	débute à la colonne	long. max.
Clé	1	30
numéro du byte	32	6

ANNEXE III : Textes Sources des programmes

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           addrvalid
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE        [0.2]   17-Nov-1987     Cree.
.COMMENTAIRES
        ce module permet de changer les flags d'une adresse E_mail
        du mailbook pour les mettre dans le status "adresse validee".

 ----------------------------------------------------------------------*
# define ProteuSubsys
#include "uif:proteus.h"
#include "uif:form.h"
#include "longueur.h"
#include "uif:help.h"
#include "uif:serv.h"
#include "uif:user.h"
#include "uif:lexa.h"
#include "uif:term.h"
#include "record.h"

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction est la fonction principale implementant la commande
'AddressValidate' du mailer.
----------------------------------------------------------------------------

F_Val_adr()
{
char choix[2],buffer[MAXIMUM];
int nb_trouve,prof_choisi,mail_choisi,code,i;
struct caracteristique car;
struct trouve result;

ENTER(F_Val_adr);
strcpy(car.surname,"");
strcpy(car.first_name,"");
strcpy(car.city,"");
strcpy(car.country,"");
strcpy(car.institute,"");
strcpy(car.network,"");
strcpy(car.choix,"");

if (MoreFields())
  {
  PROMPT("",0);
  strcpy(buffer,REPLY);
  if (buffer[0]=='#') {
                      i=1;
                      while(buffer[i-1]!='\0')
                        {
                        car.institute[i-1]=buffer[i];
                        i++;
                        }
                      }
  else strcpy(car.surname,buffer);
  if (MoreFields())
  {
```

```
  PROMPT("",0);
  strcpy(buffer,REPLY);
  if (buffer[0]=='.') {
                      i=1;
                      while(buffer[i-1]!='\0')
                        {
                        car.network[i-1]=buffer[i];
                        i++;
                        }
                      }
     }
F_Network(&car);
}
else {
     debut:F_Charast(&car);
     }
F_Lecture(&car,&nb_trouve,&result);

if(nb_trouve==0)
 {
 F_Message("recipient don't exist in the mailbook",&code);
 if (code == -1) goto fin;
 else goto debut;
}
 else {
       if(nb_trouve==4)
                  {
                  F_Message("recipient charasteristics are incomplete",&c
                  if (code == -1) goto fin;
                  else goto debut;
                  }
       else {
            if (nb_trouve==1) prof_choisi = 0;
            else {
                 F_ProfAddress(&result,nb_trouve,&prof_choisi);
                 if (prof_choisi == -1) goto fin;
                 }
            if (strcmp(car.choix,"+")!=0)
                 {F_MailAddress(&result,prof_choisi,&mail_choisi,car.choix
                  if (mail_choisi == -1) goto fin;
                 }
            else {
                 if (strcmp(car.network,"")==0) mail_choisi = 0;
                 else
                 if (strcmp(car.network,result.network[prof_choisi][0])==0
                 mail_choisi = 0;
                 else
                 if (strcmp(car.network,result.network[prof_choisi][1])==0
                 mail_choisi = 1;
                 else
                 if (strcmp(car.network,result.network[prof_choisi][2])==0
                 mail_choisi = 2;
                 else EXIT(NOK);
                 }
            F_MajFlag(&result,prof_choisi,mail_choisi);
          }
     }
fin:EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
cette fonction met les flags d'une E_mail adresse dans l'etat
"adresse validee".
-------------------------------------------------------------------------*/
F_MajFlag(result,prof_choisi,mail_choisi)
struct trouve *result;
int prof_choisi,mail_choisi;

{
FILE *fil_des1,*fil_des2;
struct un_trouve ptr;
struct prof_1_record prof1;
struct prof_2_record prof2;
struct mail_adr_record mail;
int i;
char code[CODE];
ENTER(F_MajFlag);


/* Ecriture */

fil_des1=fopen("mailbook.txt","r");
fil_des2=fopen("mailbook.txt","w");
while( ftell(fil_des1)!=result->location[prof_choisi])
  {
  fgets(code,sizeof(code),fil_des1);
  F_Recopiage(fil_des1,fil_des2);
  }
/* modification des flags */

strcpy(result->f_user[prof_choisi][mail_choisi],"+");
strcpy(result->f_node[prof_choisi][mail_choisi],"+");
strcpy(result->f_network[prof_choisi][mail_choisi],"+");
if(strcmp(result->f_ga_user[prof_choisi][mail_choisi],"")!=0)
  {
  strcpy(result->f_ga_user[prof_choisi][mail_choisi],"+");
  strcpy(result->f_ga_node[prof_choisi][mail_choisi],"+");
  strcpy(result->f_ga_network[prof_choisi][mail_choisi],"+");
  }

/*  initialisation de la structure d'ecriture */

F_Blanc(ptr.surname,SURNAME);
F_Blanc(ptr.first_name,FIRST_NAME);
F_Blanc(ptr.institute,INSTITUTE);
F_Blanc(ptr.city,CITY);
F_Blanc(ptr.country,COUNTRY);
for(i=0;i<3;++i) {
                 F_Blanc(ptr.user[i],USER_NAME);
                 F_Blanc(ptr.node[i],NODE_NAME);
                 F_Blanc(ptr.network[i],NETWORK);
                 F_Blanc(ptr.ga_user[i],USER_NAME);
                 F_Blanc(ptr.ga_node[i],NODE_NAME);
                 F_Blanc(ptr.ga_network[i],NETWORK);
                 }

/*  passage des donnees dans la structure d'ecriture */

strcpy(ptr.surname,result->surname[prof_choisi]);
strcpy(ptr.first_name,result->first_name[prof_choisi]);
strcpy(ptr.institute,result->institute[prof_choisi]);
strcpy(ptr.city,result->city[prof_choisi]);
```

```
strcpy(ptr.country,result->country[prof_choisi]);
ptr.location = result->location[prof_choisi];
for(i=0;i<3;++i) {
                 strcpy(ptr.user[i],result->user[prof_choisi][i]);
                 strcpy(ptr.node[i],result->node[prof_choisi][i]);
                 strcpy(ptr.network[i],result->network[prof_choisi][i]);
                 strcpy(ptr.ga_user[i],result->ga_user[prof_choisi][i]);
                 strcpy(ptr.ga_node[i],result->ga_node[prof_choisi][i]);
                 strcpy(ptr.ga_network[i],result->ga_network[prof_choisi][
                 strcpy(ptr.f_user[i],result->f_user[prof_choisi][i]);
                 strcpy(ptr.f_node[i],result->f_node[prof_choisi][i]);
                 strcpy(ptr.f_network[i],result->f_network[prof_choisi][i]
                 strcpy(ptr.f_ga_user[i],result->f_ga_user[prof_choisi][i]
                 strcpy(ptr.f_ga_node[i],result->f_ga_node[prof_choisi][i]
                 strcpy(ptr.f_ga_network[i],result->f_ga_network[prof_choi
                 }

F_Ecriture(&ptr,fil_des2);

fgets(code,sizeof(code),fil_des1);
fgets(&prof1,sizeof(prof1),fil_des1);
fgets(&prof2,sizeof(prof2),fil_des1);

while(strcmp(fgets(code,sizeof(code),fil_des1),"/END\n")!=0)
       fgets(&mail,sizeof(mail),fil_des1);

while(fgets(code,sizeof(code),fil_des1)!=NULL)
  F_Recopiage(fil_des1,fil_des2);

fclose(fil_des1);
fclose(fil_des2);

EXIT(OK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction recopie un enregistrement du mailbook (reference par
'fil_des1') dans un autre fichier (reference par 'fil_des2').
----------------------------------------------------------------------------*/

F_Recopiage(fil_des1,fil_des2)

FILE *fil_des1,*fil_des2;

{
  struct prof_1_record prof1;
  struct prof_2_record prof2;
  struct mail_adr_record mail;
  char code[CODE];
  ENTER(F_Recopiage);
  fputs("/BEG\n",fil_des2);
  fgets(&prof1,sizeof(prof1),fil_des1);
  fputs(&prof1,fil_des2);
  fgets(&prof2,sizeof(prof2),fil_des1);
  fputs(&prof2,fil_des2);
  while(strcmp(fgets(code,CODE,fil_des1),"/END\n")!=0)
      {
      fputs(code,fil_des2);
      fgets(&mail,sizeof(mail),fil_des1);
      fputs(&mail,fil_des2);
      }
  fputs(code,fil_des2);
```

```
  EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE            Module
.NAME            append
.LANGUAGE        C
.ENVIRONNEMENT   Easy_mail.
.AUTEUR          Bruno Loodts
.HISTOIRE         [0.1]   02-dec-1987     Cree.
.COMMENTAIRES
                 Cet module permet a un utilisateur de rentrer un nouvel
                 recipient dans son 'mailbook'.Il s'agit donc de
                 l'implementation de la commande 'AppendMailbook'.

 ------------------------------------------------------------------------*
# define ProteuSubsys
#include "uif:proteus.h"
#include "uif:form.h"
#include "longueur.h"
#include "uif:help.h"
#include "uif:serv.h"
#include "uif:user.h"
#include "uif:lexa.h"
#include "uif:term.h"
#include "record.h"
#include "uif:util.h"

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction est la fonction principale implementant la commande
'AppendMailbook'
------------------------------------------------------------------------*

F_AppendMailbook()
{
struct un_trouve car;
int i;
ENTER(F_AppendMailbook);
/* initialisation */
strcpy(car.surname,"");
strcpy(car.first_name,"");
strcpy(car.institute,"");
strcpy(car.city,"");
strcpy(car.country,"");
for(i=0;i<3;i++)
  {
  strcpy(car.user[i],"");
  strcpy(car.node[i],"");
  strcpy(car.network[i],"");
  strcpy(car.ga_user[i],"");
  strcpy(car.ga_node[i],"");
  strcpy(car.ga_network[i],"");
  /* au depart les flags sont positionnes a '?' */
  strcpy(car.f_user[i],"?");
  strcpy(car.f_node[i],"?");
  strcpy(car.f_network[i],"?");
  strcpy(car.f_ga_user[i],"?");
  strcpy(car.f_ga_node[i],"?");
  strcpy(car.f_ga_network[i],"?");
  }

F_EnterProf(&car);
```

```
F_EnterMail(&car);
F_Change(&car);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de donner l'adresse professionnelle du recipient.
----------------------------------------------------------------------------*/
F_EnterProf(car)
struct un_trouve *car;

{
ENTER(F_Charast);
InitForm("recipient","recipient characteristics",0);
AddField("surname",'a',"","recipient surname",SURNAME-1);
AddField("first_name",'a',"","recipient first_name",FIRST_NAME-1);
AddField("institute",'a',"","institute name",INSTITUTE-1);
AddField("city",'a',"","city name",CITY-1);
AddField("country",'a',"","country name",COUNTRY-1);
ChangeForm();
GetField("surname",car->surname,SURNAME-1);
GetField("first_name",car->first_name,FIRST_NAME-1);
GetField("institute",car->institute,INSTITUTE-1);
GetField("city",car->city,CITY-1);
GetField("country",car->country,COUNTRY-1);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de donner les E_Mail adresses du recipient.
----------------------------------------------------------------------------*/
F_EnterMail(result)

struct un_trouve *result;

{
ENTER(F_EnterMail);
InitForm("address","E_Mail address",0);
AddField("1_user",'a',"","",USER_NAME-1);
AddField("1_node",'a',"","",NODE_NAME-1);
AddField("1_network",'a',"","",NETWORK-1);
AddField("1_ga_user",'a',"","",USER_NAME-1);
AddField("1_ga_node",'a',"","",NODE_NAME-1);
AddField("1_ga_network",'a',"","",NETWORK-1);
AddField("2_user",'a',"","",USER_NAME-1);
AddField("2_node",'a',"","",NODE_NAME-1);
AddField("2_network",'a',"","",NETWORK-1);
AddField("2_ga_user",'a',"","",USER_NAME-1);
AddField("2_ga_node",'a',"","",NODE_NAME-1);
AddField("2_ga_network",'a',"","",NETWORK-1);
AddField("3_user",'a',"","",USER_NAME-1);
AddField("3_node",'a',"","",NODE_NAME-1);
AddField("3_network",'a',"","",NETWORK-1);
AddField("3_ga_user",'a',"","",USER_NAME-1);
AddField("3_ga_node",'a',"","",NODE_NAME-1);
AddField("3_ga_network",'a',"","",NETWORK-1);
ChangeForm();
GetField("1_user",result->user[0],USER_NAME-1);
GetField("1_node",result->node[0],NODE_NAME-1);
GetField("1_network",result->network[0],NETWORK-1);
GetField("1_ga_user",result->ga_user[0],USER_NAME-1);
```

```
GetField("1_ga_node",result->ga_node[0],NODE_NAME-1);
GetField("1_ga_network",result->ga_network[0],NETWORK-1);
GetField("2_user",result->user[1],USER_NAME-1);
GetField("2_node",result->node[1],NODE_NAME-1);
GetField("2_network",result->network[1],NETWORK-1);
GetField("2_ga_user",result->ga_user[1],USER_NAME-1);
GetField("2_ga_node",result->ga_node[1],NODE_NAME-1);
GetField("2_ga_network",result->ga_network[1],NETWORK-1);
GetField("3_user",result->user[2],USER_NAME-1);
GetField("3_node",result->node[2],NODE_NAME-1);
GetField("3_network",result->network[2],NETWORK-1);
GetField("3_ga_user",result->ga_user[2],USER_NAME-1);
GetField("3_ga_node",result->ga_node[2],NODE_NAME-1);
GetField("3_ga_network",result->ga_network[2],NETWORK-1);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction ecrit dans le mailbook un nouveau recipient et ses
E_mail adresses
REMARQUE: les donnees a ecrire sont converties en minuscules.
----------------------------------------------------------------------------*/

F_Change(ptr)

struct un_trouve *ptr;

{
FILE *file_des;
int i;
ENTER(F_Change);
file_des=fopen("mailbook.txt","a");
ptr->location = ftell(file_des);
F_Ecriture(ptr,file_des);
fclose(file_des);
/* mise-a-jour des fichiers d'index */
F_Index(ptr);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction ecrit un enregistrement dans le mailbook a l'endroit
indiquer par ptr->location.
----------------------------------------------------------------------------*/

F_Ecriture(ptr,file_des)
struct un_trouve *ptr;
FILE *file_des;
{
int i;
ENTER(F_Ecriture);


/* conversion eventuelle em minuscules */

LowerCase(ptr->surname);
LowerCase(ptr->first_name);
LowerCase(ptr->institute);
LowerCase(ptr->city);
LowerCase(ptr->country);
for(i=0;i<3;i++)
```

```
  {
  if(strcmp(ptr->user[i],"")!=0)
    {
     LowerCase(ptr->user[i]);
     LowerCase(ptr->node[i]);
     LowerCase(ptr->network[i]);
     if(strcmp(ptr->ga_user[i],"")!=0)
        {
        LowerCase(ptr->ga_user[i]);
        LowerCase(ptr->ga_node[i]);
        LowerCase(ptr->ga_network[i]);
        }
    }
 }
/* positionement au bon endroit dans le mailbook */

fseek(file_des,ptr->location,0);
/* ecriture du recipient */
F_Ajoute(ptr->surname,SURNAME);
F_Ajoute(ptr->first_name,FIRST_NAME);
F_Ajoute(ptr->institute,INSTITUTE);
F_Ajoute(ptr->city,CITY);
F_Ajoute(ptr->country,COUNTRY);
fputs("/BEG\n",file_des);
fputs(ptr->surname,file_des);
fputs(" ",file_des);
fputs(ptr->first_name,file_des);
fputs("\n",file_des);
fputs(ptr->institute,file_des);
fputs(" ",file_des);
fputs(ptr->city,file_des);
fputs(" ",file_des);
fputs(ptr->country,file_des);
fputs("\n",file_des);

for(i=0;i<3;i++)
  {
  if(strcmp(ptr->user[i],"")!=0)
   {
    F_Ajoute(ptr->user[i],USER_NAME);
    F_Ajoute(ptr->node[i],NODE_NAME);
    F_Ajoute(ptr->network[i],NETWORK);
    fputs("/MAI\n",file_des);
    fputs(ptr->user[i],file_des);
    fputs(" ",file_des);
    fputs(ptr->f_user[i],file_des);
    fputs(" ",file_des);
    fputs(ptr->node[i],file_des);
    fputs(" ",file_des);
    fputs(ptr->f_node[i],file_des);
    fputs(" ",file_des);
    fputs(ptr->network[i],file_des);
    fputs(" ",file_des);
    fputs(ptr->f_network[i],file_des);
    fputs("\n",file_des);

    if(strcmp(ptr->ga_user[i],"")!=0){
        F_Ajoute(ptr->ga_user[i],USER_NAME);
        F_Ajoute(ptr->ga_node[i],NODE_NAME);
        F_Ajoute(ptr->ga_network[i],NETWORK);
        fputs("/GAT\n",file_des);
```

```
        fputs(ptr->ga_user[i],file_des);
        fputs(" ",file_des);
        fputs(ptr->f_ga_user[i],file_des);
        fputs(" ",file_des);
        fputs(ptr->ga_node[i],file_des);
        fputs(" ",file_des);
        fputs(ptr->f_ga_node[i],file_des);
        fputs(" ",file_des);
        fputs(ptr->ga_network[i],file_des);
        fputs(" ",file_des);
        fputs(ptr->f_ga_network[i],file_des);
        fputs("\n",file_des);

        }
    }
 }
fputs("/END\n",file_des);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Cette fonction met a jour les fichiers d'index en fonction
        d'un nouveau recipient.
 ----------------------------------------------------------------------------*


F_Index(ptr)

struct un_trouve *ptr;

{
int i;
ENTER(F_Index);
/* mise-a-jour du fichier d'index "surname.ind" */
F_Maj("surname.ind",ptr->surname,ptr->location);

/* mise-a-jour du fichier d'index "first_name.ind" */
F_Maj("first_name.ind",ptr->first_name,ptr->location);

/* mise-a-jour du fichier d'index "institute.ind" */
F_Maj("institute.ind",ptr->institute,ptr->location);

/* mise-a-jour du fichier d'index "city.ind" */
F_Maj("city.ind",ptr->city,ptr->location);

/* mise-a-jour du fichier d'index "country.ind" */
F_Maj("country.ind",ptr->country,ptr->location);

/* mise-a-jour du fichier d'index "network.ind" */
for (i=0;i<3;i++)
    if (strcmp(ptr->network[i],"")!=0)
        F_Maj("network.ind",ptr->network[i],ptr->location);

EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
          Cette fonction met a jour un fichier d'index
----------------------------------------------------------------------------*/

F_Maj(file,key,location)
```

```
char *file,*key;
int location;

{
int i;
FILE *fil_des1,*fil_des2;

struct index_record enr1;
struct index_record enr2;


char key2[KEY],
     loc2[LOCATION],
     name[14+FILE_NAME];

ENTER(F_Maj);
F_Blanc(key2,KEY);
F_Blanc(loc2,LOCATION);
itoa(location,loc2,LOCATION);
strcpy(key2,key);

F_Ajoute(key2,KEY);
F_Ajoute(loc2,LOCATION);


strncpy(enr1.cle,key2,KEY-1);
strncpy(enr1.blanc," ",1);
strncpy(enr1.location,loc2,LOCATION-1);
strncpy(enr1.saut,"\n",1);
strncpy(enr1.fin,"\0",1);
strcpy(name,file);
fil_des1=fopen(name,"r");
fil_des2=fopen(name,"w");
debut:if (fgets(&enr2,sizeof(enr2),fil_des1)==NULL)
                    {
                    F_Write(fil_des2,&enr1);
                    }
else {
     if (strcmp(enr1.cle,enr2.cle)<0)
       {
       F_Write(fil_des2,&enr1);
       F_Write(fil_des2,&enr2);
       while(fgets(&enr2,sizeof(enr2),fil_des1)!=NULL)
              {
              F_Write(fil_des2,&enr2);
              }
     }
     else {
          F_Write(fil_des2,&enr2);
          goto debut;
        }
   }

fclose(fil_des1);
fclose(fil_des2);

EXIT(OK);
}
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction ecrit un enregistrement dans un fichier
------------------------------------------------------------------------------*/
F_Write(file_des,ptr)
FILE *file_des;
struct index_record *ptr;
{
ENTER(F_Write);
fputs(ptr,file_des);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction rajoute des blancs a la fin du string c'est-a-dire apres
le caractere '\0' sur longueur-1
------------------------------------------------------------------------------*

F_Ajoute(chaine,longueur)
char *chaine;
int longueur;

{
ENTER(F_Ajoute);
longueur=longueur-1;
while (strlen(chaine)<longueur) strcat(chaine," ");
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction rempli le string de blancs
------------------------------------------------------------------------------*/

F_Blanc(chaine,longueur)
char chaine[];
int longueur;

{
int i;
ENTER(F_Blanc);
longueur=longueur;
for(i=0;i<longueur;i++) chaine[i] = ' ';
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           Dir
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE       [0.2]   09-Oct-1987     Cree.
.COMMENTAIRES

ce module permet d'afficher la liste des messages recus

-------------------------------------------------------------------------------

#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:user.h"
#include        "uif:serv.h"
#include        "uif:term.h"
#include        "uif:lexa.h"
#include        "longueur.h"
#if VMS
#define NEW "newmail" /* nom du folder en VMS */
#define OLD "mail"    /* nom du folder en VMS */
#define SPAWN PutDCL("@dir.com")
#define LINE1 "$define/user Sys$output dir.list"
#define LINE2 "mail"
#define LINE3 "dir "
#define LINE4 "exit"
#define TEST  "%MAIL-E-NOTEXIST"
#else
#define NEW
#define OLD
#define SPAWN
#define LINE1
#define LINE2
#define LINE3
#define LINE4
#define TEST
#endif

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction affiche la liste des messages recus sous la forme
adresse de l'envoyeur/date/sujet du message.
Il s'agit de la fonction principale implementant la commande 'DirMessage
-------------------------------------------------------------------------------

F_Dir   ()
{
FILE *fil_ptr;
char folder_systeme[FOLDER],folder_easy_mail[FOLDER],commande[60];
ENTER(F_Dir);
strcpy(folder_systeme,NEW);     /* folder systeme par defaut */
strcpy(folder_easy_mail,"NEW"); /* folder easy_mail par defaut */
if (MoreFields()) {
                PROMPT("",0);
                if (strcmp(REPLY,"OLD")==0 || strcmp(REPLY,"old")==0)
                        {
                        strcpy(folder_systeme,OLD);
                        strcpy(folder_easy_mail,"OLD");
```

```
                              }
                    }
OpenOut("dir.com",0);
WriteOut(LINE1);
WriteOut(LINE2);
strcpy(commande,LINE3);
strcat(commande,folder_systeme);
WriteOut(commande);
WriteOut(LINE4);
CloseOut();
SPAWN;
OpenIn("dir.list",0);
ReadIn(commande,sizeof(commande));
CloseIn();
if(strncmp(commande,TEST,sizeof(TEST)-1)!=0) ShowFile("dir.list");
else {
     strcpy(commande,"THE FOLDER '");
     strcat(commande,folder_easy_mail);
     strcat(commande,"' IS EMPTY");
     ShowText(10,25,commande,_BOLD_);
     }
delete("dir.list");
delete("dir.com");
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           dmes
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE        [0.2]   23-Nov-1987     Cree.
.COMMENTAIRES

          ce module permet d'effacer un message.

 ---------------------------------------------------------------------------
#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:help.h"
#include        "uif:user.h"
#include        "uif:term.h"
#include        "uif:serv.h"
#include        "longueur.h"
#if VMS
#define SPAWN PutDCL("@dmes.com")
#define LAST  "999"
#define LINE0 "$define/user Sys$output deletemessage.list"
#define LINE1 "mail"
#define LINE2 "select mail"
#define LINE3 "del "
#define LINE4 "exit"

#else
#define SPAWN PutUNIX("rmes");
#define LAST
#define LINE1
#define LINE2
#define LINE3
#define LINE4
#endif

extern char file_name[];

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet d'effacer un message. Il s'agit de la fonction
principale implementant la commande 'DeleteMessage'.
---------------------------------------------------------------------------*

F_Dmes  ()
{
char number[NUMBER];
char commande[60];
ENTER(F_Dmes);
if(MoreFields()) {
                 PROMPT("",0);
                 if(F_Chiffre(REPLY)==OK) strcpy(number,REPLY);
                 else goto suite;
                 }
else {
     suite:PROMPT("give the number of the message ('a' to aboard)",0);
     if((strcmp(REPLY,"a")==0)||(strcmp(REPLY,"A")==0)) goto fin;
     if(F_Chiffre(REPLY)==OK) strcpy(number,REPLY);
     else goto suite;
```

```
    }

OpenOut("dmes.com",0);
WriteOut(LINE0);
WriteOut(LINE1);
WriteOut(LINE2);
strcpy(commande,LINE3);
strcat(commande,number);
WriteOut(commande);
WriteOut(LINE4);
CloseOut();
SPAWN;
delete("dmes.com");
delete("deletemessage.list");
fin : EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           forward
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE       [0.1]   01-dec-1987.
.COMMENTAIRES

        Ce module permet de renvoyer un message recu vers un ou
        plusieurs destinataires.

 ------------------------------------------------------------------------*/
#define ProteuSubsys
#include "uif:proteus.h"
#include "uif:lexa.h"
#include "pro:longueur.h"
#include "uif:serv.h"
#include "uif:user.h"
#include "pro:record.h"

extern char file_name[];
extern char sujet[];

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction est la fonction principale implementant la commande
'ForwardMessage'
------------------------------------------------------------------------------*/
F_Fmes()
{
int *nombre,i,sw;
char buffer[500],number[NUMBER];
struct caracteristique car[100];
ENTER(F_Fmes);
(*nombre) = 0;
sw = 0;
strcpy(number,"");
if(MoreFields()) {
                 PROMPT("",0);
                 if(F_Chiffre(REPLY)==OK) strcpy(number,REPLY);
                 else {
                      strcpy(buffer,REPLY);
                      strcat(buffer," ");
                      goto suite;
                      }
                 }

if(MoreFields()) strcpy(buffer,WholeInput());
else F_List(buffer);
suite:F_Extraction(nombre,buffer,car);
F_Sujet();
F_File_Name();
F_ExtractMessage(number,file_name);
for(i=0;i<(*nombre);i++)
   {
   F_Network(&car[i]);
   F_Envoi(&car[i],&sw);
   }
delete(file_name);
```

```
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE               Module
.NOM                lecture
.LANGUAGE           C
.ENVIRONNEMENT      Easy_mail
.AUTEUR             Bruno Loodts
.HISTORE            [0.3] 25-sep-1987     Cree.
.COMMENTAIRES
        Ce module lit dans le fichier mailbook.txt les E_mail adresses
        concernant une adresse professionelle.

.REMARQUES
        Cette adresse professionelle est convertie en minuscule
        pour les besoins de la recherche.
        Les parametres de recherche vides doivent commencer par ' ' ou '
        pour le bon fonctionnement de la recherche.

-----------------------------------------------------------------------------
#define             ProteuSubsys
#include            "uif:proteus.h"
#include            "uif:serv.h"
#include            "uif:lexa.h"
#include            "longueur.h"
#include            "record.h"

F_Lecture(car_ptr,nombre_trouve,trouve_ptr)
int *nombre_trouve;
struct caracteristique *car_ptr;
struct trouve *trouve_ptr;

{
ENTER(F_Lecture);
/* conversion eventuelle en minuscule */
LowerCase(car_ptr->surname);
LowerCase(car_ptr->first_name);
LowerCase(car_ptr->institute);
LowerCase(car_ptr->network);
LowerCase(car_ptr->city);
LowerCase(car_ptr->country);



if ((car_ptr->surname[0]!='\0') && (car_ptr->surname[0]!=' '))
F_Recherche("surname.ind",car_ptr->surname,car_ptr,
            nombre_trouve,trouve_ptr);

else {
if ((car_ptr->first_name[0]!='\0') && (car_ptr->first_name[0]!=' '))
F_Recherche("first_name.ind",car_ptr->first_name,car_ptr,
            nombre_trouve,trouve_ptr);
else {
if ((car_ptr->institute[0]!='\0') && (car_ptr->institute[0]!=' '))
F_Recherche("institute.ind",car_ptr->institute,car_ptr,
            nombre_trouve,trouve_ptr);
else {
if ((car_ptr->city[0]!='\0') && (car_ptr->city[0]!=' '))
F_Recherche("city.ind",car_ptr->city,car_ptr,
            nombre_trouve,trouve_ptr);
else {
```

```
if ((car_ptr->country[0]!='\0') && (car_ptr->country[0]!=' '))
F_Recherche("country.ind",car_ptr->country,car_ptr,
          nombre_trouve,trouve_ptr);
else {
if ((car_ptr->network[0]!='\0') && (car_ptr->network[0]!=' '))
F_Recherche("network.ind",car_ptr->network,car_ptr,
          nombre_trouve,trouve_ptr);
else (*nombre_trouve) = 0;
}}}}}
EXIT(OK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction fait une recherche sur une clef d'index precisee
---------------------------------------------------------------------------*/

F_Recherche(file,cle,car_ptr,nombre_trouve,trouve_ptr)
int *nombre_trouve;
char *file,*cle;
struct caracteristique *car_ptr;
struct trouve *trouve_ptr;

{
int i,location;
FILE *fil_des;
struct index_record enr;
struct un_trouve buf;
char key[KEY],name[14+FILE_NAME];

ENTER(F_Recherche);
strcpy(name,file);
fil_des=fopen(name,"r");
(*nombre_trouve) = 0;
strcpy(key,cle);
F_Ajoute(key,KEY);
debut:if((*nombre_trouve)!=3)
  {
  lecture:if(fgets(&enr,sizeof(enr),fil_des)!=NULL)
  {
  if(strncmp(enr.cle,key,KEY-1)<0) goto lecture;
  if(strncmp(enr.cle,key,KEY-1)==0)
     {
     location=atoi(enr.location);
     F_Mailbook(location,&buf);
     if(F_Compare(&buf,car_ptr)==NOK) goto lecture;
     else {
          strcpy(trouve_ptr->surname[*nombre_trouve],buf.surname);
          strcpy(trouve_ptr->first_name[*nombre_trouve],buf.first_name);
          strcpy(trouve_ptr->institute[*nombre_trouve],buf.institute);
          strcpy(trouve_ptr->city[*nombre_trouve],buf.city);
          strcpy(trouve_ptr->country[*nombre_trouve],buf.country);
          trouve_ptr->location[*nombre_trouve] = location;
          for(i=0;i<3;i++)
              {
              strcpy(trouve_ptr->user[*nombre_trouve][i],buf.user[i]);
              strcpy(trouve_ptr->f_user[*nombre_trouve][i],buf.f_user[i]
              strcpy(trouve_ptr->node[*nombre_trouve][i],buf.node[i]);
              strcpy(trouve_ptr->f_node[*nombre_trouve][i],buf.f_node[i]
              strcpy(trouve_ptr->network[*nombre_trouve][i],buf.network[
              strcpy(trouve_ptr->f_network[*nombre_trouve][i],buf.f_netw
              strcpy(trouve_ptr->ga_user[*nombre_trouve][i],buf.ga_user[
              strcpy(trouve_ptr->f_ga_user[*nombre_trouve][i],buf.f_ga_u
```

```
                strcpy(trouve_ptr->ga_node[*nombre_trouve][i],buf.ga_node[
                strcpy(trouve_ptr->f_ga_node[*nombre_trouve][i],buf.f_ga_n
                strcpy(trouve_ptr->ga_network[*nombre_trouve][i],buf.ga_ne
                strcpy(trouve_ptr->f_ga_network[*nombre_trouve][i],buf.f_g
                }
            (*nombre_trouve)++;
            goto debut;

            }
        }
      }
    }
fclose(fil_des);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction lit dans le fichier 'mailbook.txt' un enregistrement
donne par 'location'
----------------------------------------------------------------------------*/

F_Mailbook(location,buf)
int location;
struct un_trouve *buf;

{
char code[CODE];
int i;
FILE *fil_des;
struct prof_1_record prof1;
struct prof_2_record prof2;
struct mail_adr_record mail;
struct gate_adr_record gate;

ENTER(F_Mailbook);
for (i=0;i<3;i++)
  {
  strcpy(buf->user[i],"");
  strcpy(buf->node[i],"");
  strcpy(buf->network[i],"");
  strcpy(buf->ga_user[i],"");
  strcpy(buf->ga_node[i],"");
  strcpy(buf->ga_network[i],"");
  strcpy(buf->f_user[i],"");
  strcpy(buf->f_node[i],"");
  strcpy(buf->f_network[i],"");
  strcpy(buf->f_ga_user[i],"");
  strcpy(buf->f_ga_node[i],"");
  strcpy(buf->f_ga_network[i],"");
}

fil_des=fopen("mailbook.txt","r");
fseek(fil_des,location,0);
fgets(code,sizeof(code),fil_des);
fgets(&prof1,sizeof(prof1),fil_des);
strncpy(buf->surname,prof1.surname,SURNAME-1);
buf->surname[SURNAME-1]='\0';
strncpy(buf->first_name,prof1.first_name,FIRST_NAME-1);
buf->first_name[FIRST_NAME-1]='\0';

fgets(&prof2,sizeof(prof2),fil_des);
```

```
strncpy(buf->institute,prof2.institute,INSTITUTE-1);
buf->institute[INSTITUTE-1]='\0';
strncpy(buf->city,prof2.city,CITY-1);
buf->city[CITY-1]='\0';
strncpy(buf->country,prof2.country,COUNTRY-1);
buf->country[COUNTRY-1]='\0';
i = -1;
while(strcmp(fgets(code,sizeof(code),fil_des),"/END\n")!=0)
{
 if (strcmp(code,"/MAI\n")==0)
  {
   i++;
  fgets(&mail,sizeof(mail),fil_des);
  strncpy(buf->user[i],mail.user,USER_NAME-1);
  buf->user[i][USER_NAME-1]='\0';
  strncpy(buf->f_user[i],mail.f_user,FLAG-1);
  buf->f_user[i][FLAG-1]='\0';
  strncpy(buf->node[i],mail.node,NODE_NAME-1);
  buf->node[i][NODE_NAME-1]='\0';
  strncpy(buf->f_node[i],mail.f_node,FLAG-1);
  buf->f_node[i][FLAG-1]='\0';
  strncpy(buf->network[i],mail.network,NETWORK-1);
  buf->network[i][NETWORK-1]='\0';
  strncpy(buf->f_network[i],mail.f_network,FLAG-1);
  buf->f_network[i][FLAG-1]='\0';

}
else {
     fgets(&gate,sizeof(gate),fil_des);
     strncpy(buf->ga_user[i],gate.ga_user,USER_NAME-1);
     buf->ga_user[i][USER_NAME-1]='\0';
     strncpy(buf->f_ga_user[i],gate.f_ga_user,FLAG-1);
     buf->f_ga_user[i][FLAG-1]='\0';
     strncpy(buf->ga_node[i],gate.ga_node,NODE_NAME-1);
     buf->ga_node[i][NODE_NAME-1]='\0';
     strncpy(buf->f_ga_node[i],gate.f_ga_node,FLAG-1);
     buf->f_ga_node[i][FLAG-1]='\0';
     strncpy(buf->ga_network[i],gate.ga_network,NETWORK-1);
     buf->ga_network[i][NETWORK-1]='\0';
     strncpy(buf->f_ga_network[i],gate.f_ga_network,FLAG-1);
     buf->f_ga_network[i][FLAG-1]='\0';
     }

}
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction compare deux adresses professionnelles et
retourne OK si elles sont identiques
         NOK sinon
----------------------------------------------------------------------*/

F_Compare(buf,car)

struct un_trouve *buf;
struct caracteristique *car;

{
ENTER(F_Compare);
if ((car->surname[0]!='\0') && (car->surname[0]!=' '))
```

```
if (strcmp(car->surname,"")!=0)
  {
  F_Ajoute(car->surname,SURNAME);
  if(strcmp(car->surname,buf->surname)!=0) EXIT(NOK);
  }

if ((car->first_name[0]!='\0') && (car->first_name[0]!=' '))
  {
  F_Ajoute(car->first_name,FIRST_NAME);
  if(strcmp(car->first_name,buf->first_name)!=0) EXIT(NOK);
  }

if ((car->institute[0]!='\0') && (car->institute[0]!=' '))
  {
  F_Ajoute(car->institute,INSTITUTE);
  if (strcmp(car->institute,buf->institute)!=0) EXIT(NOK);
  }

if ((car->city[0]!='\0') && (car->city[0]!=' '))
  {
  F_Ajoute(car->city,CITY);
  if(strcmp(car->city,buf->city)!=0) EXIT(NOK);
  }

if ((car->country[0]!='\0') && (car->country[0]!=' '))
  {
  F_Ajoute(car->country,COUNTRY);
  if(strcmp(car->country,buf->country)!=0) EXIT(NOK);
  }

if ((car->network[0]!='\0') && (car->network[0]!=' '))
     {
     F_Ajoute(car->network,NETWORK);
     if( (strcmp(car->network,buf->network[0])!=0) &&
     (strcmp(car->network,buf->network[1])!=0) &&
     (strcmp(car->network,buf->network[2])!=0)) EXIT(NOK);
     }

EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NOM            Mailer
.LANGUAGE       C
.ENVIRONMENT    Easy_mail.
.AUTEUR         Bruno Loodts.
.HISTORE        [0.3]   25-Sep-1987     Cree.
.COMMENTAIRES   programme principal;initialisation.
-------------------------------------------------------------------------------

#define         ProteuSubsys                    /* pour eviter tous les heade
#include        "uif:proteus.h"                 /* pour les def. standards */
#include        "uif:dial.h"
#include        "uif:user.h"                    /* pour SaveArgs ()      */
#include        "uif:util.h"
#include        "stdef:pgm.h"                   /* pour la macro PGM()  */
                MONITOR(mailer);
#include        "pro:longueur.h"                    /*pour la longueur des va


/* declaration des fonctions externes */
int
F_Send(),
F_Msend(),
F_Val_list(),
F_Val_adr(),
F_Rmes(),
F_Dmes(),
F_Wmes(),
F_Fmes(),
F_Wfile(),
F_Reply(),
ExitProgram(),
F_Dir(),
F_Print(),
F_AppendMailbook();

/* declaration des variables externes */

char file_name[FILE_NAME];
char sujet[SUJET];
char numero[NUMBER];

/*
        Le titre apparaitra dans le fichier mailer.log et est centre au
        sommet de l'ecran quand le programme demarre et lorsque l'ecran
        vient d'etre efface.
*/

static  char    title[80] = " EASY_MAIL ";

PGM(mailer)
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.BUT            Initialiser le Mailer.
.RETOURNE       Rien.
-------------------------------------------------------------------------------
{       SaveParams      ();             /* Save command--line arguments
/*
```

```
        Lecture de synonyme pour les noms de fichier depuis le nom de
        fichier donne comme argument.

        LoadPaths       ("pro:mailer.fil");
/*
        "InitProteus()" ouvre un fichier .LOG qui recevra trace des sort
        de Easy_Mail par l'intermediaire du 'Program Monitoring subsyste

        call(InitProteus(PathName("Log"),       /* fichier .LOG
                         title,                 /* Title
                         PathName("Help"),      /* type du terminal
                         GetFlaggedParm('^')));  /* Terminal type
/*
        Lecture des commandes et dialogue interactif avec l'utilisateur
        pour ces commandes.
        MailCom  ();
}


/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Cette fonction travaille en dialogue interactif avec l'utilisate
et execute les fonctions correspondant aux commandes donnees.
.RETOURNE:  Rien

------------------------------------------------------------------------------

/*
        La table de structure associe les fonctions aux commandes donnee
        par l'utilisateur.
        La premiere colonne donne les nom des fonctions tandis que la de
        donne le nom de la commande pouvant etre donnee dans le mode
        interactif ainsi qu'une abreviation possible de la commande.

        STATIC  CHOICE selection[] = {
                {HostSpawn,         "Spawn ..... $"},
                {Help,              "Help .....H"},
                {SetupMode,         "Menu ..... M"},
                {F_Send,            "Send ..... S"},
                {F_Msend,           "Msend ..... X"},
                {F_Val_adr,         "AddressValid... V"},
                {F_AppendMailbook,"AppendMailbook..A"},
                {F_Rmes,            "ReadMessage.... R"},
                {F_Dir,             "DirMessage..... D"},
                {F_Dmes,            "DeleteMessage.. E"},
                {F_Wmes,            "WriteMessage... W"},
                {F_Fmes,            "ForwardMessage .F"},
                {F_Wfile,           "FileWrite...... C"},
                {F_Reply,           "ReplyMessage .. T"},
                {F_Print,           "PrintMessage .. P"},
                {ExitProgram,       "Exit ..... Q"},
                {0},              /* le zero final est obligatoire */

        STATIC  MENU menu =     {selection};

MailCom()
{
ENTER(Mailcom);
        while   (ObeyUser(&menu," Command ") NEQ EOF) ;
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE            Module
.NAME            msend
.LANGUAGE        C
.ENVIRONNEMENT   Easy_mail.
.AUTEUR          Bruno Loodts
.HISTOIRE         [0.2]   26-Nov-1987     Cree.
.COMMENTAIRES

        Ce module permet d'envoyer un message a plusieurs destinataires

 ------------------------------------------------------------------------------
# define ProteuSubsys
#include "uif:proteus.h"
#include "uif:lexa.h"
#include "longueur.h"
#include "uif:serv.h"
#include "uif:user.h"
#include "pro:record.h"

extern char file_name[];
extern char sujet[];

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction est la fonction principale implementant la commande
'Msend' du mailer.
------------------------------------------------------------------------------*

F_Msend()
{
int *nombre,i,sw,status;
char buffer[500];
struct caracteristique car[100];
ENTER(F_Msend);
(*nombre) = 0;
sw = 0;
if(MoreFields()) strcpy(buffer,WholeInput());
else F_List(buffer);
F_Extraction(nombre,buffer,car);
F_Sujet();
status=submenu();
if (status==OK)      {
                    for(i=0;i<(*nombre);i++)
                      {
                       F_Network(&car[i]);
                       F_Envoi(&car[i],&sw);
                       }
                    if(sw==0) delete(file_name);
                    }
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction est utilisee pour envoyer un message avec la commande Men
ou la commande Forward.
'car' est un pointeur vers la structure contenant les caracteristiques
du destinataire a rechercher dans le 'Mailbook' tandis que 'sw' est un
integer permettant de dire que l'envoi du message n'a pas pu se faire
```

```
Des lors 'sw' devient egal a 1.
-------------------------------------------------------------------------*/

F_Envoi(car,sw)
struct caracteristique *car;
int *sw;
{
int nb_trouve,prof_choisi,mail_choisi,code;
struct trouve result;
ENTER(F_Envoi);
debut:F_Lecture(car,&nb_trouve,&result);

if(nb_trouve==0)
 {
 F_Message("recipient doesn't exist in the mailbook",&code);
 if (code == -1) {
                 (*sw) = 1;
                 goto fin;
                 }
 F_Parametre(&result);
 F_SendMail(&result,0,0);
 }
 else {
       if(nb_trouve==4)
                  {
                  F_Message("recipient charasteristics are incomplete",
                  if (code == -1) {
                                  (*sw) = 1;
                                  goto fin;
                                  }
                  else {
                       F_Charast(car);
                       goto debut;
                       }
                  }
       else {
            if (nb_trouve==1) prof_choisi = 0;
            else {
                 F_ProfAddress(&result,nb_trouve,&prof_choisi);
                 if (prof_choisi == -1) {
                                        (*sw) = 1;
                                        goto fin;
                                        }
                 }
            if (strcmp(car->choix,"+")!=0)
                 {F_MailAddress(&result,prof_choisi,&mail_choisi,car->ch
                  if (mail_choisi == -1) {
                                         (*sw) = 1;
                                         goto fin;
                                         }
                 }
            else {
                 if (strcmp(car->network,"")==0) mail_choisi = 0;
                 else
                 if (strcmp(car->network,result.network[prof_choisi][0])
                 mail_choisi = 0;
                 else
                 if (strcmp(car->network,result.network[prof_choisi][1])
                 mail_choisi = 1;
                 else
                 if (strcmp(car->network,result.network[prof_choisi][2])
```

```
                    mail_choisi = 2;
                    else EXIT(NOK);
                    }
            F_SendMail(&result,prof_choisi,mail_choisi);
            }
      }
fin:EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet a l'utilisateur de donner la liste des
destinataires.
-----------------------------------------------------------------------*/

F_List(buffer)
char *buffer;

{
char zone[LIST];
ENTER(F_List);
InitForm("Liste","List of the recipients",0);
AddField("List1",'a',"","List of the receivers",LIST);
AddField("List2",'a',"","",LIST);
AddField("List3",'a',"","",LIST);
AddField("List4",'a',"","",LIST);
AddField("List5",'a',"","",LIST);
ChangeForm();
GetField("List1",buffer,LIST);
GetField("List2",zone,LIST);
strcat(buffer,zone);
GetField("List3",zone,LIST);
strcat(buffer,zone);
GetField("List4",zone,LIST);
strcat(buffer,zone);
GetField("List5",zone,LIST);
strcat(buffer,zone);
EXIT(OK);

}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction extrait du 'buffer' les arguments et les met dans
la table 'car'. Elle donne a '*nb' le nombre d'arguments trouves.

-----------------------------------------------------------------------*/
F_Extraction(nb,buffer,car)
char *buffer;
struct caracteristique car[];
int *nb;
{
char *ptr,variable[30];
ENTER(F_Extraction);
ptr=buffer;
while(Argument(&ptr,variable,sizeof(variable))!=0)
  F_Analyse(variable,car,nb);

EXIT(OK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction determine quel est le type du string contenu dans
'variable' et le place dans la table 'car'.
```

```
Elle donne a '*nb' le nombre d'arguments trouves.

-------------------------------------------------------------------------*/

F_Analyse(variable,car,nb)
char variable[];
struct caracteristique car[];
int *nb;
{
int i;
ENTER(F_Analyse);
if (variable[0]=='@') F_FileAnalyse(variable,car,nb);
else {
     if (variable[0]=='#') {
                           i = 1;
                           while(variable[i-1]!='\0')
                             {
                             car[*nb].institute[i-1] = variable[i];
                             i++;
                             }
                           (*nb)++;
                          }
     else if(variable[0]=='.') {
                               i = 1;
                               while(variable[i-1]!='\0')
                                 {
                                 car[(*nb)-1].network[i-1] = variable[i]
                                 i++;
                                 }
                               }
          else  {
                strcpy(car[*nb].surname,variable);
                (*nb)++;
                }
     }

EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction fait l'analyse syntaxique d'un fichier contenant une list
de destinataires.
-------------------------------------------------------------------------*/


F_FileAnalyse(variable,car,nb)
char variable[];
struct caracteristique *car[];
int *nb;
{

char buffer[500],line[160],file[FILE_NAME];
FILE *fil_des;
int i;
ENTER(F_FileAnalyse);
i = 1;
while(variable[i-1]!='\0')
  {
  file[i-1] = variable[i];
  i++;
  }
```

```
OpenIn(file,0);
while(ReadIn(line,sizeof(line))!=EOF) {
                                        strcat(buffer,line);
                                        strcat(buffer," ");
                                        }
CloseIn();
F_Extraction(nb,buffer,car);
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           print
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE        [0.2]   08-Dec-1987     Cree.
.COMMENTAIRES

        Ce module permet d'imprimer un message sur un device.
        Il implemente donc la commande 'PrintMessage' du mailer.

 ------------------------------------------------------------------------------
#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:user.h"
#include        "uif:serv.h"
#include        "longueur.h"

#if VMS
#define SPAWN PutDCL("@print.com")
#define LAST  "999"
#define LINE1  "$define/user Sys$output print.lis"
#define LINE2  "mail"
#define LINE3  "read mail "
#define LINE4  "print/queue="
#define LINE5  "exit"
#define QUEUE1 "sys$print"          /* device=1 ==> imprimante Centronix
#define QUEUE2 "diablo/form=laser" /* device=2 ==> imprimante QMS */
#define QUEUE3 "diablo"             /* device=3 ==> imprimante Diablo */
#define QUEUE4 "ln03"              /* device=4 ==> imprimante laser Digit
#else
#define SPAWN
#define LAST
#define LINE1
#define LINE2
#define LINE3
#define LINE4
#define LINE5
#define QUEUE1 "sys$print"          /* device=1 ==> imprimante Centronix
#define QUEUE2 "diablo/form=laser" /* device=2 ==> imprimante QMS*/
#define QUEUE3 "diablo"             /* device=3 ==> imprimante Diablo */
#define QUEUE4 "ln03"              /* device=4 ==> imprimante laser Digit
#endif

extern char file_name[];

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction implemente la commande 'PrintMessage' du mailer.
-------------------------------------------------------------------------------

F_Print()
{
char number[NUMBER+1],device[NUMBER+1],queue[20],commande[60];
ENTER(F_Wmes);
F_Para2(number,device);
if(strcmp(device,"1")==0) strcpy(queue,QUEUE1);
else if(strcmp(device,"2")==0) strcpy(queue,QUEUE2);
     else if(strcmp(device,"3")==0) strcpy(queue,QUEUE3);
```

```
          else if(strcmp(device,"4")==0) strcpy(queue,QUEUE4);
               else EXIT(NOK);
OpenOut("print.com",0);
WriteOut(LINE1);
WriteOut(LINE2);
strcpy(commande,LINE3);
strcat(commande,number);
WriteOut(commande);
strcpy(commande,LINE4);
strcat(commande,queue);
WriteOut(commande);
WriteOut(LINE4);
CloseOut();
SPAWN;
delete("print.com");
delete("print.lis");
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de lire les parametres [number] [#device]
rentres avec le nom de la commande conformement a la syntaxe definie.
Elle place [number] dans 'number' et [#device] dans 'device'.
-----------------------------------------------------------------------*/

F_Para2(number,device)
char *number,device[];
{
char buffer[MAXIMUM];
int i;
ENTER(F_Para2);
if(MoreFields()) {
                 PROMPT("",0);
                 strcpy(buffer,REPLY);
                 if (buffer[0]=='#') {
                                     i=1;
                                     while(buffer[i-1]!='\0')
                                          {
                                          device[i-1] = buffer[i];
                                          i++;
                                          }
                                     strcpy(number,LAST);
                                     }
                 else {
                      if (F_Chiffre(buffer)==OK) strncpy(number,buffer,N
                      else strcpy(number,LAST);
                      if(MoreFields())
                       {
                       PROMPT("",0);
                       strcpy(buffer,REPLY);
                       if (buffer[0]=='#') {
                                           i=1;
                                           while(buffer[i-1]!='\0')
                                             {
                                             device[i-1] = buffer[i];
                                             i++;
                                             }
                                           }
                       else strcpy(device,"1");
                       }
                    else strcpy(device,"1");
```

```
                    }
                }
else {
     strcpy(number,LAST);
     strcpy(device,"1");
     }
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction donne un nom du fichier par defaut
remarque:il n'y a pas de passage de parametres car 'file_name' est Exter
 ------------------------------------------------------------------------

extern char file_name[];

F_File_Name()
{
 char dateheure[31],jour[4],day[3],mois[4],annee[5],heure[3],minute[3],
     seconde[3];
 ENTER(F_File_Name);
 DateTime(dateheure,31);
 sscanf(dateheure,"%3s %2s %3s %4s %2s:%2s:%2s",
 jour,day,mois,annee,heure,minute,seconde);
 strcpy(file_name,day);
 strcat(file_name,mois);
 strcat(file_name,heure);
 strcat(file_name,minute);
 strcat(file_name,seconde);
 strcat(file_name,".MAI");
 EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE            Module
.NAME            Reply
.LANGUAGE        C
.ENVIRONNEMENT   Easy_mail.
.AUTEUR          Bruno Loodts
.HISTOIRE         [0.1]   10-Dec-1987     Cree.
.COMMENTAIRES

          ce module permet de repondre a un message recu.
           Il implemente la commande 'Reply'

 ---------------------------------------------------------------------------
#define          ProteuSubsys
#include         "uif:proteus.h"
#include         "uif:help.h"
#include         "uif:user.h"
#include         "uif:serv.h"
#include         "uif:term.h"
#include         "pro:longueur.h"
#if VMS
#define SPAWN PutDCL("@reply.com")
#define LAST  "999"
#define LINE1 "$define/user Sys$output reply.list"
#define LINE2 "mail"
#define LINE3 "read mail "
#define LINE4 "reply "
#define LINE5 "exit"
/*
   ces commandes ne doivent etre executes que lorsque le programme tourn
   sur VMS. Il s'agit en effet de la conversion de format necessaire
   pour envoyer un fichier avec PSI

*/
#define COMMANDE1 strcpy(file,file_name);
#define COMMANDE2 F_File_Name();
#define COMMANDE3 strcpy(commande,"$convert/fdl=pro:format_psi ");
#define COMMANDE4 strcat(commande,file);
#define COMMANDE5 strcat(commande," ");
#define COMMANDE6 strcat(commande,file_name);
#define COMMANDE7 WriteOut(commande);
#define COMMANDE8 delete(file);


#else
#define SPAWN PutUNIX("reply");
#define LAST "\n"
#define LINE1
#define LINE2
#define LINE3
#define COMMANDE1
#define COMMANDE2
#define COMMANDE3
#define COMMANDE4
#define COMMANDE5
#define COMMANDE6
#define COMMANDE7
#define COMMANDE8
#endif
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet de repondre a un message recu.
Il s'agit de la fonction principale implementant la commande 'Reply'.
------------------------------------------------------------------------------
extern char file_name[];

F_Reply ()
{
char number[6],commande[80],file[FILE_NAME];
int status;

ENTER(F_Reply);
if(MoreFields()) {
                 PROMPT("",0);
                 if(F_Chiffre(REPLY)==OK) strcpy(number,REPLY);
                 else strcpy(number,LAST);
                 }
else strcpy(number,LAST);
status=submenu();
if(status==OK) {
               OpenOut("reply.com",0);
               COMMANDE1
               COMMANDE2
               COMMANDE3
               COMMANDE4
               COMMANDE5
               COMMANDE6
               COMMANDE7
               WriteOut(LINE1);
               WriteOut(LINE2);
               strcpy(commande,LINE3);
               strcat(commande,number);
               WriteOut(commande);
               strcpy(commande,LINE4);
               strcat(commande,file_name);
               WriteOut(commande);
               WriteOut(LINE5);
               CloseOut();
               SPAWN;
               delete("reply.list");
               delete(file_name);
               delete("reply.com");
               COMMANDE8
               }
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           rmes
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE       [0.1]   13-Nov-1987     Cree.
.COMMENTAIRES

        ce module permet de lire un message.Il implemente la commande
        'ReadMessage'.

 ------------------------------------------------------------------------
#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:help.h"
#include        "uif:user.h"
#include        "uif:term.h"
#include        "uif:serv.h"
#include        "pro:longueur.h"
#include        <ctype.h>
#if VMS
#define SPAWN PutDCL("@rmes.com")
#define LAST  "999"
#define LINE1 "$define/user Sys$output readmessage.list"
#define LINE2 "mail"
#define LINE3 "read "
#define LINE4 "exit"
#define TEST  "%MAIL-E-NOTEXIST"
#define OLD   "mail"
#define NEW   "newmail"
#else
#define SPAWN PutUNIX("rmes");
#define LAST "\n"
#define LINE1
#define LINE2
#define LINE3
#define LINE4
#define TEST
#define OLD
#define NEW
#endif

extern char numero[];

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet d'afficher un message sur l'ecran.
Il s'agit de la fonction principale implementant la commande 'ReadMessag
------------------------------------------------------------------------------

F_Rmes  ()
{
char commande[60],folder_easy_mail[FOLDER],folder_systeme[FOLDER];

ENTER(F_Rmes);
/* lecture des parametres eventuels donnes avec la commande */
strcpy(numero,LAST);                /* numero du message par defaut */
strcpy(folder_easy_mail,"NEW");     /* folder easy_mail par defaut */
strcpy(folder_systeme,NEW);         /* folder systeme par defaut */
```

```
if(MoreFields())
  {
  PROMPT("",0);
  if(strcmp(REPLY,"OLD")==0 || strcmp(REPLY,"old")==0)
          {
          strcpy(folder_easy_mail,"OLD");
          strcpy(folder_systeme,OLD);
          }
  else if(strcmp(REPLY,"NEW")!=0 && strcmp(REPLY,"new")!=0)
              {
              if (F_Chiffre(REPLY)==OK) {
                                        strcpy(numero,REPLY);
                                        goto fin;
                                        }
              }
  if(MoreFields()) {
                   PROMPT("",0);
                   if (F_Chiffre(REPLY)==OK) strcpy(numero,REPLY);
                   }
  }
fin:
OpenOut("rmes.com",0);
WriteOut(LINE1);
WriteOut(LINE2);
strcpy(commande,LINE3);
strcat(commande,folder_systeme);
strcat(commande," ");
strcat(commande,numero);
WriteOut(commande);
WriteOut(LINE4);
CloseOut();
SPAWN;
F_Test("readmessage.list",folder_easy_mail);
delete("readMessage.list");
delete("rmes.com");
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction teste si le folder n'est pas vide. Si oui elle affiche a
l'ecran le message precedement lu et se trouvant dans le fichier 'fichie
Si non elle affiche la phrase message :
"the folder 'nom_folder' is empty" sur l'ecran.
----------------------------------------------------------------------*/
F_Test(fichier,nom_folder)
char *fichier,*nom_folder;
{
char commande[60];
ENTER(F_Fichier);
OpenIn(fichier,0);
ReadIn(commande,sizeof(commande));
CloseIn();
/* test si le folder est vide */
if(strncmp(commande,TEST,sizeof(TEST)-1)!=0)
     {
     ShowFile(fichier);
     }
else {
     strcpy(commande,"THE '");
     strcat(commande,nom_folder);
     strcat(commande,"' FOLDER IS EMPTY");
```

```
     ShowText(10,25,commande,_BOLD_);
     }
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction teste si un string de caracteres ne contient que des chif
elle renvoit 'OK' si oui
             'NOK' si non
------------------------------------------------------------------------------
F_Chiffre(chiffre)
char *chiffre;
{
char *p;
ENTER(F_Chiffre);
for(p=chiffre;isdigit(*p) ;p++);
if ((*p) == '\0') EXIT(OK);
EXIT(NOK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           send
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE        [0.6]   20-sept-1987     Cree.
.COMMENTAIRES
        ce module permet d'envoyer un mail vers un destinataire. Il s'a
        de l'implementation de la commande 'Send'.

 ------------------------------------------------------------------------
# define ProteuSubsys
#include "uif:proteus.h"
#include "uif:form.h"
#include "pro:longueur.h"
#include "uif:help.h"
#include "uif:serv.h"
#include "uif:user.h"
#include "uif:lexa.h"
#include "uif:term.h"
#include "pro:record.h"

extern char file_name[];
extern char sujet[];

int Help_Form();

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction est la fonction principale implementant la commande 'Send
------------------------------------------------------------------------

F_Send()
{
char choix[2],buffer[MAXIMUM];
int nb_trouve,prof_choisi,mail_choisi,code,i,sw;
struct caracteristique car;
struct trouve result;

ENTER(F_Send);
sw = 0;
strcpy(file_name,"");
strcpy(sujet,"");
strcpy(car.surname,"");
strcpy(car.first_name,"");
strcpy(car.city,"");
strcpy(car.country,"");
strcpy(car.institute,"");
strcpy(car.network,"");
strcpy(car.choix,"");

if (MoreFields())
  {
  PROMPT("",0);
  strcpy(buffer,REPLY);
  if (buffer[0]=='#') {
                      i=1;
                      while(buffer[i-1]!='\0')
                        {
```

```
                        car.institute[i-1]=buffer[i];
                        i++;
                        }
                      }
else strcpy(car.surname,buffer);
if (MoreFields())
{
PROMPT("",0);
strcpy(buffer,REPLY);
if (buffer[0]=='.')
   {
   i=1;
   while(buffer[i-1]!='\0')
     {
     car.network[i-1]=buffer[i];
     i++;
     }
   if (MoreFields())
     {
     PROMPT("",0);
     strcpy(buffer,REPLY);
     if (buffer[0]=='@')
       {
       i=1;
       while(buffer[i-1]!='\0')
         {
         file_name[i-1]=buffer[i];
         i++;
         }
       if (MoreFields())
         {
         PROMPT("",0);
         strcat(sujet," ");
         strcat(sujet,REPLY);
         }
       }
    else {
         strcat(sujet," ");
         strcat(sujet,REPLY);
         }
   }
 }
else if (buffer[0]=='@')
       {
       i=1;
       while(buffer[i-1]!='\0')
         {
         file_name[i-1]=buffer[i];
         i++;
         }
       if (MoreFields())
        {
        PROMPT("",0);
        strcat(sujet," ");
        strcat(sujet,REPLY);
        }
      }
   else {
        strcat(sujet," ");
        strcat(sujet,REPLY);
        }
```

```
  }
F_Network(&car);
}
else {
     F_Charast(&car);
     }
F_EnvoiSend(&car,&sw);
delete(file_name);
EXIT(OK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction est utilisee pour envoyer un message avec la commande Sen
'car' est un pointeur vers la structure contenant les caracteristiques
du destinataire a rechercher dans le 'Mailbook' tandis que 'sw' est un
integer permettant de dire que l'envoi du message n'a pas pu se faire.
A ce moment, 'sw' devient egal a 1.
----------------------------------------------------------------------*/

F_EnvoiSend(car,sw)
struct caracteristique *car;
int *sw;
{
int nb_trouve,prof_choisi,mail_choisi,code,status;
struct trouve result;
ENTER(F_Envoi);
debut:F_Lecture(car,&nb_trouve,&result);

if(nb_trouve==0)
 {
 F_Message("recipient doesn't exist in the mailbook",&code);
 if (code == -1) {
                 (*sw) = 1;
                 goto fin;
                 }
 F_Parametre(&result);
 F_SendMail(&result,0,0);
}
 else {
       if(nb_trouve==4)
                  {
                  F_Message("recipient charasteristics are incomplete",
                  if (code == -1) {
                                  (*sw) = 1;
                                  goto fin;
                                  }
                  else {
                       F_Charast(car);
                       goto debut;
                       }
                  }
       else {
            if (nb_trouve==1) prof_choisi = 0;
            else {
                 F_ProfAddress(&result,nb_trouve,&prof_choisi);
                 if (prof_choisi == -1) {
                                        (*sw) = 1;
                                        goto fin;
                                        }
                 }
            if (strcmp(car->choix,"+")!=0)
```

```
                {F_MailAddress(&result,prof_choisi,&mail_choisi,car->ch
                 if (mail_choisi == -1) {
                                        (*sw) = 1;
                                        goto fin;
                                        }
                }
           else {
                if (strcmp(car->network,"")==0) mail_choisi = 0;
                else
                if (strcmp(car->network,result.network[prof_choisi][0])
                mail_choisi = 0;
                else
                if (strcmp(car->network,result.network[prof_choisi][1])
                mail_choisi = 1;
                else
                if (strcmp(car->network,result.network[prof_choisi][2])
                mail_choisi = 2;
                else EXIT(NOK);
                }
        if (sujet[0]=='\0') F_Sujet();
        if (file_name[0]=='\0') {
                                status=submenu();
                                if(status==OK)
                                  F_SendMail(&result,prof_choisi,mail_c

                                }
        else F_SendMail(&result,prof_choisi,mail_choisi);

       }
     }
fin:EXIT(OK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet a l'utilisateur de donner des caracteristiques
de l'adresse professionnelle a rechercher dans le mailbook

----------------------------------------------------------------------*/

F_Charast(car)
struct caracteristique *car;

{
ENTER(F_Charast);
InitForm("recipient","recipient characteristics",Help_Form);
AddField("surname",'a',"","recipient surname",SURNAME-1);
AddField("first_name",'a',"","recipient first_name",FIRST_NAME-1);
AddField("network",'a',"","network name/+/?",NETWORK-1);
AddField("institute",'a',"","institute name",INSTITUTE-1);
AddField("city",'a',"","city name",CITY-1);
AddField("country",'a',"","country name",COUNTRY-1);
ChangeForm();
GetField("surname",car->surname,SURNAME-1);
GetField("first_name",car->first_name,FIRST_NAME-1);
GetField("network",car->network,NETWORK-1);
GetField("institute",car->institute,INSTITUTE-1);
GetField("city",car->city,CITY-1);
GetField("country",car->country,COUNTRY-1);
F_Network(car);
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction imprime une explication pour avoir de l'aide quand
on rempli un field
-----------------------------------------------------------------------*/
Help_Form()
{
ENTER(Help_Form);
DisplayHelp("send");
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet d'imprimer les adresses E-MAIL  correspondant
a une adresse professionelle
et retourne dans 'mail_choisi' le numero de l'adresse selectionnee.
-----------------------------------------------------------------------*/

F_MailAddress(result,prof_choisi,mail_choisi,choix)

struct trouve *result;
int prof_choisi,*mail_choisi;
char *choix;


{
int n;

ENTER(F_MailAddress);
n = 0;
InitForm("address","E_Mail address",0);
AddField("1_user",'a',result->user[prof_choisi][n],"",USER_NAME-1);
AddField("1_node",'a',result->node[prof_choisi][n],"",NODE_NAME-1);
AddField("1_network",'a',result->network[prof_choisi][n],"",NETWORK-1);
AddField("1_ga_user",'a',result->ga_user[prof_choisi][n],"",USER_NAME-1)
AddField("1_ga_node",'a',result->ga_node[prof_choisi][n],"",NODE_NAME-1)
AddField("1_ga_network",'a',result->ga_network[prof_choisi][n],"",NETWOR
n++;
if((strcmp(result->user[prof_choisi][n],"")!=0) && (strcmp(choix,"+")!=0
  {
  AddField("2_user",'a',result->user[prof_choisi][n],"",USER_NAME-1);
  AddField("2_node",'a',result->node[prof_choisi][n],"",NODE_NAME-1);
  AddField("2_network",'a',result->network[prof_choisi][n],"",NETWORK-1)
  AddField("2_ga_user",'a',result->ga_user[prof_choisi][n],"",USER_NAME-
  AddField("2_ga_node",'a',result->ga_node[prof_choisi][n],"",NODE_NAME-
  AddField("2_ga_network",'a',result->ga_network[prof_choisi][n],"",NETW
  n++;
  if(strcmp(result->user[prof_choisi][n],"")!=0)
   {
   AddField("3_user",'a',result->user[prof_choisi][n],"",USER_NAME-1);
   AddField("3_node",'a',result->node[prof_choisi][n],"",NODE_NAME-1);
   AddField("3_network",'a',result->network[prof_choisi][n],"",NETWORK-1
   AddField("3_ga_user",'a',result->ga_user[prof_choisi][n],"",USER_NAME
   AddField("3_ga_node",'a',result->ga_node[prof_choisi][n],"",NODE_NAME
   AddField("3_ga_network",'a',result->ga_network[prof_choisi][n],"",NET
   n++;
    }
 }
DisplayForm();
F_Choix(n,mail_choisi);
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction pose la question "your choice 1 to n" ou n est le
nombre max accepte et retourne la valeur choisie soustraite de 1 dans co
(soustraite de 1 car l'indice des tableau commence a zero)
elle retourne -1 pour abort.

-----------------------------------------------------------------------*/

F_Choix(n,code)
int *code,n;
{
 char comment[30];
 char valeur[2];
 ENTER(F_Choix);
 strcpy(comment,"From 1 to ");
 itoa(n,valeur,2);
 strcat(comment,valeur);
 strcat(comment," ('a' to abort)");
 debut:(*code) = -1;
 PROMPT(comment,"1");
 if ((strcmp(REPLY,"a")==0) || (strcmp(REPLY,"A")==0)) goto fin;
 *code=atoi(REPLY);
 if (((*code)<1) || ((*code)>n)) goto debut;
 (*code) = (*code) - 1;
 fin :EXIT(OK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction affiche un message sur l'ecran et retourne
code=-1 si l'utilisateur veut abandonner
code=0 sinon.
longeur max du message=40 (en comptant le '\0' )
-----------------------------------------------------------------------*
F_Message(message,code)
char *message;
int *code;
{
char comment[57];
ENTER(F_Message);
strcpy(comment,message);
strcat(comment," ('a' to aboard)");
PROMPT(comment,"");
if ((strcmp(REPLY,"a")==0) || (strcmp(REPLY,"A")==0)) (*code) = -1;
else (*code) = 0;
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet de choisir une des adresses professionelles
trouvees dans le mailbook et retourne dans nb_choisi celle
selectionnee
-----------------------------------------------------------------------

F_ProfAddress(result,nb_trouve,nb_choisi)
struct trouve *result;
int nb_trouve,*nb_choisi;

{
ENTER(F_ProfAddress);
InitForm("ProfAddress","professional address",0);
AddField("l_surname",'a',result->surname[0],"recipient surname",SURNAME-
AddField("l_first_name",'a',result->first_name[0],"recipient first_name"
```

```
AddField("1_institute",'a',result->institute[0],"institute name",INSTITU
AddField("1_city",'a',result->city[0],"city name",CITY-1);
AddField("1_country",'a',result->country[0],"country name",COUNTRY-1);
if(nb_trouve>1)
{
AddField("2_surname",'a',result->surname[1],"recipient surname",SURNAME-
AddField("2_first_name",'a',result->first_name[1],"recipient first_name"
AddField("2_institute",'a',result->institute[1],"institute name",INSTITU
AddField("2_city",'a',result->city[1],"city name",CITY-1);
AddField("2_country",'a',result->country[1],"country name",COUNTRY-1);
if(nb_trouve>2)
{
AddField("3_surname",'a',result->surname[2],"recipient surname",SURNAME-
AddField("3_first_name",'a',result->first_name[2],"recipient first_name"
AddField("3_institute",'a',result->institute[2],"institute name",INSTITU
AddField("3_city",'a',result->city[2],"city name",CITY-1);
AddField("3_country",'a',result->country[2],"country name",COUNTRY-1);
}}

DisplayForm();
F_Choix(nb_trouve,nb_choisi);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet a l'utilisateur de donner le user,node et network
pour envoyer le message
----------------------------------------------------------------------*/
F_Parametre(result)
struct trouve *result;

{
ENTER(F_Parametre);
InitForm("","E_Mail Address",0);
AddField("user name",'a',"","",USER_NAME-1);
AddField("node name",'a',"","",NODE_NAME-1);
AddField("network name",'a',"","local is the default",NETWORK-1);
ChangeForm();
GetField("user name",result->user[0][0],USER_NAME-1);
GetField("node name",result->node[0][0],NODE_NAME-1);
GetField("network name",result->network[0][0],NETWORK-1);
if (strcmp(result->network[0][0],"")==0) strcpy(result->network[0][0],"l
EXIT(OK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction initialise 'choix' en fonction de la valeur du network
----------------------------------------------------------------------*/
F_Network(car)
struct caracteristique *car;

{
ENTER(F_Network);
 if(strcmp(car->network,"?")==0)
                {
                strcpy(car->network,"");
                strcpy(car->choix,"?");
                }
 else if((strcmp(car->network,"+")==0) ||(strcmp(car->network,"")==0))
                {
                strcpy(car->network,"");
                strcpy(car->choix,"+");
```

```
            }
      else strcpy(car->choix,"+"); /* le plus performant par defaut */

EXIT(OK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction permet d'envoyer un mail en specifiant dans la structure
'*result' le destinataire choisi ( 'prof_choisi' ) et l'adresse choisie
ce destinataire ( 'mail_choisi').
-------------------------------------------------------------------------------

F_SendMail(result,prof_choisi,mail_choisi)

struct trouve *result;
int prof_choisi,mail_choisi;

{
#if VMS
#define SPAWN PutDCL
#define LOC_TCP_HEADER " \"EXOS%"
#define BIT_TCP_HEADER " \"EARN::"
#define GATE_TCP_HEADER " \"EARN::"
#define PSI_TCP_HEADER " PSI%"
#define SPAN_TCP_HEADER " "

#define LOC_TCP_SEPARATOR "@"
#define BIT_TCP_SEPARATOR "@"
#define GATE_TCP_SEPARATOR "@"
#define PSI_TCP_SEPARATOR "::"
#define SPAN_TCP_SEPARATOR "::"

#define SYSTEM_MAIL "$MAIL"
#define SUBJECT "/SUBJECT="
#define TCP_ENCLOSE "\""
#else
#define TCP_HEADER
#define SYSTEM_MAIL "mail -s "
#define TCP_SEPARATOR "@"
#define TCP_ENCLOSE "\""
#endif

char user[USER_NAME],node[NODE_NAME],network[NETWORK];
ENTER(F_SendMail);
strcpy(user,result->user[prof_choisi][mail_choisi]);
strcpy(node,result->node[prof_choisi][mail_choisi]);
strcpy(network,result->network[prof_choisi][mail_choisi]);

F_Retire(user,USER_NAME);
F_Retire(node,NODE_NAME);
F_Retire(network,NETWORK);

if (strcmp(network,"local")==0)
   F_MailLocal(user,node);
   else if (strcmp(network,"earn")==0)
             F_MailEarn(user,node);
        else if (strcmp(network,"psi")==0)
                  F_MailPsi(user,node);
             else if (strcmp(network,"span")==0)
                  F_MailSpan(user,node);
                  else F_MailGate(user,node,network);
if( (strcmp(result->f_user[prof_choisi][mail_choisi],"?")==0) ||
```

```
    (strcmp(result->f_node[prof_choisi][mail_choisi],"?")==0) ||
    (strcmp(result->f_network[prof_choisi][mail_choisi],"?")==0))
                F_Avertissement(user,node,network);

EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction envoit un mail local
-----------------------------------------------------------------------------*/

F_MailLocal(user,node)
char *user,*node;

{
char commande[100];
ENTER(F_MailLocal);
strcpy(commande,SYSTEM_MAIL);
if (sujet[0]!='\0') {
                    strcat(commande,SUBJECT);
                    strcat(commande,TCP_ENCLOSE);
                    strcat(commande,sujet);
                    strcat(commande,TCP_ENCLOSE);
                   }
strcat(commande," ");
strcat(commande,file_name);
strcat(commande,LOC_TCP_HEADER);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
strcat(commande,user);
strcat(commande,LOC_TCP_SEPARATOR);
strcat(commande,node);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
SPAWN(commande);
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction envoit un mail sur EARN

-----------------------------------------------------------------------------*/

F_MailEarn(user,node)
char *user,*node;

{
char commande[100];
ENTER(F_MailEarn);
strcpy(commande,SYSTEM_MAIL);
if (sujet[0]!='\0') {
                    strcat(commande,SUBJECT);
                    strcat(commande,TCP_ENCLOSE);
                    strcat(commande,sujet);
                    strcat(commande,TCP_ENCLOSE);
                   }
strcat(commande," ");
strcat(commande,file_name);
strcat(commande,BIT_TCP_HEADER);
strcat(commande,TCP_ENCLOSE);
```

```
strcat(commande,TCP_ENCLOSE);
strcat(commande,user);
strcat(commande,BIT_TCP_SEPARATOR);
strcat(commande,node);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
SPAWN(commande);
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction envoit un mail sur Vax_psi
.REMARQUE
Cette fonction n'etant executee que sur le systeme VMS, il n'a pas ete
necessaire de la rendre compatible a un systeme UNIX

-----------------------------------------------------------------------*/

F_MailPsi(user,node)
char *user,*node;

{
char commande[100];
char file[FILE_NAME];
FILE *fil_des;
ENTER(F_MailPsi);
strcpy(file,file_name);
F_File_name();
fil_des=fopen("psimail.com","w");
strcpy(commande,"$CONVERT/FDL=PRO:FORMAT_PSI ");
strcat(commande,file);
strcat(commande," ");
strcat(commande,file_name);
strcat(commande,"\n");
fputs(commande,fil_des);
strcpy(commande,SYSTEM_MAIL);
if (sujet[0]!='\0') {
                    strcat(commande,SUBJECT);
                    strcat(commande,TCP_ENCLOSE);
                    strcat(commande,sujet);
                    strcat(commande,TCP_ENCLOSE);
                  }
strcat(commande," ");
strcat(commande,file_name);
strcat(commande,PSI_TCP_HEADER);
strcat(commande,node);
strcat(commande,PSI_TCP_SEPARATOR);
strcat(commande,user);
strcat(commande,"\n");
fputs(commande,fil_des);
strcpy(commande,"$DELETE ");
strcat(commande,file);
strcat(commande,";*\n");
fputs(commande,fil_des);
fclose(fil_des);
SPAWN("@psimail.com");
delete("psimail.com");
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction envoit un mail sur un reseau par l'intermediaire
du gateway CERNVAX situe sur EARN

-------------------------------------------------------------------------*/

F_MailGate(user,node,network)
char *user,*node,*network;

{
char commande[100];
ENTER(F_MailGate);
strcpy(commande,SYSTEM_MAIL);
if (sujet[0]!='\0') {
                      strcat(commande,SUBJECT);
                      strcat(commande,TCP_ENCLOSE);
                      strcat(commande,sujet);
                      strcat(commande,TCP_ENCLOSE);
                    }
strcat(commande," ");
strcat(commande,file_name);
strcat(commande,GATE_TCP_HEADER);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
strcat(commande,user);
strcat(commande,GATE_TCP_SEPARATOR);
strcat(commande,node);
strcat(commande,".");
strcat(commande,network);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
strcat(commande,TCP_ENCLOSE);
SPAWN(commande);
EXIT(OK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction envoit un mail sur SPAN.

-------------------------------------------------------------------------*/

F_MailSpan(user,node)
char *user,*node;

{
char commande[100];
ENTER(F_MailSpan);
strcpy(commande,SYSTEM_MAIL);
if (sujet[0]!='\0') {
                      strcat(commande,SUBJECT);
                      strcat(commande,TCP_ENCLOSE);
                      strcat(commande,sujet);
                      strcat(commande,TCP_ENCLOSE);
                    }
strcat(commande," ");
strcat(commande,file_name);
strcat(commande,SPAN_TCP_HEADER);
strcat(commande,node);
strcat(commande,SPAN_TCP_SEPARATOR);
strcat(commande,user);
SPAWN(commande);
EXIT(OK);
```

```
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction affiche un message d'avertissement sur l'ecran pour dire
que l'adresse n'a pas ete validee et que en consequence la transmition
correcte du message n'est pas garanti.
-----------------------------------------------------------------------*

F_Avertissement(user,node,network)
char *user,*node,*network;
{
char commande[80];
ENTER(F_Avertissement);
strcpy(commande,"ADDRESS : ");
strcat(commande,user);
strcat(commande," ");
strcat(commande,node);
strcat(commande," ");
strcat(commande,network);
strcat(commande," IS NOT VALIDATED");
ShowText(17,1,commande,_BOLD_);
ShowText(18,1,"CORRECT TRANSMISSION OF THE MESSAGE IS NOT GUARANTEED",_B
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction retire les blancs a un string.
 ----------------------------------------------------------------------------
F_Retire(chaine,longueur)
char chaine[];
int longueur;

{
ENTER(F_Retire);
longueur = longueur - 2 ;
while ((longueur>=0) && (chaine[longueur]==' '))
  {
  chaine[longueur] = '\0';
  longueur = longueur - 1;
  }

EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet a l'utilisateur de donner le sujet du message
-----------------------------------------------------------------------*

F_Sujet()
{
ENTER(F_Sujet);
InitForm("sujet","subject of the mail",0);
AddField("1",'a',"","Type RETURN for no subject",SUJET-1);
ChangeForm();
GetField("1",sujet,SUJET-1);
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NOM            submenu
.LANGUAGE       C
.ENVIRONNEMENT  Easy_Mail
.AUTEUR         Bruno Loodts
.HISTORE        [0.1]   8-october-1987     Cree.
.COMMENTAIRES

Ce module permet a l'utilisateur de rediger son message.
Il offre le choix entre:
  -utiliser l'editeur EDIT
  -utiliser l'editeur EMACS
  -composer son message en mode "full-screen"
  -donner le nom du fichier dans lequel se trouve son message deja redig
  -abandonner la commande en cours.

---------------------------------------------------------------------------
#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:dial.h"
#include        "uif:user.h"
#include        "uif:help.h"
#include        "uif:serv.h"
#include        "uif:term.h"
#include        "pro:longueur.h"

int F_Edit(),F_Emacs(),F_File(),F_Compose(),F_Exit();
extern char file_name[];
        STATIC  CHOICE selection[] = {
                {F_Edit,        "Edit ..... D"},
                {F_Emacs,       "Emacs..... M"},
                {F_File,        "File  ..... F"},
                {F_Compose,     "Compose..... C"},
                {F_Exit,        "Exit..... E"},
                {0},            /* Ce zero final est obligatoire. */

        STATIC  MENU menu2 =     {selection};

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction est la fonction principale permettant a l'utilisateur de
rediger son message.
---------------------------------------------------------------------------

SubMenu()
{
        int status;
        DisplayHelp("Editor");
        status = ObeyUser(&menu2," Editor ");
        if( status == OK) EXIT(OK);
        else EXIT(NOK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de composer un message en mode "full-screen"
 ---------------------------------------------------------------------------

F_Compose()
{
char buffer[MAXIMUM+1],numero[18][3]=
```

```
{"1","2","3","4","5","6","7","8","9",
 "10","11","12","13","14","15","16","17","18"};
int i;
ENTER(F_Compose);
F_File_Name();
OpenOut(file_name,0);
InitForm("full_mode","full mode screen",0);
for(i=0;i<18;++i)
AddField(&numero[i],'a',"","",MAXIMUM);
ChangeForm();
for(i=0;i<18;++i)
  {GetField(&numero[i],buffer,MAXIMUM);
  if (strcmp(buffer,"")!=0) WriteOut(buffer);
  }
CloseOut();
EXIT(OK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet a l'utilisateur de donner le nom du fichier dans
lequel se trouve son message deja redige auparavant.
----------------------------------------------------------------------*/
F_File()
{
ENTER(F_File);
PROMPT("Type the name of the file that you will send",0);
strcpy(file_name,REPLY);
EXIT(OK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de "spawner" l'editeur EDIT.
 ----------------------------------------------------------------------*/
F_Edit  ()
{
#if VMS
#define SPAWN PutDCL
#else
#define SPAWN
#endif

char commande[60];
ENTER(F_Edit);
if (MoreFields()) {PROMPT("",0);
                   strcpy(file_name,REPLY);
                  }
  else F_File_Name();
strcpy(commande,"$EDIT ");
strcat(commande,file_name);
SPAWN(commande);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de "spawner" l'editeur EMACS.
 ----------------------------------------------------------------------*/

F_Emacs()
{
#if VMS
#define SPAWN PutDCL("@emacs_ed.com");
#else
#define
```

```
#endif

FILE *fil_des;
char commande[60];
ENTER(F_Emacs);
fil_des=fopen("emacs_ed.com","w");
if (MoreFields()) {
                  PROMPT("",0);
                  strcpy(file_name,REPLY);
                  }
else F_File_Name();
strcpy(commande,"EMACS_EXE:EMACS ");
strcat(commande,file_name);
fputs(commande,fil_des);
fclose(fil_des);
SPAWN
delete("emacs_ed.com");
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Cette fonction permet de sortir du sous-menu sans rediger
        de message et ainsi abandonner la commande en cours du mailer.
 ----------------------------------------------------------------------------

F_Exit()
{
ENTER(F_Exit);
NewScreen("");
EXIT(NOK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NAME           wfile
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE        [0.2]   20-Nov-1987     Cree.
.COMMENTAIRES

        ce module permet d'ecrire une message dans un fichier
        sans son header.
        Il s'agit de l'implementation de la commande 'FileWrite'.
 ------------------------------------------------------------------------*

#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:help.h"
#include        "uif:user.h"
#include        "uif:serv.h"
#include        "pro:longueur.h"

#if VMS
#define SPAWN PutDCL("@extract.com")
#define LAST  "999"
#define LINE0 "$define/user Sys$output "
#define LINE1 "mail"
#define LINE2 "read mail "
#define LINE3 "exit"

#else
#define SPAWN
#define LAST
#define LINE1
#define LINE2
#define LINE3
#endif

extern char file_name[];
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction est la fonction principale de la commande 'FileWrite'.
------------------------------------------------------------------------*/

F_Wfile ()
{

char number[NUMBER],nom_fichier[FILE_NAME];
ENTER(F_Wfile);
F_File_Name();
strcpy(nom_fichier,file_name);
strcpy(number,"");
F_Para(number);
F_ExtractMessage(number,nom_fichier);
F_Header(nom_fichier);
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction extrait le message numero 'number' et le met dans
'nom_fichier'
```

```
----------------------------------------------------------------------*/
F_ExtractMessage(number,nom_fichier)
char *number,*nom_fichier;
{

char commande[60];
ENTER(F_ExtractMessage);
OpenOut("extract.com",0);
if (strncmp(number,"",1)==0) strcpy(number,LAST);
strcpy(commande,LINE0);
strcat(commande,nom_fichier);
WriteOut(commande);
WriteOut(LINE1);
strcpy(commande,LINE2);
strcat(commande,number);
WriteOut(commande);
WriteOut(LINE3);
CloseOut();
SPAWN;
delete("extract.com");
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de lire les parametres [number] [@file_name]
rentres avec le nom de la commande.
Elle place [number] dans 'number' et [@file_name] dans 'file_name'.
----------------------------------------------------------------------*/

F_Para(number)
char *number;
{
char buffer[MAXIMUM];
int i;
ENTER(F_Para);
if(MoreFields()) {
                 PROMPT("",0);
                 strcpy(buffer,REPLY);
                 if (buffer[0]=='@') {
                                     i=1;
                                     while(buffer[i-1]!='\0')
                                          {
                                          file_name[i-1] = buffer[i];
                                          i++;
                                          }
                                     strcpy(number,LAST);
                                     }
                 else {
                      if(F_Chiffre(buffer)==OK) strncpy(number,buffer,NUMB
                      else strcpy(number,LAST);
                      if(MoreFields())
                       {
                       PROMPT("",0);
                       strcpy(buffer,REPLY);
                       if (buffer[0]=='@') {
                                           i=1;
                                           while(buffer[i-1]!='\0')
                                             {
                                             file_name[i-1] = buffer[i];
                                             i++;
                                             }
```

```
                                                    }
                        else F_InFile();
                        }
                    else F_InFile();
                    }
                }
else {
     F_InFile();
     }
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet a l'utilisateur de donner un nom de fichier
-------------------------------------------------------------------------------

F_InFile()

{
ENTER(F_InFile);
PROMPT("Type the name of your file",0);
strcpy(file_name,REPLY);
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction permet de supprimer le header d'un message contenu dans un
fichier et renomme ce fichier comme etant file_name.
Elle retourne comme valeur 'OK' si le header a ete supprimer
                           'NOK' si le type de message n'a pas pu
etre reconnu.( dans ce cas le header n'est pas modifie )
-------------------------------------------------------------------------------*/
F_Header(nom)
char *nom;
{
int nombre_de_ligne;
char line[100];
ENTER(F_Header);
nombre_de_ligne = 0;
OpenIn(nom,0);
ReadIn(line,sizeof(line));
ReadIn(line,sizeof(line));
CloseIn();
if (F_TestPsi(line,&nombre_de_ligne)==NOK)
    if (F_TestExos(nom,line,&nombre_de_ligne)==NOK)
       if (F_TestGate(nom,line,&nombre_de_ligne)==NOK)
             nombre_de_ligne = 4; /* header Decnet */
F_DelHeader(nom,nombre_de_ligne);
delete(nom);
EXIT(OK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction recopie le fichier 'nom' dans un fichier 'file_name'
en enlevant les n premieres lignes du fichier.
-------------------------------------------------------------------------------*

F_DelHeader(nom,n)
char *nom;
int n;
{
```

```
int i;
char line[100];
ENTER(F_DelHeader);
OpenIn(nom,0);
OpenOut(file_name,0);
i = 0;
while(i < n)  {
              ReadIn(line,sizeof(line));
              i++;
              }

while(ReadIn(line,sizeof(line))!=EOF) WriteOut(line);
CloseIn();
CloseOut();
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction teste si le message contenu dans le fichier 'nom'
est un message ayant utiliser le protocole PSI-DECnet.
Elle renvoit la valeur OK et donne a 'n' le nombre de ligne du header
si oui ou renvoit la valeur NOK si non.
----------------------------------------------------------------------*/

F_TestPsi(line,n)
char *line;
int *n;
{
#if VMS
#define TEST_PSI "From:\tPSI%"
#else
#define TEST_PSI
#endif
ENTER(F_TestPsi);
if (strncmp(line,TEST_PSI,sizeof(TEST_PSI)-1)==0) {
                                        (*n) = 4; /*nombre de ligne du hea
                                        EXIT(OK);
                                        }
else EXIT(NOK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction teste si le message contenu dans le fichier 'nom'
est un message ayant utiliser le protocole SMTP. (reseau local)
Elle renvoit la valeur OK et donne a 'n' le nombre de ligne du header
si oui ou renvoit la valeur NOK si non.
----------------------------------------------------------------------*/

F_TestExos(nom,line,n)
char *nom,*line;
int *n;
{
#if VMS
#define TEST_EXOS "From:\tEXOS%"
#define FIN_EXOS_HEADER "Message-ID:"
#else
#define TEST_EXOS
#define FIN_EXOS_HEADER
#endif
ENTER(F_TestExos);
if (strncmp(line,TEST_EXOS,sizeof(TEST_EXOS)-1)==0)
```

```
                {
                 F_LastHeader(nom,FIN_EXOS_HEADER,sizeof(FIN_EXOS_HEADER)-1,
                 EXIT(OK);
                 }
else EXIT(NOK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction teste si le message contenu dans le fichier 'nom'
est un message ayant utiliser un gateway.
Elle renvoit la valeur OK et donne a 'n' le nombre de ligne du header
si oui ou renvoit la valeur NOK si non.
-----------------------------------------------------------------------*/

F_TestGate(nom,line,n)
char *nom,*line;
int *n;
{
#if VMS
#define TEST_GATE "::GATEWAY::"
#define FIN1_GATE_HEADER "X-Vms-Mail-To:" /* pour mail de Hawaii */
#define FIN2_GATE_HEADER "To:"
#define FIN3_GATE_HEADER "Date Sent:"
#else
#define TEST_GATE
#define FIN_GATE_HEADER
#endif
ENTER(F_TestGate);
if (F_TestHeader(line,TEST_GATE,sizeof(TEST_GATE)-1)==OK)
    {
    if(F_LastHeader(nom,FIN1_GATE_HEADER,sizeof(FIN1_GATE_HEADER)-1,n)==NO
      if(F_FirstHeader(nom,FIN2_GATE_HEADER,sizeof(FIN2_GATE_HEADER)-1,n)=
            F_LastHeader(nom,FIN3_GATE_HEADER,sizeof(FIN3_GATE_HEADER)-1,n)
    EXIT(OK);
    }
else EXIT(NOK);
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction repere dans le fichier 'file' le dernier record commencant
par la chaine 'chaine' et donne a 'nb_record' le nombre d'enregistrements
parcourus depuis le debut du fichier.
Elle retourne 'OK' si la recherche est satisfaite
              'NOK' sinon.
-----------------------------------------------------------------------*/
F_LastHeader(file,chaine,longueur,nb_record)
char *file,*chaine;
int *nb_record,longueur;
{
int cpt;
char line[100];
ENTER(F_LastHeader);
cpt = 0;
OpenIn(file,0);
while(ReadIn(line,sizeof(line))!=EOF)
  {
  cpt++;
  if (strncmp(line,chaine,longueur)==0)
     (*nb_record) = cpt;
}
CloseIn();
```

```
if((*nb_record)>0) EXIT(OK);
else EXIT(NOK);
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction repere dans le fichier 'file' le premier record commencant
par la chaine 'chaine' et suivi par un record a blanc. IL donne a 'nb_reco
le nombre d'enregistrements parcourus depuis le debut du fichier.
Elle retourne 'OK' si la recherche est satisfaite
              'NOK' sinon.
----------------------------------------------------------------------*/
F_FirstHeader(file,chaine,longueur,nb_record)
char *file,*chaine;
int *nb_record,longueur;
{
int cpt;
char line1[100],line2[100];
ENTER(F_FirstHeader);
cpt = 0;
OpenIn(file,0);
ReadIn(line1,sizeof(line1));
while(ReadIn(line2,sizeof(line2))!=EOF)
  {
  cpt++;
  if ((strncmp(line1,chaine,longueur)==0) && (strcmp(line2," ")==0))
     {
     (*nb_record) = cpt;
     EXIT(OK);
     }
  else strcpy(line1,line2);
  }
CloseIn();
EXIT(NOK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction teste si la chaine de caractere 'chaine' se trouve dans
la ligne 'line'.
Elle renvoit la valeur 'OK' si oui
                       'NOK' sinon.
----------------------------------------------------------------------*/

F_TestHeader(line,chaine,longueur)
char line[],*chaine;
int longueur;
{
int cpt;
ENTER(F_TestHeader);
cpt = 0;
while(cpt<100) {
               if (strncmp(&line[cpt],chaine,longueur)==0) EXIT(OK);
               else cpt++;
               }
EXIT(NOK);
}
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE            Module
.NAME            wmes
.LANGUAGE        C
.ENVIRONNEMENT   Easy_mail.
.AUTEUR          Bruno Loodts
.HISTOIRE         [0.3]   20-Nov-1987     Cree.
.COMMENTAIRES

          ce module permet d'ecrire un message dans un fichier

 ----------------------------------------------------------------------------
#define          ProteuSubsys
#include         "uif:proteus.h"
#include         "pro:longueur.h"

extern char file_name[];

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cette fonction implemente la commande 'WriteMessage' du mailer.
----------------------------------------------------------------------------

F_Wmes  ()
{
char number[NUMBER];
ENTER(F_Wmes);
F_Para(number);
F_ExtractMessage(number,file_name);
EXIT(OK);
}
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.COMMENTAIRES

        Ce header doit etre inclus dans tout les programmes utilisant
        des longueurs 'standard'

.NOM              LONGUEUR
.TYPE             Header
.LANGUAGE         C
.ENVIRONNEMENT    Easy_mail
.AUTEUR           Bruno Loodts
.HISTORE          [0.2]   25-sep-1987     Cree.
----------------------------------------------------------------------------

#define SURNAME 21          /* le nom */
#define FIRST_NAME 13       /* le prenom */
#define NETWORK 8           /* le reseau */
#define INSTITUTE 31        /* l'institut */
#define CITY 21             /* la ville */
#define COUNTRY 4           /* le pays */
#define USER_NAME 17        /* le nom d'utilisateur */
#define NODE_NAME 36        /* le nom de noeud */
#define LIST 36             /* la liste des destinataires */
#define MAXIMUM 60          /* la plus grande  longueur d'un field*/
#define FILE_NAME 17        /* le nom du fichier */
#define KEY 31              /* une cle d'acces au mailbook */
#define LOCATION 7          /* l'adresse d'un enreg. dans le mailbook */
#define SUJET 30            /* le sujet du message */
#define FLAG 2              /* les flags du mailbook */
#define CODE 6              /* les codes dans le mailbook */
#define NUMBER 5            /* le numero d'un message     */
#define FOLDER 8            /* le nom d'un folder */
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.COMMENTAIRES

        Ce header doit etre inclus dans tout les programmes utilisant
        des structures.

.NOM              RECORD
.TYPE             Header
.LANGUAGE         C
.ENVIRONNEMENT    Easy_mail
.AUTEUR           Bruno Loodts
.HISTORE          [0.2]   04-nov-1987     Cree.
-------------------------------------------------------------------------------
#include "pro:longueur.h"

struct index_record {
                      char cle[KEY-1];
                      char blanc[1];
                      char location[LOCATION-1];
                      char saut[1];
                      char fin[1];
                    };


struct caracteristique {
                         char surname[SURNAME];
                         char first_name[FIRST_NAME];
                         char institute[INSTITUTE];
                         char network[NETWORK];
                         char city[CITY];
                         char country[COUNTRY];
                         char choix[2];
                        };

struct trouve {
               char surname[3][SURNAME];
               char first_name[3][FIRST_NAME];
               char institute[3][INSTITUTE];
               char city[3][CITY];
               char country[3][COUNTRY];
               int location[3];
               char user[3][3][USER_NAME];
               char f_user[3][3][FLAG];
               char node[3][3][NODE_NAME];
               char f_node[3][3][FLAG];
               char network[3][3][NETWORK];
               char f_network[3][3][FLAG];
               char ga_user[3][3][USER_NAME];
               char f_ga_user[3][3][FLAG];
               char ga_node[3][3][NODE_NAME];
               char f_ga_node[3][3][FLAG];
               char ga_network[3][3][NETWORK];
               char f_ga_network[3][3][FLAG];
              };

struct un_trouve {
               char surname[SURNAME];
               char first_name[FIRST_NAME];
               char institute[INSTITUTE];
```

```
                char city[CITY];
                char country[COUNTRY];
                int location;
                char user[3][USER_NAME];
                char f_user[3][FLAG];
                char node[3][NODE_NAME];
                char f_node[3][FLAG];
                char network[3][NETWORK];
                char f_network[3][FLAG];
                char ga_user[3][USER_NAME];
                char f_ga_user[3][FLAG];
                char ga_node[3][NODE_NAME];
                char f_ga_node[3][FLAG];
                char ga_network[3][NETWORK];
                char f_ga_network[3][FLAG];
               };

struct prof_1_record {
                        char surname[SURNAME-1];
                        char blanc[1];
                        char first_name[FIRST_NAME-1];
                        char saut[1];
                        char fin[1];
                        };

struct prof_2_record {
                        char institute[INSTITUTE-1];
                        char blanc1[1];
                        char city[CITY-1];
                        char blanc2[1];
                        char country[COUNTRY-1];
                        char saut[1];
                        char fin[1];
                        };

struct mail_adr_record {
                       char user[USER_NAME-1];
                       char blanc1[1];
                       char f_user[FLAG-1];
                       char blanc2[1];
                       char node[NODE_NAME-1];
                       char blanc3[1];
                       char f_node[FLAG-1];
                       char blanc4[1];
                       char network[NETWORK-1];
                       char blanc5[1];
                       char f_network[FLAG-1];
                       char saut[1];
                       char fin[1];
                      };

struct gate_adr_record {
                       char ga_user[USER_NAME-1];
                       char blanc1[1];
                       char f_ga_user[FLAG-1];
                       char blanc2[1];
                       char ga_node[NODE_NAME-1];
                       char blanc3[1];
                       char f_ga_node[FLAG-1];
                       char blanc4[1];
                       char ga_network[NETWORK-1];
```

```
  char blanc5[1];
  char f_ga_network[FLAG-1];
  char saut[1];
  char fin[1];
};
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE            Module
.NOM             initmailbook
.LANGUAGE        C
ENVIRONNEMENT    Easy_mail.
.AUTEUR          Bruno Loodts
.HISTOIRE         [0.2]   29-oct-1987     Cree.
.COMMENTAIRES

      ce programme initialise le fichier mailbook.txt
      avec un enregistrement et cree les fichiers d'index.

 ------------------------------------------------------------------------------

#define          ProteuSubsys
#include         "uif:proteus.h"
#include         "uif:serv.h"
#include         "stdef:pgm.h"
#include         "uif:term.h"
#include         "uif:user.h"
#include         "uif:form.h"
#include         "uif:util.h"
#include         "uif:help.h"
#include         "uif:lexa.h"
#include         "uif:dial.h"
#include         "longueur.h"
#include         "record.h"
                 MONITOR(initmailbook);
static char title[80]=" EASY_MAIL ";
PGM(initmailbook)
{
struct un_trouve ptr;
int i;
LoadPaths("pro:mailer.fil");
SaveParams();
call(InitProteus(PathName("Log"),
                 title,
                 PathName("Help"),
                 GetFlaggedParm("^")));
strcpy(ptr.surname,"");
strcpy(ptr.first_name,"");
strcpy(ptr.institute,"");
strcpy(ptr.city,"");
strcpy(ptr.country,"");
for(i=0;i<3;i++)
   {strcpy(ptr.user[i],"");
   strcpy(ptr.f_user[i],"?");
   strcpy(ptr.node[i],"");
   strcpy(ptr.f_node[i],"?");
   strcpy(ptr.network[i],"");
   strcpy(ptr.f_network[i],"?");
   strcpy(ptr.ga_user[i],"");
   strcpy(ptr.f_ga_user[i],"?");
   strcpy(ptr.ga_node[i],"");
   strcpy(ptr.f_ga_node[i],"?");
   strcpy(ptr.ga_network[i],"");
   strcpy(ptr.f_ga_network[i],"?");
   }
F_EnterProf(&ptr);
```

```
F_EnterMail(&ptr);
F_InitMailbook(&ptr);
ExitProgram();
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
cette fonction initialise le fichier 'mailbook.txt'
avec un enregistrement ainsi que les fichiers d'index.

-----------------------------------------------------------------------*/

F_InitMailbook(ptr)

struct un_trouve *ptr;

{
int i;
FILE *file_des;
ENTER(F_InitMailbook);

        /* ecriture du recipient dans le fichier */

file_des=fopen("mailbook.txt","w");
ptr->location = 0;
F_Ecriture(ptr,file_des);

fclose(file_des);

        /* mise-a-jour des fichiers d'index */

F_InitIndex(ptr);

EXIT(OK);
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        cette fonction cree les fichiers d'index
-----------------------------------------------------------------------*/
F_InitIndex(ptr)
struct un_trouve *ptr;

{
int loc;
char location[LOCATION];
ENTER(F_InitIndex);
loc = 0;

F_CreatIndex("surname.ind",ptr->surname,loc);
F_CreatIndex("first_name.ind",ptr->first_name,loc);
F_CreatIndex("institute.ind",ptr->institute,loc);
F_CreatIndex("city.ind",ptr->city,loc);
F_CreatIndex("country.ind",ptr->country,loc);
if (strcmp(ptr->network[0],"")!=0)
            {
            F_CreatIndex("network.ind",ptr->network[0],loc);
            if (strcmp(ptr->network[1],"")!=0)
                        {
                        F_Maj("network.ind",ptr->network[1],loc);
                        if (strcmp(ptr->network[2],"")!=0)
                            F_Maj("network.ind",ptr->network[2],loc);
                        }
            }
```

```
else EXIT(NOK);
EXIT(OK);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        cette fonction cree un fichier d'index
------------------------------------------------------------------------------*/

F_CreatIndex(fichier,key,loc)
char *fichier,*key;
int loc;
{
FILE *fil_des;
char key2[KEY],loc2[LOCATION];
struct index_record enr= {"","  ","","\n","\0"};
ENTER(F_CreatIndex);
F_Blanc(key2,KEY);
F_Blanc(loc2,LOCATION);
strcpy(key2,key);
itoa(loc,loc2,sizeof(loc2));
F_Ajoute(key2,KEY);
F_Ajoute(loc2,LOCATION);
strncpy(enr.cle,key2,KEY-1);
strncpy(enr.location,loc2,LOCATION-1);
fil_des=fopen(fichier,"w");
F_Write(fil_des,&enr);
fclose(fil_des);
EXIT(OK);
}
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.TYPE           Module
.NOM            buildindex
.LANGUAGE       C
.ENVIRONNEMENT  Easy_mail.
.AUTEUR         Bruno Loodts
.HISTOIRE       [0.3]   11-Nov-1987     Cree.
.COMMENTAIRES

      ce programme recree les fichiers d'index lorsque le fichier
      'mailbook.txt' a ete mis-a-jour avec un editeur.

 ---------------------------------------------------------------------------

#define         ProteuSubsys
#include        "uif:proteus.h"
#include        "uif:serv.h"
#include        "uif:help.h"
#include        "uif:lexa.h"
#include        "uif:util.h"
#include        "longueur.h"
#include        "record.h"

main()
{
FILE *mailbook,*surname,*first_name,*institute,*city,*country,*network;
char code[7],key[KEY],location[LOCATION];
struct prof_1_record prof1;
struct prof_2_record prof2;
struct mail_adr_record mail;
struct gate_adr_record gate;
struct index_record index;
int i,cpt_divers,cpt_network;

/* initialisations diverses */

strncpy(index.blanc," ",1);
strncpy(index.saut,"\n",1);
strncpy(index.fin,"\0",1);
cpt_divers = 0;
cpt_network = 0;

/* ouverture des fichiers */

mailbook=fopen("mailbook.txt","r");
surname=fopen("surname.ind","w");
first_name=fopen("first_name.ind","w");
institute=fopen("institute.ind","w");
city=fopen("city.ind","w");
country=fopen("country.ind","w");
network=fopen("network.ind","w");

itoa(ftell(mailbook),location,LOCATION);
F_Ajoute(location,LOCATION);
strncpy(index.location,location,LOCATION-1);

/* lecture de mailbook et ecriture des index */

while(fgets(code,sizeof(code),mailbook)!=NULL)
```

```
 {
 cpt_divers++;
 fgets(&prof1,sizeof(prof1),mailbook);
 F_Blanc(key,KEY);
 strncpy(key,prof1.surname,SURNAME-1);
 F_Ajoute(key,KEY);
 strncpy(index.cle,key,KEY-1);
 F_Write(surname,&index);
 F_Blanc(key,KEY);
 strncpy(key,prof1.first_name,FIRST_NAME-1);
 F_Ajoute(key,KEY);
 strncpy(index.cle,key,KEY-1);
 F_Write(first_name,&index);
 fgets(&prof2,sizeof(prof2),mailbook);
 F_Blanc(key,KEY);
 strncpy(key,prof2.institute,INSTITUTE-1);
 F_Ajoute(key,KEY);
 strncpy(index.cle,key,KEY-1);
 F_Write(institute,&index);
 F_Blanc(key,KEY);
 strncpy(key,prof2.city,CITY-1);
 F_Ajoute(key,KEY);
 strncpy(index.cle,key,KEY-1);
 F_Write(city,&index);
 F_Blanc(key,KEY);
  strncpy(key,prof2.country,COUNTRY-1);
 F_Ajoute(key,KEY);
 strncpy(index.cle,key,KEY-1);
 F_Write(country,&index);

 while(strcmp(fgets(code,sizeof(code),mailbook),"/END\n")!=0)
  {
  if (strcmp(code,"/MAI\n")==0)
     {
     fgets(&mail,sizeof(mail),mailbook);
     cpt_network++;
     F_Blanc(key,KEY);
     strncpy(key,mail.network,NETWORK-1);
     F_Ajoute(key,KEY);
     strncpy(index.cle,key,KEY-1);
     F_Write(network,&index);
     }
  else fgets(&gate,sizeof(gate),mailbook);
  }
itoa(ftell(mailbook),location,LOCATION);
F_Ajoute(location,LOCATION);
strncpy(index.location,location,LOCATION-1);
}
/* fermeture des fichiers */

fclose(mailbook);
fclose(surname);
fclose(first_name);
fclose(institute);
fclose(city);
fclose(country);
fclose(network);

/* tris des fichiers d'index */

DefaultSort("surname.ind",cpt_divers,KEY+LOCATION);
```

```
DefaultSort("first_name.ind",cpt_divers,KEY+LOCATION);
DefaultSort("city.ind",cpt_divers,KEY+LOCATION);
DefaultSort("institute.ind",cpt_divers,KEY+LOCATION);
DefaultSort("country.ind",cpt_divers,KEY+LOCATION);
DefaultSort("network.ind",cpt_network,KEY+LOCATION);

EXIT(OK);
}
```

```
\documentstyle[11pt]{article}

        \input  dirdoc:macrowd
        \input  dirdoc:macrogen
        \input  dirdoc:macroswc

        \def\Section#1{\section{#1}}
        \def\SubSection#1{\subsection{#1}}
        \def\C{{\bf C}}
        \def\uif{user interface}
        \def\x#1{\item{\bf#1}}

        \raggedbottom
        \topmargin=0cm
        \oddsidemargin=.5cm
        \evensidemargin=0cm
        \textheight=9.0in
        \textwidth=6.5in
        \parindent=2em
        \parskip=1ex

\begin{document}

        \pagestyle {myheadings}
        \head{The Easy-Mail Handbook}
        \pagenumbering{roman}
        \tableofcontents
        \newpage
        \setcounter{page}{1}
        \pagenumbering{arabic}
\def\screen#1#2{\section{#1 : #2}}
\def\eos#1{}
\def\ul#1{\underline{#1}}

\screen{Introduction}{First Screen}

\ \\
\BeginTable{|c|}{30}
\Rule
WELCOME \\
  TO \\
        {\bf EASY\_MAIL }       \\
\Rule
Version 0.7 \\
\Rule
\EndTable

\SkipLine

The main purpose of the development of {\bf EASY\_MAIL }is to get a
single user interface for all mailing purposes and protocols.

The different networks that are covered by the first version of this progr
 are as follow:

\BeginItems
\Item {\bf Local Ethernet}: with TCP/IP protocol
\Item {\bf EARN/BITNET}: with RJE protocol
\Item {\bf SPAN}: under DECNET protocol
```

```
\Item {\bf VAX-PSI world}: under PSI\_MAIL
\Item {\bf Gatewaying}: as currently exists with EARN only.
\EndItems
In a second version, X.400, ARPAnet, TCP/IP on X.25, Eunet will be
added.


                Type {\bf Help} to list the commands.

\eos{Introduction}

\screen{Prompts}{Question - and - Answer Dialogue}

The program puts prompts on the line which is 2nd from the bottom on the s
If the program detects an error, e.g. the response to a prompt was invalid
then an error message ({\it not} magic words or numbers) is displayed bene
the prompt, which is repeated; another response may be given.

Some prompts have {\it default values;} pressing {\bf CReturn}
 causes this value
to be taken as the response. Default values are indicated in the prompt
by \ul{underlining}:

        {\bf Topic :} \ul{commands}

--- if the response to this prompt is {\bf CReturn} then the available
commands are listed.

When it is known what the sequence of prompts will be, and the replies
to be given, they can be typed--ahead on one input line, terminated
by {\bf CReturn}, as the response to the first prompt in the sequence.
The replies may be separated by one or more spaces (which count as one).
The subsequent prompts will not appear, unless an error is detected; in th
case, the corresponding prompt is displayed, the rest of the input line
is discarded, and you may enter a valid response.

A command selects a particular function required of the program.
For example, the command {\bf Help} causes the program to provide
information on how to use the program.
A command is selected by answering the prompt

        {\bf Command :} \ul{Help}

with its name, or an {\it abbreviation;} that is, only enough first charac
of the command name are needed, to distinguish it from other commands in
the same program. Also, you may enter parameters on the same line
    as the command, instead of being prompted for them.
\eos{Prompts}

\screen{Commands}{A List of Commands}

\SkipLine
\BeginTable{|l|l|}{17,45}
\Rule
\       Command name &\ Purpose                \\
\Rule
\       {\bf AddressValidate}   &\ Validate address.\\
\       {\bf AppendMailbook}    &\ Append a new recipient in the Mailbook.
\       {\bf DeleteMessage}     &\ Delete message.        \\
\       {\bf DirMessage}        &\ Display the list of the received messag
\       {\bf Exit}              &\ Exit this program.             \\
```

```
\       {\bf FileWrite}         &\ Write message without mail header.
\       {\bf ForwardMessage}    &\ Forward message to another user. \\
\       {\bf Help}              &\ Display detailled help.                \\
\       {\bf Menu}              &\ Display menu of commands.     \\
\       {\bf Msend}             &\ Send to multiple recipients.  \\
\       {\bf PrintMessage}      &\ Print a message to a device.           \\
\       {\bf ReadMessage}       &\ Read message. \\
\       {\bf ReplyMessage}      &\ Reply to message.      \\
\       {\bf Send}              &\ Send message.                  \\
\       {\bf Spawn}             &\ Pass control to DCL or Shell.\\
\       {\bf WriteMessage}      &\ Write message to a file.\\
\       {\bf\$}{\sl command}    &\ Immediate execution of {\sl command}\\
\LastRule
\EndTable

\SkipLine
```

You may enter one of the above command names, or an abbreviation. If you know the prompts that follow a command, then you may also enter responses on the same input line.

```
\eos{Commands}

\def\screen#1#2{\subsection{#1 : #2}}

\screen{AddressValidate}{Validate an address}
     Syntax:    {\bf Addressvalidate} {\sl [recipient [.network]]}

\BeginItems
\Item {\sl Recipient} may be the surname of the person whose address you
want to validate, or
a ``\#'' followed by an institute name.
\Item {\sl Network} is the network name of which
you want validate the E\_Mail address. It must be preceded by a ``.''. The
name may be replaced by a {\bf ``+''} character, which means
the faster address or by a {\bf ``?''} in which case the user will be
given the choice among the different Electronic Mail (E\_mail) addresses f
 If the
specification of the network is omitted, the default value is {\bf ``+''}
i.e. the faster E\_mail address is choosen.
\EndItems

This command only changes the status of the flags of an E\_mail address
saying that this E\_mail address is correct.

\SkipLine

If no parameters are given, the user is prompted for the professional
address. If more than one recipient corresponding to the professional
address is found, they are displayed on the screen and the user is prompte
for a choice. If the recipient possesses more than one E\_mail address
with the specified network, they are all displayed on the screen and
the user is prompted for a choice.

\SkipLine

A procedure regularly collects all the personnal ``mailbooks'' of users
and updates the ESO's central ``mailbook'' with the new validated addresse

\eos{AddressValidate}
```

```
\screen{AppendMailbook}{append a new recipient in the Mailbook}

     Syntax:     {\bf AppendMailbook}

\SkipLine
    When this  command is called, the user is prompted for a new professio
address and its E\_Mail address(es). Theses new data are written in the
user's ''mailbook''.

\eos{AppendMailbook}

\screen{DeleteMessage}{Delete a received message}

     Syntax:     {\bf DeleteMessage} {\sl [number]}
\BeginItems
\Item {\sl number} is the number of the message that you want to delete.
\EndItems

This command deletes a received message from the ''old'' folder.
 If no number is given, the user is prompted for it.
\eos{DeleteMessage}

\screen{DirMessage}{List of received messages}

     Syntax:     {\bf DirMessage} {\sl [folder]}
\BeginItems
\Item {\sl folder} is the name of a folder.
The two folder names that you can use are ''old'' and ''new''.
The first is for the messages that were already readen. When you receive n
mail message, it automatically enter in the ''new'' folder.
The default value is ''new''.
\EndItems


This command  lists the received messages of a folder. If no folder name
is given then the ''new'' folder is used.
If the used folder contains no mails, then the message ''The folder is emp
is displayed on the screen.

\eos{DirMessage}

\screen{Exit}{Exit Easy\_Mail}

     Syntax:     {\bf Exit} or {\bf CTRL/z}

\SkipLine
    This  command terminates an Easy\_Mail session, clearing the screen
    except for the  'title'  line,  and  returning  you to the host operat
    system user
    command level (e.g. DCL).

\SkipLine
    CTRL/Z (Control key + z at the same time: \^{ }z) is equivalent to EXI

\eos{Exit}

\screen{FileWrite}{Write a received message without header to a file}

    Syntax:     {\bf FileWrite} {\sl [number] [@filename]}
\BeginItems
```

\Item {\sl number} is the number of the message that you want to write. It
default value is the last new message.
\Item {\sl filename} is the file name in wich you want to write the messag
It must be preceded by a ''@''.

\EndItems

This command allows writing a received message from the ''old'' folder
 to a file, without the
mail header. If no file name is given, the user is prompted for it.

\eos{FileWrite}

\screen{ForwardMessage}{Forward a received message to another recipient}

     Syntax: {\bf ForwardMessage} {\sl [number] [list]}
\BeginItems
\Item {\sl number} is the number of the message that you want to forward.
default value is the new read message.
\Item {\sl list} is a set of [recipient [.network]] and/or [@filename].\\
The separation character is the space.
\BeginItems
\Item {\sl Recipient} may be the surname of the person you want to reach,
a ''\#'' followed by an institute name.
\Item {\sl Network} is the network name on which
you want the message to be sent. It must be preceded by a ''.''. The netwo
name may be replaced by a {\bf ''+''} character, which means
the faster address or by a {\bf ''?''} in which case the user will be
given the choice among the different Electronic Mail (E_mail) addresses f
 If the
specification of the network is omitted, the default value is {\bf ''+''}
i.e. the faster E_mail address is choosen.
\Item {\sl filename} is the name of a ''distribution list'' file.
\EndItems
\EndItems

This command allows the forwarding of a received message from the old fold
to another(s) recipient(s).

\SkipLine
If no {\sl list} is given, the user is prompted for it.
If more than one recipient corresponding to the professional
address is found, they are displayed on the screen and the user is prompte
for a choice. If the recipient possesses more than one E_mail address
with the specified network, they are all displayed on the screen and
the user is prompted for a choice.
\eos{ForwardMessage}

\screen{Menu}{Display menu of commands}

     Syntax: {\bf Menu}

\SkipLine

Type {\bf Menu} to display the list of commands:press the key for the comm
code (after ''...''), or use arrow keys to move through the menu and pres
{\bf CReturn} to invoke underlined command.

\eos{Menu}

\screen{Msend}{Send a message to multiple recipients}

```
    Syntax:    {\bf Msend} {\sl [list]}
\BeginItems
\Item {\sl list} is a set of [recipient [.network]] and/or [@filename].\\
The separation character is the space.
\BeginItems
\Item {\sl Recipient} may be the surname of the person you want to reach,
a ``\#'' followed by an institute name.
\Item {\sl Network} is the network name on which
you want the message to be sent. It must be preceded by a ``.''. The netwo
name may be replaced by a {\bf ``+''} character, which means
the faster address or by a {\bf ``?''} in which case the user will be
given the choice among the different Electronic Mail (E\_mail) addresses f
 If the
specification of the network is omitted, the default value is {\bf ``+''}
i.e. the faster E\_mail address is choosen.
\Item {\sl filename} is the name of a ``distribution list'' file.
\EndItems
\EndItems

\SkipLine

The same message is sent to a set of people. The set is determined by
the list and the ``distribution list'' file(s).
If no {\sl list} is given, the user is prompted for it.
The message composition is done as a single send.
If a used E\_mail address is not yet validated,
a comment is displayed on the screen to say that the correct transmission
 of the
message is not guaranteed.

\eos{Msend}

\screen{PrintMessage}{Print a message to a device}

   Syntax:    {\bf PrintMessage} {\sl[number] [\#device]}
\BeginItems
\Item {\sl number} is the number of the message that you want to print. It
default value is the last new message.
\Item {\sl device} is the number of the device that you want to use to pri
 the message. Its default value is the number ``1''. The Centronix printer
the device number ``1'', the laser QMS printer ``2'', the Diablo printer `
and the laser Digital printer ``4''.
This parameter must be preceded by a ``\#''.
\EndItems

\SkipLine

   This sends a message from the ``old'' folder
 to the specified {\it device} defined in the local system.
 If no {\it number} is given, the last new message is printed.
 If no {\it device} is given, the Centronix printer is used.

\eos{PrintMessage}

\screen{ReadMessage}{Read a received message}

    Syntax:    {\bf ReadMessage} {\sl [folder]} {\sl [number]}

\BeginItems
\Item {\sl folder} is the name of a folder.
```

```
The two folder names that you can use are ''old'' and ''new''.
The first is for the messages that were already readen. When you receive n
mail message, it automatically enter in the ''new'' folder.
The default value is ''new''.
\Item {\sl number} is the number of the message that you want to read. Its
default value is the first new message.
\EndItems

This command displays a received message of a specified folder.
If no folder name is given the ''new'' folder is used.
If no number is specified the last new message is displayed.
After you read a message in the ''new'' folder, it automatically moves int
the ''old'' folder.
\eos{ReadMessage}

\screen{ReplyMessage}{Reply to a received message}

    Syntax:    {\bf Reply} {\sl [number]}
\BeginItems
\Item {\sl number} is the number of the message that you want to reply. It
default value is the last new message.
\EndItems

This command allows replying to a received message from the ''old'' folder
If no number is given, the last new message is used.
\eos{ReplyMessage}

\screen{Send}{Send a message}

     Syntax:    {\bf Send} {\sl [recipient [.network] [@filename] [subject

\BeginItems
\Item {\sl Recipient} may be the surname of the person you want to reach,
a ''\#'' followed by an institute name.
\Item {\sl Network} is the network name on which
you want the message to be sent. It must be preceded by a ''.''. The netwo
name may be replaced by a {\bf ''+''} character, which means
the faster address or by a {\bf ''?''} in which case the user will be
given the choice among the different Electronic Mail (E\_mail) addresses f
 If the
specification of the network is omitted, the default value is {\bf ''+''}
i.e. the faster E\_mail address is choosen.

\Item {\sl filename} is the name of the file that you will send.
It must be preceded by a ''@''.
\Item {\sl subject} is the subject of your message.
\EndItems

The message is reliably sent with the right protocol.
The user is prompted
for the professional address if he did not provide {\sl recipient} before.
His personal mailbook
is queried to find the E\_mail address(es) of the
recipient. If more than one recipient corresponding to the professional
address is found, they are displayed on the screen and
the user is prompted for a
choice.  If the recipient possesses more than one E\_mail address with the
specified network, they are
all displayed on the screen and the user
is prompted for a choice.
```

```
\SkipLine

If the chosen E\_mail address
is not yet validated, a comment is displayed on the screen to say that the
correct transmission of the message is not guaranteed.

\SkipLine

If a query for a specified {\sl recipient} is not successful,
the user has to
introduce the complete E\_mail address to send his mail.

\SkipLine

If no {\sl filename} is specified the message can be
composed on\-line with or
without the editor. The user may
choose the ''mailer editor''. The user can abort the command before the
send too.

\eos{Send}

\screen{Spawn}{To send commands to operating system}

    Syntax:    {\bf Spawn}

\SkipLine
    This command creates a new user process, in which DCL (Digital Com
    Language) or shell commands may be given; it must be terminated
    with the LOG or EXIT command.

\eos{Spawn}

\screen{WriteMessage}{Write a received message to a file}

     Syntax:    {\bf WriteMessage} {\sl [number] [@filename]}

\BeginItems
\Item {\sl number} is the number of the message that you want to write. It
default value is the last read message.
\Item {\sl filename} is the file name in wich you want to write the messag
It must be preceded by a ''@''.
\EndItems

This command allows the writing of a received message from the ''old'' fol
 to a file.

\SkipLine

If no number is given, the last new message is written. If no
file name is given, the user is prompted for it.

\eos{WriteMessage}
\iffalse
\screen{Editor}{this is the syntax for choice a editor}
for compose your message, choose of the following :

\BeginItems
\Item{\bf Edit} {\sl[file\_name]}: to use the EDIT editor with a specific
\Item{\bf Emacs}{\sl[file\_name]}: to use the EMACS editor.
\Item{\bf Compose}                : to use the full-screen mode.
```

```
\Item{\bf File} {\sl[file\_name]} : when your message is already in a file
\Item{\bf Exit}                   : to abort.
\EndItems

\eos{Editor}
\fi
\end{document}
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!.IDENTIFICATION       DEFINE_EASY_MAIL.COM
$!.LANGUAGE             DCL
$!.BUT                  Ce module fait les assignements necessaire
$!                      pour  Easy_mail puis le lance.
$!.AUTHOR               Bruno Loodts.
$!.HISTOIRE             Creation Sep 1987
$!----------------------------------------------------------------------
$!      Definition pour Proteus:
$!
$       DEFINE/NOLOG    public          archive
$       DEFINE/NOLOG    privat          ecf$disk
$       DEFINE/NOLOG    dirdoc          public:[doc]
$       DEFINE/NOLOG    dirsm           public:[frozen.window]
$       DEFINE/NOLOG    stdef           public:[c]
$       DEFINE/NOLOG    LNK$LIBRARY     SYS$LIBRARY:vaxcrtl.olb
$       DEFINE/NOLOG    stclib          stdef:stlib
$       DEFINE/NOLOG    uif             public:[frozen.PROTEUS]
$       ASSIGN/NOLOG    public:[c.sm.tc]TERMCAP.DAT          TERMCAPFILE
$!
$! Definition pour Easy-Mail:
$!
$ DEFINE/NOLOG pro privat:[loodts.MAILER]
$ initmailbook:=="$ecf$disk:[loodts.mailer]initmailbook"
$ buildindex:=="$ecf$disk:[loodts.mailer]buildindex"
$ r pro:mailer
```