Documentation de développement et de maintenance du gestionnaire de mémoire JEVEUX

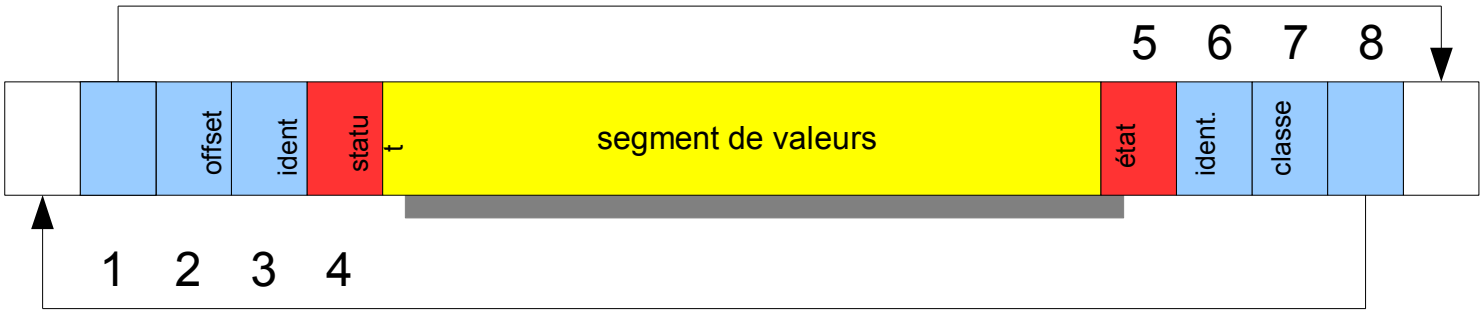
1 Introduction

Code_Aster a été développé en FORTRAN 77, ce langage ne possède pas de gestion dynamique de la mémoire et ne permet pas une très forte structuration des types. Le gestionnaire de mémoire JEVEUX a permis de pallier une partie de ces inconvénients en offrant les possibilités suivantes :

- **allocation dynamique** des zones mémoires allouées en cours des travaux,
- **gestion des débordements mémoire** sur fichier, avec archivage des résultats en fin des travaux,
- **structuration des données** du *Code_Aster*, avec accès par nom aux objets manipulés et normalisation des types Fortran utilisés.

Ce document est destiné à la documentation et à la maintenance des routines du gestionnaire de mémoire JEVEUX préfixée par JJ, JE ou JX, les routines JX font en général appel à des fonctions non portables. Par la suite, nous désignerons cet ensemble de routines sous le vocable "le logiciel". Une description précise du fonctionnement et de l'organisation interne du logiciel y est détaillée. Le lecteur pourra se reporter à la documentation [D6.02.01] Gestion Mémoire JEVEUX qui décrit l'interface des routines "utilisateur" JE.... .

2 Organisation de la mémoire



JEVEUX alloue dynamiquement chaque zone mémoire destinée à accueillir les valeurs associées à l'objet ou à la collection d'objets. Un contrôle de l'espace cumulé est effectué lors de chaque nouvelle allocation, et un mécanisme de déchargement peut éventuellement être mis en œuvre pour libérer des espaces de la mémoire associés aux objet que le développeur à déclaré comme n'étant plus utilisé.

Sur les plates-formes 64 bits, l'allocation de la zone mémoire est réalisée dynamiquement à l'aide de la routine système HPALLOC. Cette zone est vue dans le logiciel à travers le tableau de type entier (INTEGER*8 ISZON) de longueur LISZON et d'une adresse de début JISZON. Il est stocké dans le commun /IZONJE/. On utilisera dans la suite du document le terme "mot" pour désigner l'unité d'adressage. Sur plate-forme 64 bits le mot a pour longueur 8 octets et correspond à longueur du type INTEGER*8.

La zone est gérée mot par mot en unité de type INTEGER (unité d'adressage), les segments de valeurs associés aux objets JEVEUX sont encadré par 8 mots contenant, dans cet ordre, les informations suivantes :

1) l'adresse suivant le dernier mot constituant le segment de valeurs et les 8 identificateurs;
2) la valeur d'un décalage utilisé pour aligner les segments de type de longueur supérieure à l'unité d'adressage. Sur plate-forme 64 bits, cette valeur est toujours nulle pour les segments de valeurs associés à des objets de type INTEGER et REAL*8. Elle vaut 8 (octets) parfois pour les segments des valeurs associés à des objets de type COMPLEX*16 : en effet il arrive que le début du segment de valeur (position 5) ne puisse coïncider avec une position dans le tableau de référence ZK16, il faut alors se déplacer d'un mot (8 octets). C'est encore plus fréquent avec le type CHARACTER lorsqu'il vaut 16, 32 ou 80 !
3) l'identificateur entier associé aux objets simples ou aux collections ;
4) le statut du segment de valeurs qui peut prendre la valeur X ou U (codé sur un entier);
5) l'état du segment de valeurs qui peut prendre la valeur X, A ou D (codé sur un entier);;
6) l'identificateur entier associé aux objets de collection ;
7) le code de la classe associée à l'objet JEVEUX ;
8) l'adresse du premier mot précédant le segment de valeurs et les 8 identificateurs.

La gestion de la zone mémoire avec le type INTEGER ne permet pas de s'aligner correctement avec les types de longueur supérieure à cette unité d'adressage. Bien que l'ordre EQUIVALENCE présent dans le logiciel permette d'aligner l'adresse initiale des différentes variables (tableaux) de référence ZI, ZI4, ZR, ZC, ZL, ZK8, ZK16, ZK24, ZK32 et ZK80, le positionnement d'un segment de valeurs associé à un objet de type ZK32 a peu de chance d'être aligné avec un "multiple" de 4 du tableau ISZON sur plate-forme 64 bits, d'où la nécessité de gérer un décalage parmi les descripteurs.

Les valeurs permettant de coder le statut et l'état du segment de valeurs sont obtenues de façon à ce que la représentation ne corresponde à aucun type utilisé au sein du segment de valeurs et ce afin de détecter les éventuels écrasements de ces descripteurs. Ce rôle est dévolu à la routine JJLIRS qui est appelée principalement lors des requêtes de mise en mémoire du segment de valeurs. L'émission

d'un message d'erreur du type "ECRASEMENT AMONT POSSIBLE ..." indique que l'identificateur (3) ou le statut (4) ont été écrasés, l'émission d'un message d'erreur du type "ECRASEMENT AVAL POSSIBLE ..." indique que l'état (5) ou la classe (7) ont été écrasés.

Sur plate-forme 64 bits, les valeurs utilisées ont la représentation octale suivante :

01000000000000000000	pour X,	03000000000000000000	pour A,
02000000000000000000	pour U,	04000000000000000000	pour D.

Usage des segments de valeurs

Le couple état/statut permet de connaître l'usage d'un segment de valeurs en mémoire :

- XX indique que le segment de valeurs est libre, cette position est directement utilisable,
- UA indique que le segment de valeurs est utilisé en lecture (son image disque ne sera pas actualisée après libération),
- UD indique que le segment de valeurs est utilisé en écriture (son image disque sera actualisée après libération),
- XD indique que le segment de valeurs a été libéré, mais que son contenu devra être déchargé sur disque,
- XA indique que le segment de valeurs a été libéré, cette position est directement utilisable.

3 Gestion de la mémoire

Une requête d'accès en lecture ou en écriture sur le segment de valeurs associé à un objet JEVEUX provoque, s'il n'y figure pas déjà, un chargement en mémoire du contenu du segment de valeurs associé. L'adresse mémoire d'un objet JEVEUX correspond à sa position relative dans le tableau ISZON. Au préalable, il est nécessaire d'effectuer une allocation dynamique, à l'aide de la routine JJALLS pour insérer le segment de valeurs. Lorsque la requête d'allocation échoue, le système refusant d'allouer une zone mémoire, un mécanisme de libération est déclenché, il peut entraîner des accès disque lorsque des zones associées à des segments de valeurs doivent être écrites sur le fichier associé à la base. Le nouveau segment de valeurs est alloué avec une tolérance de 8 entiers qui correspondent à l'espace minimum associé à un segment de valeurs (1 entier par descripteur). Lorsque la recherche de place mémoire échoue, on provoque un arrêt de l'application en erreur <S> (arrêt par le superviseur avec sauvegarde des concepts créés).

Un appel à la fonction système LOC à travers la routine JXLOCS permet d'obtenir l'adresse relative du début du segment de valeurs par rapport au tableau ISZON en utilisant la valeur de la position de référence du début de la zone mémoire obtenue dans JXALLM et stockée dans le commun /ILOCJE/. C'est l'usage de la routine JJALTY qui permet d'aiguiller sur le tableau Z. et d'obtenir suivant le type l'adresse par rapport à la bonne référence.

L'allocation d'un segment de valeurs associé à un objet de type dont la longueur est supérieure à l'unité d'adressage utilisée (par exemple pour le type CHARACTER *24) ne permet pas automatiquement de s'aligner par rapport au tableau ISZON, il est parfois nécessaire de se décaler de quelques mots. La valeur de ce décalage est stockée dans le deuxième descripteur précédant le segment de valeurs et la taille effective du segment de valeurs est ajustée en tenant compte de son type associé.

Il reste ensuite à actualiser les descripteurs associés au segment de valeurs, cette opération est réalisée par la routine JJECRS.

Recherche de place disponible

L'appel à la routine JEDISP permet de connaître au moment de l'appel, la taille des zones mémoire disponibles, elle effectue la recherche en parcourant l'ensemble de la segmentation mémoire et

dépose au fur et à mesure la taille des zones libres ou déchargeables dans un tableau fourni par argument d'appel.

Vérification de la mémoire

Les écrasements mémoire qui affectent les descripteurs (état ou statut) ou le chaînage avant peuvent être détectés en utilisant la routine JXVERI. Cette routine examine un à un les descripteurs des segments de valeurs en mémoire. Un message d'erreur fatale est émis lors de la détection d'une anomalie, sinon la routine reste muette.

4 Gestion des libérations

La libération est le mécanisme le plus complexe mis en œuvre dans JEVEUX. On conçoit aisément que lorsque que l'on gère un espace mémoire fini, il arrive un moment où il n'est plus possible de trouver de place. Il faut alors provoquer des déchargements sur disque ou bien récupérer la place des zones devenues inutiles. JEVEUX prend en charge ces mécanismes, à condition bien sûr, que le programmeur lui ait indiqué les objets concernés ; il est nécessaire de prendre certaines précautions avant de libérer un objet, plusieurs unités de programme pouvant utiliser simultanément une adresse mémoire. La stratégie de libération fait appel d'une part à un mécanisme interne au gestionnaire de mémoire que nous allons décrire et d'autre part à des règles de programmation qui font l'objet du document [D2.06.99] "Nouvelle stratégie de libération des objets JEVEUX".

La libération d'un segment de valeurs se matérialise par le positionnement à la valeur X de l'état en lieu et place de la valeur U. Il n'y a pas d'autre effet immédiat, ce n'est que lors d'une recherche ultérieure de position en mémoire que l'on traitera effectivement le contenu du segments de valeurs.

La mise en mémoire d'un objet JEVEUX s'accompagne de l'affectation d'un attribut système : la marque. Cet attribut, de type entier, prend la valeur d'un compteur incrémenté à chaque appel à la routine JEMARQ et décrémenté à chaque appel à la routine JEDEMA. Il est possible d'obtenir la valeur de la marque courante en appelant la routine JEVEMA.

Les marques courantes ont ainsi une valeur strictement positive. Les valeurs -1, -2 et -3 sont utilisées pour traiter les exceptions suivantes.

La valeur -1 est utilisée pour garder en permanence (tout au long de l'exécution d'une commande Aster) certains objets qui seront libérés par un appel spécifique. Cette marque est utilisée lors de l'appel à la routine JEVEUT.

La valeur -2 est utilisée par JEVEUX pour ramener de façon temporaire certains objets qui seront libérés aussitôt l'action terminée (mise en mémoire des objets système de collection, ...).

La valeur -3 est utilisée pour garder en permanence (tout au long de l'exécution de *Code_Aster*) les objets utilisés par le Superviseur.

La marque -3 peut venir en remplacement de toute marque existante, la marque -1 peut remplacer une marque (positive) existante. L'objet système contenant la liste des adresses des segments de valeurs doit alors être modifié. La marque -3 est utilisée lors de l'appel à la routine JEVEUS.

On construit ainsi une hiérarchie des segments de valeurs associés aux objets. Chaque appel à la routine JEDEMA va provoquer la libération des segments de valeurs possédant la marque courante. Afin d'optimiser les libérations entraînées par un appel à JEDEMA, la mise en mémoire de chaque segment de valeurs s'accompagne du stockage de sa position (son adresse mémoire) dans un objet système (segment de valeurs de type entier). Ainsi l'ensemble des segments de valeurs associés à une marque identique est facilement identifiable et leur repérage ne nécessite qu'un simple balayage d'un vecteur d'entiers. La boucle sur les segments de valeurs est effectuée en deux temps : on traite tout d'abord globalement les collections, puis les objets simples et les segments de valeurs associés aux collections contiguës sont libérés.

L'actualisation de la marque d'un objet est réalisée lors de l'appel à la routine JJECRS, au besoin, l'objet système KDESMA est redimensionné (cet objet est désigné ici par le nom de la variable Fortran utilisée pour stocker son adresse au sein des unités de programme).

C'est la routine JJLIDE qui réalise effectivement la libération des segments de valeurs. Le premier argument de cette routine est le nom de l'appelant, il conditionne le type d'opération à effectuer :

LIBE mécanisme standard de libération avec examen de l'état, du statut et de la marque,
TASS mécanisme de libération avec écriture immédiate utilisé lors du retassage des fichiers ou en mode debug JEVEUX,
LIBF mécanisme utilisé en fin de travail lors de la fermeture d'une base JEVEUX.

Concernant les objets simples, cette libération ne pose pas de problème particulier : la routine JJLIDE vérifie que la marque associée au segment de valeurs est identique à la marque courante stockée dans le commun /IADMJE/, elle modifie les descripteurs (état et statut) du segment de valeurs et affecte à 0 la marque associée, éventuellement elle provoque un déchargement (JXECRO) et modifie le contenu des attributs adresse mémoire et adresse disque. La libération d'un objet de collection dispersée suit le même processus, les attributs étant modifiés au sein des objets système de collection. La libération d'une collection est plus délicate, les objets système devant être maintenus accessibles en mémoire tant qu'un segment de valeurs associé à l'un des objets de collection est présent en mémoire (utilisé, déchargeable, et même amovible).

5 Gestion des fichiers d'accès direct

Le gestionnaire de mémoire JEVEUX gère les déchargements mémoire sur disque, pour libérer de la place en mémoire au cours de l'exécution et pour archiver les résultats en fin de travail. On utilise à cette fin des fichiers d'accès direct. Ce sont les utilitaires OPENDR, WRITDR, READDR et CLOSDR qui sont appelés (bibc/utilitai/iodr.c).

L'adresse disque des objets JEVEUX est obtenue par combinaison du numéro de l'enregistrement du fichier d'accès direct utilisé pour déposer les valeurs, et éventuellement la position au sein de cet enregistrement.

La longueur des enregistrements est fixe, sa valeur est choisie lors de l'ouverture des bases JEVEUX par l'intermédiaire de la routine JEINIF.

Le nombre d'enregistrements qui fait partie des paramètres, est déterminé dans *Code_Aster* en fonction des conditions d'exploitation. Chaque base est découpée en unité logique de longueur 12 884 901 888octets (notion d'"extend"), cette valeur est affectée à travers la fonction ENVIMA LOFIEM et stockée dans le commun /FENVJE/. JEVEUX gère un index global qu'il découpe ensuite pour chaque extend, l'adresse disque est mesurée par rapport à l'index global, puis modulo le nombre d'enregistrements, on obtient facilement le numéro de l'extend et l'adresse relative. Les différentes unités logiques sont accessibles par un nom local qui est composé à partir des quatre premiers caractères en minuscules du nom de la base associée et du numéro d'extend.

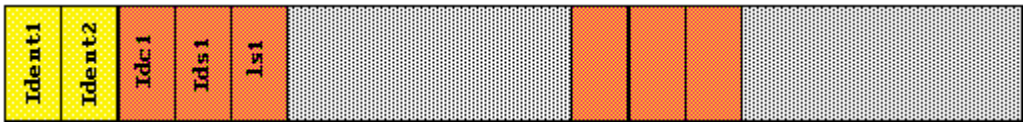
La taille des enregistrements définit deux classes d'objets JEVEUX :

1) les petits objets dont la taille est inférieure à la longueur d'un enregistrement, ils sont accumulés dans un espace de la taille d'un enregistrement avant transfert sur disque,
2) les gros objets qui nécessitent plusieurs enregistrements pour stocker leur contenu.

Lors d'une requête d'écriture sur disque, le contenu des gros objets sera directement transféré sur disque, alors qu'un tampon d'écriture sera utilisé pour les petits objets afin de les cumuler et d'atteindre environ la longueur d'un enregistrement avant leur transfert. Lors d'une requête de lecture, au minimum c'est un enregistrement qui sera utilisé, dans le cas des petits objets on utilise un tampon de lecture. Les tampons de lecture et d'écriture font partie des objets système associés à chaque base JEVEUX.

La fermeture des fichiers d'accès direct est indispensable pour actualiser l'index d'accès, c'est la routine JXFERM qui appel l'utilitaire CLOSDR.

Description des enregistrements



Chaque enregistrement est auto-décrit de façon à pouvoir facilement identifier son contenu. Comme pour la zone mémoire, les enregistrements sont vus comme une suite de mots de type entier (INTEGER*8). Les deux premiers mots donnent une information globale sur la taille des objets stockés :

- si Ident1=Ident2=0 l'enregistrement contient des petits objets, trois mots entiers (descripteurs) sont placés devant chaque segment de valeurs, il contiennent respectivement, quand ils existent, l'identificateur de collection (Idc1), l'identificateur d'objet simple ou le numéro d'objet de collection (Ids1) et la longueur du segment de valeur, suit le segment de valeurs, et on recommence pour le suivant jusqu'à ce que Idcn=Idsn=0 ;
- si Ident1 ou Ident2 est différent de 0, l'enregistrement contient tout ou partie du segment de valeurs associé à un gros objet.

Lors de la destruction d'un gros objet les identificateurs Ident1 et Ident2 sont positionnés à la valeur opposées (affectation du signe -). De même, lors de la destruction d'un petit objet, les identificateurs Idci et Idsi sont affectés du signe -.

Écriture des objets

C'est la routine JXECRO qui traite l'écriture des objets JEVEUX. Elle assure aussi, quand c'est nécessaire, l'ouverture des unités logiques associées à la partition en extend des bases. La routine examine les différents enregistrements pour trouver une suite d'enregistrements pouvant accueillir le segment de valeurs ou le tampon suivant les cas. Les enregistrements correspondant à de gros objets détruits peuvent ainsi être récupérés. L'écriture d'un petit objet se traduit par un déplacement du contenu du segment de valeurs dans le tampon d'écriture par la routine JXDEPS avec actualisation des descripteurs. Le tampon est transféré sur disque uniquement si le segment de valeurs est d'une taille supérieure à l'espace libre restant. Le segment de valeurs associé à de gros objets est transféré sur disque par la routine JXECRB. JXECRB est un chapeau faisant appel à l'utilitaire WRITDR qui actualise les descripteurs Ident1 et Ident2 ainsi qu'un compteur associé à l'enregistrement. Lors des déchargements ultérieurs, le tampon de lecture peut être utilisé pour actualiser l'image disque ; un logique indique alors ce type d'usage.

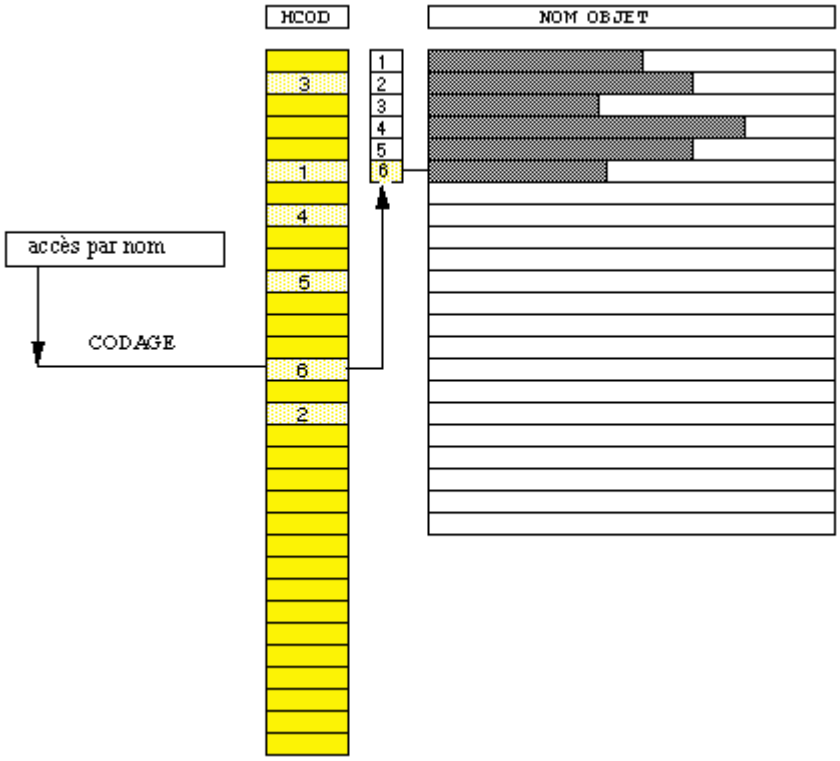
Lecture des objets

C'est la routine JXLIRO qui traite la lecture des objets JEVEUX. Le segments valeurs associé aux petits objets est rechargé en mémoire à partir d'un des tampons de lecture ou d'écriture. Le tampon de lecture peut éventuellement être déchargé sur disque avant de charger un nouvel enregistrement. Le segment de valeurs associé aux gros objets est directement relu à l'aide de la routine JXLIRB.

6 Accès par noms : adressage associatif

Les objets JEVEUX, objets simples et objets de collections, sont accessibles par nom. Les noms manipulés par les routines JE.... comportent 24 caractères, les noms utilisés en interne par JEVEUX comportent 32 caractères pour traiter le cas des collections. L'accès par nom, s'il facilite la lisibilité et permet de structurer les données, ne peut être utilisé directement en interne. On a donc recours à un

algorithme d'adressage associatif qui, à l'aide d'une fonction de codage, permet d'associer un identificateur entier à un nom. Ce système de codage est utilisé pour gérer les noms des objets JEVEUX pour chaque classe définie (associée à chaque base), mais aussi pour les noms des objets de collection.



On utilise à cette fin soit un couple d'objets formé par un vecteur d'entiers et un vecteur de chaînes de caractères pour la gestion des différents classes, soit un objet de genre répertoire de noms ayant la particularité d'être de contenu hétérogène (stockage des chaînes de caractères et des identificateurs entiers) pour les collections nommées. L'usage de ces répertoires requiert des fonctions d'accès particulières. Le dimensionnement de ces objets est effectué de façon à contenir le nombre d'identificateurs requis en minimisant leur taille et les collisions au niveau de la fonction de codage. Notre choix pour dimensionner le pointeur d'entiers associé au résultat de la fonction d'adressage s'est arrêté sur la condition suivante :

$nrep = nprem$ **où** $nprem > 1.3 \times nmax$

$nprem$ est un nombre premier et $nmax$ le nombre maximum d'identificateurs à stocker.

Le calcul de la taille des répertoires est réalisé par la fonction JJPREM dans laquelle est stockée sous forme de DATA une liste de 56 nombres premiers jusqu'à la valeur 611957 ce qui limite à cette valeur la capacité des répertoires de noms (et celle du pointeur d'entiers à environ 795 000).

La fonction de codage choisie JXHCOD fait appel à la fonction système STRMOV qui permet de transférer octet par octet une chaîne de caractères dans un tableau d'entiers et à la fonction XOR pour cumuler les résultats dans un entier (INTEGER). L'identificateur est enfin obtenu par une congruence modulo la longueur du répertoire $nrep$.

Insertion d'un nom

L'insertion d'un nouveau nom dans les répertoires associés aux différentes bases est réalisée à l'aide de la routine JJCREN. L'ensemble des répertoires des bases ouvertes est examiné de façon à assurer l'unicité du nom d'objet, les routines JEVEUX n'acceptant pas comme argument la classe associée à la base.

Si le répertoire de noms commence à être saturé, la fonction de codage, bien que choisie pour être dispersive, peut donner un identificateur identique pour deux chaînes distinctes, il y a alors collision. Un nouvel identificateur doit être déterminé en tenant compte de la valeur obtenue précédemment.

Recherche de nom dans un répertoire

Cet algorithme nécessite un certain nombre de comparaisons de chaînes de caractères et peut donc devenir coûteux. Un commun, actualisé par la dernière recherche dans les répertoires des différentes bases contient l'identificateur et la chaîne associée afin de diminuer le coût de la recherche (`/IATCJE/`). C'est la routine `JJVERN` qui effectue une comparaison avec le contenu du commun avant l'appel à `JJCREN`. Le code retour de la routine `JJCREN` dépend du type de recherche dans le répertoire : avec insertion dans le répertoire de noms (`ICRE=1`) ce code retour est obligatoirement non nul (il y a éventuellement arrêt en erreur), sans insertion il peut valoir 0, sinon 1 correspond à un objet simple et 2 à une collection. C'est la présence d'une chaîne non blanche entre les positions 25 à 32 qui indique que l'on traite un objet de collection.

Les caractères composant les noms des objets sont limités aux caractères alphanumériques complétés par les caractères spéciaux :

' '	le blanc,	'.'	le point,
'_'	le blanc souligné,	'$'	le symbole dollar,
'&'	le et commercial.		

La conformité des chaînes de caractères est vérifiée après insertion dans les répertoires (lors de la création du nom uniquement) par comparaison caractère par caractère avec le contenu du commun `/JCHAJE/` initialisé dans la routine `JEDEBU`.

Destruction d'un nom

La destruction d'un nom utilise le même algorithme que l'insertion, la position dans le répertoire ne peut être libérée en raison d'éventuelles collisions, on procède donc en rendant négatif l'identificateur et en affectant à '?' la chaîne de caractères du nom à détruire. Ainsi il sera toujours possible de récupérer cette position ultérieurement.

Redimensionnement des répertoires

Le redimensionnement des répertoires est assuré de façon automatique à l'aide de la routine `JJAREP`. La taille des répertoires des bases est doublée lors de l'opération. Cette opération reconstruit entièrement le nouveau répertoire par insertion des noms existant. L'ordre d'insertion étant conservé, les objets système ne nécessitent pas de traitement particulier, autre qu'une recopie dans un réceptacle plus grand (il s'en suit un déplacement en mémoire de ces derniers) et leur actualisation sur disque.

Cas des répertoires de collection

Les répertoires de collection sont des objets de contenu non homogène : ils stockent à la fois le résultat (de type entier) de la fonction de codage et les chaînes de caractères composant les noms. Ils sont auto-décrits et les routines les utilisant contiennent les instructions suivantes :

```
INTEGER          ILOREP , IDENO , ILNOM , ILMAX , ILUTI , IDEHC
PARAMETER       ( ILOREP=1,IDENO=2,ILNOM=3,ILMAX=4,ILUTI=5,IDEHC=6)
```

La valeur positionnée à l'adresse :	Représente :
ILOREP	la taille nécessaire à l'ensemble des codes entiers,
IDENO	l'adresse à partir de laquelle les noms sont stockés,
ILNOM	la longueur des chaînes stockées,
ILMAX	la dimension maximum du répertoire,
ILUTI	le nombre de noms effectivement stockés,
IDEHC	l'adresse à partir de laquelle les codes entiers sont stockés.

L'information stockée à l'adresse ILUTI est actualisée et utilisée dans les fonctions internes d'accès aux répertoires, sa valeur est uniquement accessible de façon externe, à l'aide de l'utilitaire JELIRA avec pour nom d'attribut NOMUTI.

Les objets de collection nommée, lors de leur création par la routine JECROC, sont insérés à l'aide de la fonction JJCODN. La routine intermédiaire JJCROC permet, suivant la valeur de son second paramètre, d'insérer un nouveau nom ou de vérifier son existence et de récupérer son ordre d'insertion.

7 Les objets système et les segments de valeurs non référencés

Le gestionnaire de mémoire JEVEUX utilise une partie de la mémoire pour gérer les attributs associés aux objets et pour traiter certaines fonctions. Afin de ne pas multiplier les routines d'accès nous avons choisi d'utiliser les mêmes structures pour les objets JEVEUX et pour la mémoire utilisée pour leur gestion. C'est pourquoi lors de l'impression de la segmentation mémoire on voit apparaître des segments de valeurs associés à des noms illicites au sens utilisateur et faisant référence aux différentes classes ouvertes à un instant donné, mais aussi des segments de valeurs qui ne sont associés à aucun nom. Les objets système associés à la base GLOBALE portent tous le préfixe suivant : _____GLOBALE_____ (le nom de la base est en position 9 à 24), le suffixe (en position 25 à 32) permet de distinguer les différents objets. Les noms des objets système sont construits de la même façon pour les autres bases.

Les objets système sont créés lors du premier appel à la routine JEINIF. JEVEUX ayant besoin en permanence d'accéder aux segments de valeurs associés, une marque spécifique (-2) leur est affectée. Un traitement particulier leur est réservé lors de la fermeture des bases.

Liste des objets système utilisés par JEVEUX :

	Suffixe du nom de l'objet système	Contenu	Type Fortran associé (64 bits)	Taille
1	$$CARA	caractéristiques de la base associée	INTEGER*8	11
2	$$IADD	adresses disque des objets	INTEGER*8	2*NREMAX
3	$$GENR	genre des objets (E,V, N ou X)	CHARACTER*1	NREMAX
4	$$TYPE	type des objets (I,R,C,L,K)	CHARACTER*1	NREMAX
5	$$DOCU	champ documentaire	CHARACTER*4	NREMAX
6	$$ORIG	champ documentaire	CHARACTER*8	NREMAX
7	$$RNOM	liste des noms d'objets	CHARACTER*32	NREMAX
8	$$LTYP	types des segments de valeurs	INTEGER*8	NREMAX
9	$$LONG	longueur mesurée dans le type des segments de valeurs	INTEGER*8	NREMAX
10	$$LONO	longueur effective mesurée dans le type des segments de	INTEGER*8	NREMAX

		valeurs		
11	$$DATE	date de première sauvegarde	INTEGER*8	NREMAX
12	$$LUTI	longueur utilisée des segments de valeurs	INTEGER*8	NREMAX
13	$$HCOD	table d'adressage associatif	INTEGER*8	NRHCOD
14	$$USADI	description du contenu des enregistrements	INTEGER*8	2*NBLMAX
15	$$ACCE	nombre d'accès en lecture/écriture aux enregistrements	INTEGER*8	NBLMAX
16	$$MARQ	marques associées aux objets	INTEGER*8	2*NREMAX
17	$$INDX	index du fichier d'accès direct associé	INTEGER*8	2*NBLMAX
18	$$TLEC	tampon de lecture	INTEGER*8	LONGBL
19	$$TECR	tampon d'écriture	INTEGER*8	LONGBL
20	$$IADM	adresses mémoire des objets	INTEGER*8	NREMAX

où

NREMAX est le nombre maximum de noms associés à une classe,
NRHCOD est obtenu à partir de NREMAX avec la fonction JJPREM,
NBLMAX est le nombre maximum d'enregistrements,
LONGBL est la longueur des enregistrements.

La dimension de la plupart des objets système est susceptible d'être réajustée en cours de calcul suivant les besoins, seul ce qui a trait à la taille des fichiers d'accès direct et à la longueur des enregistrements reste figé. Les 5 derniers objets de la liste ci-dessus n'ont pas d'image disque.

Segments de valeurs non référencés présents en mémoire

Deux segments de valeurs présents en mémoire ne possèdent pas de noms pour les identifier, nous les désignons à partir du nom des variables qui sont utilisées dans les sous-programmes.

	Contenu	Type Fortran associé	Taille
KPOSMA	ISZON(JISZON+KPOSMA+I) est la position dans le segment de valeurs associé à KDESMA des adresses associée à la ième marque	INTEGER*8	LGD
KDESMA	adresses mémoire des objets "marqués"	INTEGER*8	LGP

Les dimensions LGD et LGP sont ajustées au cours de l'exécution, leurs valeurs initiales sont respectivement la somme des longueurs des vecteurs $$RNOM de chaque classe et la valeur 50.

8 Les collections

Les collections d'objets JEVEUX sont des structures qui permettent la mise en commun des attributs et éventuellement un accès nommé à un groupe d'objets. Elles peuvent être associées à un unique segment de valeurs (collection contiguë) ou à autant de segments de valeurs que d'objets (collection dispersée). Elles sont bâties à partir des objets simples JEVEUX, et apparaissent donc sous cette forme parmi les objets associés à une classe. Le principal objet de la collection est l'objet de genre X, c'est un vecteur de 11 entiers contenant les identificateurs des différents objets composant la collection (entre autres les objets système de la collection qui contiennent un suffixe commençant par $$). Ce vecteur porte le nom attribué à l'aide de la routine JECREC (CHARACTER*24). Les objets système propres à la collection portent un suffixe commençant par $$ en position 25, s'ils sont associés à un objet partagé, ils portent un suffixe commençant par &&. Les attributs communs à l'ensemble des objets de collection sont déposés parmi les attributs de l'objet système $$DESO (genre, type, longueur, etc.).

Les objets système associés à une collection sont créés dans la classe associée (attribut identique pour l'ensemble des objets système) et chargés en mémoire par l'intermédiaire de la routine JJCREC.

	Suffixe du nom de l'objet système de collection	Contenu	Type Fortran associé	Type de collection
1	$$DESO les attributs associés à cet objet sont les attributs communs de la collection	collection contiguë : valeurs associées aux différents objets de collection	INTEGER*8	dispersée et contiguë le segment de valeurs n'existe que pour les collections contiguës
2	$$IADD	adresses disque des objets de collections dispersées	INTEGER*8	dispersée
3	$$IADM	adresses mémoire des objets de collections dispersées	INTEGER*8	dispersée
4	$$MARQ	marques associées aux objets de collections dispersées	INTEGER*8	dispersée
5	$$NOM ou bien objet partagé	liste des noms d'objets de collections nommées	suivant le répertoire associé	nommée
6	$$LONG ou bien objet partagé	longueur mesurée dans le type des segments de valeurs; reçoit les valeurs associés aux attribus LONMAX et NOMMAX	INTEGER*8	de longueur variable
7	$$LONO ou &&LONO	longueur effective mesurée dans le type des segments de valeurs; est utilisé en interne par le logiciel	INTEGER*8	de longueur variable
8	$$LUTI ou &&LUTI	longueur utilisée des segments de valeurs	INTEGER*8	de longueur variable
9	$$NUM	informations concernant les collections numérotées	INTEGER*8	numérotée

Les routines utilisant les collections, et plus précisément l'objet descripteur de la collection contiennent les instructions suivantes :

```
      INTEGER         IVNMAX     , IDDESO     ,IDIADD     , IDIADM     ,
     +                IDMARQ     , IDNOM                  , IDLONG     ,
     +                IDLONO     , IDLUTI     ,IDNUM
      PARAMETER     ( IVNMAX = 0 , IDDESO = 1 ,IDIADD = 2 , IDIADM = 3 ,
     +                IDMARQ = 4 , IDNOM  = 5              , IDLONG = 7 ,
     +                IDLONO = 8 , IDLUTI = 9 ,IDNUM  = 10 )
```

Ce qui permet de se positionner directement dans la zone mémoire pour obtenir les identificateurs des objets système (lorsqu'ils existent) dans l'ordre suivant : $$DESO, $$IADD, $$IADM, $$MARQ, $$NOM, $$LONG, $$LONO, $$LUTI et $$NUM. Le nombre maximum d'objets de collection est stocké à l'adresse IVNMAX.

Les collections nommées

Les objets associés à une telle collection sont accessibles par leur nom (fonction JEXNOM) et par leur numéro d'ordre d'insertion (fonction JEXNUM). Il est possible d'utiliser les routines JENUNO et JENONU pour passer du numéro au nom et inversement. La longueur des noms des objets est limitée à 8

caractères (CHARACTER*8) si la collection s'appuie sur un répertoire de nom "interne", ou bien peut valoir 8, 16 ou 24 si la collection s'appuie sur un répertoire de nom "externe", c'est à dire créé préalablement (répertoire de noms partagé).

Les collections numérotées

Les objets associés à une telle collection sont uniquement accessibles par leur numéro d'ordre d'insertion (fonction JEXNUM). L'objet système $$NUM est un vecteur de 2 entiers contenant respectivement le nombre maximum d'objets de collection et le nombre d'objets utilisés.

Les collections dispersées

Chaque objet est associé à un segment de valeurs, il n'est donc pas nécessaire de ramener l'ensemble de la collection pour accéder à un objet particulier. Dans ce cas il est nécessaire de gérer 3 objets système : l'un pour les adresses mémoire des segments de valeurs ($$IADM), l'autre pour les adresses disque ($$IADD) et le dernier pour gérer les libérations ($$MARQ).

Les collections contiguës

Il existe un seul segment de valeurs pour l'ensemble des objets de la collection qui est créé et dimensionné une fois pour toutes lors de la première mise en mémoire d'un des objets de collection. Ce segment de valeur est associé à l'objet système $$DESO.

Les collections de longueur variable

Chaque objet doit être dimensionné : en affectant l'attribut de longueur par la routine JEECRA ou bien en fournissant d'un vecteur de longueur (objet partagé). Dans ce cas 3 objets système sont nécessaires : pour les longueurs ($$LONG ou objet partagé), pour les longueurs dans le type des segments de valeurs associés ($$LONO ou bien &&LONO dans le cas d'un objet partagé) et enfin pour les longueurs utilisées ($$LUTI ou bien &&LUTI dans le cas d'un objet partagé).

Dans le cas des collections contiguës, il est possible accéder directement au vecteur des longueurs cumulées (LONCUM) en utilisant la fonction JEXATR combinée avec l'appel à JEVEUO pour obtenir l'adresse de ce vecteur. Cet accès permet de s'affranchir d'un appel à JEVEUO par objet de collection. La dimension de l'objet système $$LONO est incrémentée de 1 par rapport à la longueur de l'objet système $$LONG à cet effet.
Ce mécanisme a été étendu aux collections dispersées pour accéder aux attributs LONMAX et LONUTI pour chaque objet de collection, l'accès à l'attribut pour chacun des objets de la collection à l'aide de la fonction JELIRA pouvant se révélant coûteux.

Les collections de longueur fixe

Chaque objet possède la même dimension, cet attribut peut être affecté de différentes façons : en affectant directement l'attribut de longueur d'un objet de la collection ou bien l'attribut LONT de longueur totale pour une collection contiguë (appel à JEECRA).

Le mécanisme d'accès aux objets de collection

Les requêtes d'accès aux objets de collection sollicitent les objets système attachés à la collection, de la même manière qu'il est nécessaire d'avoir accès aux objets système associés à une classe lors des requêtes sur les objets simples. Il est donc nécessaire que ces objets soient présents en mémoire dès qu'une requête est effectuée sur l'un des objets de collection. La routine JJALLC est chargée de mettre en mémoire les objets système de collection. Ils obéissent à des règles particulières concernant les libérations car ils ne peuvent être déchargés de la mémoire que lorsque tous les objets de collection ont été eux-mêmes déchargés (actualisation des adresses disque et mémoire). La gestion des objets partagés est encore plus délicate car il faut pouvoir se prémunir d'une libération intempestive de ces derniers, à cette fin, ils reçoivent une marque particulière qui vaut -1.

Les différentes requêtes sur les objets de collection s'effectuent à partir des routines JE utilisées pour les objets simples, mais requièrent l'usage des fonctions de synchronisation JEXNOM, JEXNUM ou JEXATR. Ces fonctions de type CHARACTER*32 mettent à jour le contenu des communs respectifs /IDATJE/, /INUMJE/ et /KNOMJE/, de plus elles remplacent la chaîne de caractères associée au nom de l'objet JEVEUX en position 25 à 32 par les suffixes respectifs $$XNOM , $$XNUM et $$XATR. Les routines de bas niveau iront ensuite rechercher cette information au sein des différents communs suivant le type d'accès.

9 Les poursuites

L'objet système de suffixe $$CARA (contenant le nom de la base associée en position 9 à 24), contient les informations nécessaires à la réouverture du fichier d'accès direct, il contient entre autres, la position du segment de valeurs associé aux adresses disque de l'ensemble des objets contenus dans la base, ainsi que les dimensions caractéristiques des objets système. On prend donc la précaution de le stocker en tête dans le premier enregistrement. En cas de poursuite sur une base, la première action effectuée va être la relecture du contenu de cet objet. La routine JXLIR1 ouvre le fichier associé au premier "extend" (glob.1) avec des caractéristiques qui lui sont propres (ce qui peut conduire à un message d'alarme), lit les 14 premières valeurs (3 pour les descripteurs du segment de valeurs sur disque et les 11 valeurs attendues) puis referme le fichier. La longueur de l'index étant connue, il est alors possible de rouvrir proprement les fichiers (les objets système peuvent avoir été déposés sur les différents fichiers constituant la base). Les objets système n'ayant pas d'image disque sont créés et initialisés (adresse mémoire, tampon d'entrée/sortie, ...).

10 Le traitement des objets JEVEUX

Création des objets

La création des descripteurs (nom et attribut de classe, de genre et de type) des objets JEVEUX est réalisée à l'aide de la routine JECREO pour les objets simples, et par la routine JECREC pour les collections. Le décodage de la chaîne passée en argument pour affecter les attributs est effectué par la routine JJANAL. Dans le cas des objets de genre E les attributs de longueur sont directement affectés.

L'affectation des attributs

Les attributs génériques sont affectés lors de la création du nom d'objet simple ou de collection, ils figurent dans le tableau suivant :

- pour les objets simples et les collections :

CLAS	classe de rattachement de l'objet à une base de données.	V : base Volatile, G : base Globale, L : base Locale , C : base des Catalogues compilés
GENR	genre de l'objet	E : variable simple, V : vecteur, N : répertoire de noms
TYPE	type Fortran de l'objet	I, R, C, L, K8, K16,K24, K32, K80
LTYP	longueur du type	gérée automatiquement pour les types I, R, C, L, normalisée à 8, 16, 24, 32 et 80 pour les caractères

- pour les collections uniquement :

ACCES	Type d'accès : NO si nommée, NU si numérotée	NO peut être suivie du nom du répertoire de noms
STOCKAGE	CONTIG ou DISPERSE	
MODELONG	mode de définition de la longueur	VARIABLE peut être suivie du nom

	des objets de collection : CONSTANT ou VARIABLE	du pointeur de longueur
LONT	longueur totale d'une collection contiguë	
NMAXOC	nombre maximum d'objets de la collection	

Les autres attributs sont affectés à l'aide de la routine JEECRA, ces attributs figurent dans le tableau ci-dessous :

- pour les objets simples ou les objets de collection :

LONMAX	longueur de l'objet de genre V
NOMMAX	longueur de l'objet de genre N
LONUTI	longueur utilisée de l'objet de genre V
NOMUTI	longueur utilisée de l'objet de genre N
DOCU	champ documentaire (4 caractères)

La lecture des attributs

Les valeurs associées aux différents attributs peuvent être consultées à tout moment à l'aide de la routine JELIRA, y compris les attributs gérés en interne :

- attributs internes accessibles :

DATE	date de dernier déchargement disque de l'objet
ORIG	non utilisé
IADM	adresse mémoire
IADD	adresse disque
LONO	longueur mesurée dans le type des segments de valeurs et suivant leur genre
USAGE	usage (statut et état) du segment de valeurs en mémoire : UD, UA, XD, XA ou XX.

Le statut et l'état d'un segment de valeurs en mémoire peuvent être recueillis par cette routine en utilisant USAGE pour valeur de l'argument du nom d'attribut. La valeur XOUS pour ce même argument permet de déterminer si l'objet est une collection (X) ou un objet simple (S).

Remarque :

La consultation des attributs des objets de collection peut requérir la mise en mémoire des objets attribut, et leur libération en fin d'action. Une marque temporaire égale à -2 est affectée dans ce cas.

Requête d'accès aux objets

L'ensemble des requêtes d'accès aux objets JEVEUX (objets simples, objets de collection ou collections entières), qu'elles soient directes (JEVEUO, JEVEUS, JEVEUT) ou indirectes (JEEXIN, JENONU,...) suivent le processus suivant :

- traitement du nom d'objet passé en argument par JJVERN,
- éventuellement, mise en mémoire ou vérification de la présence en mémoire des objets système dans le cas d'une collection par la routine JJALLC, puis à l'aide de la routine JJCROC, détermination de l'identificateur d'objet de collection nommée ou vérification du numéro d'ordre de collection numérotée,
- en fonction du type, appel à JJALTY pour obtenir l'adresse par rapport au tableau Z* du commun de référence,
- éventuellement mise en mémoire puis affectation des identificateurs du segment de valeurs et de la marque, et détermination de l'adresse relative par la routine JXVEUO,
- dans certains cas (par exemple les consultations d'attributs), libération de l'objet et/ou de la collection par JJLIDE.

Les attributs nécessaires à la description de l'objet JEVEUX sont relus ou déterminés par la routine JXVEUO, le traitement est immédiat pour les objets simples car on a accès directement aux attributs

dans les objets système associés à la base, quelques opérations sont nécessaires pour traiter les attributs de collection ou d'objet de collection (positionnement dans les objets système de collection).

La variante JEVEUS alloue de façon permanente le segment de valeurs en mémoire.

Destruction des objets

La destruction d'un objet JEVEUX (objet simple, collection ou objet de collection dispersée) nécessite deux interventions : la destruction du segment de valeurs et la destruction des attributs. Pour un objet simple, le segment de valeurs peut posséder une image disque, dans ce cas il faut aussi détruire cette dernière, le ou les enregistrements correspondants seront marqués libres et pourront être récupérés plus tard. Le segment de valeurs en mémoire sera marqué libre. L'image disque, si elle existe sera marquée libre en utilisant, suivant le type d'objet, les descripteurs de l'enregistrement (objet système $$USADI) ou bien les descripteurs au sein d'un enregistrement (affectation du signe -). Cette fonction est assurée par la routine JXLIBD. Les attributs (nom, longueur, genre, etc.) seront libérés (routine JJMZAT) et leur position dans les objets système de la base associée sera disponible pour les nouvelles création de descripteur. L'objet système contenant l'adresse des objets marqués doit aussi être réactualisé. Le traitement d'un objet de collection est identique, l'actualisation des attributs est réalisée sur les objets système de la collection. Le segment de valeurs pour un objet de collection contiguë ne peut bien sûr pas être détruit. La destruction d'une collection est réalisée en détruisant l'ensemble des objets de collection et des objets système de la collection pourvu qu'ils ne soient pas partagés. Les routines JEDETR et JEDETC permettent de détruire des objets JEVEUX, la première travaille à partir d'un identificateur, la seconde, plus coûteuse, effectue tout d'abord une recherche de descripteur dans les répertoires des classes ouvertes à partir d'une chaîne de caractères à une position donnée. La routine JEDETV est uniquement utilisée pour détruire les objets sur la base volatile associée à la classe V entre les différentes commandes du *Code_Aster*.

Libération explicite des objets

Bien que le mécanisme de libération soit mis en œuvre avec la notion de marque et les appels obligatoires aux routines JEMARQ et JEDEMA, certaines configurations nécessitent un appel explicite aux routines de libération suivantes :

JELIBE libère l'objet demandé en respectant la marque affectée,
JELIBS libère l'objet de nom passé en argument lorsque la marque associée vaut -3,
JELIBZ libère l'ensemble des objets associés à une classe dont la marque associée vaut -1.

Recopie des objets

L'utilitaire JEDUPO permet de dupliquer un objet JEVEUX (objet simple, ou collection complète) éventuellement en déposant le résultat sur une classe différente. Les nouveaux objets sont libérés en fin d'opération. Si cette action ne pose aucune difficulté pour les objets simples, certaines précautions sont à prendre concernant les collections s'appuyant sur des pointeurs externes. Ces derniers peuvent être recréés pour devenir des objets système propres à la collection (on ne bénéficie plus de la mise en commun des attributs concernés) ou bien être conservés tels quels, mais il n'est alors pas permis de déposer le résultat de la recopie sur une autre classe. Le réceptacle peut préexister (l'utilisateur fournit un nom ou une chaîne), dans ce cas il est détruit en début d'opération. La recopie ne nécessite pas obligatoirement la présence en mémoire des segments de valeurs à copier, ils peuvent être relus directement sur disque.

Il est possible de faire usage de l'utilitaire JEDUPC qui travaille à partir d'une sous-chaîne de caractères mais nécessite en revanche une recherche préalable des noms dans le répertoire (ce qui peut s'avérer coûteux).

Impression du contenu des segments de valeurs

L'utilitaire JEIMPO est chargé d'imprimer de façon agréable le contenu du (des) segment(s) de valeur(s) associé(s) aux objets JEVEUX. Les objets système (associés à une classe ou à une collection) sont traités par la routine JEPRAT. Une mise en mémoire pouvant être effectuée, une

marque particulière (-2) est affectée aux segments de valeurs chargés. Suivant le type d'objet (objet simple, objet de collection ou collection) on récupère les attributs associés au(x) segment(s) de valeurs pour appeler la routine JJIMPO qui réalise le formatage des données.

11 Le traitement des bases

Certaines opérations traitent dans leur intégralité les bases JEVEUX, elles sont indispensables pour initier le système de gestion de mémoire, mais sont aussi utilisées en fin de processus. Attention, le contenu des bases est systématiquement enrichie lors de l'exécution de la commande Aster POURSUITE, et il est indispensable de terminer l'exécution par la commande FIN pour fermer proprement les fichiers d'accès direct. Seul un arrêt avec un message UTMESS du type <S> permet au SUPERVISEUR de valider les concepts créés et de fermer proprement les fichiers d'accès directs par appel à la routine JEFINI

L'ouverture d'une base

La longueur des enregistrements du fichier d'accès direct et la longueur initiale du répertoire de noms restent les seuls paramètres réglables associés aux bases JEVEUX. Ils sont précisés lors de l'appel à la routine JEINIF, on indique aussi le statut de la base en début de travail pour éventuellement rouvrir un fichier existant, le statut en fin de travail permet d'éviter des entrées/sorties superflues si la base n'est pas conservée. La réouverture d'une base (commande POURSUITE dans le *Code_Aster* ou lecture du catalogue des éléments compilés) requiert la connaissance de la longueur des enregistrements du fichier d'accès direct et du contenu de certains objets système, le premier enregistrement contient les données indispensables à la reconstitution de ces diverses informations. La routine JXLIR1 est chargée de relire le début du premier enregistrement : on ouvre le fichier d'accès direct (avec un index dont la taille est fixée à 11), on lit les informations en début d'enregistrement, puis on referme le fichier. On peut ensuite ouvrir le fichier d'accès direct avec un tableau d'index de longueur convenable, et relire le contenu des objets système stocké sur disque lors de la précédente exécution.

La fermeture d'une base

L'opération de fermeture d'une base, réalisée par la routine JELIBF, consiste à libérer l'ensemble des objets qui y sont rattachés, avec éventuellement écriture sur disque et à actualiser les objets système. Deux boucles sont nécessaires pour libérer les objets : la première traite les collections, la seconde traite les objets simples. Les objets système sont ensuite déchargés, les adresses disque sont traitées en dernier, les tampons d'entrée/sortie sont vidés, enfin on actualise les caractéristiques de la base sur le premier enregistrement. Le fichier d'accès direct est ensuite fermé par appel à la routine JXFERM.

Le retassage d'une base

Lors des opérations de destruction d'objet JEVEUX, l'espace disque associé est marqué libre mais n'est pas systématiquement récupéré. Le retassage permet "combler" les vides en "remontant" les enregistrements. Il est donc nécessaire de modifier l'attribut adresse disque des objets contenus dans les enregistrements à déplacer. Cette opération est immédiate pour les objets simples, concernant les collections il faut avoir accès à l'objet système contenant les adresses disque (qui lui-même peut être situé l'enregistrement à déplacer!). Il n'y a pas de réorganisation au sein des enregistrements contenant les images de petits objets. On utilise la routine JETASS et on fait appel à la variante "JETASS" de la routine de libération JJLIDE. Cet utilitaire peut être directement appelé par la commande FIN dans le *Code_Aster*.

La recopie des bases

Cette opération doit être effectuée pour prendre en compte effectivement le retassage, les fichiers d'accès direct WRITDR ne pouvant être réduits en place. La routine JXCOPY travaille à partir de bases fermées et les restitue dans le même état. Cette utilitaire peut être appelé par la commande FIN dans le *Code_Aster*.

12 Les impressions

La routine JEIMPD permet d'imprimer la liste des objets JEVEUX présents sur une ou plusieurs bases. La liste est constituée à partir du catalogue associé (objet système $$RNOM) et on imprime les informations suivantes pour chaque objet :

- l'identificateur associé au nom d'objet,
- le nom de l'objet,
- le genre de l'objet,
- le type de l'objet,
- la longueur dans le type,
- la longueur en octet du segment de valeurs,
- le numéro de l'enregistrement contenant le segment de valeurs,
- la position dans l'enregistrement pour les petits objets,
- le nombre d'accès en lecture sur l'enregistrement,
- le nombre d'accès en écriture sur l'enregistrement.

```
--------------------------------------------------------------------------------

       CONTENU DE LA BASE G

  NOM DE LA BASE               : GLOBALE
  NB D'ENREGISTREMENTS MAXIMUM : 5242
  LONGUEUR D'ENREGISTREMENT (OCTETS): 819200

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

---- NUM ------------ NOM ---------------- G T -L- -LOTY- -IADD- --LIADD- NB AC

       1 ________GLOBALE ________$$CARA    -V-I-  8     11      1       24     0
       2 ________GLOBALE ________$$IADD    -V-I-  8   4000      1      136     0
       3 ________GLOBALE ________$$GENR    -V-K-  1   2000      1    32160     0
       4 ________GLOBALE ________$$TYPE    -V-K-  1   2000      1    34184     0
       5 ________GLOBALE ________$$DOCU    -V-K-  4   2000      1    36208     0
       6 ________GLOBALE ________$$ORIG    -V-K-  8   2000      1    44232     0
       7 ________GLOBALE ________$$RNOM    -V-K- 32   2000      1    60256     0
       8 ________GLOBALE ________$$LTYP    -V-I-  8   2000      1   124280     0
       9 ________GLOBALE ________$$LONG    -V-I-  8   2000      1   140304     0
      10 ________GLOBALE ________$$LONO    -V-I-  8   2000      1   156328     0
      11 ________GLOBALE ________$$DATE    -V-I-  8   2000      1   172352     0
      12 ________GLOBALE ________$$LUTI    -V-I-  8   2000      1   188376     0
      13 ________GLOBALE ________$$HCOD    -N-I-  8   4177      1   204400     0
      14 ________GLOBALE ________$$USADI   -V-I-  8  10484      1   237840     0
      15 ________GLOBALE ________$$ACCE    -V-I-  8   5242      1   321736     0
...
```

L'impression de la mémoire

La routine JEIMPM permet d'imprimer la liste des objets JEVEUX présents en mémoire. On imprime les informations suivantes :

- la classe de l'objet,
- l'identificateur de collection associé au nom d'objet ou 0,
- l'identificateur d'objet simple ou d'objet de collection associé au nom d'objet,
- la valeur (entière) de la marque associée au segment de valeurs,
- l'adresse mémoire relative du segment de valeurs,
- le statut du segment de valeurs (X ou U),
- la longueur mesurée en unité d'adressage (entier) du segment de valeurs,
- l'état du segment de valeurs (X, A ou D),
- le nom de l'objet (éventuellement complété par le numéro d'objet de collection).

```
--------------------------------------------------------------------------------
OBJETS ALLOUES DYNAMIQUEMENT
--------------------------------------------------------------------------------

 CL-  --NUM-- -MA-  ---------IADY--------- -U- - LON UA -  -S- ------------- NOM --------------

|G|   0|       1|  -2|           69888784|U|         11| D| ________GLOBALE ________$$CARA
|G|   0|       2|  -2|          108444896|U|       4000| D| ________GLOBALE ________$$IADD
```

```
|G|    0|        3|   -2|            106261840|U|         251| D| ________GLOBALE ________$$GENR
|G|    0|        4|   -2|            106263920|U|         251| D| ________GLOBALE ________$$TYPE
|G|    0|        5|   -2|            105733504|U|        1001| D| ________GLOBALE ________$$DOCU
|G|    0|        6|   -2|            108476976|U|        2000| D| ________GLOBALE ________$$ORIG
|G|    0|        7|   -2|            108493056|U|        8004| D| ________GLOBALE ________$$RNOM
|G|    0|        8|   -2|            108557168|U|        2000| D| ________GLOBALE ________$$LTYP
|G|    0|        9|   -2|            108573248|U|        2000| D| ________GLOBALE ________$$LONG
|G|    0|       10|   -2|            108589328|U|        2000| D| ________GLOBALE ________$$LONO
|G|    0|       11|   -2|            108605408|U|        2000| D| ________GLOBALE ________$$DATE
|G|    0|       12|   -2|            108621488|U|        2000| D| ________GLOBALE ________$$LUTI
|G|    0|       13|   -2|            108637568|U|        3203| D| ________GLOBALE ________$$HCOD
|G|    0|       14|   -2|     46912496140304|U|      188742| D| ________GLOBALE ________$$USADI
|G|    0|       15|   -2|            106270864|U|       62914| D| ________GLOBALE ________$$ACCE
|G|    0|       16|   -2|            105685344|U|        4000| D| ________GLOBALE ________$$MARQ
|G|    0|       17|   -2|            105717424|U|        2000| D| ________GLOBALE ________$$INDI
|G|    0|       18|   -2|            106774256|U|      102400| D| ________GLOBALE ________$$TLEC
|G|    0|       19|   -2|            107593536|U|      102400| D| ________GLOBALE ________$$TECR
|G|    0|       20|   -2|            108412816|U|        4000| D| ________GLOBALE ________$$IADM
|G|    0|       21|    0|             96179360|X|          21| D| &FOZERO             .PROL
|G|    0|       22|    0|             85544864|X|           2| D| &FOZERO             .VALE
|G|    0|       23|    0|             71159344|X|           1| D| &&_NUM_CONCEPT_UNIQUE
|G|    0|       24|    0|            111103824|X|        5010| D| &&SYS.KRESU
|G|    0|       25|    0|             76660688|X|          11| D| &CATA.ACOUSTIQUE
|G|    0|       26|    0|            111189040|X|         136| D| &CATA.ACOUSTIQUE          $$DESO
…
|G|    0|       37|    0|            113181328|X|         379| D| &CATA.GD.NOMCMP           $$NOM
|G|    0|       38|    0|            113184432|X|         149| D| &CATA.GD.NOMCMP           $$LONG
|G|    0|       39|    0|            113185696|X|         150| D| &CATA.GD.NOMCMP           $$LONO
|G|    0|       40|    0|            113186976|X|         149| D| &CATA.GD.NOMCMP           $$LUTI
|G|    0|       41|    0|            113188240|X|         379| D| &CATA.GD.NOMGD
|G|    0|       42|    0|            113191344|X|         149| D| &CATA.GD.TYPEGD
|G|    0|       43|    0|             86518560|X|          11| D| &CATA.MECANIQUE
|G|    0|       44|    0|            113192608|X|       15776| D| &CATA.MECANIQUE           $$DESO
...
|G|    0|      135|    0|             91910000|U|          11| D| MA1      .GROUPENO
|G|  135|        1|    0|             78747056|X|           1| D| MA1      .GROUPENO                1
|G|  135|        2|    0|            106203168|X|           1| D| MA1      .GROUPENO                2
|G|  135|        3|    0|             74182160|X|           1| D| MA1      .GROUPENO                3
|G|  135|        4|    0|             79595680|X|           1| D| MA1      .GROUPENO                4
...
```

L'impression des répertoires système

La routine JEIMPR permet d'imprimer les répertoires associés aux différentes bases ouvertes, elle parcourt séquentiellement l'objet système $$RNOM et imprime les valeurs des attributs lorsque le premier caractère du nom est différent de '?'. On imprime les informations suivantes :

- l'identificateur d'objet simple,
- le nom de l'objet simple,
- le genre de l'objet,
- le type de l'objet,
- la longueur du type utilisé,
- la longueur de l'objet (attribut LONMAX ou NOMMAX),
- la longueur mesurée dans le type du segment de valeurs,
- le numéro de l'enregistrement contenant l'image disque,
- l'adresse mémoire relative du segment de valeurs
- l'adresse du pointeur alloué dynamiquement.

```
--------------------------------------------------------------------------------
------     CATALOGUE CLASSE G
------
--------------------------------------------------------------------------------

--- NUM  -------------- NOM ---------------- G T L- --LONG--- -LOTY- -IADD- --------KADM------- --------KDYN-------

       1 ________GLOBALE ________$$CARA    -V-I- 8       11      11      1            1090758           69888784
       2 ________GLOBALE ________$$IADD    -V-I- 8     4000    4000      1            5910272          108444896
       3 ________GLOBALE ________$$GENR    -V-K- 1     2000    2000      1            5637390          106261840
       4 ________GLOBALE ________$$TYPE    -V-K- 1     2000    2000      1            5637650          106263920
       5 ________GLOBALE ________$$DOCU    -V-K- 4     2000    2000      1            5571348          105733504
       6 ________GLOBALE ________$$ORIG    -V-K- 8     2000    2000      1            5914282          108476976
       7 ________GLOBALE ________$$RNOM    -V-K-32     2000    2000      1            5916292          108493056
       8 ________GLOBALE ________$$LTYP    -V-I- 8     2000    2000      1            5924306          108557168
       9 ________GLOBALE ________$$LONG    -V-I- 8     2000    2000      1            5926316          108573248
      10 ________GLOBALE ________$$LONO    -V-I- 8     2000    2000      1            5928326          108589328
```

```
     11  ________GLOBALE ________$$DATE    -V-I- 8     2000    2000      1           5930336          108605408
     12  ________GLOBALE ________$$LUTI    -V-I- 8     2000    2000      1           5932346          108621488
     13  ________GLOBALE ________$$HCOD    -V-I- 8     3203    3203      1           5934356          108637568
     14  ________GLOBALE ________$$USADI   -V-I- 8   188742  188742      2     5864054372198     46912496140304
     15  ________GLOBALE ________$$ACCE    -V-I- 8    62914   62914      1           5638518          106270864
     16  ________GLOBALE ________$$MARQ    -V-I- 8     4000    4000      0           5565328          105685344
     17  ________GLOBALE ________$$INDI    -V-I- 8     2000    2000      0           5569338          105717424
     18  ________GLOBALE ________$$TLEC    -V-I- 8   102400  102400      0           5701442          106774256
     19  ________GLOBALE ________$$TECR    -V-I- 8   102400  102400      0           5803852          107593536
     20  ________GLOBALE ________$$IADM    -V-I- 8     4000    4000      0           5906262          108412816
     21  &FOZERO            .PROL          -V-K-24        6       6      0           4377080           96179360
     22  &FOZERO            .VALE          -V-R- 8        2       2      0           3047768           85544864
     23  &&_NUM_CONCEPT_UNIQUE             -V-I- 8        1       1      0           1249578           71159344
     24  &&SYS.KRESU                       -V-K-80      500     500      0           6242638          111103824
```

...

L'impression des attributs

La routine JEIMPA imprime l'ensemble des attributs pour un objet JEVEUX.

```
 ECRITURE DES ATTRIBUTS DE "MA1     .DIME           "

JEIMPA  IMPRESSION DES ATTRIBUTS DE >MA1     .DIME           <

CLAS    G
GENR    V
TYPE    I
LTYP               4
DOCU
DATE               0
LONMAX             6
LONUTI             6
LONO               6
IADM        20357178
IADD               0
LADD               0
USAGE   X D

  ECRITURE DES ATTRIBUTS DE "MA1     .CONNEX         "

JEIMPA  IMPRESSION DES ATTRIBUTS DE >MA1     .CONNEX         <

ACCES   NU
STOCKAGECONTIG
MODELONGVARIABLE
NMAXOC           204
NUTIOC             0
LONT            1472
CLAS    G
GENR    V
TYPE    I
LTYP               4
DOCU
DATE     -1292845870
LONO            1472
IADM        20270746
IADD               0
LADD               0
USAGE   U D
```

Remarque :

L'impression des attributs des objets de collection ou de leur contenu peut requérir la mise en mémoire des objets attribut, et leur libération en fin d'action. Une marque temporaire égale à -2 est affectée dans ce cas.

L'impression du contenu d'un segment de valeurs

La routine JEIMPO permet d'imprimer le ou les segments de valeurs associés à un objet JEVEUX.

```
 IMPRESSION SEGMENT DE VALEURS >MA1     .DIME                   <
 >>>>>
      1 -          361            0          204            0            0
      6 -            3
```

```
 IMPRESSION SEGMENT DE VALEURS >MA1     .COORDO    .VALE        <
 >>>>>
      1 -  0.00000D+00  5.00000D-01  1.00000D+00  5.00000D-01  5.00000D-01
      6 -  1.00000D+00  1.00000D+00  5.00000D-01  1.00000D+00  0.00000D+00
     11 -  7.50000D-01  1.00000D+00  5.00000D-01  7.50000D-01  1.00000D+00
     16 -  1.00000D+00  7.50000D-01  1.00000D+00  0.00000D+00  5.00000D-01
     21 -  2.00000D+00  5.00000D-01  5.00000D-01  2.00000D+00  1.00000D+00
     26 -  5.00000D-01  2.00000D+00  0.00000D+00  7.50000D-01  2.00000D+00
     31 -  5.00000D-01  7.50000D-01  2.00000D+00  1.00000D+00  7.50000D-01
     36 -  2.00000D+00  5.00000D-01  1.00000D+00  1.00000D+00  1.00000D+00
     41 -  1.00000D+00  1.00000D+00  5.00000D-01  1.00000D+00  2.00000D+00
     46 -  1.00000D+00  1.00000D+00  2.00000D+00  0.00000D+00  1.00000D+00
     51 -  1.00000D+00  0.00000D+00  1.00000D+00  2.00000D+00  0.00000D+00
     56 -  5.00000D-01  3.00000D+00  5.00000D-01  5.00000D-01  3.00000D+00
     61 -  1.00000D+00  5.00000D-01  3.00000D+00  0.00000D+00  7.50000D-01
     66 -  3.00000D+00  5.00000D-01  7.50000D-01  3.00000D+00  1.00000D+00
     71 -  7.50000D-01  3.00000D+00  5.00000D-01  1.00000D+00  3.00000D+00
     76 -  1.00000D+00  1.00000D+00  3.00000D+00  0.00000D+00  1.00000D+00
     81 -  3.00000D+00  0.00000D+00  0.00000D+00  1.00000D+00  5.00000D-01
...
```

13 ANNEXE 1 : Description des communs utilisés dans le gestionnaire de mémoire JEVEUX

- /FENVJE/

```
INTEGER          LFIC,MFIC
COMMON /FENVJE/  LFIC,MFIC
```

LFIC	longueur maximun en octets d'un extend,
MFIC	taille maximum en octets d'espace disque utilisable.

- /IACCED//JIACCE/

```
 PARAMETER  ( N = 5 )
COMMON /IACCED/  IACCE(1)
COMMON /JIACCE/  JIACCE(N)
```

	variable de référence et position du segment de valeurs associé à l'objet système de suffixe :
IACCE , JIACCE	$$ACCE

- /IADMJE/

```
INTEGER          IPGC, KDESMA, LGD, LGDUTI, KPOSMA, LGP,
LGPUTI
COMMON /IADMJE/  IPGC, KDESMA, LGD, LGDUTI, KPOSMA, LGP,
LGPUTI
```

IPGC	Valeur de la marque courante (varie entre -3 et n),
KDESMA	adresse du segments de valeurs contenant les adresses des objets marqués,
LGD	longueur du segment de valeur associé à KDESMA,
LGDUTI	longueur utilisée du segment de valeur associé à KDESMA,
KPOSMA	adresse du segments de valeurs contenant les positions associées à chaque marque,
LGP	longueur du segment de valeur associé à KPOSMA,
LGPUTI	longueur utilisée du segment de valeur associé à KPOSMA.

- /IATCJE/

```
INTEGER          ICLAS ,ICLAOS , ICLACO , IDATOS , IDATCO ,
IDATOC
COMMON /IATCJE/  ICLAS ,ICLAOS , ICLACO , IDATOS , IDATCO ,
IDATOC
```

ICLAS	classe courante,
ICLAOS	classe de l'objet simple,
ICLACO	classe de la collection,
IDATOS	identificateur de l'objet simple,
IDATCO	identificateur de la collection,
IDATOC	identificateur de l'objet de collection.

- /IATRJE//JIATJE/

```
PARAMETER  ( N = 5 )
INTEGER          LTYP    , LONG    , DATE    , IADD    , IADM
,
```

```
              +                 LONO    , HCOD    , CARA    , LUTI    ,
IMARQ
               COMMON /IATRJE/  LTYP(1) , LONG(1) , DATE(1) , IADD(1) ,
IADM(1) ,
              +                 LONO(1) , HCOD(1) , CARA(1) , LUTI(1) ,
IMARQ(1)

               COMMON /JIATJE/  JLTYP(N), JLONG(N), JDATE(N), JIADD(N),
JIADM(N),
              +                 JLONO(N), JHCOD(N), JCARA(N), JLUTI(N),
JMARQ(N)
```

variable de référence et position du segment de valeurs associé à l'objet système de suffixe :

LTYPE, JLTYP	$$LTYP
LONG , JLONG	$$LONG
DATE , JDATE	$$DATE
IADD , JIADM	$$IADD
IADM , JIADD	$$IADM
LONO , JLONO	$$LONO
HCOD , JHCOD	$$HCOD
CARA , JCARA	$$CARA
LUTI , JLUTI	$$LUTI
IMARQ, JMARQ	$$MARQ

- /ICODJE/

```
       INTEGER          NUMATR
       COMMON /IDATJE/  NUMATR
```

NUMATR identificateur de l'objet système de collection $$LONO

- /IDATJE/

```
       PARAMETER  ( N = 5 )
       INTEGER          NRHCOD    , NREMAX    , NREUTI
       COMMON /ICODJE/  NRHCOD(N) , NREMAX(N) , NREUTI(N)
```

NRHCOD	taille (en entier) de l'objet système $$HCOD
NREMAX	taille (en entier) de l'objet système $$RNOM
NREUTI	longueur utilisée de l'objet système $$RNOM

- /IENVJE/

```
       INTEGER          LBIS , LOIS , LOLS , LOUA , LOR8 , LOC8
       COMMON /IENVJE/  LBIS , LOIS , LOLS , LOUA , LOR8 , LOC8
```

LBIS	longueur en bits de l'entier,
LOIS	longueur en octets de l'entier,
LOLS	longueur en octets du logique,
LOUA	longueur en octets de l'unité d'adressage,
LOR8	longueur en octets du réel,
LOC8	longueur en octets du complexe.

- /IEXTJE/

```
       PARAMETER  ( N = 5 )
```

```
      INTEGER          IDN    , IEXT    , NBENRG
      COMMON /IEXTJE/  IDN(N) , IEXT(N) , NBENRG(N)
```

IDN	n'est plus utilisé depuis l'emploi des accès nommés.
IEXT	nombre d'extends ouverts
NBENRG	nombre d'enregistrements maximum d'un extend

- /IFICJE/

```
      PARAMETER      ( N = 5 )
      INTEGER          NBLMAX    , NBLUTI    , LONGBL    ,
     +                 KITLEC    , KITECR    , KINDEF    , KIADM
,
     +                 IITLEC    , IITECR    , NITECR    , KMARQ
      COMMON /IFICJE/  NBLMAX(N) , NBLUTI(N) , LONGBL(N) ,
     +                 KITLEC(N) , KITECR(N) , KINDEF(N) , KIADM(N)
,
     +                 IITLEC(N) , IITECR(N) , NITECR(N) , KMARQ(N)
```

Pour chaque base associée à l'indice I compris entre 1 et N

NBLMAX	nombre maximun d'enregistrements,
NBLUTI	nombre d'enregistrements utilisés,
LONGBL	longueur en octets des enregistrements,
KITLEC	adresse dans K1ZON du segment de valeurs associé au tampon de lecture,
KITECR	adresse dans K1ZON du segment de valeurs associé au tampon d'écriture,
KINDEF	adresse dans ISZON du segment de valeurs associé à l'index utilisé pour les fichiers d'accès direct,
KIADM	adresse dans ISZON du segment de valeurs associé aux adresses mémoire,
IITLEC	adresse disque du tampon de lecture (numéro d'enregistrement),
IITECR	adresse disque du tampon d'écriture (numéro d'enregistrement),
NITECR	taille en octets de la portion du tampon d'écriture utilisée,
KMARQ	adresse dans ISZON du segment de valeurs associé aux marques.

- /ILOCJE/

```
INTEGER          ILOC
COMMON /ILOCJE/  ILOC
```

ILOC pointeur sur l'adresse du début de la zone mémoire allouée par JXALLM.

- /INUMJE/

```
INTEGER          NUMEC
COMMON /INUMJE/  NUMEC
```

NUMEC numéro de l'objet de collection ou numéro d'insertion dans un répertoire de noms.

- /ISTAJE/

```
INTEGER          ISTAT
COMMON /ISTAJE/  ISTAT(4)
```

ISTAT code associé à l'état et au statut des segments de valeurs
ISTAT(1) correspond à X
ISTAT(2) correspond à U
ISTAT(3) correspond à A
ISTAT(4) correspond à D

- /IXADJE/

```
INTEGER          IDINIT,IDXAXD
COMMON /IXADJE/  IDINIT,IDXAXD
```

IDINIT	vaut 5, debut de la zone mémoire gérée,
IDXAXD	position initiale dans ISZON pour la recherche.

- /IZONJE/

```
INTEGER          LK1ZON , JK1ZON , LISZON , JISZON , ISZON(1)
COMMON /IZONJE/  LK1ZON , JK1ZON , LISZON , JISZON
EQUIVALENCE    ( ISZON(1) , K1ZON(1) )
```

LK1ZON	longueur en caractère (CHARACTER*1) de la zone gérée,
JK1ZON	adresse dans K1ZON du début de la zone,
LISZON	longueur en entier (INTEGER*8 en adressage 64 bits, INTEGER*4 en

	adressage 32 bits) de la zone gérée,
JISZON	adresse dans ISZON du début de la zone,
ISZON	zone mémoire en entier (INTEGER*8 en adressage 64 bits, INTEGER*4 en adressage 32 bits) gérée dynamiquement ; ce tableau de type entier ne fait pas partie des variables déposées dans le commun, mais il est mis en équivalence avec le tableau de type caractère.

L'ordre EQUIVALENCE permet d'aligner les deux tableaux de type entier et caractère de façon à pouvoir utiliser indifféremment l'un ou l'autre suivant les besoins.

- /KUSADI//JUSADI/

```
 PARAMETER  ( N = 5 )
COMMON /KUSADI/  IUSADI(1)
COMMON /JUSADI/  JUSADI(N)
```

	variable de référence et position du segment de valeurs associé à l'objet système de suffixe :
IACCE , JIACCE	$$USADI

- /JCHAJE/

```
INTEGER          ILLICI , JCLASS(0:255)
COMMON /JCHAJE/  ILLICI , JCLASS
```

ILLICI	vaut -1,
JCLASS	est affecté au résultat de la fonction ICHAR sur les caractères licites, sinon vaut ILLICI.

- /JENVJE/

```
INTEGER          MSLOIS
COMMON /JENVJE/  MSLOIS
```

MSLOIS	masque valant la somme des poids des LOIS-1 premiers entiers, destiné à remplacer l'opération modulo (LOIS) par la fonction AND

- /JCONJE/

```
INTEGER          MSSTAT, LSSTAT
COMMON /JCONJE/  MSSTAT, LSSTAT
```

MSSTAT	n'est plus utilisé, masque valant la somme des poids des LSSTAT premiers entiers.
LSSTAT	(LBISEM - 4) où LBISEM est la longueur en bit de l'entier, est utilisé dans JELIRA pour obtenir l'équivalent sous forme de caractère du statut ou de l'état associé à un segment de valeurs.

- /KATRJE/,/JKATJE

```
PARAMETER  ( N = 5 )
CHARACTER*1      GENR    , TYPE
CHARACTER*4      DOCU
CHARACTER*8      ORIG
CHARACTER*32     RNOM
COMMON /KATRJE/  GENR(8) , TYPE(8) , DOCU(2) , ORIG(1) ,
RNOM(1)
```

```
COMMON /JKATJE/  JGENR(N), JTYPE(N), JDOCU(N), JORIG(N),
JRNOM(N)
```

variable de référence et position du segment de valeurs associé à l'objet système de suffixe :

GENR	$$GENR
GENR	$$TYPE
DOCU	$$DOCU
ORIG	$$ORIG
RNOM	$$RNOM

- /KBASJE/

```
       PARAMETER  ( N = 5 )
       CHARACTER*8      NOMBAS
       COMMON /KBASJE/  NOMBAS(N)
```

NOMBA nom de la base (utilisé pour les messages d'erreur)

- /KFICJE/

```
       PARAMETER  ( N = 5 )
       CHARACTER*2      DN2
       CHARACTER*5      CLASSE
       CHARACTER*8                  NOMFIC    , KSTOUT    , KSTINI
       COMMON /KFICJE/  CLASSE    , NOMFIC(N) , KSTOUT(N) ,
KSTINI(N) ,
      +                 DN2(N)
```

DN2	inutilisé !
CLASSE	permet de stocker l'ensemble des noms des classes ouvertes (première lettre du nom de base)
NOMFIC	nom des bases ouvertes
KSTOUT	statut en sortie des bases ('SAUVE' ou 'DETRUIT')
KSTINI	statut en poursuite des bases ('DUMMY','DEBUT' ou 'POURSUIT')

- /KNOMJE/

```
       CHARACTER*24     NOMEC
       COMMON /KNOMJE/  NOMEC
```

NOMEC nom d'objet de collection ou nom à insérer dans un répertoire

- /KZONJE/

```
       CHARACTER*1      K1ZON
       COMMON /KZONJE/  K1ZON(8)
```

K1ZON zone mémoire en caractère (CHARACTER*1) gérée dynamiquement.

- /NFICJE/

```
       INTEGER          NBCLA
       COMMON /NFICJE/  NBCLA
```

NBCLA nombre de classes ouvertes simultanément

- /NOMCJE/

```
       CHARACTER *24                       NOMCO
       CHARACTER *32    NOMUTI , NOMOS ,          NOMOC , BL32
       COMMON /NOMCJE/  NOMUTI , NOMOS , NOMCO , NOMOC , BL32
```

NOMUTI	nom utilisé (dans les appels aux routine JE....),
NOMOS	nom de l'objet simple,
NOMCO	nom de la collection,
NOMOC	nom de l'objet de collection,
BL32	chaîne blanche de longueur 32.

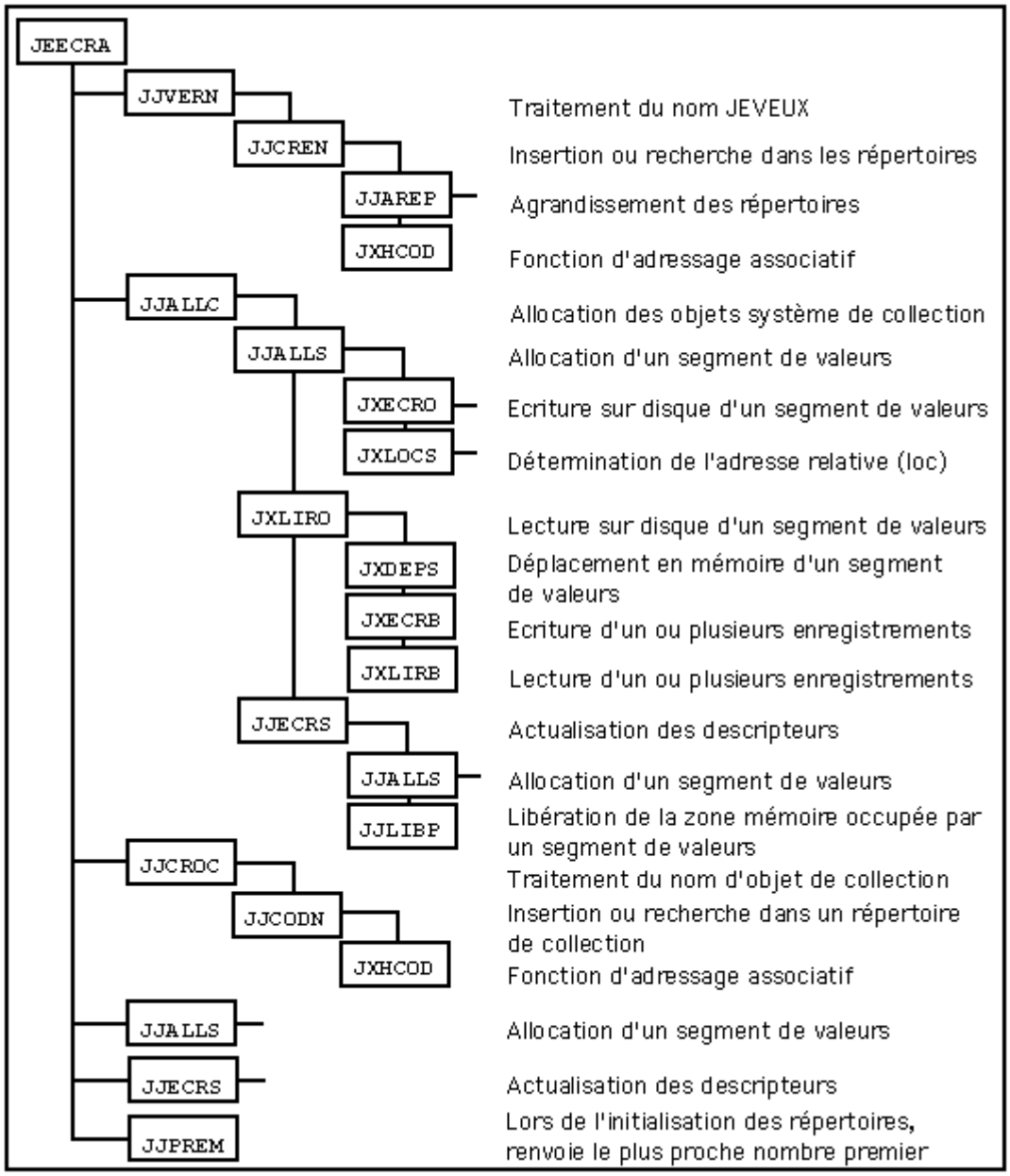
- /UNDFJE

```
INTEGER          LUNDEF,IDEBUG
COMMON /UNDFJE/  LUNDEF,IDEBUG
```

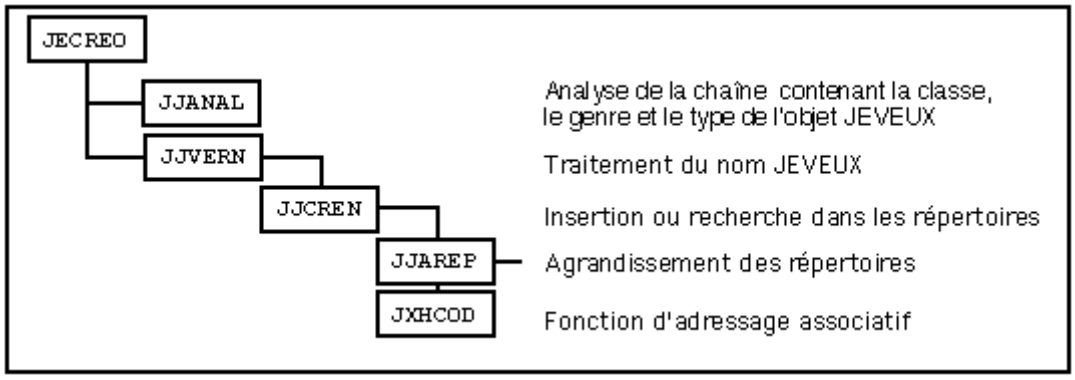
LUNDEF	affecté dans JEDEBU par la fonction envima ISNNEM (not a number),
IDEBUG	vaut 1 en mode debugg, 0 sinon.

14 ANNEXE 2 : Arbres d'appel simplifiés de quelques sous-programmes

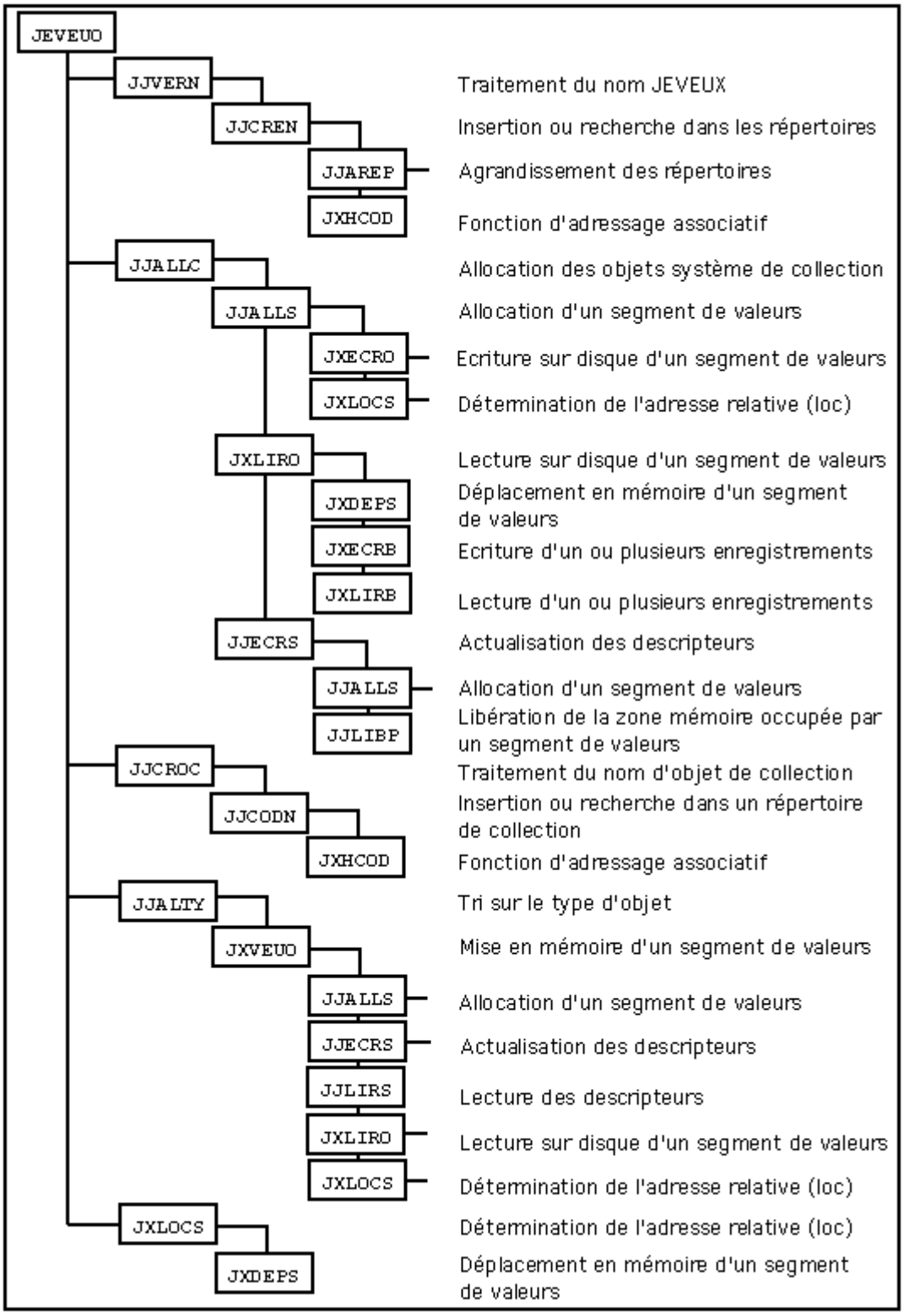
On présente ci-dessous les arbres des principaux sous-programmes JEVEUX, on a volontairement limité à trois niveaux de sous-programmes pour faciliter la compréhension. Les branches tronquées indiquent qu'il existe d'autres appels JEVEUX dans le sous-programme.



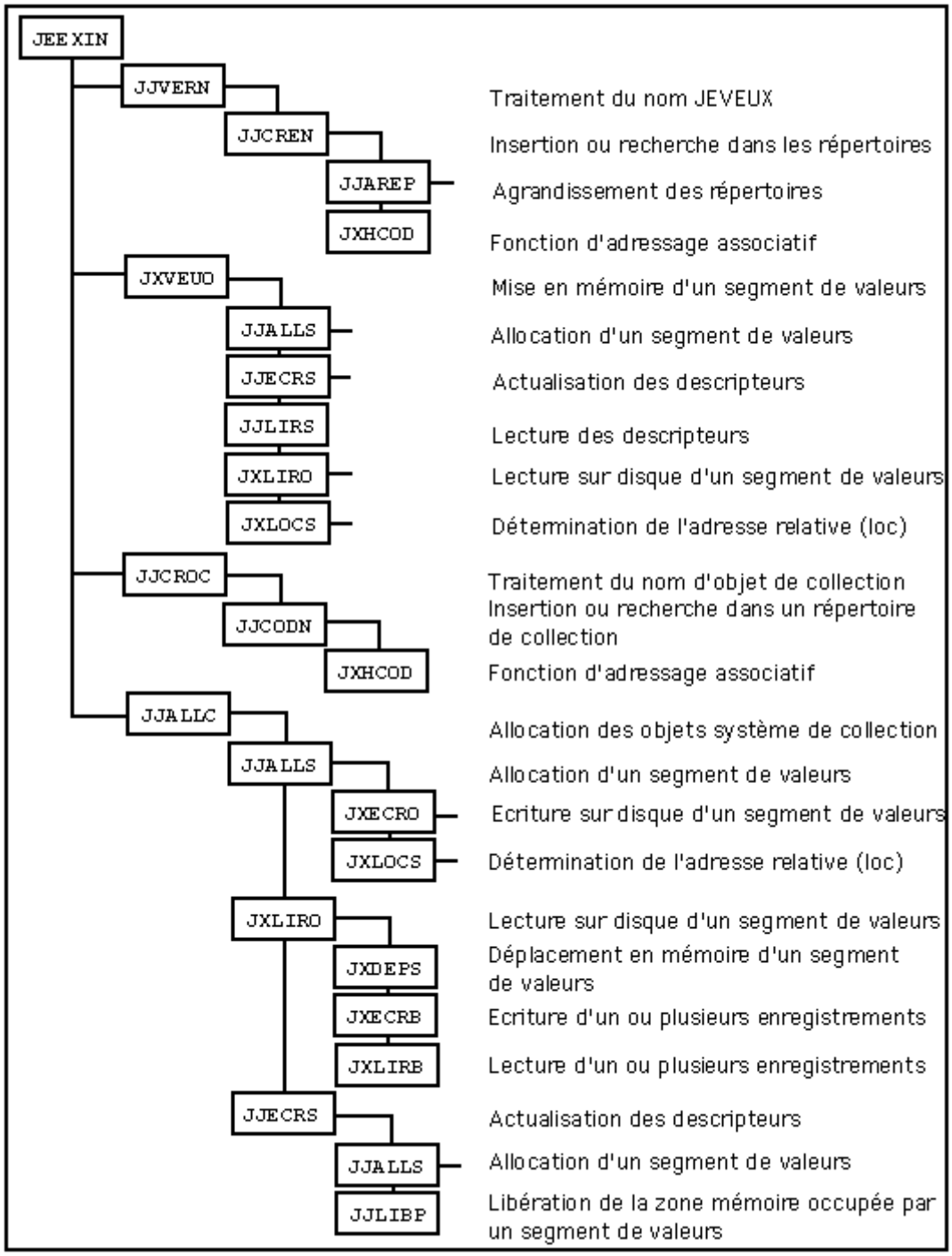
Arbre d'appel de la routine JEECRA



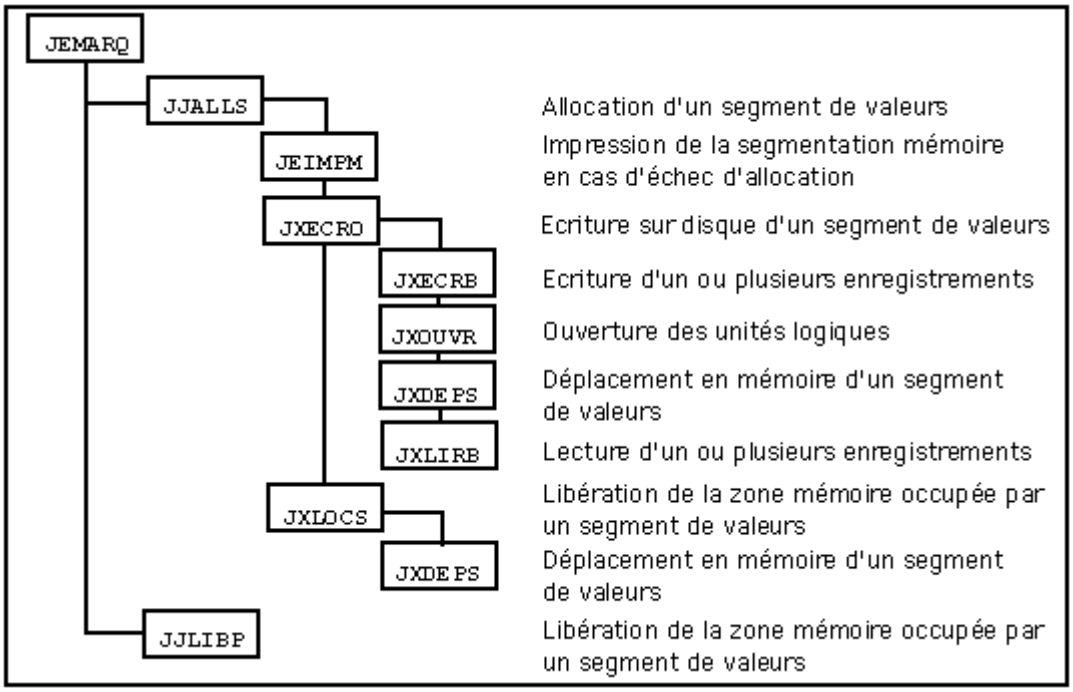
Arbre d'appel de la routine JECREO



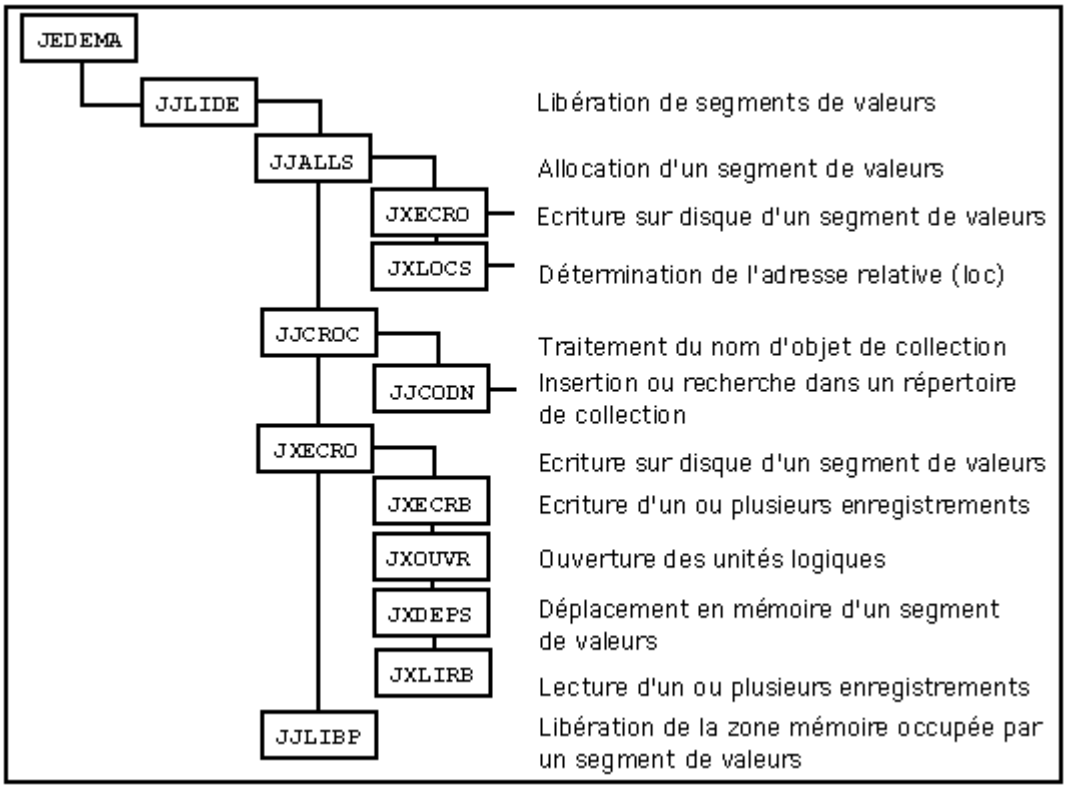
Arbre d'appel de la routine JEVEUO



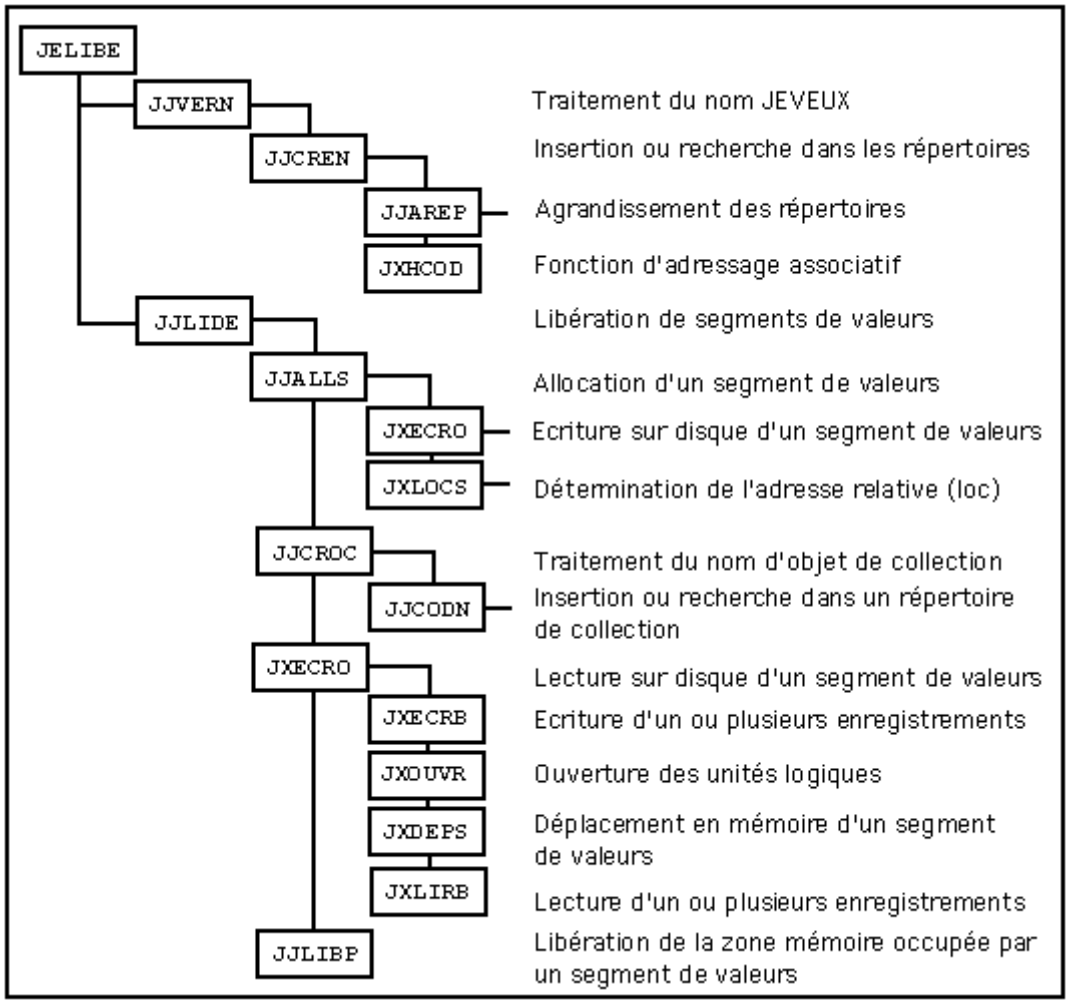
Arbre d'appel de la routine JEEXIN



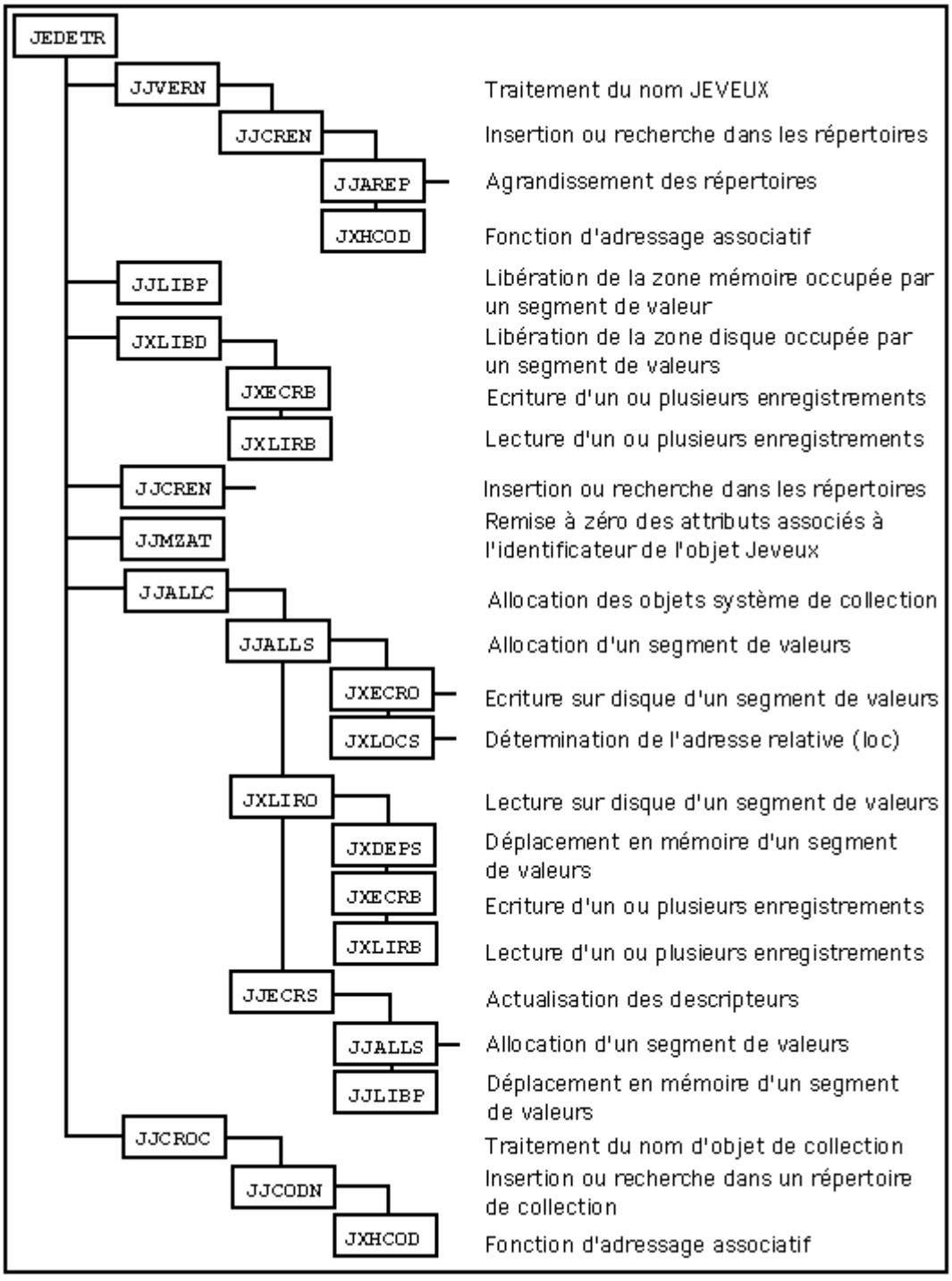
Arbre d'appel de la routine JEMARQ



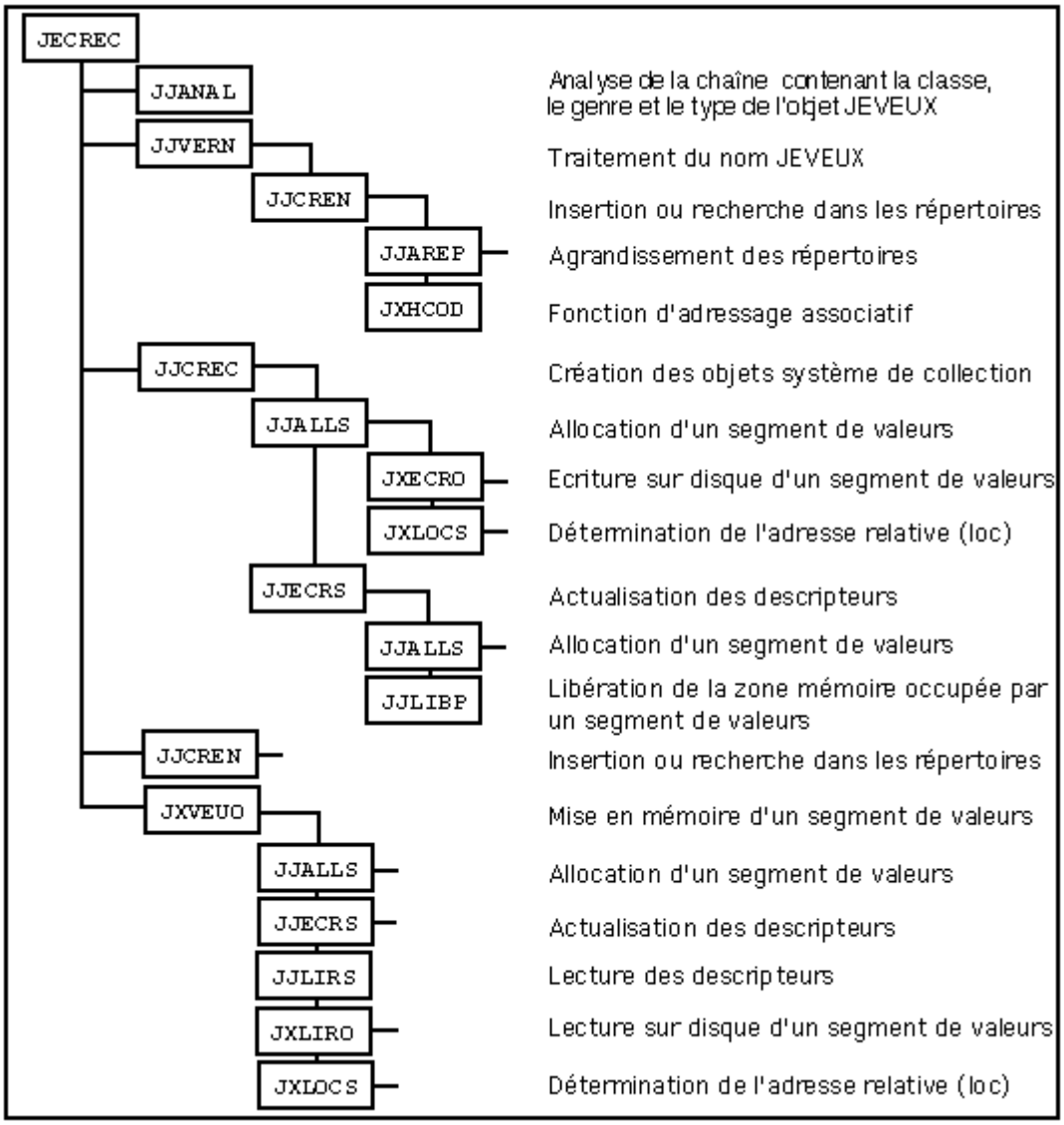
Arbre d'appel de la routine JEDEMA



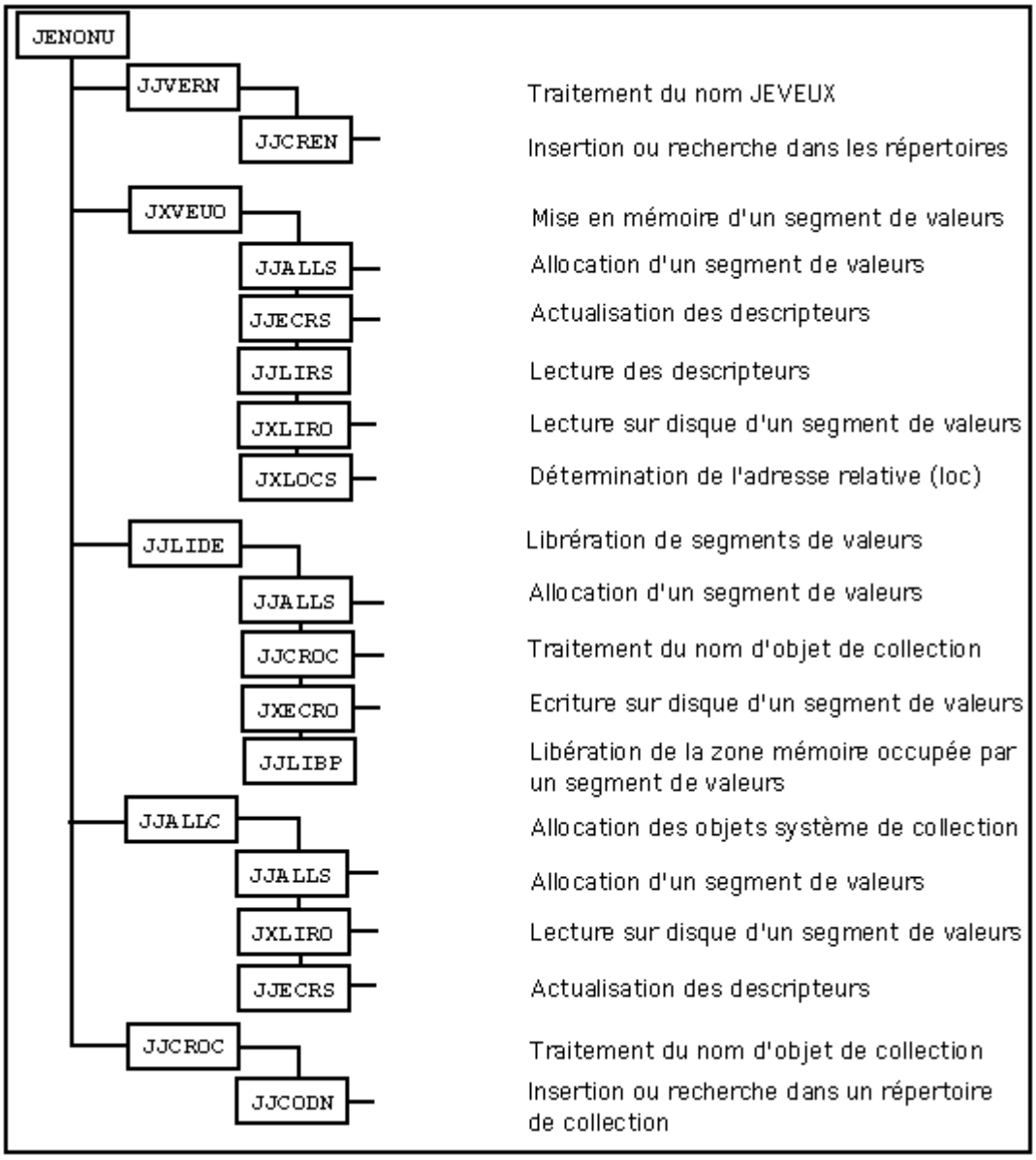
Arbre d'appel de la routine JELIBE



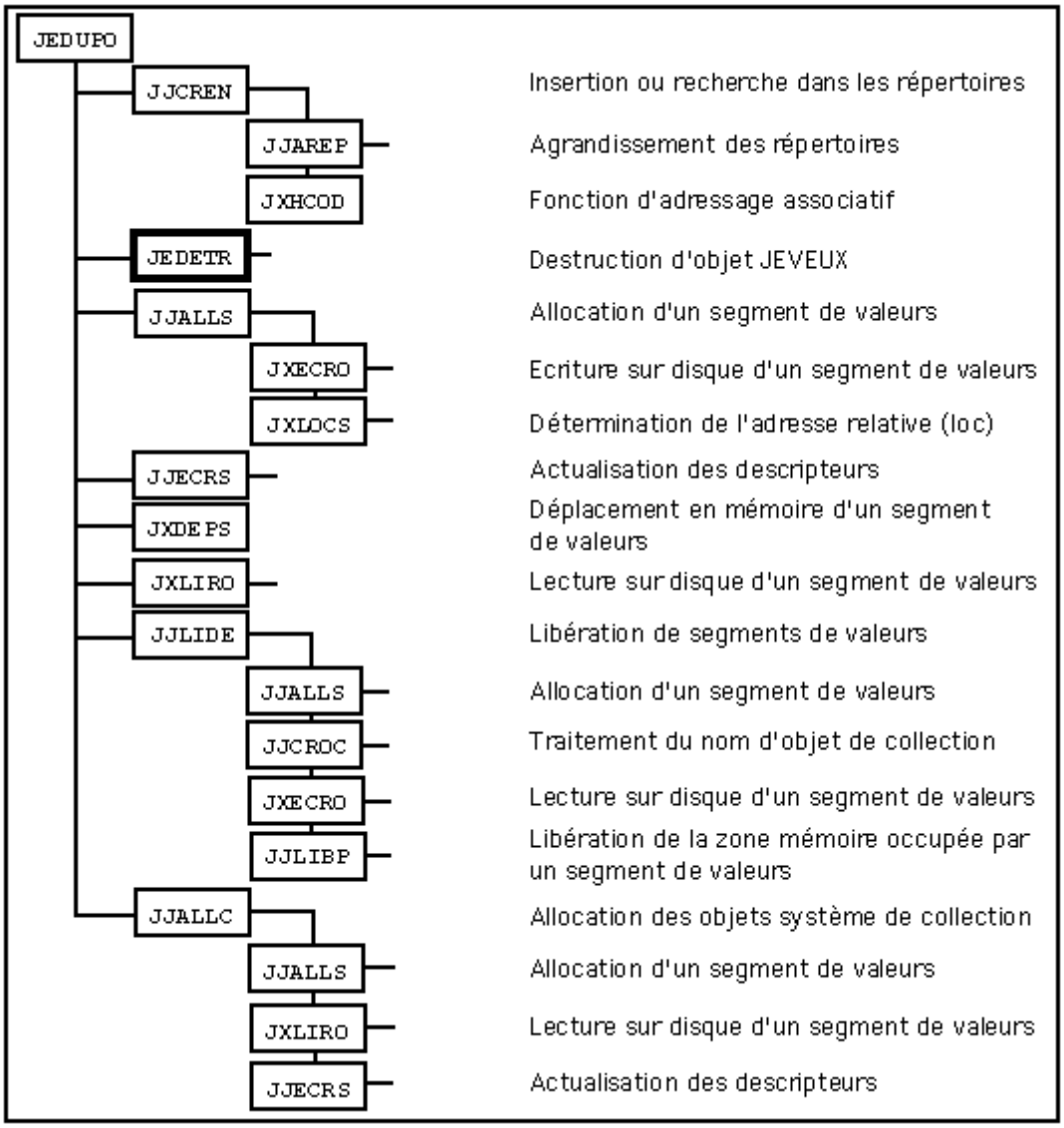
Arbre d'appel de la routine JEDETR



Arbre d'appel de la routine JECREC



Arbre d'appel de la routine JENONU



Arbre d'appel de la routine JEDUPO

15 ANNEXE 3 : Liste des sous-programmes et de leurs fonctions principales

JECREC	Création d'une collection
JECREO	Création d'un objet simple
JECROC	Création d'un objet de collection
JEDEBU	Initialisation des paramètres du logiciel
JEDEMA	Décrémente la marque et libère les objets
JEDETC	Détruit un ensemble d'objets
JEDETR	Détruit un objet
JEDETV	Détruit les objets de la base Volatile
JEDISP	Détermine les plus grands espaces disponibles en mémoire
JEDUPC	Duplique un ensemble d'objets
JEDUPO	Duplique un objet
JEECRA	Affecte les attributs d'un objet
JEEXIN	Teste l'existence d'un descripteur d'objet
JEFINI	Termine l'exécution du logiciel
JEIMPA	Imprime les attributs d'un objet
JEIMPD	Imprime la liste des objets présents sur une base
JEIMPM	Imprime le contenu de la segmentation mémoire
JEIMPO	Imprime le contenu du segment de valeur d'un objet
JEIMPR	Imprime le contenu d'un répertoire
JEINIF	Initialise les paramètres associés à une base
JELIBE	Libère un objet
JELIBF	Libère l'ensemble des objets associés à une base
JELIBZ	Libère l'ensemble des objets associés à la marque -1
JELIRA	Lecture de la valeur d'un attribut d'un objet
JELSTC	Retourne la liste des objets contenant une chaîne de caractères dans leur identificateur
JEMARQ	Incrémente la marque courante
JENONU	Détermine le numéro d'objet de collection en fonction du nom
JENUNO	Détermine le nom d'objet de collection en fonction du numéro
JEPRAT	Imprime le contenu des objets système
JERAZO	Remet à 0 le contenu d'un objet
JETASS	Déplace les enregistrements au sein du fichier d'accès direct afin de récupérer l'espace libre
JEVEMA	Délivre la valeur de la marque courante
JEVEUO	Renvoie la position dans le tableau Z. du segment de valeurs associé à un objet
JEVEUS	Renvoie la position dans le tableau Z. du segment de valeurs associé à un objet et positionne la marque à -3
JEVEUT	Renvoie la position dans le tableau Z. du segment de valeurs associé à un objet et positionne la marque à -1
JEXATR	Fonction de synchronisation permettant d'accéder au vecteur des longueurs cumulées d'une collection contiguë de longueur variable
JEXNOM	Fonction de synchronisation permettant d'accéder par nom à un objet de collection
JEXNUM	Fonction de synchronisation permettant d'accéder par numéro à un objet de collection
JJALLC	Allocation des objets système de collection
JJALLS	Allocation d'un segment de valeurs
JJALTY	Tri sur le type d'objet (avant mise en mémoire d'un segment de valeurs)
JJANAL	Analyse de la chaîne contenant la classe, le genre et le type de l'objet
JJAREP	Agrandissement des répertoires
JJCODN	Insertion ou recherche dans un répertoire de collection
JJCREC	Création des objets système de collection
JJCREN	Insertion ou recherche dans les répertoires des bases

JJCROC	Traitement du nom d'objet de collection
JJECRS	Actualisation des descripteurs
JJIMPO	Impression du contenu d'un segment de valeurs
JJLIDE	Libération des segments de valeurs
JJLIRS	Lecture des descripteurs
JJMZAT	Remise à zéro des attributs associés à l'identificateur d'un segment de valeurs
JJPREM	Lors de l'initialisation d'un répertoire, renvoie le plus proche nombre premier (figurant dans un data)
JJVERN	Traitement du nom Jeveux
JXALLM	Allocation dynamique de la zone mémoire gérée par le logiciel
JXCOPY	Recopie du fichier d'accès direct associé à une base avec élimination des enregistrement marqués inutilisés
JXDEPS	Déplacement en mémoire d'un segment de valeurs
JXECRB	Ecriture d'un ou plusieurs enregistrements
JXECRO	Ecriture sur disque d'un segment de valeurs
JXFERM	Fermeture d'un fichier d'accès direct
JXHCOD	Fonction d'adressage associatif
JXLIBD	Libération de la zone disque occupée par un segment de valeurs
JXLIBM	Libération finale de la zone mémoire allouée dynamiquement
JXLIR1	Lecture du premier enregistrement associé à une base afin de récupérer ses caractéristiques
JXLIRB	Lecture d'un ou plusieurs enregistrements
JXLIRO	Lecture sur disque d'un segment de valeurs
JXLOCS	Détermination de l'adresse relative d'un segment de valeurs
JXOUVR	Ouverture des unités logiques associées aux bases
JXVERI	Examine l'intégrité de la segmentation mémoire
JXVEUO	Mise en mémoire d'un segment de valeurs

16 ANNEXE 4 : GLOSSAIRE

Base	Ensemble de fichiers d'accès direct : la base GLOBALE est constituée des fichiers glob.1, glob.2, glob.3, ...
Classe	Nommée par la première lettre de la base, la classe permet d'associer un objet JEVEUX à un fichier.
Identificateur d'objet JEVEUX	Pour un objet simple ou une collection c'est le numéro d'ordre d'insertion dans le répertoire de la base, pour un objet de collection c'est l'identificateur de collection et le numéro d'ordre d'insertion dans la collection.
Descripteur d'un segment de valeurs	Un des 8 entiers encadrant un segment de valeurs en mémoire ou un des 3 entiers précédant un segment de valeurs sur un enregistrement (disque ou tampon)
Fichier d'accès direct	Fichier dont les enregistrements sont accessibles directement par nom ou numéro
Objet JEVEUX	Désigne à la fois les objets simples, les collections et les objets de collection
Objet simple	Objet dont les attributs sont directement accessibles parmi les objets système associés aux différentes classes
Objets de collection	Objets dont les attributs sont mis en commun et gérés dans des objets simples créés avec la collection
Segment de valeurs	Ensemble des valeurs associées à un objet JEVEUX et positionné de façon contiguë en mémoire ou sur disque
Objet système	Objets simples gérés par le logiciel et destiné à recueillir les valeurs des différents attributs.
Debug Jeveux	Option d'usage de Jeveux permettant de provoquer le déchargement immédiat des segments de valeurs lors des

libérations et l'affectation à la valeur UNDEF ou NaN du segment libéré.