Organisme(s) : EDF/IMA/MMN, GIST

Manuel de Descriptif informatique
Fascicule D1.02 : Atelier de Génie Logiciel d' *Aster*
Document D1.02.03

Spécification de l'AGLA

Résumé :

Ce document décrit les outils de l'AGLA d'un point de vue informatique. Il donne les spécifications fonctionnelles qui ont été utilisées pour la réalisation de ces outils.

Table des matières

1 Généralités

Les règles d'utilisation des outils décrits ici pour gérer la configuration logicielle du *Code_Aster* visent à ne pas trop contraindre les développeurs et à garantir que les évolutions apportées ne viennent pas altérer le bon fonctionnement des développements déjà réalisés.

Ces objectifs ne seront atteints que :

- si les développeurs ne cherchent pas à tricher avec le système (l'expérience nous a montré que les règles déjà pratiquées conduisaient à un confort tel que nul n'a tenté de le faire),
- parce que le code est géré de manière centralisée.

Ces règles s'appuient principalement sur trois notions :

- extraction du source, le système est réduit à sa plus simple expression puisqu'on se contente d'avertir la communauté que l'on souhaite modifier une partie du code donné,
- restitution, on vérifie qu'il n'y a pas de conflits entre le code rendu par un développeur, le code développé par ses collègues et le code déjà rendu dans la version de référence,
- passage systématique d'un ensemble de tests de référence avant prise en compte des modifications.

Les outils décrits dans ce document seront des scripts en C-Shell lorsque cela est possible et des programmes C pour le reste.

Pour l'instant, tous ces outils doivent être disponibles sur le `CRAY-C 98`.

Tous les outils doivent pouvoir être utilisés en batch (`qsub`) et en interactif.

Note de présentation :

- `nom_en_police_courier_minuscule` :
  - représente un fichier CRAY de l'AGLA (au sens général c'est à dire fichier ou répertoire ou commande),
  - représente également une commande ou un concept lié à CRAY et UNICOS,
- `nom_en_police_courier_minuscule_italique` :
  - représente une variable d'un outil de l'AGLA.

2 Définition du code source du *Code Aster*

Le code source du *Code Aster* est constitué :

- de routines FORTRAN stockées par BIBLIOTHÈQUEs. Nous parlerons alors de code FORTRAN.
- de fonctions en langage C.
- de modules en assembleur Cray : Cray Assembly Langage ou CAL. L'assembleur est également regroupé en bibliothèques. Comme le C et le FORTRAN le CAL est regroupé en bibliothèque.
- d'éléments de CATALOGUES constituant en quelque sorte un complément à la programmation de certaines parties du code (Langage de commande, Éléments finis). Nous parlerons alors de code CATALOGUE.

  Exemple :

```
      % MODIF DMEDX TYPELEM    DATE 12/12/91 AUTEUR
fgazzpe
      ELEMENT D_DEPL_R_DX   % DDL LIE EN MECANIQUE (CMP
DX)
      MAILLE   SEG3     001
      CARTE       4
      MTEMPSR  = INST_R    E  1  IDEN  1 INST
      MDDLMUR  = DDLM_R    E  1  IDEN  1 A1
      MDDLIMR  = DDLI_R    E  1  IDEN  1 C
      MDDLIMF  = DDLI_F    E  1  IDEN  1 C
      ...etc...
```

  Ces éléments de CATALOGUES sont stockés par CATALOGUEs.

  Définition :

  On appelle unité de source ou plus simplement unité une routine FORTRAN ou un module CAL o u une fonction C ou un élément de CATALOGUE. Ces *deux notions sont appelés unités car elles constituent effectivement des unités au sens de la gestion des sources.*

Une version d'*Aster* est constituée de code source mais aussi des cas-tests associés. Sans être du code source, un cas-test est indispensable. Les cas-tests sont également des unités et sont soumis aux mêmes contraintes de gestion que les sources.

Les sources et les cas-tests du *Code_Aster* sont stockés dans des fichiers correspondant à des versions de référence. La nécessité de pouvoir apporter des corrections sur la version livrée aux utilisateurs et de pouvoir faire évoluer la version de développement conduit effectivement à devoir considérer deux versions de référence. Actuellement, il s'agit des versions NEW2 et NEW3.

3 Instrumentation du code source

Les unités de source et de cas-test doivent posséder une ligne commentaire normalisée qui doit apparaître une et une seule fois dans chaque unité de source.

Cette ligne est appelée ligne INFO.

Il existe deux types de lignes INFO :

- la ligne AJOUT qui est destinée à informer les modules de gestion que l'unité de source est nouvelle.
- la ligne MODIF qui est destinée à informer les modules de gestion que l'unité de source vient modifier une unité de source déjà existante.

Les lignes de ce dernier type ne doivent être modifiées en **aucun cas** par le développeur.

Leur forme diffère pour du code FORTRAN, CAL, C, CATALOGUE ou CASTEST.

3.1 Code de type FORTRAN

Les lignes INFO sont de la forme :

```
SUBROUTINE nomsub
------ informations de la SUBROUTINE nomsub non prises en compte par les modules
de gestion ------
C MODIF nombib_1 DATE jj/mm/aa AUTEUR user_CRAY nom_auteur
------ informations de la SUBROUTINE nomsub non prises en compte par les modules
de gestion ------
END
INTEGER FUNCTION nomfun
------ informations de la FUNCTION nomfun non prises en compte par les modules
de gestion ------
C AJOUT nombib_2
------ informations de la FUNCTION nomfun non prises en compte par les modules
de gestion ------
END
```

Syntaxe d'une ligne INFO (les différents champs sont séparés par au moins 1 blanc) :

- "C" en première colonne,
- "MODIF" ou "AJOUT",
- nombib_i : nom de la bibliothèque,
  si "MODIF" :
  - "DATE",

- `jj/mm/aa` : date gérée au moment de la mise à jour de la version de référence,
- "AUTEUR",
- `user_CRAY` : user sur CRAY de la dernière personne qui a modifié la routine,
- `nom_auteur` : lié à l'identificateur précédent. Ces deux identificateurs sont également gérés au moment de la mise à jour de la version de référence.

Ce sont ces champs qui ne doivent pas être modifiés par le développeur.

3.2 Code de type CAL

Les lignes INFO sont de la forme :

Lorsque l'on veut rajouter une routine il faut mettre la ligne INFO suivante dans le module (cette ligne délimite le début du module) :

```
* AJOUT nom_module nom_de_bibliotheque
```

- * en première colonne,
- AJOUT (en majuscules obligatoirement !),
- le nom du module,
- le nom d'une bibliothèque CAL *Aster* où le module doit être ajouté (cette bibliothèque doit déjà exister -cf organisation Source *Aster*-).

Remarque :

Un module que l'on veut ajouter ne doit pas déjà exister dans Aster (soit en FORTRAN soit en C soit en CAL).

Une fois le module ajouté au *Code_Aster*, la ligne INFO est gérée et mise à jour par l'AGLA. Elle possède alors la forme suivante :

```
* MODIF nom_module NOM_BIB DATE jj/mm/aa AUTEUR J2BHHMB
C.MASSERET
```

Elle contient alors :

- le nom du module,
- le nom de la bibliothèque CAL où se trouve le module,
- la date de sa dernière modification,
- le user Cray et le nom de l'auteur de la dernière modification.

3.3 Code de type C

Les lignes INFO sont de la forme :

```
/* AJOUT nom_fonction nom_de_bibliotheque */
```

- /* en première colonne,
- AJOUT (en majuscules obligatoirement !),
- le nom de la fonction,
- le nom d'une bibliothèque C *Aster* où la fonction doit être ajoutée (cette bibliothèque doit déjà exister -cf organisation Source *Aster*-).

Une fois la fonction ajoutée au *Code Aster*, la ligne INFO est gérée et mise à jour par l'AGLA. Elle possède alors la forme suivante :

```
/* MODIF nom_fonc NOM_BIB DATE jj/mm/aa AUTEUR J2BHHMB
C.MASSERET */
```

Elle contient alors :

- le nom de la fonction,

- le nom de la bibliothèque C où se trouve la fonction,
- la date de sa dernière modification,
- le user Cray et le nom de l'auteur de la dernière modification.

Note :

*Pour qu'une fonction C puisse être appelée en FORTRAN il faut que son nom soit en **majuscules** .*

3.4 Code de type CATALOGU

Les lignes INFO sont de la forme :

```
      ------------- informations non prises en compte par les modules
de gestion ------
      %& MODIF nom_catalogue_1 DATE jj/mm/aa AUTEUR user_CRAY
         nom_auteur
      ------------- informations de l'élément du catalogue
nom_catalogue_1
non prises en compte par les modules de gestion ------
      %& AJOUT nom_catalogue_2
      ------------- informations de l'élément du catalogue
nom_catalogue_2
non prises en compte par les modules de gestion ------
```

Syntaxe d'une ligne INFO (les différents champs sont séparés par au moins 1 blanc) :

- "%" en première colonne,
- "MODIF" ou "AJOUT",
- & en deuxième colonne,
- nom_catalogue : nom du catalogue,
  si "MODIF" :
  - "DATE",
  - jj/mm/aa : date gérée au moment de la mise à jour de la version de référence,
  - "AUTEUR",
  - user_CRAY : user sur CRAY de la dernière personne qui a modifié la routine,
  - nom_auteur : lié à l'identificateur précédent. Ces deux identificateurs sont également gérés au moment de la mise à jour de la version de référence.

On constate, sur cet exemple que la ligne INFO joue le rôle de délimiteur **contrairement** au cas du FORTRAN.

3.5 Fichier de type CASTEST

Pour les cas-test deux lignes INFO sont obligatoires AJOUT ou MODIF et TITRE.

Les lignes INFO sont de la forme :

- pour la modification d'un cas-test :

```
        ------------ informations non prises en compte par les
modules de gestion ------
        % MODIF  DATE jj/mm/aa  AUTEUR user_CRAY nom_auteur
        % TITRE explication, en clair, du rôle de ce cas-test
        ------------ informations non prises en compte par les
modules de gestion ----
        DEBUT(CODE:(NOM:'nom_du_cas_test',
```

- pour l'ajout d'un cas-test :

```
        ------------ informations non prises en compte par les
modules de gestion ------
        % AJOUT
        % TITRE explication, en clair, du rôle de ce cas-test
        ------------ informations non prises en compte par les
modules de gestion ----
        DEBUT(CODE:(NOM:'nom_du_cas_test',
```

Il est à noter que :

- le nom du fichier doit correspondre à celui du cas-test,
- il n'y a qu'un seul cas-test par fichier,
- la ligne `INFO 'TITRE'` est à renseigner par le développeur.

3.6 Fichier de type UNIGEST

Un fichier de type UNIGEST ne contient pas expressément du code source, mais des consignes concernant les unités de source pour les modules de restitution.

Dans ce fichier on indique les unités de source ou de cas-test que l'on souhaite déplacer d'une bibliothèque à une autre ou supprimer. Ce type de fichier contient des lignes du type suivant :

- FORSUPPR nom_de_routine nom_de_bibli
  pour les routines que l'on souhaite supprimer, en indiquant leur nom et la bibliothèque d'appartenance,
- CATSUPPR nom_de_sous_catalogue nom_de_catalogue
  pour les éléments de catalogues que l'on souhaite supprimer, en indiquant leur nom et la catalogue d'appartenance,
- TESSUPPR nom_de_cas_test
  pour les cas-tests que l'on souhaite supprimer, en indiquant leur nom,
- FORDEPLA nom_de_routine bibli1 bibli2
  pour les routines que l'on souhaite déplacer d'une bibliothèque bibli1 vers une bibliothèque bibli2.

Il est à noter que pour ce type de fichier :

- il n'y a pas de notion de commentaire,
- pour qu'il n'y ait pas d'incohérence il ne peut pas y avoir plus d'un ordre UNIGEST concernant la même unité,
- il n'est pour l'instant pas prévu d'ordres équivalent pour le source C et CAL.

3.7 Types de fichiers

On appelera fichier de type :

- FORTRAN un fichier contenant du source de type FORTRAN,
- CAL un fichier contenant du source de type assembleur Cray,
- C un fichier contenant du source de type langage C,
- CATALOGU un fichier contenant du source de type CATALOGU,
- CASTEST un fichier contenant du source de type CASTEST,
- UNIGEST un fichier contenant du source de type UNIGEST.

Remarques :

On ne peut pas mélanger dans un même fichier les différents types (FORTRAN, CAL, C, CATALOGU, CASTEST, UNIGEST),

Pour les ajouts, la date et le nom de l'auteur ne sont évidemment pas nécessaires dans la ligne INFO.

4 Fichier d'identification `ident` *Aster*

L'identification du développeur s'effectue par son `user` Cray. Le fichier `ident` *Aster*, résidant sur `Cray`, permet de faire la correspondance avec son nom. La mise à jour de `ident` *Aster* est assurée par l'administrateur *du Code_Aster*. Un `user` Cray ne peut apparaître qu'une seule fois dans ce fichier. Pour pouvoir utiliser les outils de l'AGLA il est obligatoire que le `user` appelant soit défini dans ce fichier.

Syntaxe du fichier `ident` *Aster* (les champs sont séparés par au moins un blanc) :

- user Cray,
- nom,
- adresse eMail,
- département,
- groupe (pour les statistiques concernant l'EDA),
- rôle dans *Aster* parmi (un ou plusieurs séparé par un /) :
  - EDA,
  - CUA,
  - ACUA,
  - CEDA,
  - ASA,
  - CPA,
- type de machine pour l'interface `asterix` (si CUA) parmi :
  - SUN_OS,
  - SUN_SOLARIS,
  - HP,

Exemple :

```
%user Cray nom          eMail          Dpt
gjbhhts         Pascal.Mialon@der.ed  MMN  I75  EDA/CEDA
P.MIALON        f.fr                  MMN  I74  SUN_OS/HP
f6bhhbu D.BUI   Danièle.Bui.der.edf.           CUA/ACUA SUN
                fr
```

5 Fichiers du code noté (`quirou`, `quicat` et `quites`)

Les outils de notation s'appuient sur :

- le fichier `quirou` résidant sur Cray sur lequel sont inscrits :
  - les noms des routines `FORTRAN`, modules `CAL`, et fonctions `C` notées (nous préciserons cette notion par la suite) par les développeurs,
  - le nom de la version de référence concernée ou le mot-clé **`RAYE`** (si l'unité a été dénotée par l'administrateur,
  - hh:mm:ss : l'heure de notation,
  - jj/mm/aa : la date de notation,
  - user_CRAY : l'identificateur du développeur sur Cray,

- le nom du développeur,
- jj/mm/aa : la date de dé-notation par l'administrateur (lorsqu'il y a le mot-clé **RAYE** dans le champ version).

Ces fichiers sont triés par date de notation croissante.

Exemple :

```
      -------------  ROUTINES SORTIES ACTUELLEMENT:
      TE0010   NEW1   17:32:29   07/11/90   g8bhhxd
X.DESROCHES
      CHARGR   NEW2   17:00:55   22/02/91   vabhht2   C.MENGONI
      CORDDL   NEW2   17:01:21   22/02/91   vabhht2   C.MENGONI
      TE0040   NEW2   17:05:30   04/02/91   b8bhhh1
D.SELIGMANN
      LEC151   RAYE   14:16:55   20/03/91   c4gffhj   A.ZSCDR
25/03/91
      LEC151   NEW2   13:06:43   24/04/91   d6bhhhm
A.M.DONORE
      LEC151   NEW2   15:26:17   24/04/91   c4gffhj   A.ZSCDR
      LEC164   NEW2   15:10:41   24/04/91   d6bhhhm
A.M.DONORE
      LEC747   NEW2   17:26:13   24/04/91   d6bhhhm
A.M.DONORE
```

- le fichier `quicat` résidant sur Cray sur lequel sont inscrits les mêmes informations que décrites ci-dessus mais concernant les éléments de catalogue du *Code Aster*.

  Exemple :

```
      ------------  CATALOGUES SORTIS ACTUELLEMENT:
      MELIE2DL   NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
      MELIE3DL   NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
      MELIE6DL   NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
      THLIE2D    NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
      THLIE3D    RAYE   08:59:08   13/03/91   c4gffhj   A.ZSCDR
13/03/91
      THLIE3D    NEW2   09:47:35   13/03/91   vabhhts
J.PELLET
      MECYSE3    NEW2   17:53:39   29/05/91   gjbhhts
P.MIALON
```

- le fichier `quites` résidant sur Cray sur lequel sont inscrits les mêmes informations que décrites ci-dessus mais concernant les cas-tests du *Code_Aster*.

  Exemple :

```
      ------------  CAS-TESTS SORTIS ACTUELLEMENT:
      ADLV100A   NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
      HPLV100A   NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
      HSNV100A   NEW2   08:51:08   13/03/91   vabhhts
J.PELLET
```

```
      SDLL15A     NEW2    08:51:08    13/03/91    vabhhts
J.PELLET
      TPLA01A     RAYE    08:59:08    13/03/91    c4gffhj    A.ZSCDR
13/03/91
```

Par définition, nous dirons qu'une unité de source ou de cas-test est notée si son nom est inscrit dans un des fichiers de code noté (`quirou` ou `quicat` ou `quites`) et qu'elle n'est pas "rayée".

Remarques :

Les notations de plus de trois mois seront supprimées des fichiers de notation lors de l'utilisation de `majnew`,

Les notations "rayées" de plus de trois mois seront également supprimées lors de l'utilisation de `majnew`.

6 Fichiers listes des cas-tests

Ces fichiers se situent dans le répertoire `/aster/astest/vers` où `vers` désigne la version de développement associée (`NEW3` par exemple).

6.1 Liste de gestion des cas-tests (`Liste`)

Cette liste décrit tous les cas-tests de validation d'*Aster*. Elle permet de classer chaque cas-test parmi une des catégories suivante :

liste restreinte : le cas-test sera soumis pour valider une restitution (attribut '.'),
liste complète : le cas-test n'est soumis qu'une fois par semaine par l'administrateur *Aster* (attribut 'S'),
liste performance : le cas-test est géré individuellement (attribut 'P').

Cette liste est réactualisée automatiquement par la mise à jour d'une nouvelle version d'*Aster* : `majnew`.

Elle comprend les champs suivants (le séparateur étant le caractère '|') :

- attribut d'Ajout 'A' ou de Modification 'M' lors de la dernière mise à jour et '.' dans le cas contraire,
- type de liste :
  '.' liste restreinte (type de liste par défaut lorsque l'on ajoute un cas-test),
  'S' liste complète,
  'P' liste Performance,
- nom du cas-test,
- temps total d'exécution en secondes,
- mémoire demandée en Mw,
- un champ commentaire pour justifier le classement du cas-test dans un type de liste (sur 55 caractères),
- le titre du cas-test (débutant obligatoirement par le caractère '%').

En fin de liste et pour chaque type il y a :

- le nombre de cas-tests dans la liste,
- le temps cumulé d'exécution de tous les cas-tests (en s),
- le coût cumulé d'exécution de tous les cas-tests en job de jour, de nuit et de week-end.

Après une mise à jour d'*Aster*, l'administrateur peut modifier la liste en modifiant la répartition des cas-tests dans les différentes listes ainsi que la modification de la zone commentaire. Cette opération s'effectue à l'aide d'un éditeur de texte.

L'administrateur a ensuite la possibilité de mettre à jour les coûts des différentes listes. A partir de cette liste il peut également mettre à jour les listes restreintes et totales décrites ci-dessous.

Exemple :

```
| . | P | sdnl102a | 00068 | 008 |                    | % TITRE
Portee de cab …
| . | . | ssll101b | 00009 | 008 |                    | % TITRE
PROBLEME DE HP…
| . | S | ssll11c  | 00004 | 008 |                    | % TITRE
treillis de b…
| A | . | ssll11d  | 00004 | 016 |                    | % TITRE
Treillis de b…
| . | . | ssll12a  | 00011 | 008 |                    | % TITRE
treillis de b…
| . | S | ssls101d | 00020 | 008 |    cf. ssls100     | % TITRE
Plaque circul…
| M | S | sslv101g | 00031 | 016 |    Trop long       | % TITRE
SSLV101/G    …
| M | . | sslv101h | 00028 | 016 | Affe_char_cine     | % TITRE
SSLV101/H CTJ…
| . | S | ttll01e  | 00015 | 008 | cf. tpll01         | % Choc
thermique sur …
| . | . | ttnl02a  | 00115 | 008 |    Trop cher cf.   | % TITRE
transitoire t…
|   |   |          |       |     |    hsnv102a        |
| . | . | zzzz100a | 00014 | 008 | Teste des plantages | % TITRE
fonctions, na…
                              possible
#################################################################
#################                  NB cas-tests   temps (s)
jour (F)   nuit (F)   week-end (F)
# Liste Restreinte    0007         000265     000589
000228      000128
# Liste Complete      0013         000512     001338
000455      000669
# Liste Performance   0001         000068     000151
000060      000034
#################################################################
################
```

6.2 Liste de tous les cas-tests (`liste_ct.tout`)

Cette liste contient le nom et la description de tous les cas-tests officiels d'une version. Elle permet de passer tous les cas-tests à chaque nouvelle version. Cette liste est mise à jour à partir de la liste de gestion des cas-tests décrite ci-dessus. Les cas-tests de performances sont insérés dans cette liste mais sont mis en commentaire.

Toute ligne débutant par '`%`' est un commentaire.

Le titre du cas-test débute obligatoirement par '`%`'.

Classée par ordre alphabétique .

Exemple :

```
   %  Liste de tous les cas-tests officiels de la version NEW3
      adlv100a % Piston couplé à une colonne de fluide
      ahlv100b % Guide d'onde à sortie anéchoique (ondes planes)
      hplv100a % Parallélépipède. Module d'YOUNG fonction de la
température.
      hsnv10a  % Thermo-plasticité en traction simple
      mtlp100a % Trempe d'un barreau infini à section carrée
   %  sdnl102a % TITRE Portee de cables triphasee avec
charpentes et descentes
```

6.3 Liste restreinte de cas-tests (`liste_ct.rest`)

Cette liste, de même forme que la précédente, ne contient que les cas-tests les plus pertinents au sens de la non regression du code. Elle permet de passer les cas-tests définis lors de chaque restitution de code par un développeur. Cette liste est mise à jour par l'administrateur *Aster* à partir de la liste de gestion des cas-tests.

7 Fichier d'autorisation des restitutions (lien_dvp)

Ces fichiers se situent dans le répertoire /aster/astest/vers où vers désigne la version associée.

L'AGLA permet de prendre en compte la restitution groupée pour les développements dépendants. Par défaut chaque développeur est seul habilité à rendre ses développements. Lorsque des développements sont liés, un des développeurs doit centraliser les restitutions. Les autres développeurs devront alors lui déléguer leur droit à rendre leurs développements. Tant que le bénéficiaire de cette délégation (ou l'administrateur) n'aura pas redonné leurs droits aux autres développeurs, ils ne pourront pas faire de restitution.

A chaque nouvelle version les droits sont réinitialisés (chaque développeur est alors le seul habilité à rendre ses développements).

Le fichier lien_dvp ne contient que les délégations de restitution (pas les droits par défaut). Il est composé de quatre champs par enregistrement :

- user Cray du développeur délégant son droit de restitution,
- user Cray du bénéficiaire de cette délégation,
- version concernée par la délégation (NEW1 ou NEW2 ou NEW3),
- jj/mm/aa date de la délégation,
- (nom_developpeur_appelant -> nom_developpeur_beneficiaire) déterminés à partir du fichier ident.

Exemple :

```
g8bhhhh  j2bhhmb  NEW3  05/08/92 (R.Michel -> C.Masseret)
```

Pour des raisons de cohérence, avant la construction de la nouvelle version,

- les user Cray doivent exister dans le fichier ident,
- un développeur ayant bénéficié d'au moins une délégation ne pourra pas déléguer son propre droit de restitution à un autre développeur,
- un développeur ne peut déléguer qu'une seule fois son droit de restitution.

8 Outils de notation du code source

Ces outils doivent être utilisés par toute personne souhaitant effectuer un développement dans une version de référence du *Code_Aster*.

Ils sont situés dans le répertoire /aster/astest/vers où vers désigne la version associée et sont également disponibles via l'environnement d'utilisation asterix [D1.02.01 §12].

D'une manière générale tous les outils décrits par la suite renvoient un code d'erreur par l'intermédiaire du code de retour d'un processus UNIX (consultable par $status en C-SHELL et $? en Bourne-SHELL).

Ce code est le suivant :

- 0 si tout est OK,
- 2 si un message d'alarme a été émis,
- 4 si une erreur a été rencontrée.

8.1 asno : notation d'unités

Noter dans les fichiers de code notée (quirou ou quicat ou quites) des unités de source.

8.1.1 Mode d'appel

asno *mon_fichier* [*version* [*message*]]

mon_fichier désigne un fichier étant soit du type FORTRAN ou CAL ou C ou CATALOGUE ou UNIGEST ou CASTEST.

version est un paramètre qui porte le nom de la version de référence concernée (NEW1, NEW2, NEW3). La valeur prise actuellement par défaut est NEW3.

message désigne un fichier en sortie où seront inscrits les messages d'alarmes et d'erreurs émis par asno. Le nom pris par défaut est : message_asno.

8.1.2 Tâches réalisées

Trois tâches principales :

A Recherche du type du fichier code source FORTRAN ou CAL ou C, code source CATALOGUE, UNIGEST, ou CASTEST sur les critères suivants :

- Si le fichier contient une ligne commençant par :
  C AJOUT ou C MODIF
  le C étant en première colonne, le nombre de blanc entre C et AJOUT ou entre C et MODIF n'étant pas significatif,
  alors le fichier sera considéré comme contenant du code FORTRAN
- Si le fichier contient une ligne commençant par :
  /* AJOUT ou /* MODIF
  le /* étant en première colonne, le nombre de blanc entre /* et AJOUT ou entre /* et MODIF n'étant pas significatif,
  alors le fichier sera considéré comme contenant du code C
- Si le fichier contient une ligne commençant par :
  * AJOUT ou * MODIF
  l'* étant en première colonne, le nombre de blanc entre * et AJOUT ou entre * et MODIF n'étant pas significatif,
  alors le fichier sera considéré comme contenant du code CAL
- Si le fichier contient une ligne commençant par :
  % AJOUT ou % MODIF

le % étant en première colonne, le nombre de blanc entre % et AJOUT ou entre % et MODIF n'étant pas significatif,

alors le fichier sera considéré comme contenant du code CATALOGUE

- Si le fichier contient une ligne commençant par :

  % TITRE

  le % étant en première colonne, le nombre de blanc entre % et TITRE n'étant pas significatif,

  et une ligne débutant par :
  DEBUT (CODE: (NOM: 'nom_cas_test'

  alors le fichier sera considéré comme contenant du code CASTEST

- Si le fichier contient une ligne commençant par :

  FORSUPPR, CATSUPPR, TESSUPPR, FORDEPLA, alors, le fichier sera considéré du type UNIGEST.

Si le fichier contient un seul type de source *Aster* :

la notation peut continuer

Sinon

un message d'erreur est inscrit et FIN.

B Recherche du nom des unités de sources et du nom des bibliothèques ou catalogues :

- Dans le cas d'un fichier de type FORTRAN :

  Le nom des routines est trouvé selon les normes de déclarations FORTRAN :

```
PROGRAM nom_unite
SUBROUTINE nom_unite
FUNCTION nom_unite
INTEGER FUNCTION nom_unite
...etc...
```

  Le nom des bibliothèques est trouvé par la ligne INFO.

- Dans le cas d'un fichier de type CATALOGUE ou CAL ou C :

  Les noms des unités et du catalogue (ou de la bibliothèque) sont trouvés grâce aux lignes INFO :

```
%&      MODIF nom_catalogue DATE jj/mm/aa AUTEUR
nom_auteur
```

  ou

```
%&      AJOUT nom_catalogue
```

  Le % étant en colonne 1 & en 2ème colonne et le nombre de blanc n'étant pas significatif. (respectivement avec /* pour le C et * pour le CAL).

- Dans le cas d'un fichier de type CASTEST :

Les noms des cas-tests sont trouvés grâce au nom du cas-test dans l'instruction DEBUT(CODE:(NOM:'nom_cas_test',

- Dans le cas d'un fichier du type UNIGEST :

  Les noms des éléments d'unité sont trouvés de manière évidente.

C Pour chacune des unités à noter :

- Si elle est déjà notée dans un fichier :

  Si la première notation de cette unité est le fait du développeur appelant asno :
  il n'y a rien a faire.

  Sinon :

  une alarme est émise.

  Le nom des personnes ayant noté cette unité ainsi que les heures et dates de notation sont inscrites dans le fichier *message*.

  Si le développeur appelant asno n'a pas déjà noté cette unité :
  l'unité est notée

  Sinon :

  l'unité de source est notée dans le fichier de code noté avec l'identificateur appelant, l'heure et la date.

8.2 asdeno : dénotation d'unités

Dénoter des unités de source à partir d'un fichier source.

8.2.1 Mode d'appel

asdeno *mon_fichier* [*version* [*message*]]

mon_fichier désigne un fichier contenant soit du code FORTRAN ou CAL ou C, soit du code CATALOGUE, soit de type UNIGEST ou CASTEST.

version est un paramètre qui porte le nom de la version de référence concernée (NEW1, NEW2, NEW3). La valeur prise actuellement par défaut est NEW3.

message désigne un fichier en sortie où seront inscrits les messages d'alarmes et d'erreurs émis par asdeno. Le nom pris par défaut est : message_asdeno.

8.2.2 Tâches réalisées

- Tâche A identique à asno.
- Tâche B identique à asno.
- Recherche, dans le fichier de code noté, des unités de source déjà notées par l'identificateur appelant, et suppression des lignes correspondantes.

8.3 xasdeno : dénotation d'unités à partir de leur nom

Dénoter des unités de source à partir d'un fichier contenant des noms de module et d'un type de module.

8.3.1 Mode d'appel

```
xasdeno mon_fichier type [version [ message ]]
```

mon_fichier désigne un fichier contenant un nom de module par ligne les noms de module doivent faire référence à des unités du même type.

version est un paramètre qui porte le nom de la version de référence concernée (NEW1, NEW2, NEW3). La valeur prise actuellement par défaut est NEW3.

type : pour désigner le type d'unité à dénoter parmi f (fortran), c, cal(assembleur), cata (catalogue), test (cas-test).

message désigne un fichier en sortie où seront inscrits les messages d'alarmes et d'erreurs émis par xasdeno. Le nom pris par défaut est : message_xasdeno.

8.3.2 Tâches réalisées

Suppression, dans le fichier de code noté, des unités de source déjà notées par l'identificateur appelant avec la version donnée des modules dont le nom est dans le fichier.

8.4 asdeno_adm : dénotation par l'administrateur

Possibilité donnée seulement à l'administrateur *Aster* d'invalider toutes les notations concernant des unités. Les unités 'dénotées' auront leur champ version remplacé par la mention 'RAYE' et la date de cette dénotation sera rajouté dans les fichiers de notation (pour que les développeurs ayant noté ces unités soient prévenus).

Les unités rayées ne seront 'visibles' qu'avec asqui et asquit.

8.4.1 Mode d'appel

```
asdeno_adm mon_fichier [version [ message ]]
```

mon_fichier désigne un fichier contenant soit du code FORTRAN ou CAL ou C, soit du code CATALOGUE, soit du type UNIGEST ou CASTEST.

version est un paramètre qui porte le nom de la version de référence concernée (NEW1, NEW2, NEW3). La valeur prise actuellement par défaut est NEW3.

message désigne un fichier en sortie où seront inscrits les messages d'alarmes et d'erreurs émis par asdeno_adm. Le nom pris par défaut est : message_asdeno_adm.

8.4.2 Tâches réalisées

- Tâche A identique à asno
- Tâche B identique à asno
- Recherche, dans le fichier de code noté, des unités de source déjà notées, remplacement du champ version par 'RAYE' et ajout de la date de dénotation.

8.5 asqui : liste de notation d'unités

Connaître les développeurs qui ont notés des unités de source.

8.5.1 Mode d'appel

```
asqui mon_fichier [ message ]
```

mon_fichier désigne un fichier contenant soit du code FORTRAN ou CAL ou C, soit du code CATALOGUE, soit du type UNIGEST ou CASTEST.

message désigne un fichier en sortie où seront inscrits les messages d'alarmes et d'erreurs émis par asqui. Le nom pris par défaut est : message_asqui.

8.5.2 Tâches réalisées

- Tâche A identique à asno.
- Tâche B identique à asno.
- Recherche dans le fichier de code noté si les unités de source sont déjà notés (toutes les versions sont concernées).
  Si tel est le cas :
  - Le nom des unités de source et des personnes ainsi que les heures et dates de notation sont inscrits dans *message* et FIN.
    (tâche identique à la tâche C de asno).

8.6 asquit : liste de toutes les notations

Connaître toutes les unités de source notés.

8.6.1 Mode d'appel

```
asquit [ message ]
```

message : définition identique à asno. Le nom pris par défaut est : message_asquit.

8.6.2 Tâche réalisée

Recopie des fichiers de code noté (quirou et quicat et quites) dans le fichier *message*.

9 Outils de vérification et de restitution

Ils sont situés dans le répertoire `/aster/astest/vers` où `vers` désigne la version associée et sont également disponible via l'environnement d'utilisation asterix [§12 D1.02.01].

9.1 `aslien` : délégation du droit de restitution

Lorsque des développements sont dépendants il faut pouvoir les rendre simultanément afin de tester le code. Cela oblige un des développeurs à centraliser les développements et à effectuer la restitution pour les autres développeurs.

Pour éviter les problèmes, les développeurs doivent déléguer leurs droits de restitution à celui qui centralise les développements.

Cette délégation est exclusive pour une seule version et valable jusqu'à la mise à jour de la nouvelle version. La délégation peut cependant être rendue par le bénéficiaire, ou par l'administrateur.

Un développeur ne peut déléguer qu'une seule fois ce droit (pour une version donnée). Le bénéficiaire d'une délégation ne peut pas déléguer lui même son droit de restituer.

9.1.1 Mode d'appel

```
aslien user_beneficiaire [ version [ message ] ]
```

user_beneficiaire : user Cray du développeur bénéficiant de la délégation de restitution.

version : est un paramètre qui porte le nom de la version de référence concernée (`NEW1`, `NEW2`, `NEW3`). Par défaut actuellement : `NEW3`.

message : définition identique à `asno`. Par défaut : `message_aslien`.

9.1.2 Tâches réalisées

Vérifier que le *user_beneficiaire* existe dans ident et est différent de l'appelant (*iret* = 4).

Vérifier que l'appelant n'a pas déjà délégué son droit de restitution (*iret* = 4).

Vérifier que le *user_beneficiaire* n'a pas déjà délégué son droit de restitution (*iret* = 4).

Vérifier que l'appelant n'est pas déjà bénéficiaire de droits de restitution (*iret* = 4).

Si *iret* < 4 compléter le fichier lien_dvp avec :

- user de l'appelant,
- user_beneficiaire,
- version,
- jj/mm/aa date d'utilisation de aslien,
- (nom_developpeur_appelant -> nom_developpeur_beneficiaire) déterminés à partir du fichier ident.

9.2 aslibe : libération du droit de restitution

Libération de la délégation de restitution par le bénéficiaire ou l'administrateur.

9.2.1 Mode d'appel

```
aslibe user [ version [ message ] ]
```

user : user Cray du développeur qui a délégué son droit de restitution.

version : est un paramètre qui porte le nom de la version de référence concernée (NEW1, NEW2, NEW3). Par défaut actuellement : NEW3.

message : définition identique à asno. Par défaut : message_aslibe.

9.2.2 Tâches réalisées

Vérifier que l'appelant et *user* sont différents et définis dans ident (*iret* = 4).

Si l'appelant est l'administrateur alors suppression de la ligne de lien_dvp contenant en premier champ *user* pour la version *version* (si elle n'existe pas *iret* = 4).

Si l'appelant est un développeur alors suppression de la ligne de lien_dvp contenant en premier champ *user* et en deuxième champ le user de l'appelant pour la version *version* (si elle n'existe pas *iret* = 4).

9.3 asquil : liste des délégations de restitution

Liste dans le fichier de message d'asquil de toutes les délégations de restitution accordées.

9.3.1 Mode d'appel

```
asquil [ message ]
```

message : définition identique à asno. Par défaut : message_asquil.

9.3.2 Tâches réalisées

Recopier le fichier lien_dvp dans *message*.

9.4 asverif : vérification de source

Vérifier qu'un ensemble de source peut être introduit dans une version de référence du *Code Aster*.

Cet outil effectue une partie des tâches réalisées par le module de restitution de source asrest qui, devra, être utilisé obligatoirement par toute personne souhaitant effectivement restituer un développement.

asverif permet de vérifier une restitution groupée, dans le cas où les restitutions de plusieurs développeurs sont dépendantes.

asverif travaille dans un répertoire temporaire.

9.4.1 Mode d'appel

```
asverif repertoire [version [ message ]]
```

repertoire est le nom d'un répertoire qui contient un répertoire par développeur désirant restituer du source. Les noms de ces répertoires doivent être le user-Cray du développeur pour déterminer l'auteur du source restitué. Dans chacun de ces répertoires on peut trouver, suivant les cas, les fichiers :

repertoire/user(s)/

histor	dans lequel on trouve un descriptif des évolutions apportées (obligatoire).
fortran	dans lequel se trouve du source FORTRAN.
cal	dans lequel se trouve du source assembleur.
catalogu	dans lequel se trouve des éléments de catalogues.
unigest	dans lequel on indique les unités de source que l'on souhaite déplacer d'une bibliothèque à une autre ou supprimer.
test/	répertoire dans lequel se trouvent des fichiers de

commande de cas-tests `*.comm` et éventuellement, des fichiers `*.mail` ou `*.para`

`c/` répertoire dans lequel se trouvent des fichiers des fonctions `C *.c`.

message : définition identique à asno. Par défaut : `message_asverif`.

version est un paramètre qui porte le nom de la version de référence concernée (`NEW1`, `NEW2`, `NEW3`). Par défaut actuellement : `NEW3`.

9.4.2 Tâches réalisées

Les tâches décrites ci-dessous le sont pour tous les fichiers des répertoires `user(s)` contenus dans *repertoire*. Ces répertoires ont le nom d'un `user` Cray contenu dans le fichier `ident` et pour lequel l'appelant détient une délégation de restitution (fichier `lien_dvp`).

Seul(s) le(s) fichier(s) `histor` est(sont) obligatoire(s), les actions décrites ci-dessous ne sont, bien entendu, réalisées que si les fichiers auxquels elles font référence existent. Dans certain cas, des actions de substitution seront effectuées, elles seront clairement spécifiées.

- Destruction de tous les répertoires et de tous les fichiers se trouvant dans le(s) répertoire(s) "`$TMPDIR/`*`user(s)`*". Recopie du contenu des *`repertoire/user(s)`* dans les répertoires "`$TMPDIR/`*`user(s)`*".

  Le module travaille alors à partir des fichiers qui se trouvent dans `$TMPDIR/user(s)`.

  Les répertoires `$TMPDIR/user(s)"` *sont protégés en écriture à tout utilisateur sauf à l'utilisateur administrateur*. `asverif` *(qui a les priorités de l'administrateur) travaille alors uniquement dans les fichiers de* `$TMPDIR/ser(s)`.
- Vérification du type des fichiers `fortran`, `catalogu`, `unigest`, `test/*.comm` selon les critères énoncés dans la tâche 1 de `asno`.

  Vérification que `histor` est non vide (pour chaque *`user`*).
- Vérifications d'ordre syntaxique
  - Pour fortran et histor :

    Détection des lignes de plus de 72 caractères et inscription des lignes correspondantes dans *`message`* (*`iret`* `= 4`).

    Détection des lignes et inscription des lignes correspondantes dans *`message`* (*`iret`* `= 2`) contenant :

    des caractères minuscules,
    des caractères autres que l'alphabet A-Z, les chiffres 0-9, l'espace, la tabulation et les signes : `, ) ( * + − = / ' & § ! % : " . ? $ < > _`

    Détection des lignes et inscription des lignes correspondantes dans *`message`* (*`iret`* `= 4`) contenant : des caractères autres que l'alphabet A-Z et a-z, les chiffres 0-9, l'espace, la tabulation et les signes : `, ) ( * + − = / ' & § ! % : " . ? $ < > ; é è # \ [ ]`
  - Pour catalogu :

    Détection des lignes de plus de 72 caractères et inscription des lignes correspondantes dans *`message`* (*`iret`* `= 4`).

    Détection des lignes et inscription des lignes correspondantes dans *`message`* (*`iret`* `= 4`) contenant :

    des caractères autres que l'alphabet A-Z, les chiffres 0-9 et les signes : `, ) ( * + − = / ' & § ! % ; é è à _`
  - Pour `unigest` :

Détection des lignes de plus de 72 caractères et inscription des lignes correspondantes dans *message* (*iret* = 4).

Vérification que chaque unité n'est référencée qu'une seule fois sur l'ensemble des développeurs.

- Pour test/*.comm :

  Détection des lignes de plus de 72 caractères et inscription des lignes correspondantes dans *message* (*iret* = 4).

  Détection des lignes et inscription des lignes correspondantes dans *message* (*iret* = 4) contenant :

  des caractères autres que l'alphabet A-Z et a-z, les chiffres 0-9 et les signes :
  ,) (* + − = / ' & § ! % ; é è à _

- Pour histor :

  Détection des lignes de plus de 72 caractères et inscription des lignes correspondantes dans *message* (*iret* = 4).

  Détection des lignes contenant un caractère autre que : tout caractère autorisé dans les différents types de source sauf les caractères accentués et inscription des lignes correspondantes dans *message* (*iret* = 4).

- Recherche du nom des unités de source selon les critères énoncés pour la tâche B d'asno.
- Recherche dans le fichier de code noté des unités de source déjà notées.
  - Si user n'est pas le seul à avoir noté l'unité de source :

    si c'est le premier un message indiquant le nom de l'unité est inscrit dans *message* (*iret* = 0),

    sinon un message indiquant le nom de l'unité et le nom de l'auteur premier noté est inscrit dans *message* (*iret* = 4),

    Si une unité n'a pas été notée un message indiquant le nom de l'unité est inscrit dans *message* (*iret* = 0) et (très gentiment) on la note.
- Vérification de cohérence avec la version de référence *version* :

  - Vérification que les unités de source de fortran ou cal ou c/*.c ou catalogu :

    du type MODIF sont restituées dans la bonne bibliothèque et à la même date que la version de référence *version*.,

    du type AJOUT n'existent pas déjà dans toutes les bibliothèques d'*Aster* et sont restituées dans une bibliothèque existante.
  - Vérification que les bibliothèques spécifiées des unités de source de unigest à détruire sont correctes pour la version *version*..
  - Vérification que les bibliothèques spécifiées des unités de source de unigest à déplacer sont correctes pour la version *version*..
  - Vérification que les cas-tests :

    du type MODIF existent et à la même date que la version de référence *version*,

du type AJOUT n'existent pas déja

Si tel n'est pas le cas, les unités de source incriminées sont signalées dans le fichier *message*(*iret = 4*).

- Destruction de tous les répertoires et de tous les fichiers se trouvant dans le répertoire "$TMPDIR/*user(s)*".

9.5 asrest : restitution de source

Vérifier qu'un ensemble de source peut être introduit dans une version de référence du *Code Aster* et préparer la mise à jour éventuelle.

Cet outil effectue un certain nombre de vérifications puis il lance le passage des cas-tests de référence dans un job (de nuit par défaut mais modifiable).

Cet outil doit être utilisé par toute personne souhaitant restituer un développement.

Comme asverif, asrest permet d'effectuer une restitution groupée pour les développements dépendants.

9.5.1 Mode d'appel

```
asrest repertoire [version [ message [classe_job]]]
```

repertoire est le nom d'un répertoire. Il a la même fonction que dans asverif. On peut donc y trouver, suivant les cas, dans des répertoires user-CRAY les fichiers :

- histor,
- fortran,
- cal,
- catalogu,
- unigest,
- test/
  - *.comm Fichier de commande du cas-test,
  - *.mail Maillage Aster du cas-test,
  - *.mali Maillage Ali-Baba du cas-test,
  - *.mgib Maillage Gibi du cas-test,
  - *.msup Maillage Super-Tab du cas-test,
  - *.para Paramètres d'exécution du cas-test.
- c/*.c Fichiers des fonctions C.

Par rapport à asverif on doit trouver dans le répertoire test/ tous les fichiers nouveaux ou modifiés concernant les cas-tests ajoutés ou modifiés.

version : est un paramètre qui porte le nom de la version de référence concernée (NEW1, NEW2, NEW3). La valeur prise par défaut est NEW3.

message : définition identique à asno. La valeur prise par défaut est message_asrest.

classe_job : classe des jobs Cray soumis par as.tout.

9.5.2 Tâches réalisées

Seul(s) le(s) fichier(s) histor est(sont) obligatoire(s), les actions décrites ci-dessous sont, bien entendu, réalisées que si les fichiers auxquels elles font référence existent. Dans certain cas, des actions de substitution seront effectuées, elles seront clairement spécifiées.

- Tâches de asverif sauf la destruction des répertoires de travail
- Si *iret* ≥ 4 FIN.
- Enregistrement du répertoire complet de *repertoire/user(s)* dans le(s) fichier(s) eda/*version*/*user(s)*/identifi
- Insertion dans user(s)/histor de l'auteur et de la date courante.
- Mise à jour des dates et des auteurs dans les lignes INFO :
  - les lignes AJOUT sont changées en MODIF,
  - les dates sont mises à la date d'utilisation d'asrest,
  - l'auteur porté par la ligne est mis à *user* plus le nom du développeur.

  Chaque mise à jour fait l'objet d'une inscription dans le fichier de récapitulation eda/version/user/recap
- Eclatement des routines de *user(s)*/fortran dans le répertoire eda/*version*/*user(s)*/bibfor/*bibli* avec une routine par fichier, *bibli* étant la bibliothèque spécifiée par la ligne INFO.
  Recopie des routines à déplacer de FORDEPLA de l'ancienne bibliothèque de référence dans eda/*version*/*user*/bibfor/*bibli* où *bibli* est la nouvelle bibliothèque.
- Compilation des routines de eda/*version*/*user(s)*/bibfor, avec : cft77_aster dvpdbg
  les ERRORS de compilation sont recopiées dans message (*iret* = 4), les WARNINGS également (*iret* = 2).
  Le module objet est conservé dans fortran.o.
  (Dans le cas de développements dépendants il y a une copie dans chaque eda/*version*/*user* concerné.)
- Si *iret* ≥ 4 FIN.
- segldr avec les bibliothèques objets de la version version. Le module exécutable porte le nom execut.
  (Dans le cas de développements dépendants il y a une copie de execut dans chaque eda/*version*/*user* concerné).
- Compilation des catalogues (tâches définies pour ccat92 moins les vérifications, -qui sont déjà effectuées dans asrest).
- Si *iret* ≥ 4 FIN.
- Recopier dans eda/*version*/*user(s)* la liste des cas-tests a passer (liste_ct.rest) et l'enrichir des cas-tests présents dans cette évolution.
- Lancement des cas-tests (par défaut en classe nuit).
  Tâches identiques à as.tout avec les fichiers crées par asrest (execut, catalogue de commandes et d'éléments s'ils sont présents).
  La classe des jobs de cas-test par défaut le batch de nuit (sinon paramètre *classe_job*).
  Le lancement des cas-tests est interrompu si au moins 5 cas-tests ne se terminent pas par OK.
  Pour passer les tests un repertoire eda/*version*/*user_de_reference*/astout est créé. Dans ce repertoire se trouvent tous les fichiers de cas-tests nécessaires. (Ceux qui surchargent la référence).

Les résultats des cas-tests non OK seront également dans ce répertoire sous resu_NOOK/ et resu_NOOK/flash. (cf as.tout). Le *user_de_reference* est le user de l'appelant s'il rend lui même du source, sinon c'est le premier user par ordre alphabétique des développeurs qui restituent. Les fichiers .code et .resu des cas-test ajoutés sont conservés. Les informations pour la mise à jour de la liste des cas-tests (temps total, mémoire requise) des cas-tests ajoutés ou modifiés sont conservé.

- Recopie du fichier *message* dans le(s) fichier(s) eda/*version*/*user(s)*/message.
  Recopie de la valeur de *iret* dans le(s) fichier(s) eda/*version*/*user(s)*/iret.

9.5.3 Fichiers créés

Les fichiers créés sont :

```
/      /eda/      /       /histor
aster  version    user(s
                  )
                          /
                          fortran
                          /cal
                          /
                          catalog
                          u
                          /test    / nom_cas_test
                                   .comm
                                   / nom_cas_test
                                   .para
                                   / nom_cas_test
                                   .mail
                                   / nom_cas_test
                                   .mali
                                   / nom_cas_test
                                   .msup
                                   / nom_cas_test
                                   .mgib
                          /
                          unigest
                          /bibfor  / bibli          / routine.f
                          /bibcal  / bibli          / module.cal
                          /bibc    / bibli          /
                                                    fonction_C.c
                          /obj     /fortran.o
                                   /cal.o
                                   / fonction_C .o
                          /execut
                          /catalo  / bibli          /
                                                    subcat.cata
                          /catobj  /commande
                                   /elements
                          /
                          identif
                          i
                          /
                          message
                          /iret
                          /recap
                          /ls_ct
```

```
/astout  /*.comm
         /*.para
         /*.mail
         /*.mali
         /*.msup
         /*.mgib
         /liste_ct
         /resu_NOOK      /*.resu
                         /*.code
                         /
                         flash/*error

                         *output
```

9.6 as.tout : passage de cas-tests

Enchaînement d'une liste de cas-tests avec analyse des résultats.

Caractéristiques principales de cet outil :

- construction des scripts de commandes Unicos de chaque cas-test à partir des fichiers de données *Aster*,
- soumission de ces cas-tests en utilisant la possibilité du Cray d'exécuter plusieurs jobs en même temps,
- collecte des résultats des différents jobs,
- réalisation d'un document synthétique de l'ensemble des cas-tests envoyés et éventuellement exécution d'une action en fonction du résultat des cas-tests.

Exemple de synthèse :

```
                          -- A S T E R -- VERSION  2.05.11 --

   --- Fin de tous les cas tests       :  le 12/07/93 a
15:21:01
   --- Voici les resultats :

                                        Temps CPU  Temps SYS  Temps
Total   Cout
   adlv100a    OK                         5.28       0.92       6.20
13.79 F
   ahlv100b        NOOK_TEST_RESU         6.70       0.65       7.35
16.33 F
   hplv100a    OK                         7.80       0.63       8.42
18.72 F
   hplv101a    OK                         4.34       0.72       5.05
11.23 F
   hplv101b        CPU_TIME_LIMIT         6.29       0.82       7.11
15.80 F
   hsnl100a    OK                         3.04       0.67       3.71
8.24 F
   hsnv100a    OK                        10.81       0.69      11.50
25.56 F
   --------  ----  ---------------  --------  --------
--------  --------
      7        5          2          44.26       5.09      49.35
109.67 F


                        PROBLEME AVEC 2 CAS-TESTS
```

as.tout vérifie la cohérence des arguments transmis puis construit et soumet un job (qastout) de la même classe que celle des cas-tests. Ce job construit et soumet des chaînes de cas-tests et attend la fin de tous les cas-tests lancés pour mettre en forme les résultats et les envoyer dans le fichier message.

9.6.1 Mode d'appel

as.tout *fichier_parametre* [*message*]

fichier_parametre : Fichier contenant les différents paramètres d'exécution d'as.tout.

message : fichier des messages. Par défaut message_astout.

fichier_parametre contient 12 valeurs.

Un paramètre est le premier champ d'une ligne ne débutant pas par le caractère '%'. Il ne peut donc pas y avoir plusieurs paramètres sur la même ligne.

La reconnaissance des paramètres se fait sur l'ordre de déclaration. Si l'on veut avoir la valeur par défaut d'un paramètre il faut le remplacer par le caractère `'.'`. Les lignes vides sont autorisées.

Les paramètres sont les suivants :

- nom du fichier contenant la liste des cas-tests (pas de valeur par défaut),
- répertoire contenant les fichiers référence des cas-tests (fichiers de commandes, de maillages et paramétres avec la même organisation que dans la base de référence par défaut `/aster/astest/`*`version`*`/-`),
- répertoire contenant des fichiers de cas-tests qui surchargeront les fichiers de référence (pas de valeur par défaut),
- répertoire recevant les résultats (pas de valeur par défaut), les fichiers `nom_cas-test.resu` et `nom_cas-test.code` y sont crées et, s'il n'existe pas déjà, un répertoire `flash/` est créé, il reçoit les fichiers `nom_cas-test.error` et `nom_cas-test.output`),
- nom de l'exécutable *Aster* (par défaut `/aster/`*`version`*`/execut`),
- fichier contenant le catalogue des commandes compilées (par défaut `/aster/`*`version`*`/catobj/commande`),
- fichier contenant le catalogue des éléments compilés (par défaut `/aster/`*`version`*`/catobj/elements`),
- la classe de job Cray pour passer les cas-tests : `batch` ou `nbatch` ou `wbatch` (par défaut `nbatch` -batch de nuit-),
- le nombre maximum de cas-tests qui se terminent mal avant interruption du lancement des autres cas-tests (pas de défaut),
- la version *Aster* pour les cas-tests (sert à déterminer les noms par défaut, `NEW3` par défaut),
- si "`DATE`" la date et l'heure est rajoutée en extension des résultats des jobs, sinon les extensions `.error` et `.output` sont ajoutées au nom du cas-tests pour le résultat des jobs Cray,
- si "`RESOK`" les fichiers de résultat de tous les jobs lancés sont envoyés dans le répertoire résultat ; pour toute autre valeur, seuls les résultats des jobs non `OK` sont conservés.

9.6.2 Tâches réalisées

Par `as.tout` :

- Vérification des paramètres :
  - Existence des différents répertoires et fichiers.
  - Vérification des domaines de validité des paramètres classe du job, version et datation des fichiers du "flasheur".
  - Vérification que les noms de cas-test respectent la régle de nommage (des lettres puis des chiffres et une lettre).
  - Si `iret` $\geq$ 4 FIN.
- Vérification de l'existence des fichiers `.para` des cas-test à passer.
- Construction du job lanceur de cas-test de même classe que les cas-tests.
- Soumission de ce job par `qastout`.

Par le lanceur de cas-test `qastout` :

- Vérifications des deux premières tâches de `as.tout` .
- Création d'un répertoire dans `/tmp` pour collecter les résultats de tous les cas-tests.
- Construction des jobs de tous les cas-tests.

  Chaque job vérifie les principes suivants :

  - si les paramètres `NQS` sont incompatibles avec la machine fin de `as.tout` (`iret=4`),
  - si le cas-test n'est pas le dernier d'une chaîne :
    - teste le code retour du `qsub` du cas-test suivant,
    - vérifier par `qstat` que le cas-test est bien pris en compte par `NQS`,
    - envoyer un `mail` sur Cray à l'administrateur *Aster* (`d6bhhhh`) en cas de problème détecté par `NQS`,
    - en cas de problème `NQS` appel au script `Dernier_Job`.
  - si le cas-test est le dernier d'une chaîne :
    - appel au script `Dernier_Job`.
- Soumission des cas-tests sous la forme de 16 chaînes séquentielles lancées en parallèle.

Le script `Dernier_Job` est appelé par tout cas-test détectant un problème `NQS` et le dernier job de chaque chaîne. Ce script doit déterminer si le job qui l'appelle est le dernier cas-test en exécution sur Cray parmi tous les cas-tests lancés par cet `as.tout`. Si c'est le cas `Dernier_Job` invoque le job `Dernier_Job.btc`

Tâches du job `Dernier_Job.btc` :

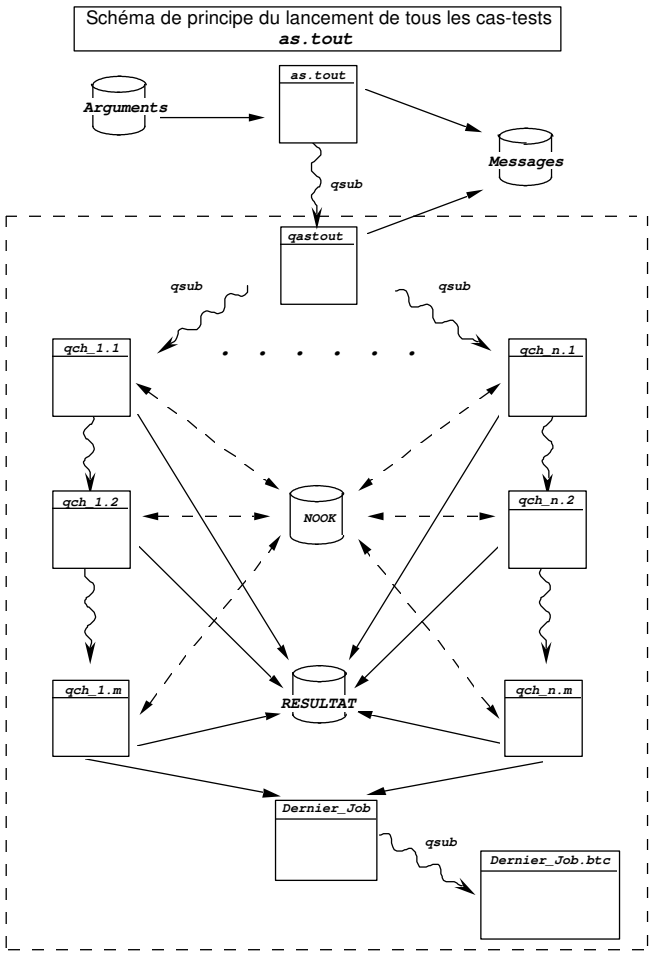
Mise en forme du résultat des cas-tests

Pour chaque cas-test soumis affichage

- d'un diagnostic :
  - OK,
  - `ARRET_ANORMAL` (arrêt non géré par *Aster*),
  - `<F>_SUPERVISEUR` (*Aster* interrompu par le superviseur),
  - `ERREUR_<F>` (autres erreurs fatales détectées par *Aster*),
  - `ERREUR_<S>` écrasement ?????? sans conséquence visible,
  - `<E>_VOLATILE` il reste un objet dans la base volatile,
  - `NOOK_TEST_RESU` (non respect de la valeur de référence),
  - `DEFAUT_FICHIER` (il manque un fichier ou un répertoire indispensable au cas-test),
  - `CPU_LIMITE` (manque de temps pour finir le cas-test),
  - INTERRUPTION_JOB,
- du temps CPU,
- du temps système,
- du temps total du cas-test,
- du coût du cas-test.

Pour l'ensemble des cas-tests soumis affichage :

- du temps CPU total,
- du temps système total,
- du temps total de tous les cas-tests,
- du coût de tous les cas-tests.



9.7 ccat92 : compilation des catalogues

Compilation d'unités de catalogue dans des fichiers de commandes et d'éléments compilés.

Pas de vérification de cohérence, ni d'identité du développeur.

9.7.1 Mode d'appel

ccat92 *catalogu catalo catobj* [*execut* [*para* [*version* [*message*]]]]

catalogu : fichier contenant les unités de catalogue à compiler.

catalo : est un répertoire en sortie qui est organisé et qui contient les sources éclatés de catalogu en éléments de CATALOGUE conforme au répertoire des versions en exploitation.

catobj : est un répertoire en sortie conforme au répertoire des versions en exploitation. catobj contient les catalogues compilés de la version *version* d'*Aster* "surchargées" par les éléments contenus dans catalogu.

execut : fichier en entrée qui contient le module exécutable qui doit effectuer la compilation de catalogu (par défaut c'est /aster/*version*/execut).

para : paramètre qui vaut "prive" pour l'utilisation faite de ccat92 dans asrest et qui permet de marquer les catalogues compilés de la mention PRIVE.

version et *message* : définitions identiques à asno (par defaut *message* = message_ccat92).

9.7.2 Tâches réalisées

Vérification que *catalogu* est bien du type CATALOGU (première tâche d'asno).

Eclatement des sous-catalogues dans catalo/
Si *iret* ≥ 4 FIN

Dans répertoire de travail établir un lien symbolique avec tous les catalogues de la version de référence.

Réaliser la "surcharge" avec les unités du fichier *catalogu* en recopiant les fichiers de *catalo*.

- Compilation du catalogue des commandes :
  - Concaténation de tous les sous-catalogues de commande dans fort.3,
  - Assignation du fichier 'COMMANDE_PRIVEE' à fort.3,
  - Compilation du fichier 'COMMANDE_PRIVEE' avec *Aster* (*execut* ou l'exécutable par défaut de *version*),

- Si `iret ≥ 4 FIN`,
  Copie des messages de compilation par *Aster* dans `message (fort.6)`
  Copie du fichier d'erreur (`fort.9`) dans *`message`*.
- Sinon sauvegarde du résultat de la compilation des commandes dans `catobj/commande.`

.

- Compilation du catalogue des éléments :
  - recopier `compelem/grandeurs_1ere.cata` dans `fort.4`,
  - compter le nombre de sous-catalogues dans `options/` (=*nb_opt*),
  - rajouter '`OPTION_SIMPLE` *nb_op*' dans `fort.4`,
  - rajouter tous les sous-catalogues de `options/` dans `fort.4`,
  - rajouter '`FIN`' dans `fort.4`,
  - rajouter tous les sous-catalogues de `typelem/` dans `fort.4`,
  - rajouter '`FIN`' dans `fort.4`,
  - rajouter `compelem/type_maille.cata` dans `fort.4`,
  - rajouter `compelem/modes_partages.cata part` dans `fort.4`,
  - rajouter `compelem/phenomenes_modelisation.cata` dans `fort.4`.

Si *version* est `NEW1` ou `STA1` il n'y a pas de compilation avec *Aster* . Le résultat est juste la concaténation décrite ci-dessus.

Sinon compilation avec *Aster* en utilisant le catalogue de commande précédemment compilé, s'il existe.

Si `iret` < `4` sauvegarde du résultat de la compilation des éléments dans `catobj/elements`.

Sinon :

- Copie des messages de complation par *Aster* dans *message* (`fort.6`).
- Copie du fichier d'erreur (`fort.9`) dans *message*

10 Outils de mise à jour de la version

Ces outils ne seront disponibles que sur la machine de l'administrateur des sources *d'Aster*.

10.1 majnew : mise à jour de la version NEW

Mettre à jour une version de référence `NEW` du *Code Aster* par incrémentation de niveau.

10.1.1 Mode d'appel

Ce module protégé en lecture, en écriture et en exécution n'est accessible que par les administrateurs des versions.

`majnew` *l_developpeur para version message iret_max user_IBM password_IBM*

l_developpeur est un fichier dans lequel on trouve, ligne par ligne, la liste des `identificateurs` restituant leur source.

para est un paramètre qui vaut "`COPOLD`" ou "`PASCOPOLD`" permettant de sauvegarder ou de simplement remplacer la version mise à jour.

version est la version (`NEW1` `NEW2` `NEW3`) à mettre à jour.

message **:** définition identique à asno. Par défaut message_majnew.

iret_max **:** code d'erreur maximum de asrest pour que la restitution soit prise en compte (0 ou 2).

user_IBM **:** user IBM pour la sauvegarde.

password_IBM : mot de passe du user IBM.

10.1.2 Tâches réalisées

Si *version* = NEW2, est de la validité du mot de passe sur IBM. Si problème *iret* = 4 et fin de majnew

Constitution dans /aster/eda/*version*/majnew/ des fichiers :

```
/aster/ eda/ version /majnew/ fortran
                             histor
                             recap
                             liste_ct
                             ls_ct
                             catalo/ cata/ sous-
catalogue.cata
                             unigest
                             bibfor/ bibli/  routine.f
                             obj/    fortran_user(s).o
                                     cal_user(s).o
                                     fonctions_C.o
                             test/ *.comm
                                   *.para
                                   *.mail
                                   *.mali
                                   *.msup
                                   *.mgib
                                   *.code
                                   *.resu
```

par concaténation des fichiers correspondants dans les répertoires eda/*version*/*user* de tous les identificateurs présents dans *l_developpeur*.

Ceci à condition que le fichier eda/*version*/*user*/iret existe et contienne au plus la valeur *iret_max*. Sinon un message recopiant le eda/*version*/*user*/message est émis (*iret* = 4) et fin de majnew.

Si tous les eda/*version*/*user*/iret de la *l_developpeur* sont incorrects alors FIN (*iret* = 4).

Les sources FORTRAN compilés (en mode debug) de chaque *user* sont copiés sous le nom fortran_*user*.o, l'assembleur CAL compilé est copié sous cal_*user*.o et les fonctions C sous *nom_fonction*.o.

Incrémentation du numéro de niveau trouvé dans :

/aster/*version*/versio.

Ce fichier sera mis à jour juste à la fin du module.

Recopie avec mise à jour de la date de la ligne INFO et des variables contenant le numéro de version de la routine FORTRAN versio dans bibfor.

Compilation en majdbg le module objet est placé dans :

/aster/ eda/ /*version* /majnew/ versio.o

Sauvegarde des bibliothèques compilées bibobj en bibold et catobj en catold si *param* = COPOLD.

Compilation de FORTRAN en majobj :

- Appel à bld avec le résultat pour la mise à jour de la bibliothèque bibobj avec les suppressions de routines de unigest.
- Appel à bld avec les fortran_*users(s)*.o pour la mise à jour de la bibliothèque bibdbg avec les suppressions de routines de unigest.

Compilation de CAL en majobj :

- Appel à bld avec le résultat pour la mise à jour de la bibliothèque bibobj.
- Appel à bld avec les cal_*users(s)*.o pour la mise à jour de la bibliothèque bibdbg.

Compilation de BIBC en majobj :

- Appel à bld avec le résultat pour la mise à jour de la bibliothèque bibobj.
- Appel à bld avec les cal_*users(s)*.o pour la mise à jour de la bibliothèque bibdbg.

segldr avec la bibliothèque bibobj de la version *version*..

Mise à jour de /aster/*version*/execut

Destruction des unités de CATSUPPR dans /aster

/aster/ *version*/ catalo/ *bibli*/ *sub_cat*
diffcat/*bibli*/ *sub_cat*

Compilation des catalogues avec le paramètre *para* = "public" :

ccat92 *version* execut catalogu catalo catobj message_ccat iret_ccat *para*

diff-b/aster/*version*/bibfor/*bibli*/*routine* /aster/eda/*version*/bibfor/*bibli*/*routine* pour toutes les routines de majnew/bibfor

Concaténation, en tête, dans /aster/*version*/diffsub/*bibli*/*routine* (avec écriture d'une identification de la mise à jour).

diff-b/aster/*version*/catalo/*bibli*/*sub_cat* /aster/eda/*version*/catalo/*bibli*/*sub_cat* pour toutes les routines dans majnew/catalo

Concaténation, en tête, dans /aster/version/diffcat/bibli/sub_cat (avec écriture d'une identification de la mise à jour).

Copie de

/aster/	eda/	version/majnew/	catalo/	bibli/	sub_cat
			bibfor/	bibli/	routine
			bibcal/	bibli/	module
			bibc/	bibli/	fonction

dans :

/aster/	version/	catalo/	bibli/	sub_cat
		bibfor/	bibli/	routine
		bibcal/	bibli/	module
		bibc/	bibli/	fonction

Destruction des routines à détruire de `unigest` et des anciennes bibliothèques de routines à déplacer dans :

/aster	/ *version*	/bibfor	/ *bibli* / *routine*
		/diffsub	/ *bibli* / *routine*
		/diffcal	*bibli* / *module*
		/diffc	*bibli* / *fonction*
		/fort	/ *routine*

Créations des liens symboliques entre *:*

/aster	/ *version*	/bibfor	/ *bibli* / *routine*

et

/aster	/ *version*	/fort	/ *routine*

Copie de `histor` dans

/aster	/ *version*	/histor	/v.sv.n

où `v.sv.n` est le numéro de version/sous-version/niveau avec en-tête : `Version v.sv.nv` *date*

Ajout de `recap` trié alphabétiquement en fin de `histor`.

Détermination et ajout des statistiques en fin de `histor`.

Préparation de l'`histor` version IBM (par concaténation à la fin du `histor` existant), pour une version `NEW2` uniquement.

Suppression de toutes les notations des unités de `fortran`, `catalo`, `test/` et `unigest` dans `quirou`, `quicat` et `quites` quels que soient les développeurs.

Suppression de toutes les notations de plus de 3 mois (y compris les 'RAYE').

Destruction des unités `TESSUPPR`, et mise à jour des listes de cas-tests (`liste_ct.tout` et `liste_ct.rest`).

Copie des cas-tests ajoutés ou modifiés et mise à jour de la liste de cas-tests pour gérer les listes, restreinte, totale et de performance : `/aster/astest/`*version*`/Liste`.
L'ancienne liste est conservée dans le fichier `/aster/astest/`*version*`/Liste.old`

Réinitialisation du fichier de délégation de restitution `lien_dvp`

Mise à jour de /aster/*version*/versio

Si*version* `NEW2` :

- copie sur IBM des fichiers ajoutés ou modifiés.
  Sauvegarde des ordres dispose de ces copies dans le fichier dispIBM (pour pouvoir relancer cette opération indépendamment du reste de la mise à jour en cas de problème avec l'IBM).
- suppression sur IBM des fichiers supprimés sur Cray (FORTRAN , CATALOGU et CASTEST).
  Sauvegarde du job IBM de suppression dans le fichier suppIBM (pour pouvoir relancer cette opération indépendamment du reste de la mise à jour en cas de problème avec l'IBM).

Envoyer le fichier de messages de majnew à l'ASA par mail (Administrateur des sources *Aster*).

Envoyer le fichier histor à tous les développeurs ayant un compte sur une station de travail (dont l'adresse est dans le fichier ident *Aster*).

10.1.3 Fichiers de /aster/NEWn

```
/       version /        histor/        v.sv.n
aster
/
                         fort/          routine
                         bibfor/        bibli /   routine.f
                         diffsub/       bibli /   routine.f
                         bibcal/        bibli /   module.ca
                                                  l
                         diffcal/       bibli /   module.ca
                                                  l
                         bibc/          bibli /   fonction.
                                                  c
                         diffc/         bibli /   fonction.
                                                  c
                         diffcat/       bibli /   subcat.ca
                                                  ta
                         catobj/        command
                                        e
                                        element
                                        s
                         versio
```

```
/       astest/ version  Liste
aster   /                Liste.old
/                        liste_ct.rest  castest
                         liste_ct.tout
                         diffct/
                         castest .comm
                         castest .mail
                         castest .mali
                         castest .mgib
                         castest .msup
                         castest .para
                         castest .resu
                         castest .code
```

10.2 finmaj : fin de mise à jour de la version NEW

Détruire les fichiers des développeurs ayant restitué leur source (même ceux n'ayant pas étés retenus pour la mise à jour).

Sauvegarder sur IBM les fichiers des jobs de la mise à jour si la version = NEW2 .

Ce module protégé en lecture, en écriture et en exécution n'est accessible que par les administrateurs des versions.

10.2.1 Mode d'appel

```
finmaj version message user_IBM password_IBM repertoire
```

version est la version (NEW1 NEW2 NEW3) qui vient d'être mise à jour.

message: définition identique à asno.

user_IBM est le user IBM pour la sauvegarde.

password_IBM est le mot de passe du user précédent.

repertoire est le répertoire qui contient les fichiers de résultat de majnew qui doivent être sauvegardés sur IBM.

10.2.2 Tâches réalisées

- Destruction des /aster/eda/*version*/*user(s)*
- Concaténation des fichiers présents dans *repertoire*.
- Si *version* = NEW2 : sauvegarde sur IBM de ce fichier dans ASTER.CRB(CVSVNV)
- Si la sauvegarde s'est bien passée, destruction de tous les fichiers de *repertoire*.
- Soumission de la liste compléte des cas-tests, de nuit et en ne conservant que le résultat des jobs Non OK (sous /c/gr1/d6bhhhh/finmaj).

10.3 actuliste : actualisation du coût des différentes listes de cas-tests

10.3.1 Mode d'appel

actuliste [*sauv* [*version* [*message*]]]

sauv : si ce paramètre = "sauv" le résultat de l'actualisation remplace le fichier Liste.

version : version d'*Aster* concernée (NEW1 ou NEW2 ou NEW3 par défaut NEW3).

message : fichier de message de la commande (par défaut message_actuliste).

10.3.2 Tâches réalisées

Pour pouvoir récupérer les tarifs des différentes classes de job sur Cray actuliste est "enrobé" dans un job qui :

- détermine les tarifs des jobs de jour, nuit, week-end,
- Calcul des temps totaux et les coûts des différentes listes de cas-tests (restreinte, compléte, performance),
- donne le résultat dans le fichier de message,
- si sauv="sauv" :
  copie du fichier /aster/astest/*version*/Liste
  dans /aster/astest/*version*/Liste.old.

  Sauvegarde du nouveau fichier Liste :

  copie du fichier /aster/astest/*version*/liste_ct.rest
  dans
  /aster/astest/*version*/liste_ct.rest.old,
  copie du fichier
  /aster/astest/*version*/liste_ct.tout
  dans
  /aster/astest/*version*/liste_ct.tout.old,

  construction du fichier /aster/astest/*version*/liste_ct.rest en extrayant les cas-tests ayant l'indicateur "." en dernière colonne du fichier liste.
- construction du fichier /aster/astest/*version*/liste_ct.tout en extrayant les cas-tests ayant l'indicateur "." ou "S" en deuxième colonne du

fichier liste. Les cas-tests ayant l'indicateur "P" sont dans le fichier avec un "%" en première colonne (pour les mettre en commentaire).

10.4 majliste : mise à jour des listes complétes et restreintes

10.4.1 Mode d'appel

```
majliste [version [message] ]
```

version : version d'*Aster* concernée (NEW1 ou NEW2 ou NEW3 par défaut NEW3).

message : fichier de message de la commande (par défaut message_majliste).

10.4.2 Tâches réalisées

- Mise à jour de la liste restreinte `/aster/astest/version/liste_ct.rest` en extrayant du fichier `/aster/astest/version/Liste` les champs nom de cas-test et titre, des lignes dont la deuxième colonne (le séparateur étant le caractère |) contient un point.
- Mise à jour de la liste complète `/aster/astest/version/liste_ct.tout` en extrayant du fichier `/aster/astest/version/Liste` les champs nom de cas-test et titre de toutes les lignes décrivant un cas-test. Les cas-tests de la liste de performance (un `P` en deuxième colonne) sont mis en commentaire (un `%` en premier caractère de la ligne).

10.5 `majsta` : stabilisation d'une version

Avant la stablilisation d'une version, prévenir par courrier toutes les personnes concernées de la date de l'opération.

Permet de stabliser la version `NEWn` en `STAn`.

10.5.1 Mode d'appel

`majsta numero_version message para`

`numero_version` : numéro de la version d'*Aster* à stabliser.

`message` : fichier de message de la commande.

`para` : parametre `OLD` pour copier la `STA` en `OLD` et `PASOLD` pour ne pas la copier.

10.5.2 Tâches réalisées

Incrémentation de 1 du numéro de sous-version dans la routine `versio`

Compilation et édition des liens :

- maj de /aster/NEW*n*/bibobj
- maj de /aster/NEW*n*/bibdbg
- maj de /aster/NEW*n*/execut

Si `para` = `OLD` Copie de `/aster/STAn` dans `/aster/OLDn` :

- lien entre /aster/OLD*n*/execut et /aster/OLD*n*/EXECUT
- lien entre /aster/OLD*n*/catobj/elements et /aster/OLD*n*/CATOBJ/ELEMENTS
- lien entre /aster/OLD*n*/catobj/commande et /aster/OLD*n*/CATOBJ/COMMANDE

Copie de :

- /aster/NEW*n*/mode_exec
- /aster/NEW*n*/execut dans /aster/STA*n*/execut
- /aster/NEW*n*/catobj/* dans /aster/STA*n*/catobj

lien entre /aster/STA*n*/execut et /aster/STA*n*/EXECUT
lien entre /aster/STA*n*/catobj/elements et /aster/STA*n*/CATOBJ/ELEMENTS
lien entre /aster/STA*n*/catobj/commande et /aster/STA*n*/CATOBJ/COMMANDE

- Conservation des cas-tests et de leurs résultats
  /aster/astest/NEW*n*/*.* dans /aster/astest/STA*n*

Création d'un histor de stabilisation.

Mise à jour de /aster/NEW*n*/versio .

Envoie, par e-mail, de l'histor à tous les développeurs.

Sauvegarde sur cassette Sun de :

```
/aster/NEWn/bibobj/
/aster/NEWn/bibdbg/
/aster/NEWn/fort/
/aster/NEWn/c/
/aster/NEWn/cal/
/aster/NEWn/catalo/
/aster/astest/NEWn/
```

10.6 Changement de version NEW

Pas d'outils spécifiques pour le changement de version NEW.

Détail des opérations "manuelles" à effectuer :

- Effectuer une version STA avec majsta.
- Sauvegarder sur cassette la version STA notamment les fichiers bibobj et bibdbg.
- Copie des sources de NEW*n* a NEW*n+1* :
  ```
  mkdir  /aster/NEW n+1
  rcp -r /aster/NEW n /* /aster/NEW n+1
   mkdir  /aster/astest/NEW n+1
  rcp -r /aster/astest/NEW n /* /aster/astest/NEW n+1
  ```
- Copie des restitutions effectuées en NEW*n* pour les transferer en NEW*n+1* :
  ```
   mkdir  /aster/eda/NEW n+1
  rcp -r /aster/eda/NEW n /* /aster/eda/NEW n+1
  ```
- Changer la valeur par défaut du paramètre *version* pour tous les outils de l'AGLA (modification de la routine /aster/adm/tool/init_defaut.csh).
  Faire la même chose pour asterix.
- Transformer les notations NEW*n* en NEW*n+1* dans les fichiers :
  ```
  /aster/agla/qui/quirou
  /aster/agla/qui/quicat
  /aster/agla/qui/quites
  ```
- Modifier le fichier /aster/eda/NEW*n+1* pour mettre : *n+1*.0.0.
- Faire un asrest sur le cas-test zzzz* pour pouvoir faire une mise à jour qui officialisera le nouveau numéro de version. (Ne pas oublier de faire un histor de circonstance).
- Faire une mise à jour avec majnew.
- Restreindre l'accés à /aster/NEW*n* et /aster/astest/NEW*n* avec aclaut.

11 Compilation et édition des liens

11.1 `cft77_aster` : module de compilation officiel d'*Aster* pour le Fortran

Effectuer les différents modes de compilation en vigueur pour *Aster*. Cette centralisation permet de modifier facilement la compilation officielle d'*Aster*.

Si code retour = `2` il y a des "warnings" fortran pour *Aster* ou des "warnings" non tolérés par *Aster*.

Si code retour = `4` il y a des erreurs de compilation fortran ou de mauvais arguments.

Les messages d'erreur sont envoyés dans le fichier standard de sortie (il est bien évidemment possible de rediriger dans un fichier avec >)

11.1.1 Mode d'appel

```
cft77_aster   mode_compil   fichier   objet
```

mode_compil : mode de compilation d'*Aster* développement (= `dvpdbg`, `dvpobj`, `majdbg`, `majobj`).

fichier : fichier source FORTRAN à compiler.

objet : fichier objet résultat.

11.1.2 Tâches réalisées

Options de compilation en mode `dvpdbg` :

```
/bin/cft77_4.0 -i64 -dp -en -m3 -M 408, 118, 881 -eiz
```

Options de compilation en mode `dvpobj` :

```
/bin/cft77_4.0 -i64 -dp -en -m3 -M 408, 118, 881 -ez
```

Options de compilation en mode `majdbg` :

```
/bin/cft77_4.0 -i64 -dp -en -m3 -M 408, 118, 881 -ez
```

Options de compilation en mode `majobj` :

```
/bin/cft77_4.0 -i64 -dp -en -m3 -M 408, 118, 881
```

En plus des erreurs de syntaxe détectées par le compilateur, un certain nombre de "warning" sont détectés et entraînent un code de retour 2 ou 4 selon la gravité.

Listes des "warnings" avec leur numéro donné par `cft77.`

Le code retour qu'elles engendrent et une description succinte.

Leur type : w warning

```
          A  ANSI

342   A2 :  Boolean  values  are  nonstandard.  (Interdit  sauf
DEFVEM)
720   A2 :  Equivalence caractere/non caractere
726   A2 :  Equivalence tableau de longueur 1 et common
753   A2 :  The use of edit descriptor "item" is nonstandard.
(Interdit          sauf JEIMPO)
890   A2 :  Utilisation des minuscules
895   A2 :  Expression   has   an   assumed-length   character
variable and is          an actual argument
```

Tous les autres warning entrainent un code retour = 4.

11.2 `cc_aster` : module de compilation officiel d'*Aster* pour le langage `C`

Effectuer les différents modes de compilation en vigueur pour *Aster*.

Le fichier objet est le même que le fichier source avec l'extension `.o`

Si code retour = `2` il y a des "warnings" `C`.

Si code retour = `4` il y a des erreurs de compilation `C` ou de mauvais arguments.

Les messages d'erreur sont envoyés dans le fichier standard de sortie (il est bien évidemment possible de rediriger dans un fichier avec >).

11.2.1 Mode d'appel

```
cc_aster   mode_compil   fichier
```

mode_compil : mode de compilation d'*Aster* développement (= `dvpdbg`, `dvpobj`, `majdbg`, `majobj`).

fichier : fichier source `C` à compiler.

11.2.2 Tâches réalisées

Options de compilation en mode `dvpdbg` :

```
cc -c -Gn
```

Options de compilation en mode `dvpobj` :

```
cc -c
```

Options de compilation en mode `majdbg` :

```
cc -c -Gn
```

Options de compilation en mode `majobj` :

```
cc -c
```

11.3 as_aster : module de compilation officiel d'*Aster* pour l'assembleur Cray (CAL)

Effectuer les différents modes de compilation en vigueur pour *Aster*.

Si code retour = 2 il y a des "warnings" as.

Si code retour = 4 il y a des erreurs de compilation as ou de mauvais arguments.

Les messages d'erreur sont envoyés dans le fichier standard de sortie (il est bien évidemment possible de rediriger dans un fichier avec >)

11.3.1 Mode d'appel

```
as_aster   mode_compil   fichier   objet
```

mode_compil : mode de compilation d'*Aster* développement (= dvpdbg, dvpobj, majdbg, majobj).

fichier : fichier source asembleur à compiler.

objet : fichier objet résultat.

11.3.2 Tâches réalisées

Options de compilation en mode dvpdbg :
as

Options de compilation en mode dvpobj :
as

Options de compilation en mode majdbg :
as

Options de compilation en mode majobj :
as

11.4 segldr_aster: module d'édition des liens officiel d'*Aster*

Constitution d'un nouvel exécutable *Aster* à partir de bibliothèques et d'éventelles bibliothèques (ou fichiers objets) de surcharge.

Si code retour = 2 il y a des "warnings" segldr.

Si code retour = 4 il y a des erreurs d'édition des liens ou de mauvais arguments.

Les messages d'erreur sont envoyés dans le fichier standard de sortie (il est bien évidemment possible de rediriger dans un fichier avec >)

11.4.1 Mode d'appel

```
segldr_aster bib_aster bib_fermeture nv_execut [ fichier(s)_surcharge ]…
```

11.4.2 Tâches réalisées

Appel à `segldr` avec les bibliothèques dans l'ordre suivant (priorité décroissante) :

- bin=*fichier_surcharges* s'ils existent,
- `bin=bib_aster`
- `bin=bib_fermeture`
- `bin=/usr/lib/sysz%nag12`
- `bin=/usr/lib/sysz%generale`

Options de `segldr` :

- `-f zero` Initialisation des variables à zéro.
- `-D 'DUPENTRY=NOTE,NOTE,NOTE'` Pas d'affichage des messages pour les entrées de modules en double (à cause de la bibliothèque de surcharge `fermetur`)

12 Outils de gestion de l'AGLA par l'administrateur

12.1 verrou : verrouillage de l'AGLA pour les développeurs

Sert à interdire l'accès aux outils de l'AGLA qui modifient les fichiers de référence. Cela permet de suspendre l'utilisation de l'AGLA pour la mise à jour de l'AGLA.

12.1.1 Mode d'appel

```
verrou [ "message explicatif de la cause du verrouillage" ]
```

12.1.2 Tâches réalisées

Met en place le mécanisme de verrouillage de l'AGLA. Le message explicatif sera transmis à tout développeur qui tentera d'utiliser un des outils de l'AGLA utilisant le mécanisme de verrouillage standard.

12.2 deverrou : déverrouillage de l'AGLA

Permet de deverrouiller le mécanisme d'accès exclusif aux outils de l'AGLA. Sert après un "plantage" d'un de ces outils ou après un verrouillage volontaire de l'Administrateur (avec `verrou`).

Permet aussi à chaque développeur de supprimer un verrou concernant son `user`.

12.2.1 Mode d'appel

```
deverrou
```

12.2.2 Tâches réalisées

Déverrouillage de l'AGLA.

12.3 fgrep_agla : "fast-grep" dans les sources de l'AGLA

Rechercher une chaîne de caractères simple (pas une expression régulière) dans les sources en C-shell.

12.3.1 Mode d'appel

```
fgrep_agla 'chaine de caracteres'
```

12.3.2 Tâches réalisées

```
fgrep "chaine de caracteres" /aster/outils/*[a-z0-9]
/aster/adm/tool/*[a-z0-9]
```

12.4 `stat_agla.qsub` : arbres d'appel statique des scripts-shells

Donner pour chacun des outils de l'AGLA l'arbre d'appel des modules C-shell et C.

Résultats dans `/aster/adm/g_agla/resu`.

Bug : ne marche pas pour des appels dynamiques (par variables).

12.4.1 Mode d'appel

```
qsub stat_agla.qsub
```

Cette opération étant relativement longue il est préférable de l'envoyer en batch (avec `qsub`).

12.4.2 Tâches réalisées

Pour chacun des modules de l'AGLA, rechercher les appels :

- `source` : du shell en "include",
- `csh` : du C-shell en module autonome,
- `qsub` : du C-shell en batch,
- les appels directs : des programmes écrits en C.

12.5 `app_agla.qsub` : appelants des scripts-shell

Rechercher pour chaque module de l'AGLA les modules qui lui font référence.

Résultats dans `/aster/adm/g_agla/resu/liste_appelant`.

Bug : ne marche pas pour des appels dynamiques (par variables).

12.5.1 Mode d'appel

```
qsub app_agla.qsub
```

12.5.2 Tâches réalisées

Pour chaque module de `/aster/adm/tool` recherche dans `/aster/adm/tool` et `/aster/outils` des modules qui leur font référence.

13 Outils pour *Aster*

13.1 Sauvegarde des sources sur IBM : limité à `NEW2`

13.1.1 `dispose_fort_agla` : sauvegarde du source FORTRAN

Copie des fichiers contenus dans `/aster/NEW2/bibfor` sur `ASTER.NEW2.FORT()`.

Attention :

Il faut mettre à jour le mot de passe du user IBM `GJMHHTS`.

Il est préférable d'envoyer la commande en batch.

13.1.2 `dispose_cat_agla` : sauvegarde du source CATALOGU

Copie des fichiers contenus dans `/aster/NEW2/catalo/`*`catalogue`* sur :

ASTER.NEW2.CATALO.*catalogue* ().

Attention :

Il faut mettre à jour le mot de passe du user IBM GJMHHTS .

Il est préférable d'envoyer la commande en batch.

13.2 Sauvegarde des sources sur cassette Sun

13.3 aclaut Contrôles d'accès de fichiers et répertoires

Utilisation des fonctionnalités d'Acces Control List (ACL) de l'environnement Multilevel Security (MLS) du système UNICOS 7.0.

Avec l'ACL il est possible de gérer le contrôle d'accès à des fichiers et des répertoires avec d'autres outils que les permissions read, write, execut pour user, group ou other.

En étant propriétaire d'un fichier il est possible d'invalider ou de rajouter l'une de ces permissions à un user ou un ensemble de user, sans avoir à modifier leur appartenance à un group (ce qui est réservé au super-utilisateur).

La commande aclaut permet d'appliquer à un ou plusieurs fichiers ou répertoires ce contrôle d'accès.

13.3.1 Mode d'appel

Se positionner dans le répertoire /aster/agla/acl
```
cd /aster/agla/acl
aclaut nom_fichier_aut
```

Le fichier aut est un fichier de type texte où l'on donne :

- la liste des fichiers ou répertoires sur lesquels on doit appliquer un contrôle d'accès,
- la liste des autorisations ou interdictions.

Les lignes blanches ou débutant par % sont considérées comme des commentaires.

Les lignes donnant les fichiers d'application sont de la forme :

```
FIC>nom_de_fichier_ou_repertoire
```

Un seul nom de fichier par ligne. Si l'on donne un nom de répertoire, le contrôle d'accès sera appliqué récursivement à tous les fichiers et répertoires qu'il contient.

Les lignes donnant les autorisations ou interdiction ont de la forme :

```
ACL>autorisation_au format_acl
```

Une seule autorisation par ligne.

Exemple de fichier `aut` pour ne donner l'accès qu'aux utilisateurs `gjbhhts` et `j2bhhmb` (en dehors du propriétaire `d6bhhhh`) aux fichiers sources de la version `NEW2` d'*Aster*.

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
        % Fichier de description du controle d'acces aux
sources de NEW2
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

        %----------------------------------------------------
        % Liste des fichiers dont il faut contrôler l'acces
        %----------------------------------------------------
        % Un par ligne precede de FIC>
        % Si l'on donne un répertoire le controle d'acces est
applique
        % recursivement a toutes ses entrees.
        FIC>/aster/NEW2
        FIC>/aster/astest/NEW2

        %----------------------------------
        % Description du controle par user
        %----------------------------------
        % Un controle d'acces par ligne, sous la forme
        % ACL>a: user : group : permissions_d_acces
        % permissions d'acces : r read, w write, x execut, n
none

        % Interdire tous les users
        ACL>a: * :   : n

        % Autoriser les users suivants :
        ACL>a: gjbhhts : * : rx :
        ACL>a: j2bhhmb : * : rx :
```

13.3.2 Tâches réalisées

Vérification de l'existence du fichier *`fichier_aut`*.

Construction d'un fichier `/aster/agla/acl/`*`fichier_aut`*`.acl` par la récupération des ordres de contrôle `ACL` et l'utilisation de la commande `spacl`.

Appliquer récursivement à tous les fichiers et répertoires demandés ce fichier `aut` par la commande `spset`.

Conserver la trace de ces commandes dans le fichier `/aster/agla/acl/`*`fichier_aut`*`.log`.

Page laissée intentionnellement blanche.