Comprendre le listing de l'outil `asverif / asrest`

Résumé :

L'outil (`asverif / asrest`) de vérification des sources (avant leur intégration dans les sources officiels) produit un listing parfois long et complexe à comprendre. Le but de ce document est de décrire la structure générale du listing ainsi que certains détails qui peuvent dérouter l'utilisateur de cet outil.

Table des matières

1 Structure générale du listing

L'outil (que nous appellerons plus simplement `asverif` par la suite) est en réalité une suite d'outils élémentaires qui sont enchaînés les uns aux autres. Chaque outil élémentaire a pour fonction de contrôler une partie des sources (Fortran, catalogues, tests, ...). Le listing produit par `asverif` est donc la concaténation des listings produits par ces outils élémentaires.

L'enchaînement des outils élémentaires n'est pas totalement arbitraire. Par exemple, l'édition des liens ne peut se faire avant la compilation.

Parfois, certaines erreurs détectées par l'outil l'empêchent de poursuivre l'analyse ; par exemple, si le fortran modifié (ou ajouté ou supprimé) est inacceptable (par l'outil), il n'y aura pas de compilation des catalogues car cette compilation doit se faire avec l'exécutable aster qui dépend du fortran modifié. Le listing sera alors tronqué.

Dans l'exemple suivant, nous avons essayé d'enchaîner le plus d'outils élémentaires possibles. Pour cela, nous avons utilisé `asverif` sur une "surcharge" du code comprenant :

- des routines `fortran`
- des catalogues de commande
- des catalogues d'éléments
- des sources `python`
- des suppressions de source (`unigest`)

Le listing aura alors la forme générale suivante :

```
=== Vérification du code source de J.PELLET (vabhhts)
================================================================================
...   <VERIF_1> (vérification du fichier "histor")

*** Vérification du source FORTRAN
************************************************************
...   <VERIF_2>

*** Vérification du source CATALOGU
************************************************************
...   <VERIF_3>

*** Vérification du source CATALOPY
************************************************************
...   <VERIF_4>

*** Vérification du source PYTHON
************************************************************
...   <VERIF_5>

*** Vérification du source UNIGEST
************************************************************
...   <VERIF_6>

=== Restitution du code source de J.PELLET (vabhhts)
================================================================
...   <VERIF_7>

=== Compilation des sources FORTRAN en majobj                 ===
================================================================
...   <VERIF_8>  (compilation f90 + outil CRS)
```

```
    Compilation des sources FORTRAN avec /aster/outils/g77.csh
================================================================
...   <VERIF_9>  (compilation g77)

=== Vérification des sources FORTRAN avec CRP                ===
================================================================
...   <VERIF_10>

=== Edition des liens des sources FORTRAN C et CAL           ===
================================================================
...   <VERIF_11>

=== Construction du catalogue de commandes python
================================================================
...   <VERIF_12>

=== Compilation du catalogue d'éléments
================================================================
...   <VERIF_13>
--- Fin de asverif.exesh avec le code de retour : 4
----------------------------------------------------------
--- CODE RETOUR = 4
----------------------------------------------------------
----------------------------------------------------------
--- DIAGNOSTIC JOB : ERREUR_AGLA
----------------------------------------------------------
```

Chaque partie du listing (dénommée ici " <VERIF_n> ") est produite par un outil élémentaire. Elle est introduite par une chaîne de caractères qui lui est propre. Par exemple, la vérification du fichier "histor" (<VERIF_1>) est introduite par la chaîne "=== Vérification du code source de J.PELLET (vabhhts)"

Dans les paragraphes suivants, nous allons revenir sur certaines parties de ce listing, celles concernant les sources FORTRAN.

2 Que faut-il lire dans ce listing ?

Globalement, le listing comporte beaucoup d'impressions inutiles (celles qui disent que "tout va bien"). Ces informations ne facilitent pas la tâche de lecture, mais elles permettent de vérifier que les sources pris en compte sont bien les bons.

Donc, il ne faut pas lire tout le listing ! Les informations à ne pas manquer sont les erreurs fatales et les alarmes.

Les erreurs fatales (celles qui rendront impossible la restitution) commencent toutes par la chaîne "(F/U-". Par exemple :

```
(F/U-015-bis): Le fichier /cluster/members/member2/tmp/519552/vabhhts/histor
n'est pas conforme au canevas
(F/U-h01b) : les clés documentaires X0.00.00 sont interdites.
(F/U-046): acou_face3 : SUPPRESSION SOUS-CATALOGUE IMPOSSIBLE
(F/U-091): ERROR(S) et WARNING(S) FORTRAN (détecté par g77):
```

Les alarmes commencent toutes par la chaîne "(A/U-". Par exemple :

```
(A/U-040): MODIF OK de : calculel/calcul.f
```

```
(A/U-558): grandeur_simple__ : modification d'une unité dont le responsable
est : JMBHH01
(A/U-043): debcal : FORSUPPR ROUTINE OK
(A/U-044): sslx101a : SUPPRESSION CAS-TEST OK
```

Ces exemples montrent que malheureusement, ces “alarmes” ne sont pas toujours des alarmes !

Pour aider l’utilisateur à s’y retrouver, l’outil `asverif` fait un récapitulatif des alarmes ayant entraîné un code retour non nul. Attention : ce récapitulatif n’est produit que si aucune erreur fatale n’est détectée.

Exemple :

```
########################################################################
### Récapitulatif des Alarmes entraînant un code retour = 2 :
(A/U-h10) :  Modification du document : U4.02.01

(A/U-222): calcul : Modification de la carte TOLE
           Les erreurs précédées d'un A ont été ajoutes dans la carte
           Les erreurs précédées d'un S ont été supprimées de la carte
A CRP_20 CALCUL
########################################################################
```

2.1 Résumé

Lorsque l’on utilise un des outils de `l’agla (asverif / asrest / pre_eda)`, il faut d’abord rechercher la chaîne `"(F/U"` dans le listing. Cette chaîne “pointe” les erreurs fatales. Lorsque toutes les erreurs fatales sont corrigées, il faut alors regarder le récapitulatif des Alarmes.

Remarque :

> *Depuis quelques temps, une nouvelle vérification est faite concernant la cohérence entre les identifieurs de message utilisés dans les sources (par exemple : `ALGELINE_12`) et la présence de ces messages dans les catalogues de messages `(bibpyt/Messages/*.py)`. Cette vérification (en fin de listing) n’émet malheureusement de message `"(F/U"` dans le listing.*

Les problèmes détectés : message “manquant” ou message “inutilisé” apparaissent sous la forme suivante dans le listing :

```
Messages inutilisés :
CALCULEL4_4 PREPOST6_37
Messages inexistants :
PREPOST6_36
```

3 Problèmes liés au fortran

Les sources fortran sont ceux qui posent le plus de problèmes aux utilisateurs de l'agla. La raison en est la multiplicité des outils qui leurs sont appliqués :

- script de contrôle du fortran
- compilateur(s)
- outil de vérification de la norme ANSI77 (CRS)
- outil d'intercompilation (CRP)
- édition de liens

Nous allons illustrer le listing de ces outils.

3.1 Script de contrôle du fortran

Ce script est exécuté au début de l'outil. Son listing se trouve dans la partie <VERIF_2>. Plusieurs vérifications sont effectuées :

- que le source ressemble à du fortran
- que le source ne contient pas de caractères interdits (ou déconseillés) ou de lignes trop longues
- que le source provient bien de la dernière version officielle (vérification de la carte "C MODIF")
- que l'utilisateur n'a pas modifié les cartes "C TOLE" (on comprendra cette phrase après avoir lu le paragraphe concernant l'outil CRS)

```
--- Vérification que le source est de type FORTRAN

/tmp/519797/aster/eda/vabhhts/identifi/fortran contient du source FORTRAN

--- Vérification que le source ne contient pas
--- de caractères génants
  17/-:C WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF  MERCHANTAB
  17/+:ILITY OR FITNESS FOR A PARTICULAR PURPOSE. SEE THE GNU  GENERAL PUBLIC
LICENSE FOR MORE DETAILS.


(A/U-014): Les lignes précédentes (du fichier
/tmp/519797/aster/eda/vabhhts/identifi/fortran)
        : sont précédées de leur numéro dans le fichier et d'un diagnostic :
            num_ligne/-: les 72 premiers caractères d'une ligne (ERREUR FATALE)
            num_ligne/+: la suite de la ligne précédente        (ERREUR FATALE)
            num_ligne/!: contient des caractères déconseillés   (ALERTE)
            num_ligne/?: contient des caractères INTERDITS      (ERREUR FATALE)


(F/U-014): Le fichier /tmp/519797/aster/eda/vabhhts/identifi/fortran contient :
         :     des caractères INTERDITS
         :     et/ou des lignes TROP LONGUES

--- Vérification de cohérence avec la version de référence (NEW7)

(A/U-222): calcul : Modification de la carte TOLE
           Les erreurs précédées d'un A ont été ajoutes dans la carte
           Les erreurs précédées d'un S ont été supprimées de la carte
A CRP_20 CALCUL
```

3.2 Compilateur(s)

Seules les erreurs fatales du compilateur sont à prendre en compte. Les éventuels "warnings" ne sont pas interceptés par l'agla. Sur la machine actuellement utilisée par l'agla (en Août 2003 : l'alphaserver), on compile 2 fois les sources à l'aide de 2 compilateurs différents f90 (<VERIF_8>) et g77 (<VERIF_9>).

extrait de <VERIF_8> :

```
================================================================
=== Compilation des sources FORTRAN en majobj                ===
================================================================

(F/U-090): ERROR(S) et WARNING(S) FORTRAN :
+ f90 -v -c -fast -i8 -r8 -omp /tmp/519797/tmp_asverif/bibfor/calculel/ ...
f90: Warning: /tmp/519797/tmp_asverif/bibfor/calculel/calcul.f, line 268:
     Variable IACHIX is used before its value has been defined
            EXICH = ((IPARIN.GT.0) .AND. ZL(IACHIX-1+IPARIN))
-----------------------------------------^
```

Remarque :

asverif s'appuie sur le code retour de la commande de lancement de l'outil de compilation pour émettre le message d'erreur ((F/U-090) : ERROR(S) et WARNING(S) FORTRAN). Il faut donc ensuite consulter les messages émis par l'outil de compilation.

3.3 Outil de vérification de la norme ANSI77 (CRS)

L'outil CRS a été récupéré dans le domaine public (il s'agit en réalité d'une version à peine modifiée de l'outil ftncheck). L'exécutable de cet outil produit un listing (dans <VERIF_8>) pouvant contenir un grand nombre de types d'erreurs ou de warnings. Chaque type de message est précédé d'un label de la forme CRS_nnn ; par exemple :

```
CRS_206 routine CALCUL "/tmp/519808/tmp_asverif/bibfor/calculel/calcul.f" ,
        line 93 col 15: Warning: non standard:(declaration hors norme:
REAL*8,...)

CRS_513 routine CALCUL : Warning:Variables used before set: IACHIX
```

Le "piège" de cet outil est que dans les messages imprimés (en anglais), il utilise les mots "error" et "warning" alors que pour les outils de l'agla ces mots ne sont pas forcément synonymes d'erreur fatale ou d'alarme.

Il y a des "Warnings" CRS qui entraînent des erreurs fatales de l'agla et des "errors" CRS sans importance !!!

Le principe de l'utilisation de CRS dans l'agla est le suivant :

- On exécute le programme CRS et on récupère son listing : l1
- On regarde si dans le listing l1, il existe des lignes commençant par des labels "CRS_nnn" recherchés par l'agla (remarque : certains messages CRS_nnn sont totalement ignorés ; par exemple ceux concernant les types hors norme REAL*8 ou COMPLEX*16).
- Dans les lignes retenues, on retire les erreurs CRS_nnn qui sont tolérées (mécanisme du "C TOLE CRS_nnn" décrit ci-dessous).

- S'il reste des lignes, on imprime le texte suivant :
- ==⇒ Compilation refusée pour le Code_Aster a cause des erreurs FORTRAN CRS suivantes (sans carte 'C TOLE') :
- CRS_513 CALCUL
- Puis on imprime le listing brut l1

3.3.1 Mécanisme du "C TOLE CRS_nnn"

L'agla considère que certaines "erreurs" de CRS peuvent souffrir d'exceptions. Par exemple, l'interdiction de l'équivalence entre variables numériques et de type CHARACTER rend impossible l'usage de JEVEUX. Pour qu'une exception soit admise par l'outil, l'utilisateur doit indiquer dans son source qu'il viole délibérément la règle de programmation. Pour cela, il écrira par exemple dans une de ses routines : "C TOLE CRS_513"

3.3.2 Résumé

Pour exploiter le résultat de l'outil CRS, il faut rechercher la chaîne "==⇒ Compilation refusée ... " Si elle est présente, il ne faut regarder que les messages CRS_nnn qui sont indiqués au dessous.

3.4 Outil d'inter-compilation (CRP)

L'outil CRP fonctionne de la même manière que l'outil CRS. Il accepte également des exceptions gérées par les cartes "C TOLE CRP_nnn".

La spécificité de cet outil est de vérifier l'inter-compilation des sources fortran. C'est-à-dire par exemple:

- vérifier que le nombre et le type des arguments lors d'un appel à une procédure sont les mêmes que ceux définis lors de la définition de la procédure,
- vérifier la cohérence des différentes occurrences d'un COMMON,
- …

```
Exemple de listing de CRP  (<VERIF_10>) :
================================================================
=== Vérification des sources FORTRAN avec CRP                ===
================================================================
(F/U-222): Erreur(s) détectée(s) par crp :
===> Erreurs crp et inter-compilation :
CRP_150 CALCUL
CRP_3 CALCUL
```