Manuel de Descriptif Informatique
Fascicule D6.04 :
Document D6.04.01

Utilitaires d'impression de messages

Résumé

Ce document complète les règles concernant les Entrées/Sorties [D2.07.01], il décrit les routines (paquet "UTMESS") permettant de réaliser de "brèves" écritures de message.

Table des matières

1 Types de messages

Les unités logiques d'impression sont établies en début d'exécution du travail, par le superviseur, les impressions émises par les routines décrites dans les paragraphes suivants seront effectuées sur ces unités logiques sans possibilité de redirection.

Les messages émis seront dirigés uniquement en fonction de leur type :

Code	Type de message	Fichiers de sortie
F	message d'erreur fatale, l'exécution s'arrête après diverses impressions. Les concepts créés au cours de l'exécution sont perdus.	ERREUR MESSAGE RESULTAT
E	message d'erreur, l'exécution continue (un peu cf. [§1.2]).	ERREUR MESSAGE RESULTAT
S	message d'erreur, les concepts créés au cours de l'exécution sont validés par le superviseur, l'exécution s'arrête.	ERREUR MESSAGE RESULTAT
A	message d'alarme.	MESSAGE RESULTAT
I	message d'information du superviseur (cf. [§1.5]).	MESSAGE

1.1 Les messages de type F

Ce type de message est suivi d'un arrêt immédiat de l'application, il est utilisé dans le cadre de la détection d'erreur grave ne pouvant permettre la poursuite normale d'une commande *Aster*.

L'émission d'un message d'erreur `F` provoquera l'arrêt par `CALL JEFINI ('ERREUR')`, cet appel déclenche l'impression du contenu de la mémoire et des répertoires associés aux différentes bases, de plus, un appel à `JXVERI` est effectué pour contrôler le chaînage des segments de valeurs.

Le Superviseur ajoute devant le message émis une impression contenant la date et le numéro de la version.

Remarque :

- *les concepts créés au cours de l'exécution ne sont pas validés par le Superviseur, les objets `JEVEUX` y étant attachés pouvant être partiellement remplis ou écrasés, la base `GLOBALE` n'est pas sauvegardée (dans ce cas, l'interface `asterix` ne la recopie pas vers le répertoire utilisateur),*
- *Il est fait appel à une routine système (`CALL ABORT( )`) provocant la remontée d'erreur : impression des noms des routines appelantes et, suivant le type de compilation, contenu des variables de ces routines.*

1.2 Les messages de type E

Ce type de message permet d'analyser une série d'erreurs avant l'arrêt du programme. Par exemple, l'analyse syntaxique du fichier de commandes par le Superviseur ou l'analyse du fichier de maillage par la commande LIRE_MAILLAGE.

L'émetteur d'un message de type E **devra émettre un message de type** F à la fin de son analyse, à moins qu'une des routines appelante l'émette à sa place (paramètre d'erreur d'un OPxxxx rendu non nul au superviseur)

1.3 Les messages de type S

Ce type de message provoque un arrêt immédiat de l'application avec validation des concepts créés lors de l'exécution et fermeture "propre" de la base GLOBALE. Son emploi se trouve justifié dans des portions de programmation instrumentées pour mesurer le temps machine écoulé (cf. [D6.08.01] Mesure du temps CPU). Il permet de se prémunir d'un arrêt système au cours d'un processus itératif. Le développeur doit tout de même prendre quelques précautions afin de ne pas stocker des champs non valides, par exemple une solution non convergée qui pourrait être ensuite imprudemment utilisée comme solution initiale.

1.4 Les messages de type A

Les messages d'alarme de type A sont à éviter tant que cela est possible. En effet, il n'est pas recommandé d'inquiéter inutilement l'utilisateur du code, et il est préférable de ne pas remplir le fichier MESSAGE par des informations superflues. Le nombre de messages d'alarme est limité automatiquement à 5 messages **successifs** identiques.

Il est recommandé aux utilisateurs qui ont des messages de type A de "réparer" leur fichier de commandes pour les faire disparaitre.

1.5 Les messages de type I

Les messages d'information sont définis dans [D2.07.01 §6].

Rappelons que :

- ils sont commandés par le **mot clé 'IMPR'** des commandes, sauf ceux émis par le superviseur (qui est "au dessus" des commandes),
- ce ne sont **pas** les **résultats** de la commande, ni l'**écho des données** de l'utilisateur, ni des **alarmes** affaiblies,…
- ils font partie de la fourniture **contractuelle** d'*Aster*. On ne les change pas sans en parler en EDA et aux utilisateurs.

Ces messages **ne doivent pas être imprimés** par les routines UTMESS, … présentées dans ce document. **Seul le superviseur** peut utiliser UTMESS <I>.

Les commandes voulant émettre des messages d'information le feront avec des WRITE et les utilitaires décrits dans [D6.04.02] "utilitaires d'impression d'informations par IMPR des commandes".

Remarque :

> *Nous avons choisi d'imprimer ces messages avec des "WRITE" car cette méthode est plus souple que les UTMESS : il est plus facile de gérer les écritures en colonnes. De plus, chaque message n'est plus systématiquement précédé d'une ligne blanche et du nom de la commande (et de la routine) émétrice.*

2 Fonctionnement général

Deux méthodes sont proposées afin d'émettre un message :

- si le message ne comporte qu'un simple texte, on utilise UTMESS [§3.5],
- si le message comporte à la fois des textes et des valeurs, l'émission comporte trois étapes :
  - l'initialisation du message UTDEBM [§3.1],
  - le corps du message (répétable) [§3.3],
  - la fin du message [§3.2].

Les sous-programmes d'impression décrits ci-après doivent être appelés entre l'appel à un sous-programme d'initialisation UTDEBM et un sous-programme de terminaison UTFINM.

UTDEBM, par lequel on spécifie le type du message, imprime un entête normalisé indiquant ce type et identifiant l'émetteur du message.

Etant donnée la variété des situations cet émetteur n'est pas encore défini de manière normalisée. Le programmeur cherchera à définir l'émetteur afin que l'on puisse retrouver facilement où le message a été émis et en prévoyant que la classification par nom d'émetteur puisse servir à établir une documentation sur les messages d'erreurs. On propose de nommer cet émetteur par la chaîne de caractères formée par le nom du sous-programme appelant et un numéro d'ordre.

Exemple d'appel :

```
CALL UTDEBM ('F','NMELNL_02',' VALEUR DE NU NON TROUVEE')
```

Cet appel initie l'émission d'un message de type F, l'émetteur est identifié comme étant la routine NMELNL, c'est le deuxième message figurant dans le corps de la routine.

Les impressions sont effectuées par l'intermédiaire d'un tampon d'accumulation. La taille de ce tampon représente une dizaine de lignes d'impression. Il est imprimé chaque fois qu'il se trouve rempli.

Le sous-programme de terminaison UTFINM indique au logiciel que le message est terminé. Dans ce cas, le contenu courant du tampon est vidé.
Le tampon possède des tabulations permettant d'aligner les impressions, ces tabulations ainsi que la longueur en nombre de caractères par ligne sont définies dans le superviseur.

Quatre types de sous-programmes sont disponibles :

sous-programmes d'initialisation et de fin d'utilisation	UTDEBM UTFINM
sous-programmes d'impression de valeurs : • de type integer • de type real • de type complex • de type character • de type logical	 UTIMPI UTIMPR UTIMPC UTIMPK UTIMPB
sous-programmes de positionnement du message	UTPOSI UTSAUT
sous-programme particulier équivalent à l'appel de UTDEBM et de UTFINM	UTMESS

Remarque :

Toutes les routines sauf UTMESS doivent être appelées entre un UTDEBM et un UTFINM.

3 Sous-programmes à l'usage des développeurs

3.1 Sous-programme d'initialisation

```
SUBROUTINE UTDEBM ( ch1 , spg , texte )
```

in	ch1	K1	type de message (F , E , S , A ou I)
in	spg	K*	nom de l'émetteur
in	texte	K*	texte à imprimer

3.2 Sous-programme de fin d'impression

```
SUBROUTINE UTFINM ( )
```

Cette routine sans argument permet de déclarer la fin de l'impression et de vider le contenu courant du tampon.

Remarque :

L'appel à cette routine est obligatoire pour vider le tampon d'écriture.

3.2 Sous-programmes d'impression de valeurs

```
SUBROUTINE UTIMPx ( cara , kvar , nbval , val )
```

in	cara	K1	caractère d'édition, prend les valeurs suivantes : • S le texte et les valeurs sont mis à la suite dans le tampon, • L un retour à la ligne est effectué avant de placer le texte et les valeurs dans le tampon.
in	kvar	K*	texte
in	nbval	I	nombre de valeurs à imprimer
in	val	I R C K* L	vecteur de valeurs de longueur nbval à imprimer de type correspondant à x : • INTEGER pour UTIMPI • REAL pour UTIMPR • COMPLEX pour UTIMPC • CHARACTER pour UTIMPK • logical pour UTIMPB

3.3 Sous-programmes de positionnement dans le tampon

```
SUBROUTINE UTPOSI ( ipos )
```

in	ipos	I	position dans la ligne

Cette routine permet de positionner un texte ou une valeur à une colonne donnée.

```
SUBROUTINE UTSAUT ()
```

Cette routine sans argument permet de sauter une ligne.

3.4 Sous-programme de messages simples

```
SUBROUTINE UTMESS ( ch1 , spg , texte )
```

in	ch1	K1	type de message
in	spg	K*	nom de l'émetteur
in	texte	K*	texte à imprimer

Cette routine permet d'imprimer l'entête et un texte. L'entête est le même que celui écrit par UTDEBM. Ce sous-programme vide le tampon, un appel à UTFINM est donc inutile.

4 Exemples d'utilisations

Premier exemple

L'appel suivant :

```
    CALL UTMESS('F','NMDOME_05','IL Y A PLUSIEURS CHARGES'
&                //' THERMIQUES ')
```

Imprime le message suivant :

```
<F> <NMDOME_02> 'IL Y A PLUSIEURS CHARGES THERMIQUES'
```

Deuxième exemple

Les appels suivants :

```
    CALL UTDEBM('F','FOLOCA_02','ON DEBORDE A GAUCHE')
    CALL UTIMPR('L','   VALEUR A INTERPOLEE: ',1,X)
    CALL UTIMPR('L','      BORNE INFERIEURE: ',1,VAL(1))
    CALL UTFINM( )
```

Impriment les messages suivants :

```
<F> <FOLOCA_02> ON DEBORDE A GAUCHE
     VALEUR A INTERPOLEE: 1.2E-3
     BORNE INFERIEURE: 1.5E-3
```

Troisième exemple

Les appels suivants :

```
    CALL UTDEBM('S','OP0070','ARRET PAR MANQUE DE TEMPS CPU')
    CALL UTIMPI('S',' AU NUMERO D''ORDRE: ',1,NUMORD)
    CALL UTIMPR('L',' TEMPS MOYEN PAR INCREMENT DE CHARGE: ',1,
&       TPS1(4) )
    CALL UTIMPR('L',' TEMPS CPU RESTANT: ',1,TPS1(1))
    CALL UTFINM()
```

Impriment les messages suivants :

```
<S> <ASTER  2. 3.31 10/03/93 >
<S> <OP0070> ARRET PAR MANQUE DE TEMPS CPU AU NUMERO D'ORDRE:  2
     TEMPS MOYEN PAR INCREMENT DE CHARGE:  3.1856E+03
     TEMPS CPU RESTANT:    1.5562E+03
```

5 Description du sous-programme d'initialisation interne

Ce sous-programme est uniquement appelé par le Superviseur, il permet l'initialisation des tables internes de correspondance classe de message / numéro d'unité logique et des paramètres de gestion du tampon.

```
SUBROUTINE UTINIT ( nbfica , mcol , itb )
```

in	nbfica	I	nombre de fichiers d'alarme: - nbfica = 1 MESSAGE - nbfica = 2 MESSAGE et RESULTAT
in	mcol	I	nombre de colonnes utilisées pour l'impression (<133)
in	itb	I	pas des tabulations (prendre 1 par défaut)

Remarque :

- *le superviseur édite les messages sur 80 colonnes,*
- *cette routine fait appel à la fonction IUNIFI.*