Manuel de Descriptif Informatique
Fascicule D1.01 : Environnement de développement et outils
Document D1.01.02

Environnement de développement sur plate-forme autonome : run_aster

Résumé :

Ce document décrit l'environnement de développement sur plate-forme autonome (SOLARIS, HPUX, IRIX, linux ou Windows NT4). Il comporte notamment une notice d'utilisation de la procédure run_aster et la description des fichiers de paramètres associés.

1 Présentation générale et pré requis

1.1 Objectifs

La constitution d'une version locale et la surcharge du *Code_Aster*, outre les procédures classiques de compilation du code source (C et FORTRAN) et d'édition des liens, nécessite quelques opérations particulières comme la constitution des catalogues compilés associés au langage de commandes et à la description des éléments finis. Les procédures de validation par l'exécution d'une série de tests de références constituent un point de passage indispensable à la validation d'un développement, au moins pour s'assurer de la non régression.
Le souci d'assurer la portabilité sur différentes plates-formes et différents systèmes d'exploitation a conduit à écrire un ensemble de procédures présentant un minimum d'adhérence système. Le choix s'est orienté vers le langage de script tcl, ce qui nous assure une large diffusion et permet de toucher à la fois le monde unix et windows. L'appel de procédures n'étant pas vraiment convivial, il a semblé utile d'enrober leurs appels, sans pour autant développer une interface complexe. Nous avons donc choisi un moyen simple pour appeler de façon générique les différentes opérations de construction et d'utilisation d'un exécutable local en utilisant une unique commande et un fichier de paramètre à mots clés facilement éditable.
Ce document décrit les différentes opérations accessibles, la syntaxe du fichier de paramètre associé, ainsi que toute les actions préliminaires d'organisation des sources et librairies.

1.2 Pré requis

La machine cible doit permettre d'accéder directement ou par montage de type NFS aux sources, aux librairies et aux fichiers de tests, il n'est pas prévu d'accès distant de type `rsh`, `rcp` sous unix. Ces différents fichiers doivent être déposés sous une arborescence figée organisée (sous unix) de la façon suivante :

sources FORTRAN	bibfor/algeline
	bibfor/algorith
	bibfor/assembla
	bibfor/calculel
	bibfor/cataelem
	bibfor/elements
	bibfor/jeveux
	bibfor/modelisa
	bibfor/postrele
	bibfor/prepost
	bibfor/soustruct
	bibfor/stbtrias
	bibfor/supervis
	bibfor/utilifor
	bibfor/utilitai
	fermetur
sources C	bibc/algeline
	bibc/include
	bibc/supervis
	bibc/utilitai
sources CATALOGUE	catapy/commande
	catalo/compelem
	catalo/options
	catalo/typelem
sources Python	bibpyt/Accas
	bibpyt/Cata
	bibpyt/Lecture_Cata_Ele
	bibpyt/Superviseur
sources des tests	tests/

L'installation préalable du logiciel TCL/TK est nécessaire, tous les scripts de construction s'appuyant sur ce langage. D'autre part il est indispensable d'avoir procédé à l'installation de la version 2.1 de python. La machine doit disposer d'un compilateur FORTRAN (FORTRAN 77 pour le *Code_Aster*) et d'un compilateur C.

L'édition de liens requiert les librairies MED version 2.0 et HDF version 5.0 qui devront être construites au préalable à l'aide de la procédure d'installation fournie.

librairie MED	`libmed.a`
librairies HDF	`libhdf5.a`
librairie python	`libpython2.1.a`
librairies TclTk	`libtcl8.3.a libtk8.3.a libX11.a`
librairie BLAS *	`/usr/lib64/libcomplib.sgimath.so`
détection des signaux *	`lfpe`

* uniquement sous SGI – IRIX64

1.3 Description du résultat de la procédure d'installation

La procédure d'installation outre les fichiers sources décrits ci-dessus, dépose les scripts de construction et quelques répertoires sous le répertoire désigné lors de l'installation. Le fichier de configuration et les script sont adaptés en fonction des renseignements fournis (compilateurs utilisés, librairies, …).

scripts de lancement	`tcl/run_aster.tcl`
	`tcl/aster.tcl`
	`tcl/run`
	`tcl/run_aster`
	`tcl/conf/init_conf`
	`tcl/make_aster`
	`tcl/make_exec`
	`tcl/make_cata`
	`tcl/make_etude`
	`tcl/make_lib`
	`tcl/make_test`

répertoires de développement (utilisé par les scripts)	`dvp/`
	`dvp1/`
	`dvp2/`

La procédure `run_aster` est constituée de trois fichiers de commandes tcl (`aster.tcl`, `run_aster.tcl` et `run`) et d'un (ou plusieurs) fichier(s) de paramètres (par exemple `make_exec`, `make_cata`, `make_etude`...).
`aster.tcl` contient les différentes procédures tcl associées à chaque action.
`run_aster.tcl` contient la procédure d'analyse du fichier de paramètre et une procédure générique lançant les appels aux différentes actions.
`run` permet d'enrober l'appel des précédentes procédures directement depuis un shell script (unix ou windows) sans passer par une fenêtre Tcl (de type `wish`).
Le script `run_aster` permet de positionner l'environnement python.
Par ailleurs, il est indispensable de fournir un fichier de configuration décrivant l'ensemble des adhérences liées à la machine d'exécution (plates-formes SOLARIS, HP, IRIX, linux, ou NT). Ce fichier (`tcl/conf/init_conf`) permet d'indiquer les différents outils (compilateur, éditeur de liens, librairies, etc.), leurs chemin d'accès et toutes les options propres à la plate-forme.

L'exécution de `make_aster` permet de construire une version complète du code en mode nodebug ou debug et conduit à la création de l'arborescence suivante :

modules objets nodebug	obj/ obj_f/
modules objets debug	dbg/ dbg_f/
librairies nodebug	lib_obj/lib_aster.lib lib_obj/ferm.lib lib_obj/asterm.obj
librairies debug	lib_dbg/lib_aster.lib lib_dbg/ferm.lib lib_dbg/asterm.obj
exécutable nodebug	asteru.exe
exécutable debug	asterd.exe
catalogues compilés	elements commande/ cata_ele.pickled

2 Utilisation de la procédure

2.1 Généralités

Le script de lancement est enrobé dans un fichier exécutable (run_aster sous unix, run_aster.bat sous windows) appelant directement le shell script tcl. Ce script doit être lancé sur la machine d'exécution, éventuellement après connexion en rlogin (sous unix).

```
% run_aster mon_fichier_para
```

Le fichier de paramètres mon_fichier_para est analysé globalement :

- tout ce qui suit le caractère % est ignoré,
- les mots-clés sont tous en minuscules, leur ordre est indifférent, mais c'est la dernière occurrence dans le fichier qui est prise en compte,
- les mots-clés sont situés à gauche du caractère **:** (les blancs sont ignorés autour),
- il a deux types de mots-clés :
  - ceux servant à activer une tâche : make_aster, make_exec, make_cata, make_lib, make_test, make_etude, ... ils exécutent les différentes tâches dans un ordre figé (mot-clé **action**),
  - ceux servant à renseigner les fichiers et les différents paramètres (mot-clé **paramètre**),
- toute ligne ne contenant pas un mot-clé est ignorée,
- tous les noms de fichiers **doivent être indiqués en absolu**.

Les noms de fichier ou de répertoire fournis, le sont toujours en absolu (par exemple /home/utilisa/mon_fichier_de_donnees.dat).

2.2 Description du fichier de paramètres

Le symbole : sépare le mot-clé de la valeur associée. Les mot clés action sont tous de la forme make_**action** ou maj_**action**, ils prennent uniquement la valeur oui ou non. Les mots-clés paramètres dépendent du mot-clé **action**, ils sont décrits pour chaque commande au paragraphe [§2.4].
La syntaxe est commune à l'ensemble des mots-clés **action**, excepté pour le mot clé make_etude. Dans ce cas, pour renseigner les différents fichiers de données ou de résultats associés à une étude, le mot-clé est constitué du nom de l'unité logique suivi d'au moins l'une des lettres D ou R indiquant que le fichier est en donnée ou/et en résultat.
Exemple :

```
fort.20 DR : /home/user/toto.mail
```

Le mot clé rep_tcl est obligatoire dans tout fichier de configuration, il permet de charger les différentes procédures figurant dans le fichier aster.tcl.

2.3 Description du fichier de configuration

Ce fichier permet de définir les différents compilateurs, leur localisation, les paramètres utilisés, les librairies externes et les librairies système. C'est un fichier à mots-clés qui suit la syntaxe des fichiers de paramètres (on utilise le même analyseur). Il n'y a pas de vérification lors de l'analyse de ce fichier de configuration, aucun mot-clé n'est obligatoire, la valeur par défaut est mise à " " (chaîne réduite à un blanc).

Mot clé	Paramètre associé	
f77	nom de fichier	Compilateur FORTRAN 77
cc	nom de fichier	Compilateur C
lib	nom de fichier	Commande d'archivage des librairies
link	nom de fichier	Editeur de liens
python	Nom de fichier	Exécutable python (interpréteur de commandes python)
rep_incl	nom de répertoire	Contient les includes C ; les noms de fichiers doivent posséder l'extension .h
opcc	chaîne de caractères	Décrit les options de compilation C
opcc_dbg	chaîne de caractères	Décrit les options de compilation C en mode debug
opf77	chaîne de caractères	Décrit les options de compilation FORTRAN
opf77_dbg	chaîne de caractères	Décrit les options de compilation FORTRAN en mode debug
path_hdf	nom de répertoire	Contient les librairies HDF
path_med	nom de répertoire	Contient les librairies MED
op_link	chaîne de caractères	Décrit les options d'édition de liens
lib_sys	chaîne de caractères	Liste des librairies systèmes
lib_hdf	chaîne de caractères	Liste des librairies HDF
lib_med	chaîne de caractères	Liste des librairies MED

Le fichier de configuration associé à la plate-forme d'installation (init_conf) fait partie de la fourniture et est adapté lors de l'installation.

2.4 Description des commandes disponibles

On décrit ci-dessous l'ensemble des actions possibles, en précisant ce qui est attendu derrière chaque mot-clé. Les mots-clés doivent impérativement figurer dans le fichier de paramètres, ils peuvent éventuellement être affectés à la valeur " " (chaîne réduite à un caractère blanc).

Dans les tableaux ci-dessous, on met en caractères gras les paramètres qui sont susceptibles d'être modifiés à chaque exécution du script. En principe, les autres paramètres (type de plate-forme, localisation des sources / catalogues / librairies *Aster* de référence, des scripts tcl, des fichiers de configuration) sont inchangés d'une exécution à l'autre et peuvent donc être renseignés une fois pour toutes.

2.4.1 Construction initiale d'une version du *Code_Aster* : `make_aster`

Cette procédure permet de construire les librairies, un modules exécutables et les catalogues compilés de référence. Il est recommandé de construire la version optimisée et la version instrumentée pour le déboggage.

Mot clé	Paramètre associé	
make_aster	oui	
rep_tcl	nom de répertoire	Répertoire contenant les fichiers tcl fournis
fic_conf	nom de fichier	Fichier de configuration propre à la plate-forme utilisée
plate-forme	IRIX, HPUX, SOLARIS, P_LINUX ou PPRO_NT	Plate-forme cible (utilisée pour la compilation conditionnelle) IRIX : plate-forme SGI HPUX : plate-forme HP SOLARIS : plate-forme SUN P_LINUX : linux sur plate-forme x86 PPRO_NT : windows sur plate-forme x86
debug	debug ou nodebug	debug : les sources sont compilés en mode debug (-g) nodebug : les sources sont compilés avec le niveau d'optimisation précisé dans le fichier de configuration

2.4.2 Construction d'un exécutable - mot clé `make_exec`

Cette procédure permet de constituer un module exécutable, en surchargeant éventuellement les librairies de référence par des sources Fortran ou C " personnelles " (routines ajoutées, ou modification des routines de référence).

Mot clé	Paramètre associé	
make_exec	oui	
rep_tcl	nom de répertoire	Répertoire contenant les fichiers tcl fournis
fic_conf	nom de fichier	Fichier de configuration propre à la plate-forme utilisée
plate-forme	IRIX, HPUX, SOLARIS, P_LINUX ou PPRO_NT	Plate-forme cible (utilisée pour la compilation conditionnelle) IRIX : plate-forme SGI HPUX : plate-forme HP SOLARIS : plate-forme SUN P_LINUX : linux sur plate-forme x86 PPRO_NT : windows sur plate-forme x86
debug	debug ou nodebug	Debug : les sources sont compilés en mode debug (-g) nodebug : les sources sont compilés avec un certain niveau d'optimisation
rep_ref	nom de répertoire	Répertoire contenant les librairies et l'environnement d'exécution de référence (obtenu par make_aster)
source	liste de noms de répertoires	Répertoires contenant les fichiers FORTRAN et/ou C " personnels " avec lesquels on veut surcharger la version de référence. Les noms de fichiers doivent posséder l'extension .f ou .c . Les différents noms de répertoires doivent être séparés par le caractère | .
rep_obj	nom de répertoire	Répertoire contenant des modules objets utilisateur, les fichiers doivent posséder l'extension .o (unix) ou .obj (windows)
rep_lib	nom de répertoire	Répertoire contenant des librairies utilisateur, les fichiers doivent posséder l'extension .lib
exec	nom de fichier	Nom du fichier exécutable produit

Les mots clés `rep_obj` et `rep_lib` sont facultatifs.

2.4.3 Construction des catalogues compilés - mot clé make_cata

Cette procédure permet de construire les catalogues de commandes et d'éléments compilés à partir des catalogues de référence du *Code_Aster* et des fichiers catalogues sources (en complément ou en surcharge).
Les opérations de compilation de catalogue s'appuie sur la présence d'une carte utilisée lors de la gestion de configuration sur la machine de référence, il est donc indispensable de conserver cette carte lorsqu'elle existe et de l'ajouter dans les nouveaux catalogues.

Mot clé	Paramètre associé	
make_cata	oui	
rep_trav	nom de répertoire	répertoire utilisé lors de l'exécution du *Code_Aster* (pour compiler les catalogues) ; ce répertoire est normalement détruit en fin de procédure
plate-forme	IRIX, HPUX, SOLARIS, P_LINUX ou PPRO_NT	plate-forme cible (utilisée pour la compilation conditionnelle) IRIX : plate-forme SGI HPUX : plate-forme HP SOLARIS : plate-forme SUN P_LINUX : linux sur plate-forme x86 PPRO_NT : windows sur plate-forme x86
fic_conf	nom de fichier	fichier de configuration propre à la plate-forme utilisée
rep_tcl	nom de répertoire	répertoire contenant les fichiers tcl fournis
rep_catalo	nom de répertoire	répertoire contenant l'arborescence des sources des catalogues de référence
catalo	liste de noms de répertoires	répertoire de fichiers catalogues d'éléments "personnels " avec lesquels on veut surcharger les catalogues de référence. Pour les catalogues de commande, les noms des fichiers doivent posséder l'extension .capy et inclure la carte #& MODIF ou #& AJOUT. Pour les catalogues d'éléments, les noms des fichiers doivent posséder l'extension .cata et inclure la carte %& MODIF ou %& AJOUT. Les différents noms de répertoires doivent être séparés par le caractère | .
unigest	nom de fichier	Fichier utilisé lors de la gestion de configuration et dont on exploite les lignes CATSUPPR pour détruire les catalogues d'éléments correspondants [D1.02.01]
exec	nom de fichier	nom de l'exécutable aster utilisé pour compiler les catalogues (il doit exister)
rep_coco	nom de répertoire	catalogue de commandes compilées produit
elco	nom de fichier	catalogue d'éléments compilés produit

Remarque :

La compilation des catalogues d'éléments est une opération incrémentale qui s'appuie sur le fichier cata_ele_pickled *(construit par* make_aster*).*

2.4.4 Construction d'une librairie - mot clé make_lib

Cette procédure permet de construire une librairie personnelle à partir de fichiers sources (FORTRAN ou C).

Mot clé	Paramètre associé	
make_lib	oui	
debug	debug ou nodebug	debug : les source sont compilés en mode debug (-g) nodebug : les source sont compilés avec un certain niveau d'optimisation
plate-forme	IRIX, HPUX, SOLARIS, P_LINUX ou PPRO_NT	Plate-forme cible (utilisée pour la compilation conditionnelle) IRIX : plate-forme SGI HPUX : plate-forme HP SOLARIS : plate-forme SUN P_LINUX : linux sur plate-forme x86 PPRO_NT : windows sur plate-forme x86
fic_conf	nom de fichier	Fichier de configuration propre à la plate-forme utilisée
rep_tcl	nom de répertoire	Répertoire contenant les fichiers tcl fournis
source	liste de noms de répertoires	Répertoire contenant les fichiers FORTRAN et/ou C " personnels " avec lesquels on veut surcharger la version de référence. Les noms de fichiers doivent posséder l'extension .f ou .c (séparés par 1 "L")
rep_lib	nom de répertoire	Répertoire destinataire de la librairie produite
lib_aster	nom de la librairie	Nom de la librairie produite déposée sous rep_lib

2.4.5 Passage d'une liste de tests - mot clé make_test

Cette procédure permet de lancer une série de tests. Un test *Aster* est constitué d'un certain nombre de fichiers (dont a minima un fichier de commandes et un fichier de paramètres) dont les noms sont constitués à partir du nom du test et d'un suffixe. Les principaux suffixes sont les suivants :

Suffixe	Unité logique	Type de fichier
.comm	fort.1	commandes *Aster*
.mail	fort.20	maillage au format *Aster*
.mgib	fort.19	maillage au format Gibi
.msup	fort.19	maillage au format Ideas
.para	-	paramètres d'exécution
.mess	fort.6	fichier MESSAGE
.resu	fort.8	fichier RESULTAT
.erre	fort.9	fichier ERREUR
.27	fort.27	fichier en données associé pour ce test à l'unité logique 27

Le fichier contenant la liste des tests doit contenir un nom de test par ligne, tout ce qui suit le caractère % est ignoré.

Mot clé	Paramètre associé	
make_test	oui	
rep_tcl	nom de répertoire	Répertoire contenant les fichiers tcl fournis
rep_trav	nom de répertoire	Répertoire utilisé lors de l'exécution du *Code_Aster* (pour compiler les catalogues) ; ce répertoire est normalement détruit en fin de procédure
liste_test	nom de fichier	Liste des tests à lancer
rep_test	nom de répertoire	Répertoire des fichiers de tests
rep_ref	nom de répertoire	Répertoire de référence des sources
rep_py	nom de répertoire	Sources python surchargeant l'environnement d'exécution
exec	nom de fichier	Nom de l'exécutable utilisé
rep_coco	nom de répertoire	Catalogue de commandes compilées utilisé
elco	nom de fichier	Catalogue d'éléments compilés utilisé

2.4.6 Passage d'une étude - mot clé make_etude

Cette procédure permet de lancer une étude. Une étude *Aster* est constituée d'un certain nombre de fichiers auxquels il faut associer une unité logique et un indicateur D ou R pour préciser si le fichier ou le répertoire est en données et/ou en résultats. Les paramètres d'exécution sont passés globalement derrière le mot clé para (la liste peut-être obtenue en lançant l'exécutable avec le paramètre -h ou -help).
Si la valeur du mot clé make_etude est positionnée à non, le répertoire d'exécution est préparé avec les catalogues compilés, l'environnement python et les fichiers de données de façon à pouvoir y lancer le debugger. Lors de l'exécution de la procédure, la ligne de commande de lancement de l'exécutable *Code_Aster* est affichée avec tous les paramètres nécessaires.

Mot clé	Paramètre associé	
make_etude	oui ou non	oui : l'exécution du Code_aster sera lancée après recopie de l'environnement et des données dans le répertoire de travail. non : le répertoire de travail est préparé afin de pouvoir lancer une exécution avec le debugger.
rep_tcl	nom de répertoire	Répertoire contenant les fichiers tcl fournis
rep_trav	nom de répertoire	Répertoire utilisé lors de l'exécution du *Code_Aster* (pour compiler les catalogues) ; ce répertoire est normalement détruit en fin de procédure
exec	nom de fichier	Nom de l'exécutable utilisé
rep_coco	nom de répertoire	Contient le catalogue de commandes compilées utilisé
elco	nom de fichier	Catalogue d'éléments compilés utilisé
base	nom de répertoire	Répertoire d'accueil de la base Aster (données et résultats)
d_base	nom de répertoire	Répertoire d'accueil de la base Aster (données uniquement)
r_base	nom de répertoire	Répertoire d'accueil de la base Aster (résultats uniquement)
d_ensi	nom de répertoire	Répertoire de fichiers au format Ensight en données
r_ensi	nom de répertoire	Répertoire de fichiers au format Ensight en résultats
para	chaîne de caractère	Liste des paramètres d'exécution

Les différentes unités logiques sont indiquées sous la forme suivante :
fort.ij DR : nom_de_fichier
où ij désigne le numéro d'unité associée, au moins l'un des indicateurs D ou R doit être présent.

Parmi les paramètres d'exécution, il est indispensable de fournir la taille de la zone mémoire gérée dynamiquement lors de l'exécution, l'unité utilisée est le méga mot d'entier.
-memjeveux 16 correspond donc à 64 méga octets sur une plate-forme 32 bits.

2.5 Fichiers de paramètres livrés lors de l'installation

Plusieurs fichiers de paramètres directement utilisables sont livrés sous le répertoire tcl lors de l'installation, ils ont été automatiquement adaptés à la configuration de l'utilisateur et peuvent être immédiatement passés en paramètre de la procédure run_aster.

make_aster	permet de construire une version complète en mode debug ou non des librairies, de l'exécutable et des catalogues compilés.
make_exec	permet de construire un exécutable à partir des fichiers sources FORTRAN et C (extensions .f ou .c) déposés sous les répertoires dvp1/ et dvp2/, le résultat est déposé sous dvp/asteru.exe
make_cata	permet de construire les catalogues de commandes et d'éléments compilés à partir des fichiers sources catalogue (extension .cata et .capy) déposés sous le répertoire dvp/, les résultats sont déposés sous dvp/commande et dvp/elements
make_etude	permet d'effectuer une exécution du *Code_Aster* avec les fichiers asteru.exe, commande et elements avec les données déposées sous etude/

`make_test` permet de lancer l'exécution des déposés sous le répertoire `/etude` avec les fichiers `asteru.exe`, `commande` et `elements`. La liste des tests utilisée est déposée dans le fichier `etude/liste_ct`.

`make_lib` permet de construire une librairie à partir de sources situés sous `/dvp`, le résultat est déposé sous `/dvp/lib`

Ces fichiers peuvent être recopiés (comme exemples) puis modifiés.

3 Exemple d'utilisation

3.1 Fichier de configuration sur plate-forme x86 sous linux

```
% Fichier de configuration associe a aster.tcl
%
% version PPRO_linux avec superviseur python
% ==================
%
f77  : /usr/bin/g77
cc   : /usr/bin/gcc
lib  : /usr/bin/ar -rv
link : /usr/bin/g77

python     : /usr/local/bin/python2.1

rep_incl  : /home/aster/STA6.2/include
path_hdf  : /home/aster/MED2.0/hdf/lib
path_med  : /home/aster/MED2.0/lib

opcc     : -c    -I /usr/local/include/python2.1
opcc_dbg : -c -g -I /usr/local/include/python2.1

opf77     : -c -O3
opf77_dbg : -c -g

op_link : -Xlinker -export-dynamic -lieee -ldl -lpthread -lutil
lib_sys : -L/usr/local/lib/python2.1/config -lpython2.1 -L/usr/local/lib -ltk8.3 -ltcl8.3 -lX11
lib_hdf : /home/aster/MED2.0/hdf/lib/libhdf5.a -lm -lz
lib_med : /home/aster/MED2.0/lib/libmed.a
```

3.2 Fichier de paramètres utilisé pour construire un exécutable sous linux

```
% parametres obligatoires :
% ==========================
rep_tcl     : /home/aster/STA6.2/tcl

% construction d'un exécutable surchargeant des librairies :
% ============================================================
fic_conf    : /home/aster/STA6.2/tcl/conf/init_conf
make_exec   : oui
source      : /home/aster/STA6.2/dvp
rep_ref     : /home/aster/STA6.2
exec        : /home/aster/STA6.2/dvp/asteru.exe
debug       : nodebug
plate-forme : P_LINUX
```

3.3 Fichier de paramètres utilisé pour effectuer une exécution sous linux

```
parametres obligatoires :
==========================
rep_tcl     : /home/aster/STA6.2/tcl

 construction d'un exécutable surchargeant des librairies :
============================================================
fic_conf    : /home/aster/STA6.2/tcl/conf/init_conf

make_etude  : oui
rep_ref     : /home/aster/STA6.2
rep_trav    : /tmp/trav_aster
rep_coco    : /home/aster/STA6.2/commande
elco        : /home/aster/STA6.2/elements
exec        : /home/aster/STA6.2/dvp/asteru.exe

para        : -memjeveux 16 -rep none
fort.1   D  : /home/aster/STA6.2/tests/ttnl100a.comm
fort.19  D  : /home/aster/STA6.2/tests/ttnl100a.mgib
fort.6   R  : /home/aster/STA6.2/tests/ttnl100a.mess
fort.8   R  : /home/aster/STA6.2/tests/ttnl100a.resu
fort.9   R  : /home/aster/STA6.2/tests/ttnl100a.erre
r_base      : /home/aster/BASE
plate-forme : P_LINUX
```

3.4 Exemple d'appel

```
% run_aster make_exec

% run_aster make_etude
```

Page laissée intentionnellement blanche.