Procedure DEBUT

Usage

Usage affects Memory, Disc and files.

The execution of a command file consists of a set of commands starting with DEBUT and ending in FIN [U4.11.02], (also see procedure PORSUIT [U4.11.03])

When the command DEBUT is read by Supervisor, it carries out the following tasks:

- Definition of the characteristics of databases (managed by JEVEUX) and assignment of the associated files,
- Reading of the catalogues of the elements and the commands.

User should not worry about the apparent complex syntax of this procedure as a simple default call of DEBUT() is sufficient in most cases.

Operands for the procedure DEBUT are to be used in the study case which require a more important size of the database or to divert the various files on number of logical unit different from the numbers provided by default.

Any commands placed before DEBUT are ignored if they are syntactically correct.

Table of Contents

Syntax

```
DEBUT(
      ◇PAR_LOT        = / 'OUI',                              [DEFAULT]
                        / 'NON',
      ◇IMPR_MACRO     = / 'NON',                              [DEFAULT]
                        / 'OUI',
      ◇LANG = Lang,                                           [TXM]
      ◇BASE = _F ( ◆FICHIER = / 'GLOBALE',
                              / 'VOLATILE',
                              /| LONG_ENRE = lenr,            [I]
                               | NMAX_ENRE = nenr,            [I]
                               | LONG_REPE = lrep,            [I]
                 ),
      ◇CODE = _F ( ◆NIV_PUB_WEB = / 'INTERNET',
                                  / 'INTRANET',
                    ◇VISU_EFICAS = / 'OUI',                   [DEFAULT]
                                  / 'NON',
                 ),
      ◇ERREUR = _F ( ERREUR_F = / 'ABORT',                    [DEFAULT]
                               / 'EXCEPTION',
                   ),
      ◇IGNORE_ALARM = l_vale,                                 [l_Kn]
      ◇DEBUG = _F (  JXVERI =          / 'OUI',
                                       / 'NON',
                      ENVIMA =           'TEST',              [l_Kn]
                     ◇JEVEUX =         / 'OUI',
                                       / 'NON',
                     ◇SDVERI =         / 'OUI',
                                       / 'NON',
                     ◇HIST_ETAPE =     / 'NON',
                                       / 'OUI',
                   ),
      ◇MESURE_TEMPS = _F ( ◇NIVE_DETAIL = / 0,                [DEFAULT]
                                          / 2, /1, /3
                           ◇MOYENNE = / 'NON',                [DEFAULT]
                                      / 'OUI',
                         ),
      ◇MOMOIRE = _F ( ◇TAILLE_BLOC = / 800.,                  [DEFAULT]
                                     = / tbloc,               [R]
                    ),
      ◇CATALOGUE = _F ( ◆FICHIER = nfic,                      [l_Kn]
                        ◇UNITE = unit,                        [I]
```

```
                    ),
  ◇RESERVE_CPU = _F ( /VALE = vale,                    [R]
                      /POURCENTAGE = pcent,            [R]
                      ◇BORNE = / bv,                   [R]
                               / 180,                  [DEFAULT]
                     ),
  );
```

Output

At the beginning of the execution of Code_Aster, a heading is displayed. Following is found in there:

- Version, Model number and date of last amendments of Code_Aster,
- Date and hour of the beginning of the execution,
- Name, Architecture and the operating system of the machine,
- Language used for the display of the messages,
- Type of parallelism available (MPI/OpenMP), the number of allocated processors,
- Version of the libraries used (when it is available) for hdf5, med, mumps, tape,
- Several information on the distribution of the memory.

For example:

```
Limiting memory for the execution: 256.00 Mo
consumed by initialization: 148.68 Mo
by the objects of the command set: 17.48 Mo
remains for the dynamic allocation: 89.84 Mo
Limiting Taille of the files of exchange: 48.00 Go
```

This means:

- `256 Mo` is the quantity of memory required by the user, it is the total quantity which should not be exceed.
- `148.68 Mo` is consumed simply by booting the execution (loading of executable, the assocated dynamic libraries, etc)
- `17.48 Mo` is consumed by reading of the command file (**Note**: in mode `PAR_LOT='NON'`, the command set being read progressively, this value will be null)
- `89.84 Mo` is the quantity of memory available (at this time) for the objects of computation (which is equal to `256 - 148.68 - 17.48`). If this value is too low, computation cannot begin.

During the execution, according to the dynamic allocations carried out, when this value varies more than 10% (upwards or downwards) a message, shown below, informs the user:

```
The memory currently consumed except JEVEUX is of 214.08 Mo.
The limit of dynamic allocation JEVEUX is fixed at 41.92 Mo.
```

At the end of execution, an assessment indicates if same computation can be started again with

less memory:

```
The memory requested from launching is over-estimated, it is of
256 Mo.
The peak report used is of 216.02 Mo.
```

or so more memory is necessary (depending on the platforms, the maximum limit can be exceeded without the system stopping computation):

```
The memory requested from launching is underestimated, it is of
256 Mo.
The peak report used is of 273.22 Mo.
```

Operands

Operand PAR_LOT

PAR_LOT =

Mode of processing of the commands:

'OUI': (default choice); the supervisor analyses all the commands before asking for the execution of it

'NON': after having analysed a command the supervisor asks for execution, then passes to the analysis (and the execution) of the following command (processing order by command)

Keyword IMPR_MACRO

IMPR_MACRO =

Shows or hides the output produced by the macros in the message file. Reading of the message file can be painful when it contains all outputs of the subcommands generated by macro itself. By default, only the output of the command explicitly called by the user in the command file will appear.

Keyword LANG

This keyword makes it possible to choose the language of display of the messages transmitted by the code. If this keyword is not indicated, the environment variables determines the language of the message (reference: http://www.gnu.org/software/gettext/manual/gettext.html#Users). This can be defined in the file `~/.bashrc : LANG=fr_FR.UTF-8 export`.

Encoding (`UTF-8 or ISO-8859-1`) makes it possible to correctly display the accentuated characters.

The `LANG` keyword expects a value in two letters, for example `'FR'` (for French) or `'EN'` (for English).

When the language is selected (by the environment or LANG), it is still necessary that the file of the translated message (file .mo) is available. This file is expected under this name:
$ASTER_ROOT/share/local/"Lang"/LC_MESSAGES/aster_"version".mo
where $ASTER_ROOT/ is the main directory of Code_Aster (e.g.: /aster or /opt/aster), Lang is the name of language in small letters (e.g.: en, fr etc.) and version is the name of the version of Code_Aster used (e.g.: stable, testing, unstable).

If the translation file cannot be read, French is used as default.

Note: even if the file of translation exists, when a message was not translated, there is displayed in French (language used of the messages in the source code).

Keyword BASE

BASE =
This keyword redefines the values of the parameters of the files of random access associated with "databases" if the user does not wish to use the Default built in parameters.

Default values of the parameters associated with databases.

```
GLOBALE
      NMAX_ENRE       62914
      LONG_ENRE       100        Kmots
      LONG_REPE       2000
VOLATILE
      NMAX_ENRE       62914
      LONG_ENRE       100        Kmots
      LONG_REPE       the 2000
```

word is worth 8 bytes in 64 bits under LINUX64, TRU64 an IRIX64. It is worth 4 bytes in 32 bits under SOLARIS, HP-UX, WINDOWS-NT and LINUX.

Procedure DEBUT will allocate a file of random access for base 'GLOBALE' a record of 100 Kmots (the K is worth 1024) to more than 62914 records on a LINUX64 machine by Default.

Note:

The real size of the file is dynamic and it depends on the volume of information to store. But this size is limited by the operating conditions and parameters present among the values characterising the platform.
On a 64 bit platform, the maximum size is fixed at 48 Go. This value can be modified while passing an argument on the command line behind the key word "- max_base size" where size is an actual value measured out of Mo.

On a 32 bit platform, the maximum size is fixed at 2,047 Go (2,147,483,647), but the code manages several files to go beyond this limit when the parameter "- max_base" is passed in the argument.
For the base GLOBALE, which can be saved and re-used in computation data, the maximum size in "POURSUITE" is preserved such as it is if the parameter "- max_base" is not used, but perhaps redefined with the need for this manner.

Operand FICHIER

◆FICHIER =
Symbolic name of the base considered.

Operands LONG_ENRE / NMAX_ENRE / LONG_REPE

Definitions of the parameters

/ | LONG_ENRE = lenr
lenr is the length of the records in Kmots of the files of random access used.

Note:

The memory manager JEVEUX uses this parameter to determine two types of objects, the large object which will be cut out in as many records a snecessary, and the small objects which will be accumulated in a buffer of the size of a record before being discharged.

| NMAX_ENRE = nenr
nenr is the number of records per defect, if this value is not modified by the use of the keyword "- max_base" then this value is given starting from LONG_ENER and of an operating parameter on the reference platform of LINUX64 it is fixed at 48 Go (51,539,607,552 bytes)

Note:

Two operands LONG_ENER and NMAX_ENER must be used with precaution. Bad use can lead to a critical stop of the program by saturation of the files of random access. Coherence between the maximum size of the file and the value resulting from the product of two parameters LONG_ENRE and NMAX_ENRE is checked at the beginning of the execution.

| LONG_REPE = lrep
lrep is the initial length of the directory (maximum number of addressable objects by JEVEUX). It is managed dynamically by the memory manager which extends the size of the directory and all the associated system objects as needs.

Note:

The user choice to modify these various parameters determines certain characteristics of GLOBALE database which no longer cannot be modified in POURSUITE.

Keyword CODE

CODE =
This keyword is intended only for the command files of the tests of non regression managed with the source code.

The presence of this keyword automatically positions the debugging mode DEBUG(JXVERI='OUI',) which implements checks on objects JEVEUX, which can bring a overcost to the execution. The behaviour in the event of error can be modified.

Operand NIV_PUB_WEB

NIV_PUB_WEB = 'INTRANET'
Indicates the level of publication. INTRANET means that the test is only diffusible on the internal network.

NIV_PUB_WEB = 'INTERNET'
INTERNET means that the test is diffusible on the external network.

Operand VISU_EFICAS

VISU_EFICAS = 'OUI'
Indicates whether the command file can be open without problems with the tool EFICAS. This keyword is primarily used for the tests and at the end of receipt of the new versions of the tools.

VISU_EFICAS = 'NON'
Indicates the presence of python source in the command file and thus not allowing its editing with tool EFICAS.

Keyword ERREUR

This keyword permits to modify the behaviour of the code in the event of a FATAL <F> error.

Operand ERREUR_F

In case of an error, the code stops the normal execution of the command.
By default, an exception is then raised (for the detailed definition of a Python exception, one will refer to the documentation of Python or that of the supervisor [U1.03.01]). In this case, the code carries out the command FIN [U4.11.02]) which closes the base in order to allow the possible continuation of computation. It should be noted that although the initial error is known as “fatal”

<F>, the diagnosis is <S>_ERROR since the exception “is recovered” by FIN. This base will be then recopied by the driver of studies. This is the behaviour with ERREUR_F='EXCEPTION'.

If ERREUR_F='ABORT' is used, it means that user has explicitly asked the code to stop execution of the commands in the event of fatal error <F>. Command FIN is not carried out and the base is thus not closed correctly, which means that it is not copied and further resumption of computation is not possible.

Note:

For the execution of the benchmarks by the developers, the stop by ABORT is automatic and by default. This is activated by the presence of the keyword factor CODE (except if ERREUR_F specifies another parameter)

In the event of lack of CPU time, of memory, for all errors of the type <S> and the exceptions, the behaviour that is described with ERREUR_F='EXCEPTION' is executed.

Keyword IGNORE_ALARM

IGNORE_ALARM =

This keyword allows the user to remove the display of certain alarms (of which the user knows the origin) in order to easily identify other alarms which could appear.

During the execution of the command FIN, a summary chart of the alarms emitted during the execution (and the number of occurrences) is systematically displayed. The alarms ignored by the user are preceded by “*” to distinguish them (and they appear even if they were not emitted).

Alarms are indicated starting from the bill of materials appearing between the characters < and >, for example: IGNORE_ALARME = (‘MED_2’, ‘SUPERVIS_40’, …)

Keyword DEBUG

DEBUG =

Option to start debugging (this is reserved for the developers and the maintenance of the code).

Operand JXVERI

JXVERI =

This operand permits to control the integrity of the segments of the memory between two executions of consecutive commands. By default the execution is carried out without DEBUG. This option is systematically activated in the presence of keyword CODE.

Operand ENVIMA

ENVIMA = ‘TEST’

This operand permits printing of the parameters present in software package ENVIMA [D6.01.01] characterising the machine in the file RESULTAT

Operand JEVEUX

◇ JEVEUX =

This operand permits to activate the operating mode in debug of the memory manager JEVEUX, saving on disc not differed and assignment from the segment values to an indefinite value.

Operand SDVERI

SDVERI = 'NON'

This operand is only for Developers. Attention, this functionality causes a considerably little overcost during the execution.

This operand starts the checking of data structures produced by the operators. It is used in the frame of the procedures of development of the code in the tests of non regression. If the keyword CODE is present, this operand takes the default value 'OUI'.

Operand HIST_ETAPE

HIST_ETAPE = 'NON'

This operand makes it possible to preserve all the history of the stages of commands used. This operand is memory hungry and must only be used for a particular cases (the commands which require it, indicates it in their documentation). By default, the history is not preserved.

Keyword MESURE_TEMPS

This keyword makes it possible to choose the level of detail of the impressions of CPU time which will be displayed in the file of messages by the commands carrying out elementary computations, of the resolutions of system linear, the unloading of objects on disc or communications MPI.

Operand NIVE_DETAIL

By default, following line will be printed at the end of each command

```
#1.Resolution.des.systemes.lineaires CPU. (USER+SYST/SYST/ELAPS): 7.52 0.79 11.22
# 2.Calculs.elementaires.et.as semblages CPU. (USER+SYST/SYST/ELAPS): 15.07 0.70 15.77
```

◇ NIVE_DETAIL = 0 no printing
= 1 default printing
= 2 more detailed printing

```
#1.Resolution.des.systemes.lineaires CPU (USER+SYST/SYST/ELAPS): 7.72 0.82 8.72
```

```
#1.1.Numerotation, .connectivity .de.la.ma trice CPU (USER+SYST/SYST/ELAPS): 0.21 0.02
0.31
#1.2.FACTORISATION.SYMBOLIC CPU (USER+SYST/SYST/ELAPS): 0.58 0.05 1.28
#1.3.Factorisation.numerique. (ou.precond.) CPU (USER+SYST/SYST/ELAPS): 6.78 0.73 7.71
#1.4.Resolution CPU (USER+SYST/SYST/ELAPS): 0.15 0.02 0.35
# 2.Calculs.elementaires.et.as semblages CPU (USER+SYST/SYST/ELAPS): 28.87 0.64 29.47
#2.1.Routine.calcul CPU (USER+SYST/SYST/ELAPS): 26.61 0.56 26.61
#2.1.1.Routines.te00ij CPU (USER+SYST/SYST/ELAPS): 24.58 0.07 25.78
#2.2.Assemblages CPU (USER+SYST/SYST/ELAPS): 2.26 0.08 3.36
#2.2.1.Assemblage.matrices CPU (USER+SYST/SYST/ELAPS): 2.02 0.06 3.12
#2.2.2.Assemblage.seconds.membres CPU (USER+SYST/SYST/ELAPS): 0.24 0.02 0.37
```

= 3 more detailed printing and incremental printing for each time step

During parallel computations (MPI), the time spent in the communications is also displayed

```
#4 Communications MPI CPU (USER+SYST/SYST/ELAPS) : 12.67 0.50 12.68
```

Operand MOYENNE

◇ MOYENNE = 'NON', DO NOT display statistics
= 'OUI', display statistics

This keyword makes it possible to exclusively control the display of additional statistics for parallel computations. It is the average of measurements on all the processors as well as the standard deviation of these measurements.

Each display time is then supplemented as follows:

```
#1 Résolution.des.systèmes.linéaires CPU (USER+SYST/SYST/ELAPS): 0.29 0.00 0.35
 (average….diff. .procs) CPU (USER+SYST/SYST/ELAPS): 0.30 0.00 0.47
 (variation-type.diff. .procs) CPU (USER+SYST/SYST/ELAPS): 0.01 0.00 0.05
```

Keyword MEMOIRE

The assignment of resources to various data structures is a dynamic allocation. The user can indicate the limits of these resources during launching of executable in the interface of access.

Operand TAILLE_GROUP_ELEM

TAILLE_GROUP_ELEM = tgrel [default: 1000]

This operand gives the maximum number of finite elements of the same type which will be gathered in a group of elements.

This operand influences the performance of memory and CPU of elementary computation and assemblies.

When tgrel is increased, the user in general will save CPU time, but on the other hand, objects JEVEUX are larger, which can require more memory.

Operand TAILLE_BLOC

TAILLE_BLOC = tbloc [default: 800]

This parameter gives the size of te blocks of the matrices factorized for solver LDLT. This size is given in kiloR8 (1 kiloR8 = 1024 realities). This parameter influences the number of operations of input/output and thus over the time of assembly and resolution. By default, this value is fixed at 800 kiloR8, that is to say 8 records per default on the file of random access associated with base JEVEUX.

Keyword CATALOGUE

This keyword is reserved for the developers, it is used at the time of the operation of compilation of the catalogues of elements to obtain the file in the form of base JEVEUX.

Operand FICHIER

◆ FICHIER = nfic

This operand can only take the value 'CATALEM'.

Operand UNITE

◇ UNITE = logical

Numerical digits associated with the catalogues of elements. In the procedures of construction of the catalogue of elements, the user uses the value 4. The file fort.4 is obtained starting from the directory contents of the catalo sources using a python procedure.

Keyword RESERVE_CPU

This keyword permits to reserve a share of the CPU time allocated to the job to finish the execution in the event of a proper stop for lack of CPU time detected by Code_Aster command. This mechanism is useful only in the case of batch execution of Code_Aster. The value can be indicated as absolute value or in the form of percentage of total CPU time. The values of this keyword is limited by the keyword LIMITS.

When the keyword CODE is present i.e. for all the tests of non regression, a reserve time of 10 seconds is imposed systematically if the keyword RESERVE_CPU is absent.

Operand VALE

This is the value expressed in seconds taken from the total CPU time over which certain total command is based to stop the execution properly.

Operand POURCENTAGE

This is the percentage of the total CPU time withdrawn over which certain total command is based to stop the execution properly.

Operand LIMITS

This is the maximum value of the reserve time. Default is 180 seconds.