
Procedure CONTINUATION

1 Goal

To continue a study starting from the safeguard with format JEVEUX or format HDF of its base 'TOTAL'.

The apparently complex syntax of this procedure should not worry the user, the call with the operands by default, is sufficient in most case: CONTINUATION ()

The use of this order is completely similar to that of BEGINNING. The keywords whose behavior is identical in BEGINNING and CONTINUATION are only described in [U4.11.01].

2 Syntax

```
CONTINUATION
(
  ◇ PAR_LOT = / 'YES' , [DEFECT]
              / 'NOT' ,
  ◇ IMPR_MACRO = / 'NOT' , [DEFECT]
                 / 'YES' ,

  ◇ LANG = Lang, [TXM]

  ◇ BASE = _F (
              ◆ FILE = 'VOLATILE',
                ◇ / | LONG_ENRE= lenr, [I]
                  | NMAX_ENRE= nenr, [I]
                  | LONG_REPE= lrep, [I]
                ),
  ◇ CODE = / 'NOT' , [DEFECT]
           / 'YES' ,

  ◇ ERROR = _F ( ERREUR_F = / 'ABORT' , [DEFECT]
                 / 'EXCEPTION' ,
                 ),
  IGNORE_ALARM = l_vale, [l_KN]

  ◇ DEBUG = _F (
              ◇ JXVERI = / 'YES' ,
                / 'NOT' ,
                ◇ ENVIMA = 'TEST' , [l_Kn]
                ◇ JEVEUX = / 'YES' ,
                  / 'NOT' ,
                ◇ SDVERI = / 'YES' ,
                  / 'NOT' ,
                ◇ HIST_ETAPE = / 'NOT' ,
                  'YES' ,
              ),

  ◇ MESURE_TEMPS = _F (... to see order BEGINNING),
  ◇ MEMORY = _F (... to see order BEGINNING),

  ◇ RESERVE_CPU = _F ( / VALE = vale [R]
                      / PERCENTAGE = pcent [R]
                      ◇ LIMIT = / bv, [R]
                        / 900. [DEFECT]
                    ),
  ◇ FORMAT_HDF = / 'NOT' , [DEFECT]
                 / 'YES' ,
)
)
```

3 Principle of operation

This procedure affects, moreover, the resources memory necessary to the continuation of calculation.

The operands of the order are homologous with those of the procedure `BEGINNING` [U4.11.01]. They make it possible to specify certain resources allocated to the new execution.

The study carried out previously continues with a set of orders starting with `CONTINUATION` and ending in `END` [U4.11.02].

Orders placed front `CONTINUATION` (except obviously `BEGINNING`) or afterwards `END`, if they are syntactically correct, are ignored.

Procedure `CONTINUATION` who is carried out, as of his reading by the supervisor, carries out the following tasks:

- definition of the logical units of the files used in impression,
- allowance of the files associated with the databases managed by `JEVEUX`,
- reading of the catalogues of orders but not of the catalogues of the elements which were recopied on the database during the first execution.

The operands are to be used to divert the various files on numbers of logical unit different from the affected numbers by default or to adjust certain parameters of files.

The simple concepts of python (of variable type) created during a preceding execution are preserved in a file associated with base `JEVEUX` (`pick.1`). During the execution of the procedure `CONTINUATION` these concepts are regenerated and can thus be used under the name under which they were created.

Notice

*Are not saved in `pick.1`, the python objects of the type `classifies`, `function` and `type`.
See the paragraph `Examples` for a possible mode of use.*

4 Operands

The operand `PAR_LOT` and keywords `LANG` and `DEBUG` are identical to those of the procedure `BEGINNING` [U4.11.01].

The keyword `BASE` is different for the procedure `CONTINUATION`.

The keyword `HDF` allows the second reading of a base stored the format "Hierarchical Dated Format".

4.1 Keyword `BASE`

`BASE` =

The functionality of this keyword is to redefine the values of the parameters of the files of direct access associated with the "database" if one does not wish to use those fixed by default. The maximum size of the associated files, and consequently the maximum number of recordings, can be redefined using the parameter passed on the command line behind the keyword

In mode `CONTINUATION`, certain characteristics of the base `TOTAL` cannot be modified any more. Values by default of the parameters associated with the databases

BIRD		
<code>NMAX_ENRE</code>	62914	
<code>LONG_ENRE</code>	100	K words

LONG_REPE	2000
-----------	------

The word is worth 8 bytes out of platform 64 bits under LINUX 64, TRU64 and IRIX 64.4 bytes out of platform 32 bits under SOLARIS, HP-UX and WINDOWS-NT, LINUX.

Under Linux 64, procedure CONTINUATION, with the values by default, a file of direct access of at the maximum will allocate 62914 recordings of 100 *Kmot* (it *K* is worth 1024) for the base 'VOLATILE'.

Note:

The real size of the file is dynamic; it depends on the volume of information to store indeed. Cette size is limited by the conditions of operating and a parameter preset among the values characterizing the platform. On the platform of Linux reference 64 bits, the size initial at 48 Go is fixed. This value is used to dimension 2 objects used by the manager of memory, it will be modified automatically in the course of execution if need be. It is possible of modifier this value while passing an argument on the command line of achievable behind the keyword "- max_base size" where size is an actual value measured out of Mo.

On the platforms 32 bits, the size initial at 2,047 Go (2 147,483,647) is fixed, but the code manages several files to go beyond this limit when the parameter "- max_base" is modified.

For the Total base, which can be saved and re-used in data of a calculation, maximum size initial in "CONTINUATION" is preserved such as it is if the parameter "- max_base" is not used, but perhaps redefined with the need for this manner.

4.1.1 Operand FILE

◆ FILE =

Reference symbol of the base considered.

Only the parameter of the database 'VOLATILE' can be redefined.

4.1.2 Operands LONG_ENRE/NMAX_ENRE/LONG_REPE

Definition of the parameters of the database (files of direct access).

/ | LONG_ENRE = lenr

lenr is the length of the recordings in *Kmots* files of direct accesses used.

Note:

The manager of memory JEVEUX uses this parameter to determine two types of objects: the large objects which will be cut out in as many recordings as necessary, and the small objects which will be accumulated in a plug of the size of a recording before being discharged.

| NMAX_ENRE = nenr

nenr is the number of recordings by default, this value is given from LONG_ENRE and of an operating parameter under LINUX64 fixed at 12 Go (51 539 607 552 bytes) for the maximum size of the file associated with a database.

Note:

Two operands LONG_ENRE and NMAX_ENRE must be used with precaution, a bad use which can lead to the brutal stop of the program by saturation of the files of direct access. Coherence enters maximum size of the file and the value resulting from the product of the two parameters LONG_ENRE and NMAX_ENRE is checked at the beginning of execution.

| LONG_REPE = lrep

`lrep` is the initial length of the repertoire (maximum number of addressable objects by JEVEUX), it is managed dynamically by the manager of memory which extends the size of the repertoire and all the associated system objects as needs.

4.2 Keyword CODE

This keyword makes it possible to activate the generation of the file `.code`. Held with the CAS-tests, this keyword is a version simplified of the keyword `CODE` of `BEGINNING`.

4.3 Keyword FORMAT_HDF

```
FORMAT_HDF = 'YES'
```

Allows to read again a base `TOTAL` saved in a file with format HDF (cf orders `END` [U4.11.02]). The base is then rebuilt starting from objects JEVEUX stored in the file, this file can be built on a different platform (operating system, platform 32 or 64 bits). The characteristics of the original base are read again in the file and the base is rebuilt with identical (one preserves for example the length of the recordings).

The file associated with the base `TOTAL` with format HDF is named `bhdf.1` in the repertoire of execution of the code.

5 ExampleS of use

The standard use of this procedure is:

```
CONTINUATION ()
```

The tests `yyy100a` and `yyy100b` illustrate the use of `RESERVE_CPU`
The tests `forma04b`, `ssnv156a`, `ssnv166b`, `yyy108`, illustrate the use of `MEMORY`.

Python does not save the functions and classes defined in the main file. It is thus the case of the command file of `code_aster`.

To use functions or classes and to find them in continuation, they should be defined in an external module:

- That is to say the module `userpkg/usermod.py`:

```
def pyfunc (X):  
    return X * 2
```

- One would make in the first command set:

```
BEGINNING ()  
from userpkg import usermod  
forms = FORMULA (NOM_PARA=' X', VALE=' pyfunc (X) ',  
pyfunc=usermod.pyfunc)  
assert forms (2) == 4  
END ()
```

- Then:

```
CONTINUATION (PAR_LOT=' NON')  
assert forms (2) == 4  
END ()
```

For that, it is enough to add in the study the repertoire `userpkg` of type `name`.