Utilities of impression of messages

Summary:

One presents in this document the utilities of emission of messages of information, error or alarm in FORTRAN (or in python) (routine U2MESG in particular) as well as the mechanism of lifting of exception (routine UTEXCM).

Contents

1 Example of message

```
! ---------------------------------------------------------!
! <A> <MODELISA4_9>                                         !
!                                                           !
!                                                           !
!   - > Phase of checking of the grid: presence of       !
!      flattened meshs.                                  !
!                                                           !
!   - > Risk & the Council:                                  !
!      Check your grid. The presence of such meshs!
!      can lead to problems of convergence and harm!
!      with the quality of the results.                        !
!                                                           !
!                                                           !
!                                                           !
! This is an alarm. If you do not understand the direction of   !
! this alarm, you can get results           !
! unexpected                                                !
! ---------------------------------------------------------!
```

2 Type of messages

The logical units of impression are established at the beginning of completion of the work, by the supervisor, the impressions emitted by the routines described in the following paragraphs will be carried out on these logical units without possibility of redirection.

The transmitted messages will be directed only according to their type:

Code	Type of message	Output files
F	Error message fatal, the execution stops after the impression of the message.	MESSAGE ERROR RESULT
E	Error message, the execution continues (a little) cf [§2.2].	MESSAGE ERROR RESULT
With	Message of alarm.	MESSAGE RESULT
I	Message of information.	MESSAGE
Z	Recoverable lifting of exception in python	

2.1 Messages of the type F

This kind of message is followed of a dead halt of the application, it is used within the framework of the serious detection of error which cannot allow the normal continuation of an order Aster. The emission of an error message **F** cause the stop of the execution.

By default for “truths” users (if the keyword BEGINNING/CODE is not used), the execution stops after having validated the concepts which can the being:

- concepts produced by the orders already carried out

- The concept in the course of creation if the order envisaged it (for example, for the order STAT_NON_LINE, the steps of already filed times are validated).

On the other hand, if the command file uses BEGINNING/CODE, the code stops immediately and calls the routine abort() in order to create a “core slips by” usable by the debugger post-mortem.

2.2 Messages of the type E

This kind of message makes it possible to analyze a series of errors before the program stop. For example, syntactic analysis of the command file by the Supervisor or analysis of the file of grid by the order LIRE_MAILLAGE.

The transmitter of a message of the type **E** must transmit a message of the type **F** at the end of its analysis.

2.3 Messages of the type A

One should not misuse the messages of alarm of the type **With** who can worry the users unnecessarily. The message must be clear and comprise tracks to avoid this alarm.

The number of messages of alarm is limited automatically to 5 identical successive messages.

It is recommended to the users who have messages of the type With “to repair” their command file to make them disappear.

2.4 Messages of the type I

Messages **I** are messages of information.

2.5 Messages of the type Z

Messages **Z** are messages allowing the mechanism of the “exceptions”. This kind of message is used only in the 2 routines utexcm and utexcp. and do not owe the being elsewhere! See [§8] for the exceptions in general.

3 Operation general

We start from an example to present the operating process of the utilities of impression of message:

3.1 Example

Printed message

```
! --------------------------------------------------!
! <A> <CHATON_2>                                    !
!                                                   !
!    The small kitten is of pink.              !
!                                                   !
! This is an alarm. If you do not understand the direction!
! of this alarm, you can get results    !
! unexpected                                        !
! --------------------------------------------------!
```

This message user the small kitten is of pink. consists of two parts:

- A: the text fixes the small kitten is of color.
- B: the pink variable text which represents the contents of a variable of type character.

To implement in FORTRAN or the python the impression of this message, it is necessary:

- to describe in a catalogue python the part fixes (A) of the message, with the format of the variables to be printed
- to instrument the subroutine FORTRAN or the function python to cause the impression of the message.

Call FORTRAN

```
SUBROUTINE KITTEN (…,…)
…
…
IF (I.EQ.1) THEN
  VALK = 'ROSE'
  CAL U2MESK ('WITH', 'CHATON_2', 1, VALK)
ENDIF
…
…
END
```

Arguments of subroutine U2MESK:

- 'With' : specify that one prints a message of the type alarms
- 'CHATON_2' : is the identifier of the message: KITTEN is the name of the catalogue of message chaton.py and quantifies it 2 indicate that one will take in this catalogue the message n°2.
- 1: only one variable character is printed
- VALK : variable character to be printed.

Catalogue message: chaton.py

```
cata_msg = {

1: _(U """
    The kitten is well of green color Phew!
 """),

2: _(U """
    The kitten is of color % (k1) S.
 """),

3: _(U """
   The kitten has % (i1) D legs and % (i2) D eyes.
 """),

4: _(U """
   The kitten weighs % (r1) F kilogrammes.
```

```
 """),

5: _(U """
   CAUTION: Your cat is odd, it:
      - has more than 4 legs, it has % (i1) of them D,
      - is of color % (k1) S and % (k2) S,
      - is too large, it weighs (% (r1) F kilogrammes.

  One stops the expenses, it is not a cat!!
"""),

 }
```

4 Utilities FORTRAN

Name	Function
U2MESS	Print a simple message
U2MESK	Print at the same time a message and values of type character
U2MESI	Print at the same time a message and values of the whole type
U2MESR	Print at the same time a message and values of the real type
U2MESG	Print at the same time a message, values of type nature, whole and real

4.1 Impression of simple messages

SUBROUTINE U2MESS (STANDARD, IDMESS)

TYPE	IN	Type of message (E, F, A, I)
IDMESS	IN	identifier of the message

Example:

```
CAL U2MESS ('WITH', 'CHATON_1')
```

Print the following message:

```
<A> <CHATON_1>
The kitten is of green color Phew!
```

4.2 Impression of messages and the values of type character

SUBROUTINE U2MESK (STANDARD, IDMESS, NK, VALK)

TYPE	IN	Type of message (E, F, A, I)
IDMESS	IN	Identifier of the message
NK	IN	Parameters number of type character
VALK	IN	Values of the parameters of the characters type

Example:

```
VALK = 'ROSE'
CAL U2MESK ('WITH', CHATON_2', 1, VALK)"
```

Print the following message:

```
<A> <CHATON_2>
The kitten is of pink.
```

4.3 Impression of messages and the values of the whole type

SUBROUTINE U2MESI (STANDARD, IDMESS, NI, VALI)

TYPE	IN	Type of message (E, F, A, I)
IDMESS	IN	Identifier of the message
Ni	IN	Many parameters of the whole type
VALI	IN	Values of the parameters of the whole type

Example:

```
VALI (1) = 4
VALI (2) = 2
CAL U2MESI ('WITH', 'CHATON_3', 2, VALI)"
```

Print the following message:

```
<A> <CHATON_3>
The kitten has 4 legs and 2 eyes.
```

4.4 Impression of messages and the values of the real type

SUBROUTINE U2MESR (STANDARD, IDMESS, NR, VALR)

TYPE	IN	Type of message (E, F, A, I)
IDMES	IN	Identifier of the message
NR	IN	Parameters number of the real type
VALR	IN	Values of the parameters of the real type

Example:

```
VALR (1) = 130.
CAL U2MESR ('WITH', 'CHATON_4', 1, VALR)"
```

Print the following message:

```
<A> <CHATON_4>
The kitten weighs 130.0 kilogrammes.
```

4.5 Impression of messages of the values of the characters type, whole and real

SUBROUTINE U2MESG (STANDARD, IDMESS, NK, VALK, NI, VALI, NR, VALR)

TYPE	IN	Type of message (E, F, A, I)
IDMESS	IN	Identifier of the message
NK	IN	Many parameters of type character

VALK	IN	Values of the parameters of type character
Ni	IN	Many parameters of the whole type
VALI	IN	Values of the parameters of the whole type
NR	IN	Many parameters of the real type
VALR	IN	Values of the parameters of the real type

Example:

```
VALK (1) = 'blue'
VALK (2) = 'pink'
VALI = 5
VALR  = 130.
U2MESG ('WITH', 'CHATON_5', 2, VALK, 1, VALI, 1, VALR)"
```

Print the following message:

```
<E> <CHATON_5>
CAUTION: Your cat is odd, it:
  - has more than 4 legs, it has 5 of them,
  - is of blue and pink,
  - is too large, it weighs 130.0 kilogrammes.
```

One stops the expenses, it is not a cat!!

4.6 Impression of a message in several "pieces"

It is sometimes practical to transmit a message "per pieces" i.e. to call several times the routine U2MESG. This is in pricipe impossible for the error messages fatal (F/E) because the code stops as of the first message!
In addition, it is wished that the complete message appear to the user like a single message. In particular it is wanted that it appears within the same framework.

For that, one has the mechanism 'A+','F+',... the principle is to add one'+'with the type of the message as long as the message is not finished. The last message of the group (without'+') the message finishes.

One will be able for example to write a message in 2 pieces while writing:

```
CAL U2MESK ('F+', 'FONCT0_11', 1, NOMF)
CAL U2MESR ('F',  'FONCT0_26', 3, VALR)
```

What, associated with the following catalogue:

```
11: _(U """
The interpolation of the function '% (k1) was not authorized.
The type of interpolation of the function is worth 'NOT'

  - > Risk & the Council:
    See keyword Interpol of the orders which create functions.
"""),

26: _(U """
   X-coordinate requested: % (r1) F
   found interval: [% (r2) F, % (r3) F]
"""),
```

will lead to a message resembling:

```
! ---------------------------------------------------------------------!
! <F> <FONCT0_11>                                                      !
!                                                                      !
! The interpolation of the function fonc3 is not authorized.           !
!   The   type   of   interpolation   of   the   function   is   worth   'NOT'
!
!                                                                      !
!  - > Risk & the Council:                                               !
!    See keyword Interpol of the orders which create functions.  !
!                                                                      !
!   X-coordinate requested: 105.                                          !
!   found interval: [0. , 100.]                                    !
! ---------------------------------------------------------------------!
```

Note:

- In a group of messages producing a message “per pieces”, the type of the various messages dowry being the same one (F/A/I).
- The identifier of the printed message is that of the first message of the group.

5 Utility Python

The impression of the messages in the files python is carried out by method UTMESS.

Name	Function
UTMESS	Print at the same time a message, values of the characters type, whole and real

def UTMESS (standard, idmess, valk, vali, valr)

type	IN	Type of message (E, F, A, I)
idmess	IN	identifier of the message
valk	IN	Values of the parameters of the characters type
vali	IN	Values of the parameters of the whole type
vali	IN	Values of the parameters of the real type

Contrary to FORTRAN, there is only one function UTMESS because arguments valk, vali and valr are optional (cf Python Tutorial, keyword arguments). The function of U2MESI is realized in this manner:

```
UTMESS ('with', 'MESSAGE_1', vali= (1, 2.3))
```

6 Format of impression in the catalogues of message

Format	Function
% (kN) S	Format of impression of N^{ième} value of type character
% (in) D	Format of impression of N^{ième} value of the whole type
% (RN) F	Format of impression of N^{ième} value of the real type (one uses sometimes the format %g, %e, %F, %E, %G)

More on the codes of formattage of the chains: to see G-string formatting operations documentation Python.

It is necessary that the messages are chains “unicode” (note “U in front” """) because the messages contain characters not-ASCII.
Some simple rules must be followed in order to preserve a maximum of homogeneity between the few 6,000 messages:

- To make sentences, not abbreviations, not of names of routines, structure of data which are possibly known developers but not users.
- Not to make a “page layout”: no the superfluous jump of line, to avoid forming columns, etc because it will be difficult to preserve them during the translation.

7 Examples

Identifier message	Catalogue message (chaton.py)	Call FORTRAN/python
-	cata_msg = {	-
CHATON_1	1: _(U" "" - The Kitten is well of green color Phew! """),	→ FORTRAN CAL U2MESS ('WITH', 'CHATON_1') → Python UTMESS ('WITH', 'CHATON_1')
CHATON_2	2: _(U" "" - The kitten is of color % (k1) S """),	→ FORTRAN CAL U2MESK ('I', 'CHATON_2', 1, COUL) → Python UTMESS ('I', 'CHATON_2', valk=…)
CHATON_3	3: _(U" "" - The kitten has % (i1) D legs and % (i2) D eyes. """),	→ FORTRAN VALI (I) = NBPATT VALI (2) = NBYEUX CAL U2MESI ('I', 'CHATON_3', 2, VALI,) → Python UTMESS ('I', 'CHATON_2', vali=…)
CHATON_4	4: _(U" "" - The kitten weighs % (r1) F kilogrammes. """),	→ FORTRAN VALR = WEIGHT CAL U2MESR ('I', 'CHATON_4', 1, VALR,) → Python UTMESS ('I', 'CHATON_4', valr=…)
CHATON_5	5: _(U" "" - CAUTION: Your cat is odd, it: → has more than 4 legs, it has % (i1) of them D, → is of color % (k1) S and % (k2) S, → is too large, it weighs (% (r1) F kilogrammes. One stops the expenses, it is not a cat!! """),	→ FORTRAN VALI = FIVE VALK (1) = ' BLEU' VALK (2) = 'ROSE' VALR= 130. CAL U2MESG ('E', 'CHATON_4', 2, VALK, 2, VALI, 1, VALR,) → Python UTMESS ('E', 'CHATON_5', valk, vali, vakr)
-	}	-

8 Utilities of lifting of exception

In the course of writing

Utilities UTEXCP and UTEXCM allow to raise an exception python since FORTRAN and to transmit it to the command set. This exception can be recovered in the command file (or an macro-order) by a block try/except.

The exceptions being typical cases of <F> error, if they are not recovered, the supervisor finishes calculation as after an ordinary <F> error (closing and recopy of the base). Only exception FatalError escape the rule since it is equivalent to a <F> error (and still it will be a little further seen this behavior is modifiable).

The mechanism of exception makes it possible to try an order then to take again the hand if this one fails while raising a particular exception.

This operation is particularly used in the macro-orders, either to behave differently according to the context, or to transmit a message more speaking to the user that which would have been emitted by an order girl.

Note:

The use of the exceptions and especially of the blocks try/except *has direction only in mode* PAR_LOT=' NON'

8.1 Implementation in the source

One presents here the implementation of the mechanism of the exceptions in source FORTRAN and python from the point of view of the developer. One will a little low approach the exceptions with the vision “user” (§8.4).

For the description of what is an exception, mechanisms of lifting (raise) and of interception (except), one will refer to the Description of the exceptions of the Tutorial Python.

“To raise an exception” means to emit a signal with (often) elements of context to allow the program inviting to react as follows:

- I understand the signal and I make such decision consequently
- I do not understand the signal and I stop (by emitting another signal with the program calling of the higher level)
- I am unaware of the signal and I leave the program inviting of the higher level to treat the signal

And so on, knowing that with highest, it is the interpreter Python itself which will make the decision to stop the execution.

Syntactically and schematically, operation is the following:

```
try:

   …
   Instructions to be carried out which can for example raise 3 exceptions
   different: UneExceptionParticulière, UneAutreException, EncoreUneAutre
   …

except UneExceptionParticulière, arguments:
```

…
These instructions will be carried out if exception "UneExceptionParticulière"is raised. One can use the "arguments" used at the time of the lifting of the exception.
In the case simplest (and frequent), "arguments"is a message which can be printed with"print"or used as character string with"str (arguments)".

```
   …

except UneAutreException, arguments2:

   …
   Instructions to be carried out if the UneAutreException exception is
raised.
   …
```

Let us suppose that the block of instructions under try raise the exception EncoreUneAutre (which is not intercepted by a block except), the execution stops with a message of the type:

```
Traceback (most recent cal last):
  File ".......", line 3, in < ....... >   <<<    allows to say where is
                                                produced exception
EncoreUneAutre: ......                   <<<   elements of context
                                                (arguments) providing
                                                details on the reasons
                                                lifting of exception
```

For the other mechanisms, try/finally or try/except/else, one will refer to documentation Python.

8.1.1 Lifting of exception in Code_Aster

Simplest: in Python, i.e. in the command file or an macro-order, it is enough to make:

```
yew nbre_iterations > nbre_maxi_autorise:
   raise aster.NonConvergenceError, "Absence of convergence with the
iteration count authorized"
```

In source FORTRAN (example in nmerro.f who treats the errors in STAT_NON_LINE) :

```
…
ELSE IF (ITEMAX) THEN
  CAL UTEXCP (22, 'MECANONLINE_83')
…
```

where

- 22 is the number of the exception
- 'MECANONLINE_83' is the identifier of the message to be printed

Indeed, the exceptions suitable for Code_Aster are numbered to be accessible easily since FORTRAN:

number of the exception	Description	Name of the exception in the module aster	Standard error
20	Fatal error	FatalError	F
21	Nonfatal error	error	F

22	Not convergence	NonConvergenceError	F
23	Failure integration of the behavior	EchecComportementError	F
24	Empty waveband	BandeFrequenceVideError	F
25	Singular matrix	MatriceSinguliereError	F
26	Failure of treatment of the contact	TraitementContactError	F
27	Noninvertible matrix of contact	MatriceContactSinguliereError	F
28	Lack of time CPU	ArretCPUError	F
29	Failure of piloting	PilotageError	F

Remarks

The exceptions are defined in the module aster (astermodule.c). Thus, one reaches it by aster.error .
Except for FatalError , all the exceptions derive from error who thus represents the envelope of the <F> errors .
The behavior in the event of fatal error (FatalError) is modifiable by the user [U1.03.01] , this choice is given in the order DEBUT/POURSUITE [U4.11.01], [U4.11.03] keyword factor ERREUR_F . In the macro-orders, the function will be used aster.onFatalError (arg) to modify this behavior.

8.2 Lifting of an exception with impression of a message

CAL UTEXCP (CODE, IDMESS)

CODE	IN	Number of the exception
IDMESS	IN	Identifier of the message

Example:

```
CAL UTEXCP (23, 'CHATON_7')
```

8.3 Lifting of exception with impression of messages and values of the characters type, whole and real

CAL UTEXCM (CODE, IDMESS, NK, VALK, NI, VALI, NR, VALR)

CODE	IN	number of the exception
IDMESS	IN	Identifier of the message
NK	IN	Many parameters of type character
VALK	IN	Values of the parameters of type character
Ni	IN	Many parameters of the whole type
VALI	IN	Values of the parameters of the whole type
NR	IN	Many parameters of the real type
VALR	IN	Values of the parameters of the real type

Example:

```
VALI (1) = NDIM1
VALI (2) = NDIM2
VALR     = 5.D0
CAL UTEXCM (24, 'CHATON_12', 0, '', 2, VALI, 1, VALR)
```

8.4 Example of use

In this example, one wishes:

- in the event of not-convergence of STAT_NON_LINE, to start again calculations by increasing the iteration count,
- to stop calculations if another error occurs.

This use is illustrated in the CAS-test ssnp125a.

Programming

The exception is raised NonConvergenceError who carries the number 22:

```
…
ELSE IF ((.NOT.CONVER) .AND. ITEMAX .AND. (.NOT.ARRET)) THEN
    ITAB (1) = NUMORD
    ITAB (2) = ITERAT
    CAL UTEXCM (22, 'MECANONLINE_85', 0, K8B, 2, ITAB, 0, RTAB)
END IF
```

Command file

```
try:
     STATNL=STAT_NON_LINE (...)

except aster.NonConvergenceError, message:
     # not convergence
     print "one continues by increasing the iteration count"
     STATNL=STAT_NON_LINE (reuse=STATNL,
                           …
                           CONVERGENCE=_F (ITER_GLOB_MAXI=400,)
                           …)
```

File message: mecanonline.py

```
85: _(U """
   Stop: absence of convergence to the number of moment: % (i1) D
                                  at the time of the iteration: % (i2) D
"""),
```

Printed message

```
   !
-------------------------------------------------------------------------!
   ! <EXCEPTION> <MECANONLINE_85>                                          !
   !                                                                       !
   !                                                                       !
   !    Stop: absence of convergence to the number of moment:
123           !
```

```
   !                                  at the time of the iteration:  51
!
   !                                                                  !
   !                                                                  !
   !
---------------------------------------------------------------------!
```