LA GESTION DES ERREURS DANS SAS

Caractéristiques :

Catégorie : Base
OS : tous
Version : 9.x
Vérifié en septembre 2016

La gestion des erreurs peut se faire à plusieurs niveaux dans SAS. Nous proposons des options générales qui permettent de changer le comportement par défaut de SAS face à certains types d'erreur (format ou variable non trouvés, erreur de syntaxe simple, etc). Ensuite, nous avons des macro-variables pour tester le bon déroulement d'un programme, et permettant donc de prendre des actions soit immédiatement lors de l'exécution du programme, ou lorsqu'elle est terminée.

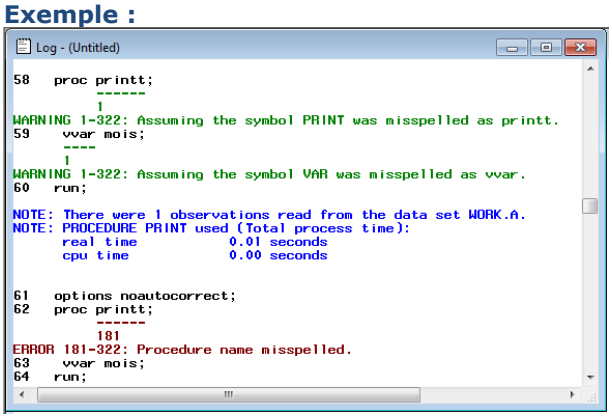
Sommaire

1. DECIDER DU NIVEAU D'ERREUR A CONSERVER, LORS DE L'ABSENCE DE TABLES, VARIABLES, FORMATS...

Commençons par un rapide panorama d'options pour gérer les erreurs du type : table, variable ou format manquants. Ces options permettent de choisir si cette absence conduira à un message d'erreur, ou simplement un message d'avertissement. Ce sont des options générales qui sont actives automatiquement dans la session SAS avec une valeur par défaut. Elles se gèrent, quand on souhaite modifier ce comportement par défaut, via une instruction options ou la fenêtre de gestion des options (dans la session SAS déjà démarrée), le fichier autoexec, en ligne de commande de démarrage de SAS ou encore dans le fichier de configuration.

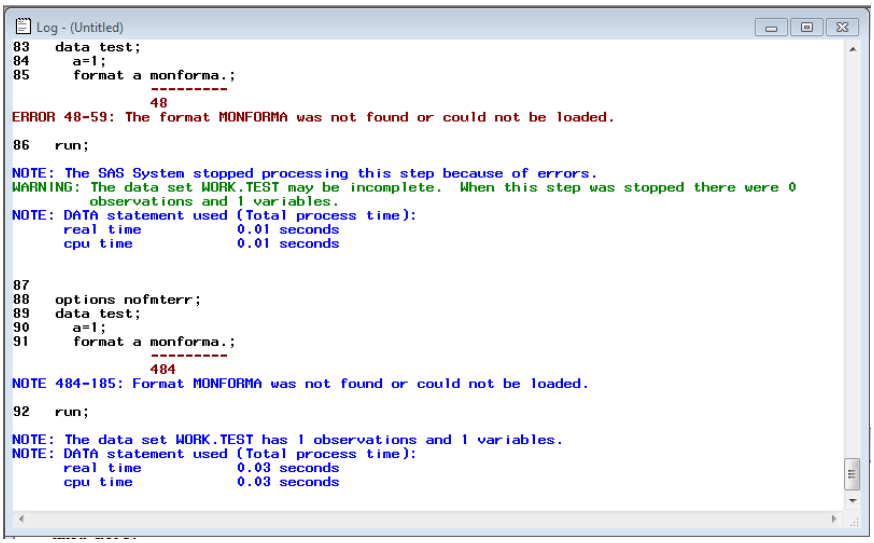
1.1. Présentation d'options de gestion des erreurs

Nom : autocorrect
Utilité : Elle autorise les fautes de frappe simples dans les noms de procédures, d'instructions.
Par défaut : active
Exemple :



Impact : Quand l'option est active, malgré deux fautes de frappe, la procédure print est exécutée. Mais quand l'option est désactivée, la syntaxe SAS doit être strictement utilisée. Les fautes de frappe ci-dessus ne sont plus gérées. En conséquence, la procédure print n'est pas exécutée.

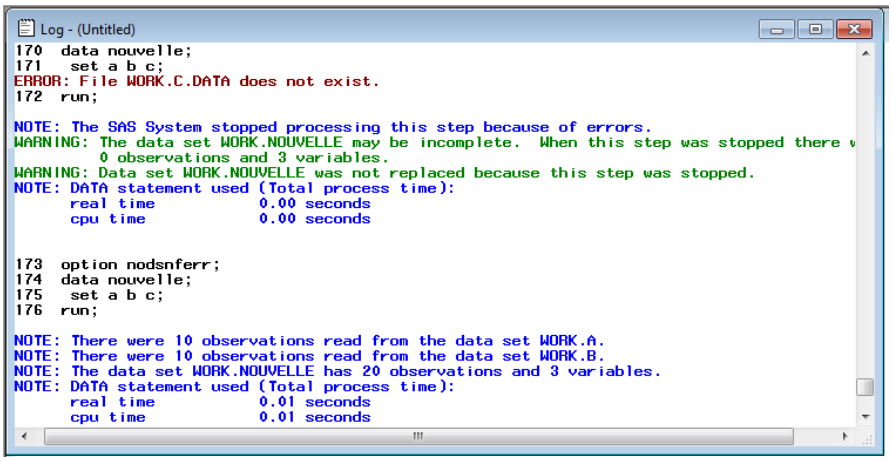
Nom : fmterr
Utilité : Permet de gérer le comportement de SAS quand un format n'est pas trouvé (soit avec une erreur, soit en ignorant le fait que le format est inconnu)
Par défaut : active
Exemple :



Impact : Lors de la création d'une table SAS, quand l'option est positionnée, un format inconnu provoque une erreur et l'interruption de l'étape data. En conséquence, la table n'est pas créée.
En désactivant l'option, une note cette fois est affichée, ce qui a pour effet de laisser la table se créer, simplement sans format associé.

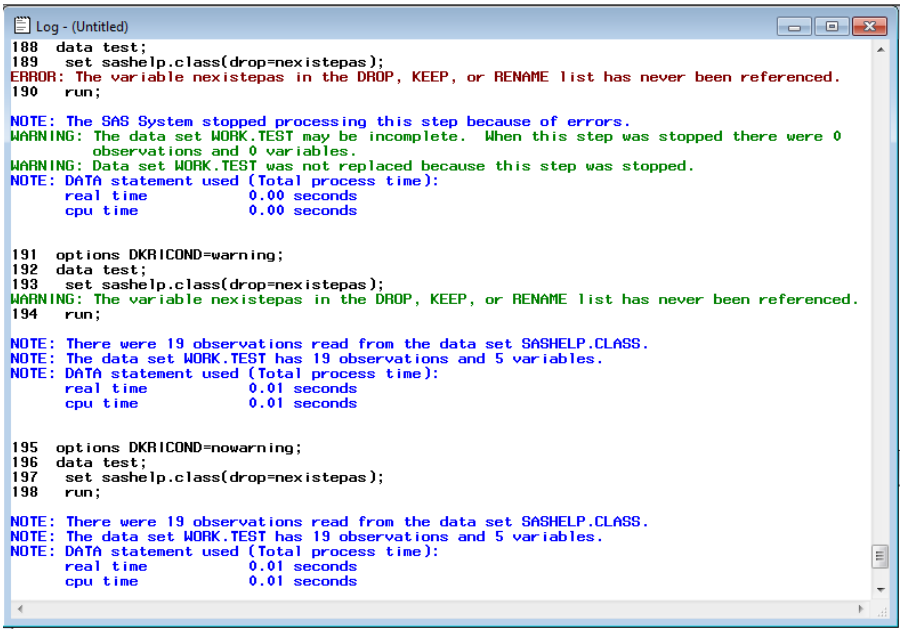
Astuce : L'option nofmterr est particulièrement utile pour pouvoir consulter le contenu d'une table, où des formats sont appliqués, mais quand le catalogue de formats n'est pas disponible (non fourni lors de l'échange entre deux utilisateurs par exemple). Elle permet au moins de visualiser les données brutes. Sinon, l'ouverture de la table est impossible.

Nom : dsnferr
Utilité : Permet de gérer comment SAS réagit quand une table SAS n'existe pas (soit par une erreur, soit en ignorant le fait que la table n'existe pas).
Par défaut : active
Exemple :



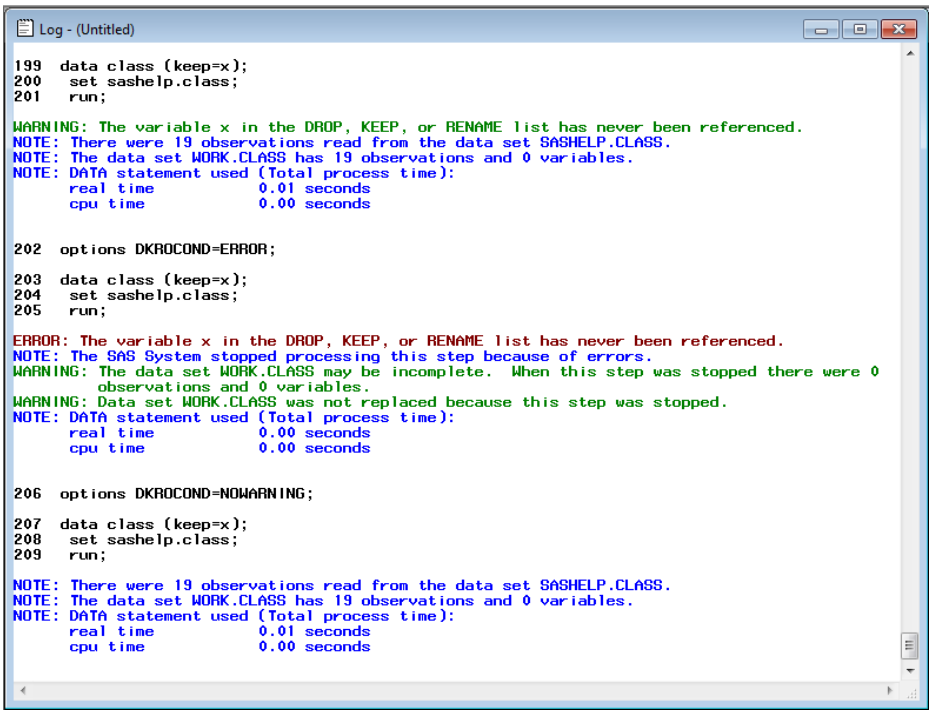
Impact : quand une table SAS n'existe pas, que ce soit lors de son appel dans une étape data ou par une procédure, alors une erreur est retournée. Dans cet exemple, la table « nouvelle » n'est pas créée puisqu'une des tables utilisées pour la créer n'existe pas.
L'exemple montre ensuite qu'en positionnant l'option nodsnferr, la table « nouvelle » est néanmoins créée, même si l'une des trois tables de l'instruction set n'existe pas.

Nom : dkricond
Définition : Permet de gérer le comportement de SAS quand une variable, utilisée dans une option rename, drop ou keep, n'existe pas, dans une table qui est lue.
Par défaut : ERROR
Exemple :



Impact :
Prenons un exemple simple : lors de la lecture d'une table SAS, l'instruction drop est utilisée pour ne pas conserver une variable. En fait, cette variable n'existe pas. Par défaut, SAS retourne un message d'erreur. L'étape data est interrompue. La table SAS, qui devait être créée, ne l'est pas.
En utilisant DKRICOND=warning, le message d'erreur est remplacé par un message d'avertissement. La table est donc créée, en ignorant donc le fait qu'une variable, qui n'existe pas, aurait dû être supprimée.
Enfin, en utilisant DKRICOND=nowarning, plus aucun message n'est affiché, et la table est créée.

Nom : dkrocond
Définition : Permet de gérer le comportement de SAS quand une variable, utilisée dans une option rename, drop ou keep, n'existe pas, dans une table qui est créée.
Par défaut : WARN
Exemple :



Le même principe s'applique ici. Les options DKR**I**COND et DKR**O**COND se distinguent uniquement sur le fait qu'elles s'appliquent sur des tables en entrée (**I**nput) ou en sortie/création (**O**utput).
Dans une étape data, créer une table alors qu'une variable, qui n'existe pas, est spécifiée dans une option drop, keep ou rename ne remet pas en jeu l'intégrité des données. C'est pourquoi elle est positionnée à WARN par défaut, ce qui a pour effet que malgré l'absence de variable(s), la table est néanmoins créée.

Nom : syntaxcheck
Définition : Dès qu'une erreur survient, deux options sont impactées :
- l'option obs est positionnée à 0
- l'option noreplace est activée

Dans les faits, le reste du programme est uniquement compilé, mais les tables SAS ne sont pas remplacées (si elles existaient déjà) et les nouvelles tables sont créées avec 0 observation (donc seulement avec leur « header » (ou description)).
En résumé, SAS se positionne en mode « syntax check », autrement dit, seule la syntaxe est vérifiée, le code n'étant plus exécuté.
Par défaut : activée en batch ou en client/serveur (n'est pas valide dans une session SAS interactive, voir dmssynchk plus bas)
Exemple : Un programme exécuté en batch contient une erreur. Voici un extrait de la log :

```
1          data table1;
2          x=fontionquinexistepas();
                    ____________________
                    68
ERROR 68-185: The function FONTIONQUINEXISTEPAS is unknown, or cannot be accessed.
3          run;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: SAS set option OBS=0 and will continue to check statements.
      This might cause NOTE: No observations in data set.
WARNING: The data set WORK.TABLE1 may be incomplete.  When this step was stopped there were
0 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds
4
5          data table2;
6          set table1;
7          run;
NOTE: The data set WORK.TABLE2 has 0 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds
8
9          data test;
10         x=1;
11         run;
NOTE: The data set WORK.TEST has 0 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds
12
13         proc options;run;
NOTE: PROCEDURE OPTIONS used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
ERROR: Errors printed on page 1.
NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
```

Impact : SAS se positionne en mode "syntax check" (cf la note surlignée). Toutes les tables sont créées, sans observation. La proc options ne retourne aucune information (alors qu'elle les affiche normalement dans la log).

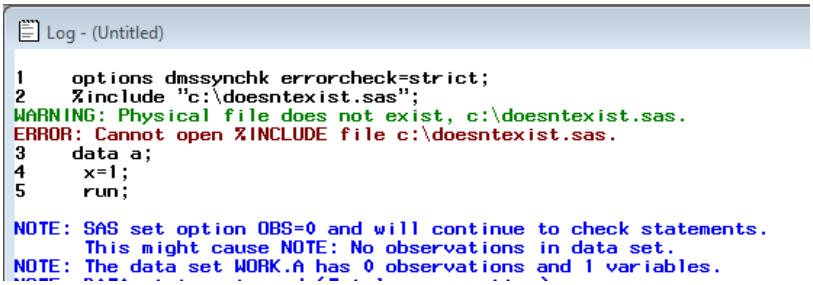
Pour changer ce comportement par défaut du mode batch, il faut ajouter l'option –nosyntaxcheck à la ligne de commande.
Exemple :
«C:\Program Files\SASHome\SASFoundation\9.4\sas.exe» -sysin c:\prog\monprogramme.sas -nosyntaxcheck

Nom : dmssynchk
Définition : C'est le pendant de l'option syntaxcheck, en mode interactif. Donc dès qu'une erreur survient, SAS se positionne en mode « syntax check ».
Par défaut : nodmssynchk

Nom : errorcheck
Définition : Permet de définir si le mode « syntax check » sera activé ou non lors d'une erreur dans l'une de ces instructions : libname, filename, lock (SAS/Share), %include.
En mode *Normal*, le mode « syntax check » n'est pas activé dans le cas d'une telle erreur. Il l'est en mode *Strict*.
Par défaut : Normal
Exemple : Un fichier qui n'existe pas est appelé dans une instruction %include. En mode *normal*, le code retour est 0 (le code retour est stocké dans une macro-variable automatique : syscc). Par contre, en mode *strict*, il sera valorisé à une valeur différente de 0, soit 1012 ici.

```
1    %include "c:\doesntexist.sas";
WARNING: Physical file does not exist, c:\doesntexist.sas.
ERROR: Cannot open %INCLUDE file c:\doesntexist.sas.
2    %put &syscc;
0
3    options errorcheck=strict;
4    %include "c:\doesntexist.sas";
WARNING: Physical file does not exist, c:\doesntexist.sas.
ERROR: Cannot open %INCLUDE file c:\doesntexist.sas.
5    %put &syscc;
1012
```

Dans un second exemple, voyons comment l'application de l'utilisation du mode strict de l'option errorcheck. Le mode syntax check sera activé en cas d'erreur, si l'option dmssynchk est activée en interactif, ou syntaxcheck dans les autres cas d'utilisation de SAS en client/serveur ou en batch.

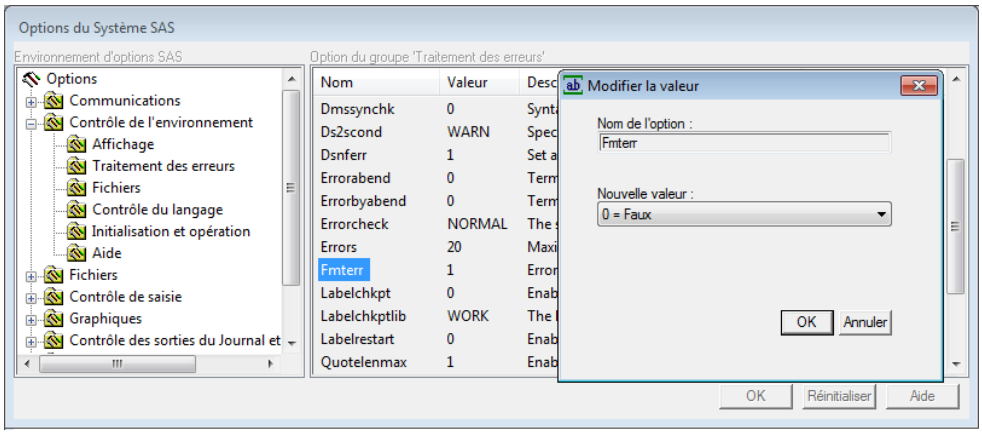


Donc avec dmssynchk activée, et errorcheck=strict, si une instruction %include référence un programme SAS qui n'existe pas, SAS passe en mode « syntax check » et les tables SAS ne sont plus créées, les instructions SAS ne sont plus exécutées.

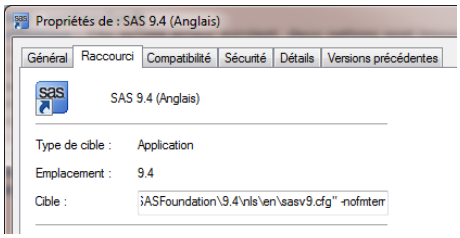
1.2. Mise en place des options

Si toutes les valeurs par défaut vous conviennent, aucune autre action n'est évidemment nécessaire. Au cas où vous voudriez changer une des options, voici les approches possibles :

- Dans la session SAS déjà ouverte :
  - En exécutant simplement l'option (comme vu dans les exemples présentés en 1.1). Exemple : `options nofmterr;`
  - En utilisant la fenêtre options, qui s'appelle avec la commande « options » en ligne de commande



- Avant de démarrer la session SAS :
  - Dans un fichier autoexec.sas, avec la même instruction option qu'en interactif
  - Dans la cible du raccourci utilisé pour démarrer SAS :



- Dans le fichier de configuration, avec le nom de l'option précédé d'un tiret (-) et sans autre caractère :
  -nofmterr

1.3. Vérification du positionnement des options

Les valeurs des options abordées jusqu'ici peuvent être consultées en exécutant la proc options :

```
PROC OPTIONS GROUP=ERRORHANDLING;
run;
```

L'option group=errorhandling permet de restreindre l'affichage aux options catégorisées dans ce groupe. Le résultat est affiché dans la log.

Il existe également quelques options intéressantes dans la proc options, à savoir :

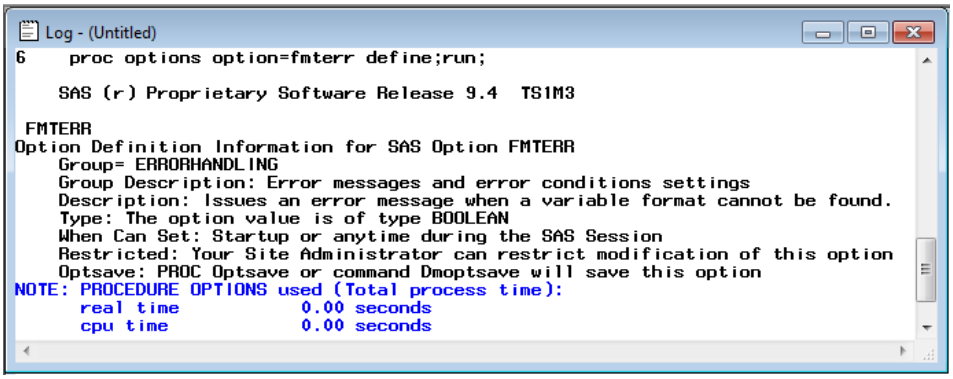
- *define*, qui permet d'écrire dans la log des informations complémentaires sur une option donnée.
- *value*, qui permet de rappeler la valeur courante de l'option

Ainsi :

```
proc options option=fmterr define;
run;
```

affiche dans la log, entre autres, que l'option fmterr :

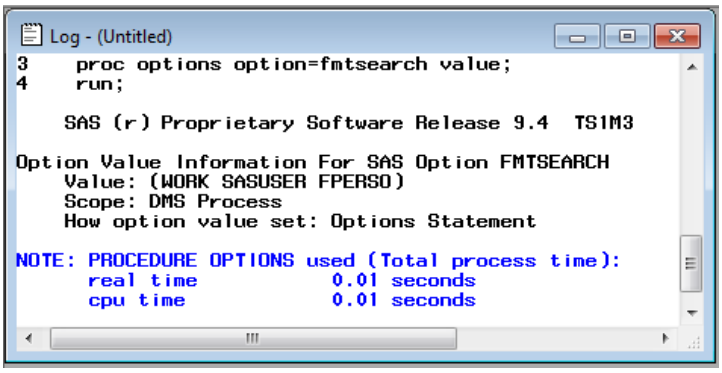
- fait partie du groupe errorhandling, donc du groupe qui répertorie les options pour gérer les messages d'erreur et le paramétrage des conditions dans lesquelles elles sont gérées
- est de type booléen (fmterr/nofmterr)
- peut être modifiée au démarrage ou interactivement dans une session SAS



Et :

```
libname Fperso 'c:\MesFormats';
options fmtsearch=(work sasuser fperso);
proc options option=fmtsearch value;
run;
```

permet de rappeler aisément comment a été alimentée l'option fmtsearch :



Cela a peu d'intérêt dans cet exemple mais peut être intéressant quand l'option est manipulée à plusieurs reprises dans une session SAS, et donc pour s'assurer des bibliothèques qu'elle référence en cas de doute.

Plutôt que d'être affichée dans la log, la valeur d'une option globale peut également être retrouvée via du code SAS, stockée dans une macro-variable, et utilisable à toutes fins utiles.

```
%let stockage_fmtsearch=%sysfunc(getoption(fmtsearch,keyword));
%put &stockage_fmtsearch;
```

Ici, la macro stockage_fmtsearch contient le texte : `FMTSEARCH=(WORK SASUSER FPERSO)` réutilisable dans une instruction options.

2. TESTER LA VALEUR DE MACRO-VARIABLES AUTOMATIQUES SAS

Une fois le choix fait sur les valeurs des options globales à SAS, donc comment SAS réagit par exemple à l'absence d'un format, d'une table, d'une variable, ou tout simplement lors de la présence d'une erreur, il est intéressant de prévoir des conditions de sortie lors de l'exécution d'un programme, ou de tester le code retour global du programme. Les macro-variables SYSERR et SYSCC sont alors utiles.

2.1. La macro variable syserr

Elle est utilisée pour détecter une erreur, que ce soit dans une étape data ou une procédure. Elle est réinitialisée à chaque nouvelle étape d'un programme (donc après chaque étape data et chaque procédure). Il convient donc d'en tester la valeur aux éventuels points critiques d'un programme.

```
/* 3 macros sont définies, puis sont appelées au sein d'une quatrième macro */
/* Si une macro ne s'exécute pas correctement, on souhaite le gérer          */

/* Commençons par créer une macro, qui teste la valeur de la macro syserr    */
/* pour décider ici de stopper la session SAS en cas d'erreur                */

%macro arret_si_erreur;
   %if &syserr > 0 %then %do;
      endsas;
   %end;
%mend;

/* 3 macros sont définies                                                    */
/* A la fin de chacune d'entre elles, la macro arret_si_erreur est appelée */
/* Donc si une erreur est rencontrée, SAS s'interrompt                       */

%macro MalereMacro;
 data test;
  x=1;
 run;
%arret_si_erreur;
```

```
%mend;

%macro Ma2emeMacro;
 proc print data=nexistepas;run;
 %arret_si_erreur;
%mend;

%macro Ma3emeMacro;
 proc print data=test;run;
  %arret_si_erreur;
%mend;

%macro appel3macros;
  %Ma1ereMacro;
  %Ma2emeMacro;
  %Ma3emeMacro;
 %mend;

%appel3macros;

/* Ici, une erreur est rencontrée dans la 2eme macro, en raison de l'inexistence
de la table appelée par la proc print. Le reste du code ne sera pas exécuté.     */
```

La macro variable syserr vaut 0 quand l'exécution s'est correctement déroulée, ou prend d'autres valeurs, en fonction du type d'erreur rencontrée. Vous pouvez vous référer à la documentation sur syserr pour plus de détails.

Terminer complètement la session SAS (comme ici avec la commande « endsas ; » dans la macro « ***arret_si_erreur*** ») n'est pas forcément le but recherché, surtout dans le cadre d'une utilisation en mode interactif. Il existe une alternative, pour interrompre la suite de l'exécution, sans fermer la session SAS. Elle consiste à utiliser `%abort cancel ;` à la place de `endsas ;`
La documentation sur %abort décrit les options disponibles et leurs impacts en mode interactif et batch.

2.2. La macro variable syscc

La macro syscc contient le code retour que SAS renvoie au système d'exploitation, une fois qu'un programme a été exécuté. La valeur 0 correspond à un succès. Dès lors qu'une erreur aura été rencontrée dans le programme, syscc sera valorisée à une valeur autre que 0, à savoir 4 en cas de « Warning » et une valeur supérieure en cas d'erreur.
Un cas de valorisation de cette macro-variable automatique a été vu lors de la présentation de l'option errorcheck.

Syscc est également la macro qui est utilisée pour vérifier le statut final d'exécution d'un programme SAS exécuté en batch. Sur Windows, un batch SAS se programme avec une instruction de ce type :
"C:\Program Files\SASHome2\SASFoundation\9.4\sas.exe" –sysin "c:\batch\monbatch.sas" –log "c:\batch\logs\monbatch.log" –icon –nosplash
L'option –log est optionnelle mais permet de spécifier dans quel répertoire le fichier log sera créé.
Les options icon et nosplash permettent d'éviter de voir apparaître des fenêtres SAS lors de l'exécution du batch.
La commande d'exécution d'un batch SAS peut ensuite être intégrée dans un script Windows, où la valeur de la variable %ERRORLEVEL% pourra être testée, pour décider de telle ou telle action, en fonction de sa valeur. Par exemple, on peut décider d'envoyer un mail en cas d'erreur.

Les macro-variables syserr et syscc sont donc bien différentes et à utiliser dans des contextes différents. Syscc aura une valeur non nulle dès lors qu'un warning ou une erreur sera intervenue dans le programme. Syserr peut avoir une valeur non nulle à un moment donné lors de l'exécution du programme, et une valeur nulle en fin de programme (valeur correspondante à l'exécution de la dernière étape data ou procédure).

3. EN CAS DE PROBLEME

3.1. Eléments à transmettre au Support Clients

Si vous rencontrez des problèmes lors de l'utilisation de ces options et macro-variables, vous pouvez nous écrire à support@sas.com, en attachant à votre message l'erreur reçue.

4. LIENS UTILES

Dans la documentation SAS 9.4 :

- SAS 9.4 System Options : Reference, fourth edition
- Exécuter SAS en batch

5. CONCLUSION

La connaissance des options et macro-variables décrites ici est un plus, lors du développement de vos programmes, en particulier lorsque ces derniers sont amenés à être déployés en batch. En effet, la syntaxe aura pu être validée lors de l'écriture et le test du code SAS, mais l'exécution en batch peut potentiellement parfois échouer, pour une cause externe à SAS. Par exemple, une bibliothèque qui ne peut pas être affectée (en raison d'une indisponibilité du disque physique où elle est définie par exemple, ou une indisponibilité de la base de données où elle est référencée, etc), un répertoire supprimé ou déplacé par erreur, et donc un appel à un programme SAS qui ne peut se faire, etc. Prévoir des clauses de sortie de programme, et tester le code retour du batch SAS est donc très important, pour éviter des problèmes d'intégrité de données (tables SAS remplacées, mais de façon partielle par exemple) ou pour être informé de tout problème (via un envoi de mail en cas de souci).

Karine CHRILLESEN
Consultante Support Clients SAS France