Statistical Analysis System ou **Système SAS**

Rappels & Mise à niveau

Michel Dubois

Michel.Dubois@univ-ubs.fr

Remerciements

- **Anthony GUILLOUZO**

Pour la partie SAS BASE de ce document

- **Hugues Gérard**

Pour la partie SAS CONNECT

PLAN

- **Chapitre 1**

SAS BASE

- **Chapitre 2**

SAS Macro

- **Chapitre 3**

SAS CONNECT

Formation

SAS/BASE

Objectifs de la formation

- **Rappel du « noyau » de SAS**
- **Connaissance de l'environnement SAS**
- **Gestion des données**
- **Analyse des données**
- **Présentation des résultats**

Sommaire

- **Présentation de l'environnement Windows & unix**
- **Les éléments SAS**
- **Le langage SAS**
- **La proc SQL et le dictionnaire de données**
- **Introduction à l'ODS**
- **Méthodologie**

Présentation de SAS

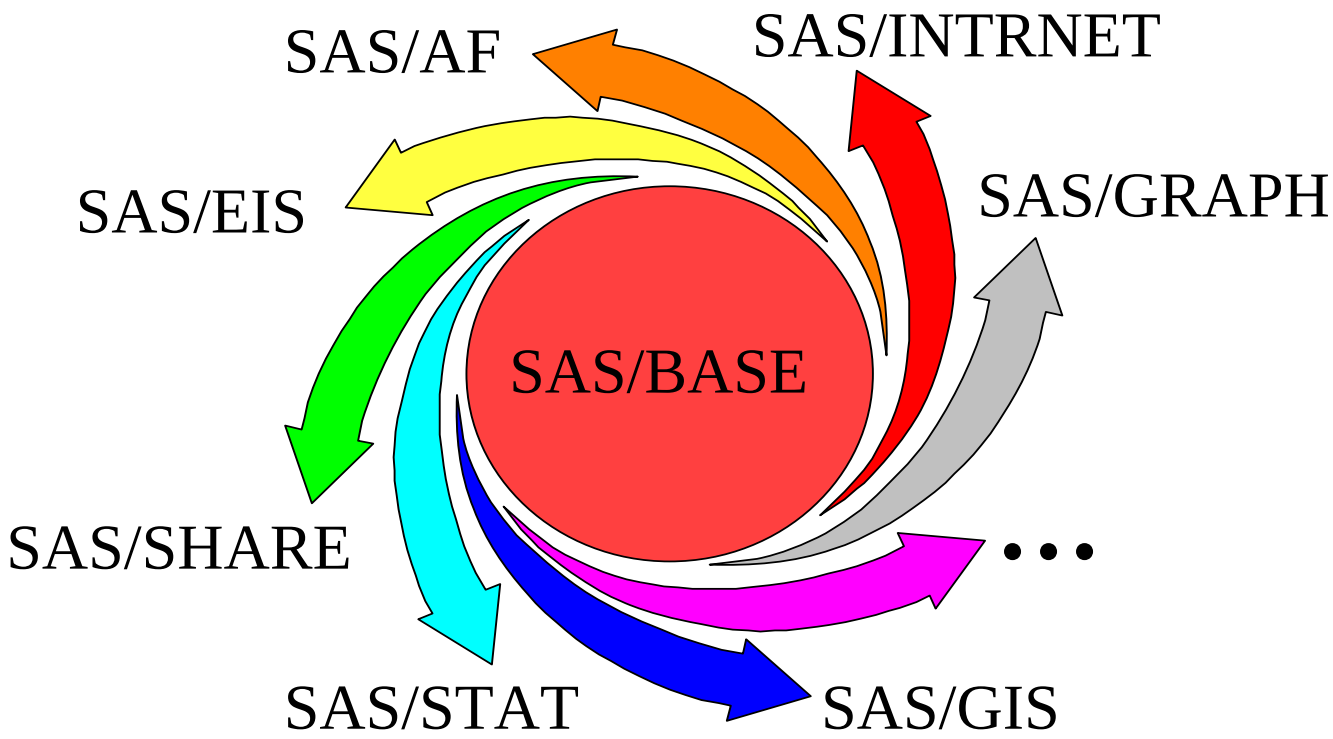
Introduction à SAS

- Créée en 1976 sous l'impulsion de Jim GOODNIGHT, actuel CEO de la société.
- A son origine, SAS signifiait Statistical Analysis System : logiciel de traitement et d'analyse de données
- Aujourd'hui, SAS n'est plus considéré comme un logiciel « Statistique », nous parlons du Système SAS.

Présentation de SAS

- SAS permet :
  - De lire, traiter, créer des fichiers externes
  - De traiter des tables SAS
  - D'effectuer des calculs de statistiques descriptives et de l'analyse de données
  - De représenter des données sous forme graphique : tableaux, histogrammes, courbes, camemberts, cartes, …
  - De développer des applications (/client/serveur, internet, …)
  - …
- Outil d'aide à la décision

Les modules


SAS/AF
SAS/INTRNET
SAS/EIS
SAS/GRAPH
SAS/BASE
SAS/SHARE
• • •
SAS/STAT
SAS/GIS

Sommaire

- **Présentation de l'environnement**
- Les éléments SAS
- Le langage SAS
- La proc SQL et le dictionnaire de données
- Introduction à l'ODS
- Méthodologie

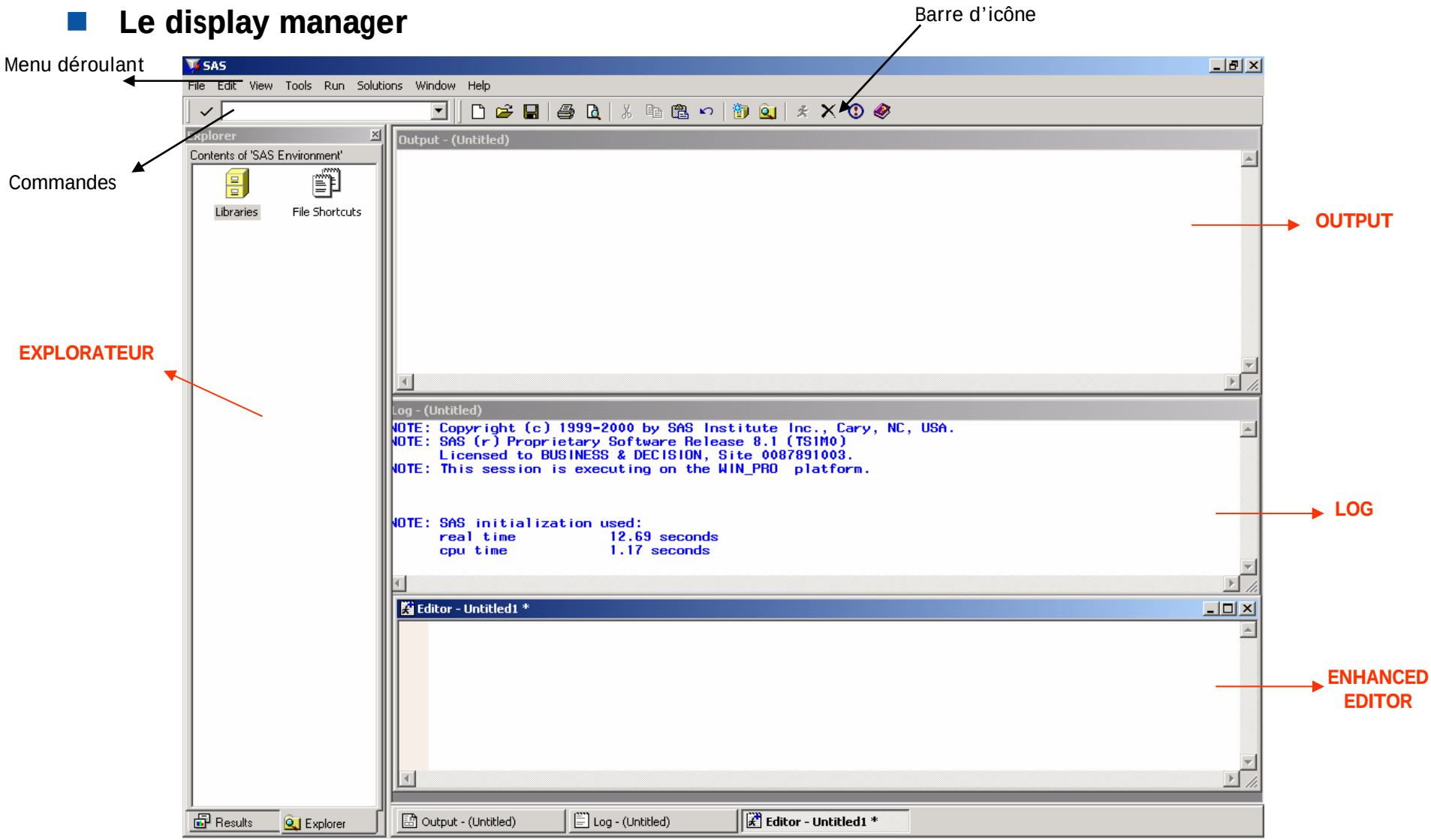
Présentation de l'environnement

- **Le lancement de sas**

- **Le display manager**
  - Présentation générale :
    - PROGRAM EDITOR Écriture et exécution des programmes
    - ENHANCED EDITOR Version améliorée du PROGRAM EDITOR
    - LOG Compte rendu de l'exécution du programme
    - OUTPUT Résultat de l'exécution
    - EXPLORER Explorateur pour les données
    - RESULT Explorateur pour les sorties
  - Comment retrouver une commande
  - Les touches de fonctions

Présentation de l'environnement

Le display manager



Présentation de l'environnement

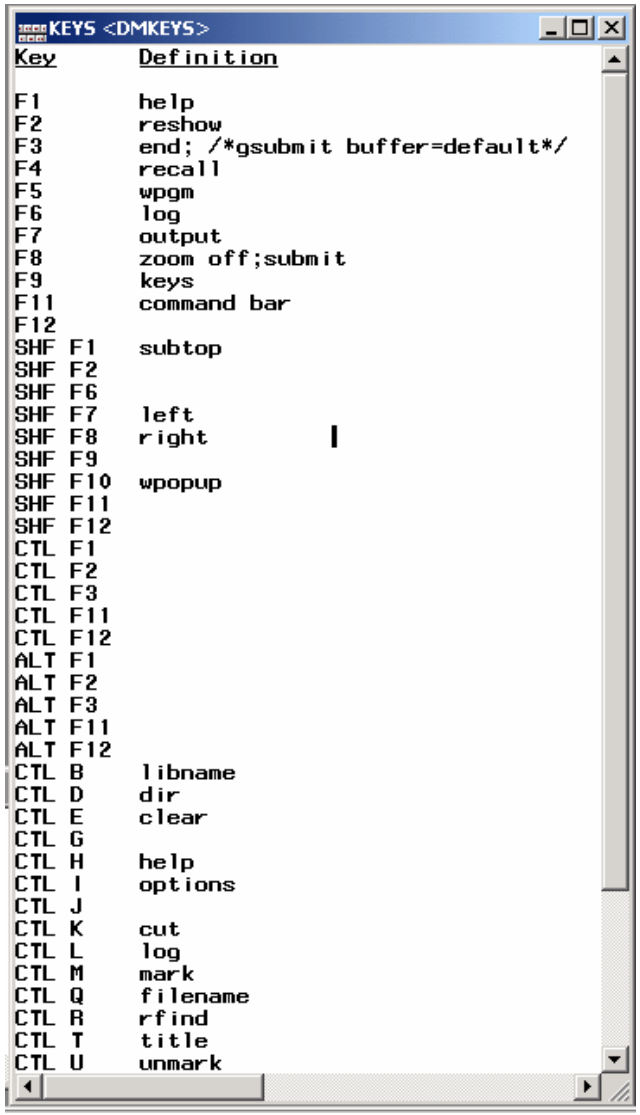
Les commandes (quelques exemples)

Clear	Vide le contenu de la fenêtre active
Find	Permet de trouver un chaîne de caractère
Open	Ouvre une table avec l'éditeur Viewtable
Fsedit	Ouvre une table avec l'éditeur Fsedit (mode ligne)
Fsview	Ouvre une table avec l'éditeur Fsview (mode table)
Help	Aide
Keys	Ouvre la fenêtre des touches de fonctions
Libassign	Ouvre la fenêtre pour assigner un nouveau libname
Rfind	Permet de retrouver un chaîne de caractères
Option	Ouvre la fenêtre des options de l'environnement SAS
Submit	Exécute le contenu de l'éditeur
X	Commande système

Souvent les 3 premières lettres de la commande suffisent (Exemple : Lib au lieu de libassign)

Présentation de l'environnement

- **Les touches de fonctions**

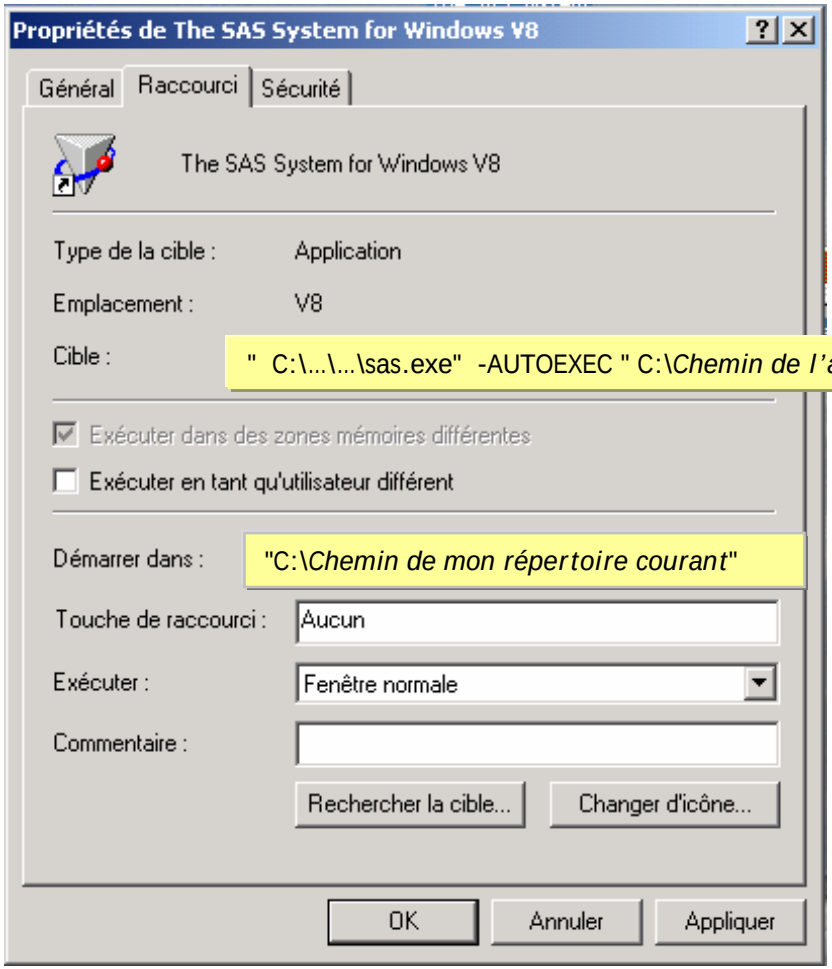


Présentation de l'environnement

- **Les fichiers de configuration d'une session**
  - L'AUTOEXEC
    - Il se nomme normalement « autoexec.sas »
    - Permet d'exécuter automatiquement un programme SAS stocké dans un fichier séquentiel
    - Permet l'allocation automatique des librairies nécessaires pour une session
    - Permet de modifier les options de l'environnement
    - N'est pas obligatoire
  - Le CONFIG
    - Il se nomme normalement « sasv8.cfg »
    - Contient les paramètres logiciels nécessaires pour lancer SAS
    - Permet le positionnement d'options et personnalise un environnement applicatif particulier.
    - Est obligatoire

Présentation de l'environnement

- **Leur paramétrage se fait au niveau des propriétés du raccourci.**



Présentation de l'environnement

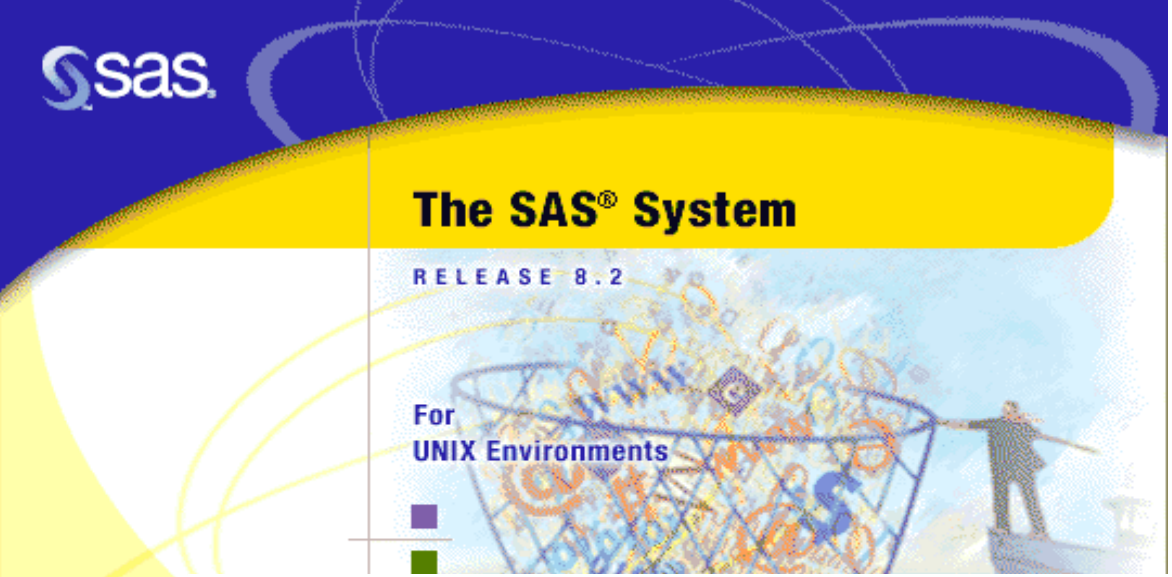
- **Le mode de recherche de ces fichiers est différent selon le positionnement d'options ou non**

- **La démarche est la suivante :**

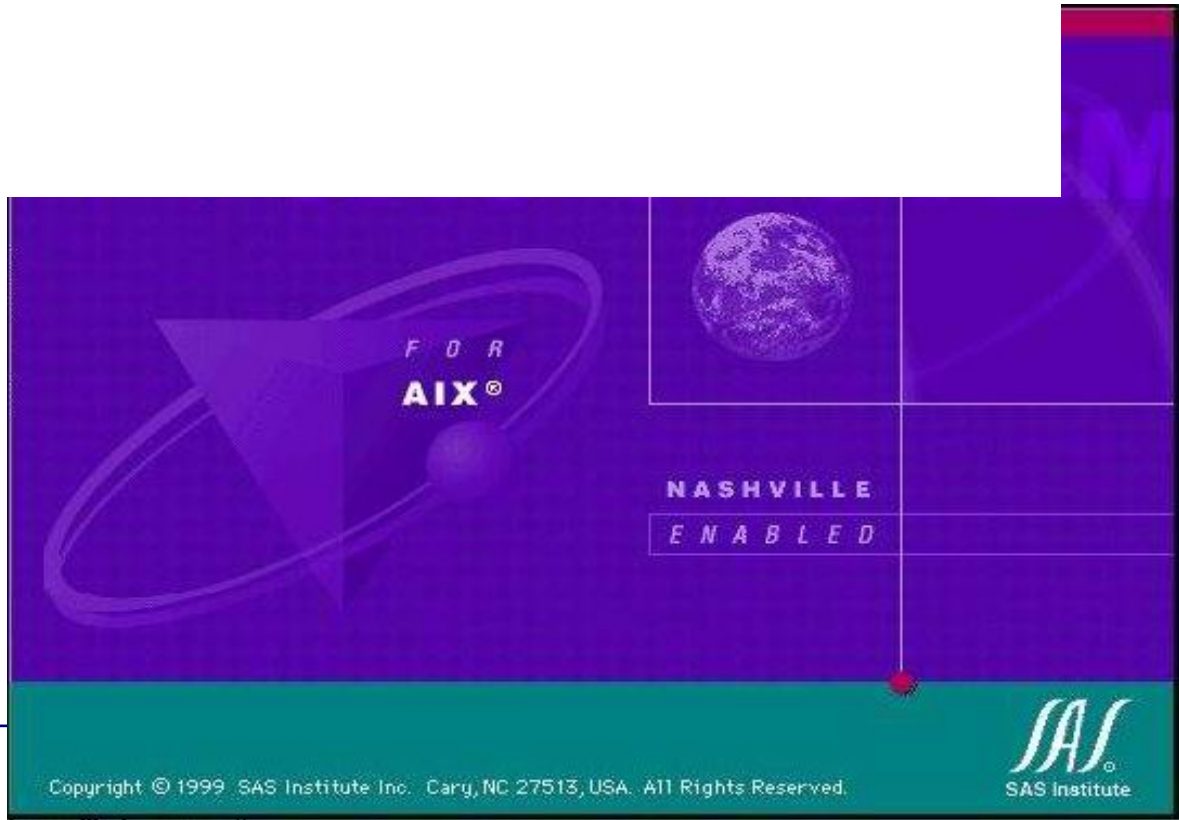
  1. Recherche des fichiers dans la ligne de commande
  2. Recherche des fichiers dans le répertoire nommé dans « Démarrer dans »
  3. Recherche des fichiers dans la racine du logiciel SAS

- **A noter :**

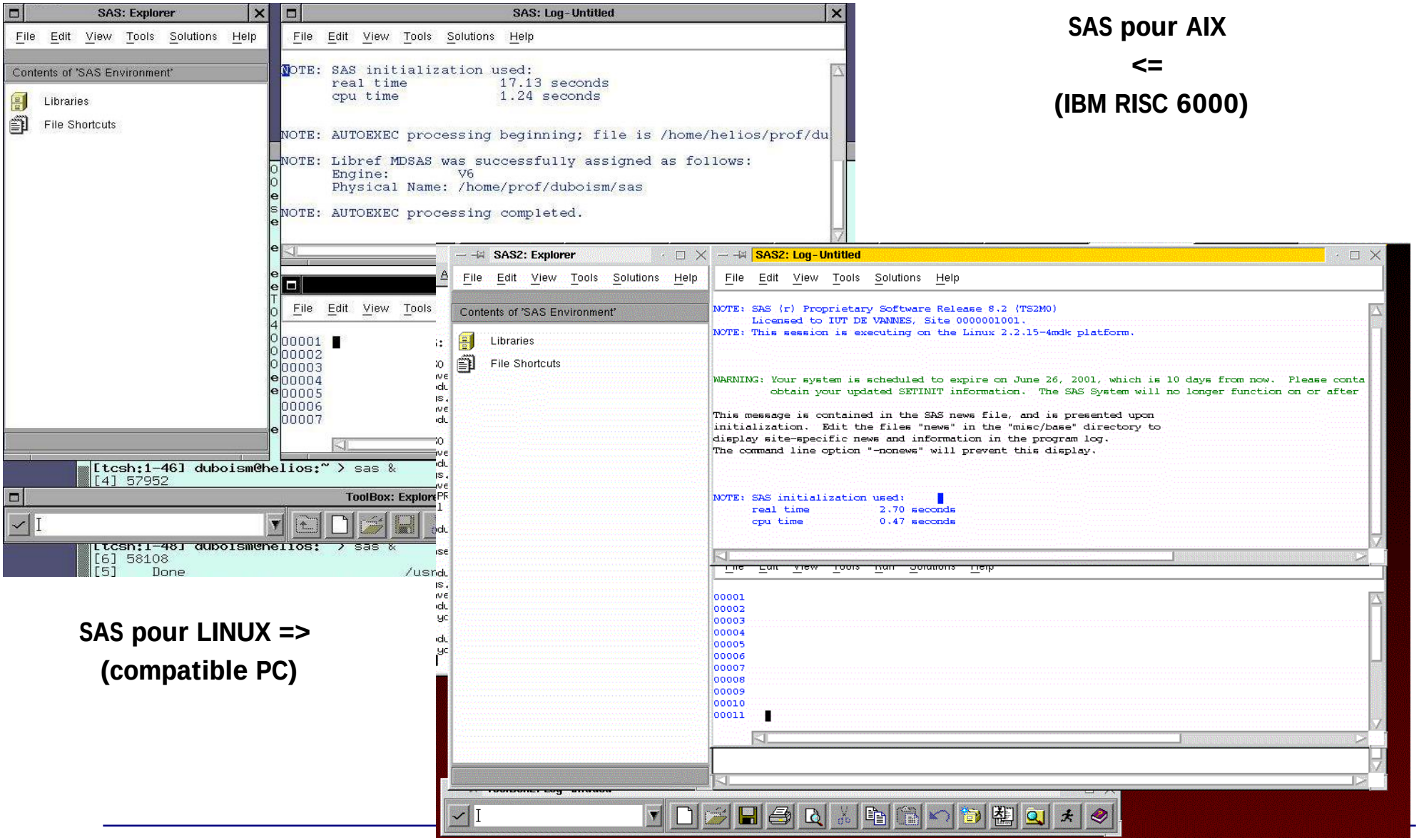
  - Le paramétrage du fichier « autoexec » (autoexec.sas) peut être réalisé dans le fichier « config » (sasv8.cfg)
  - Les fichiers « autoexec.sas » et « sasv8.cfg » peuvent changer de nom



SAS pour UNIX



SAS sous UNIX



SAS pour AIX

<=

(IBM RISC 6000)

SAS pour LINUX =>

(compatible PC)

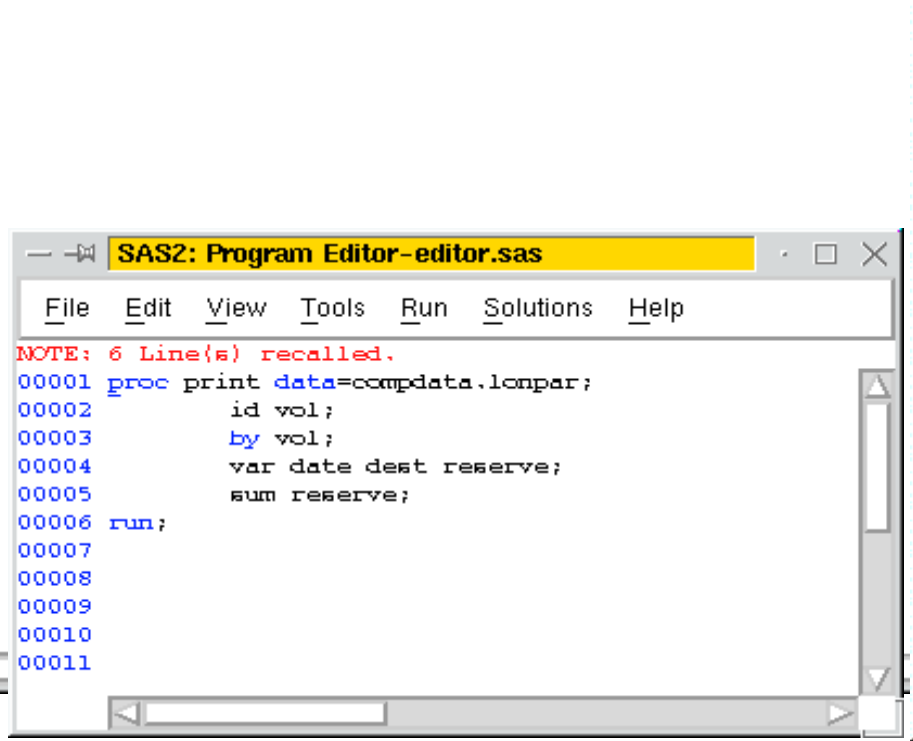
PLAN

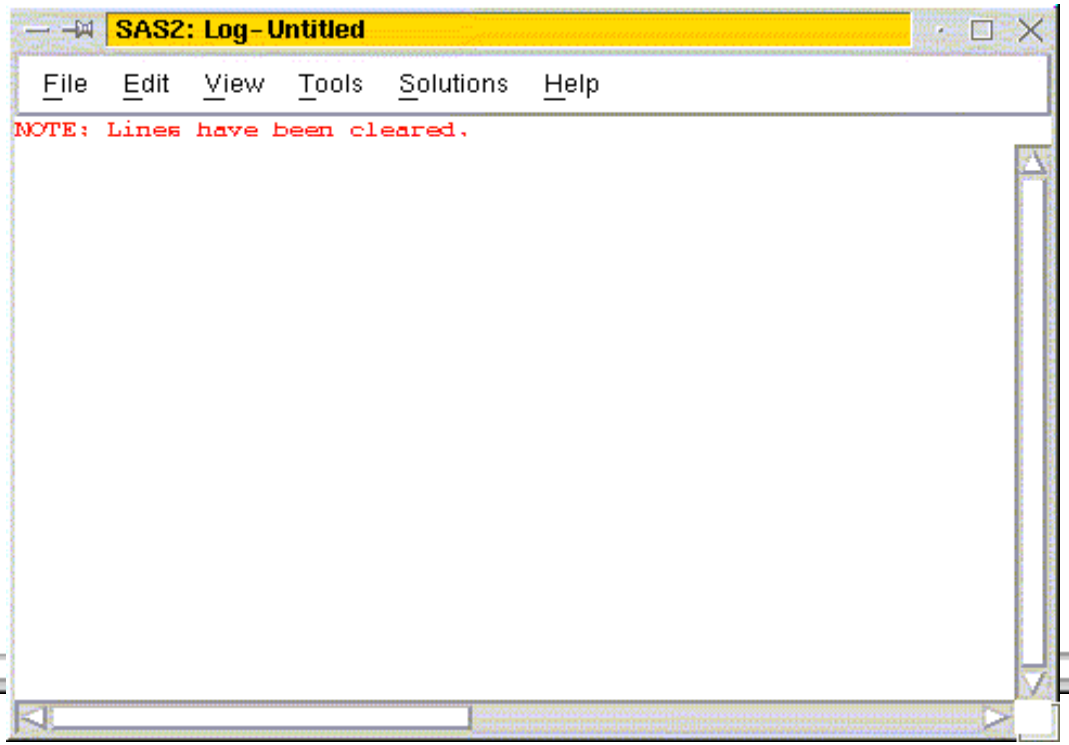
- **Editer / Corriger un programme**
- **Porter/Importer des bibliothèques**
- **Mode client/serveur**

Editer et corriger un programme

Le programme doit être soumis puis corrigé ou complété.

Son fonctionnement est suivi dans les fenêtres de programmation.





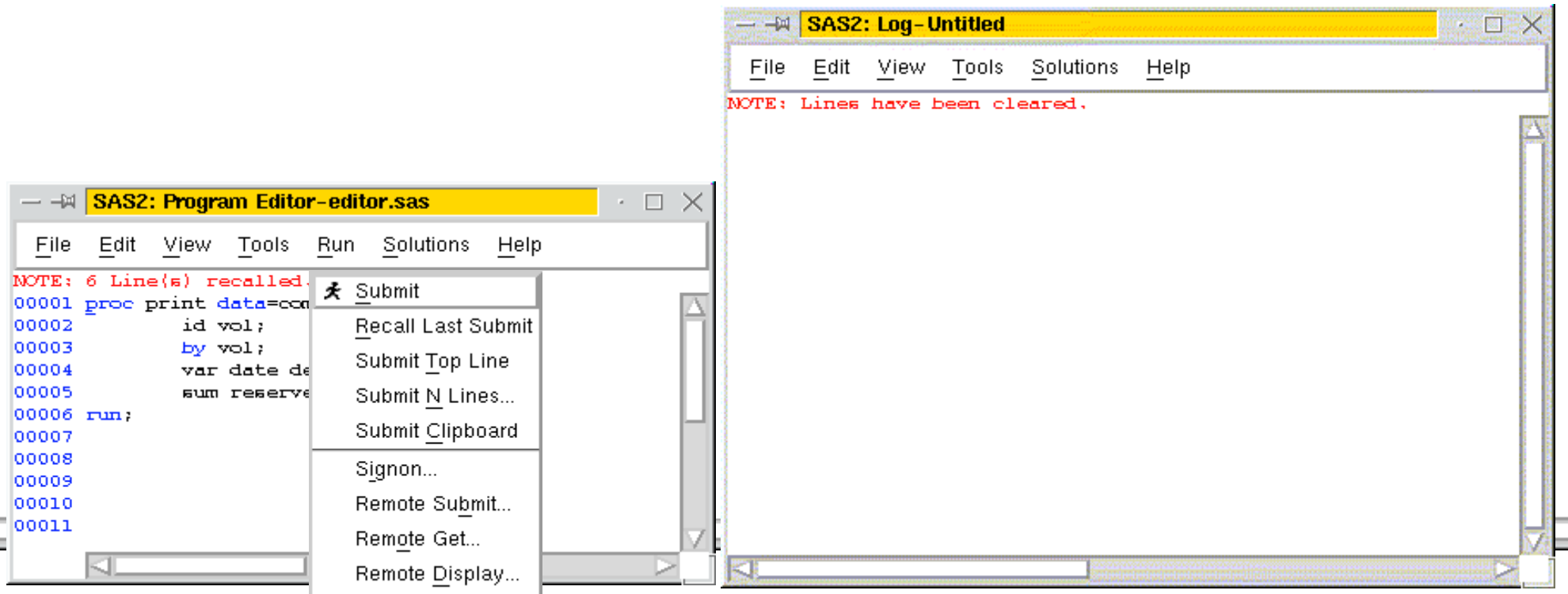
Editer et corriger un programme

SAS2: Output-Untitled

Edit View Tools Solutions Help

Pour soumettre votre programme :

RUN->SUBMIT ou la commande SUBMIT (SUB) ou la touche F3.



Editer et corriger un programme

SAS2: Output - Untitled

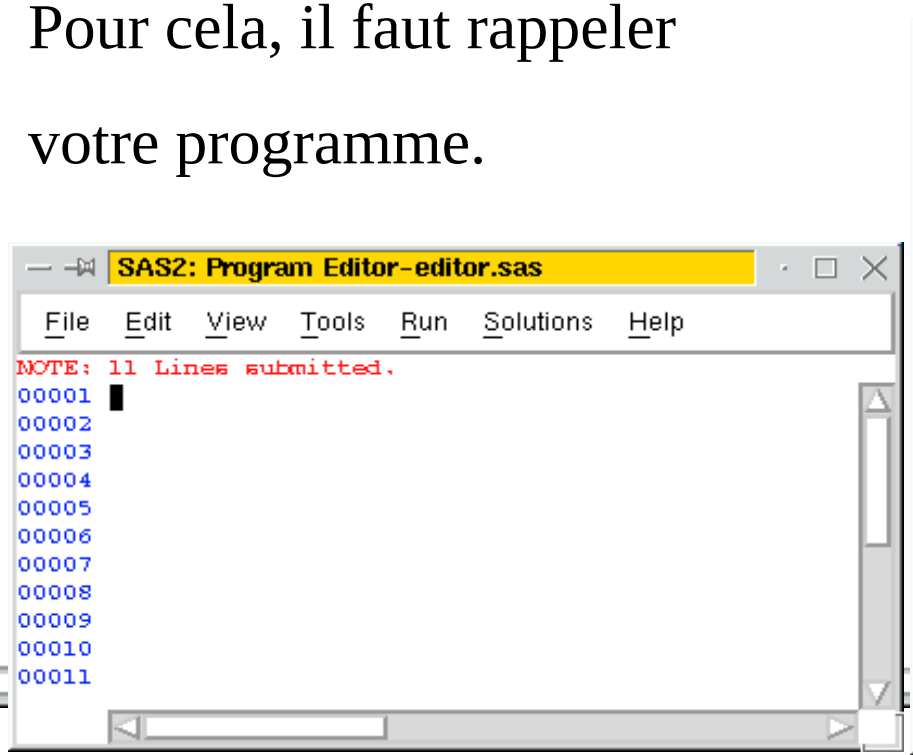
Edit View Tools Solutions Help

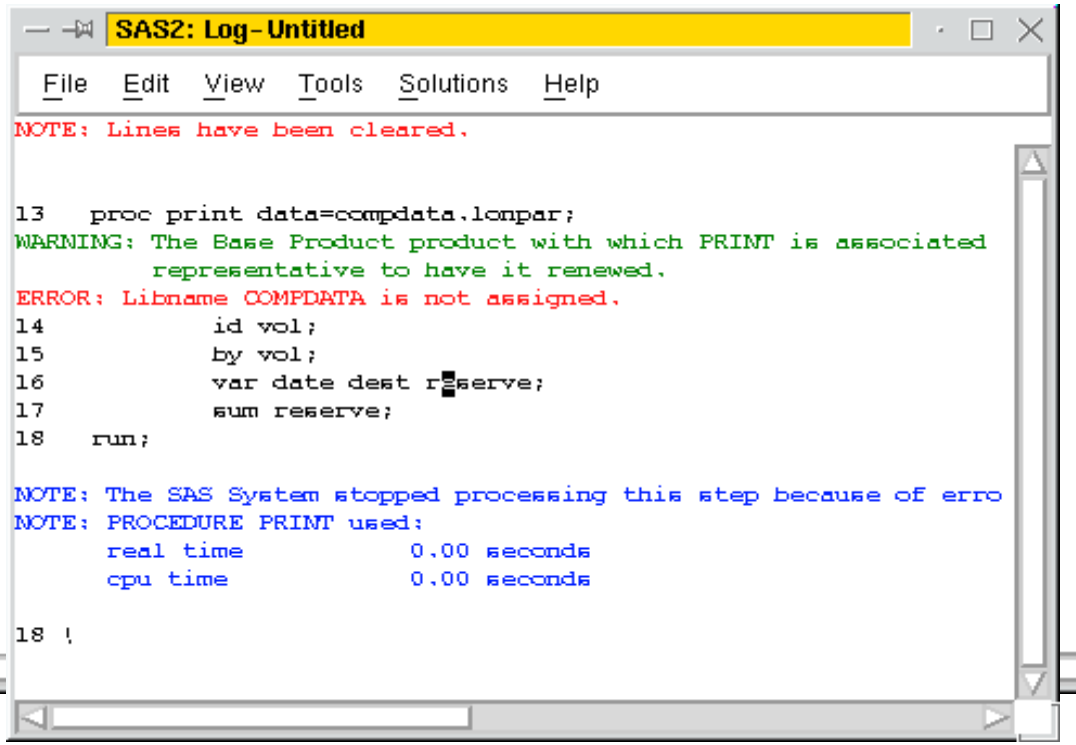
Des erreurs sont signalées (ERROR en Rouge).

Elles doivent être corrigées pour que le programme fonctionne.

Pour cela, il faut rappeler

votre programme.



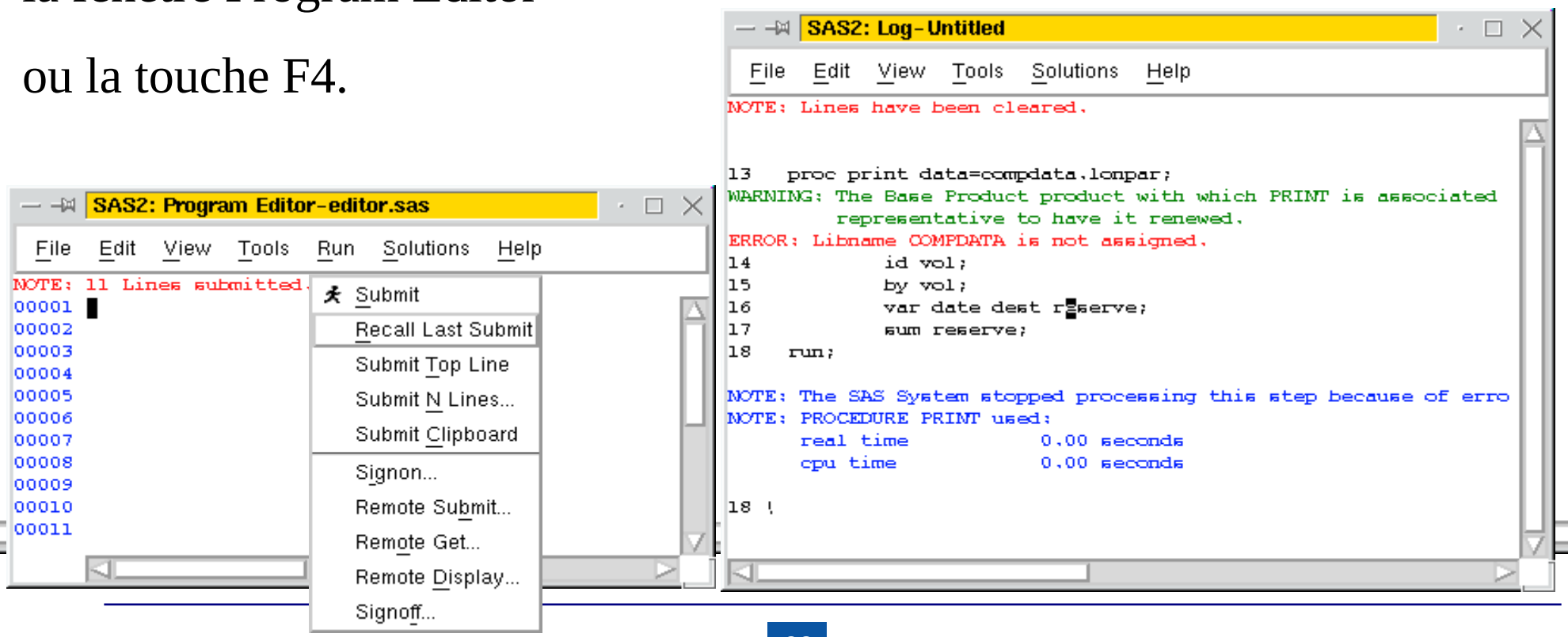


Editer et corriger un programme

Pour rappeler votre programme :

RUN->RECALL LAST SUBMIT ou la commande RECALL (REC) dans la fenêtre Program Editor

ou la touche F4.

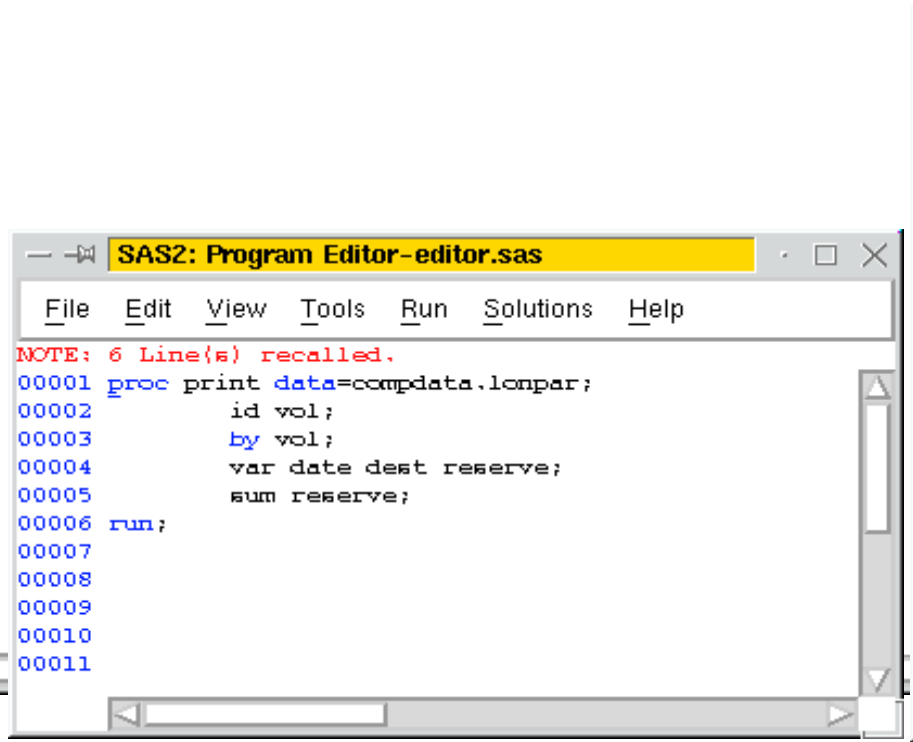


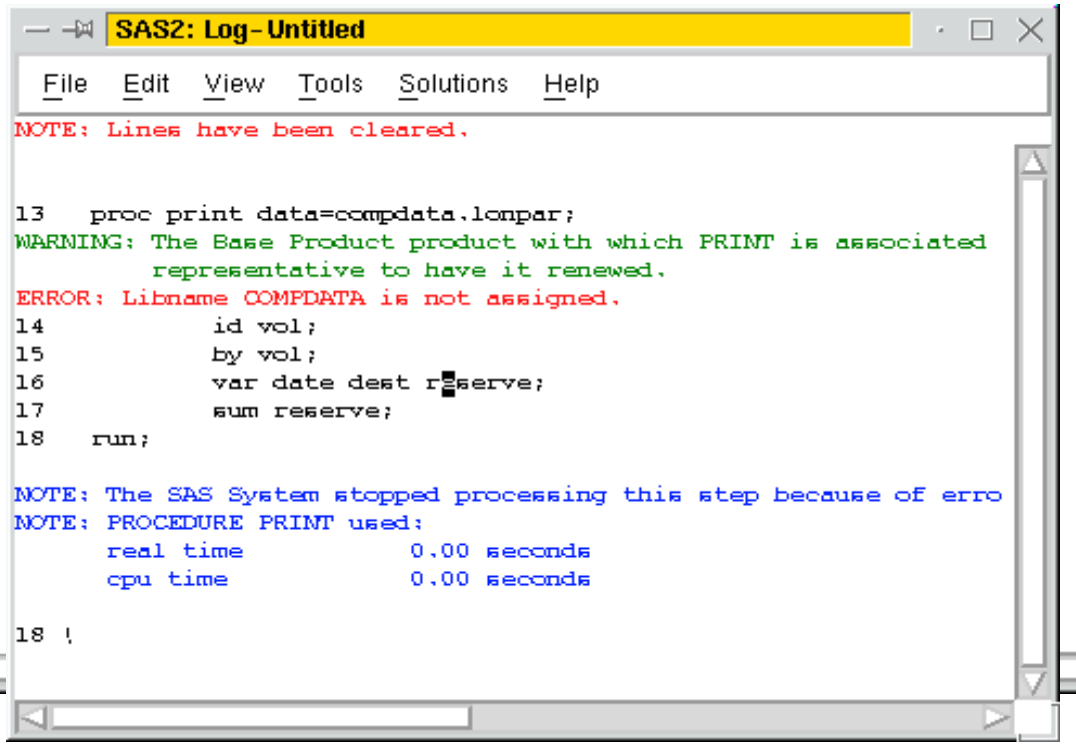
Editer et corriger un programme

SAS2: Output - Untitled

Edit View Tools Solutions Help

Diagnostic : Le libname COMPDATA n'est pas assigné.





Editer et corriger un programme

SAS2: Output - Untitled

Edit View Tools Solutions Help

Diagnostic : Le libname COMPDATA n'est pas assigné.

SAS doit l'assigner avant d'exécuter la proc print.

La dernière ligne doit être soumise.

SAS2: Program Editor - editor.sas

File Edit View Tools Run Solutions Help

```
NOTE: 7 line(s) included.
00001 proc print data=compdata.lonpar;
00002          id vol;
00003          by vol;
00004          var date dest reserve;
00005          sum reserve;
00006 run;
00007 libname compdata '/root/CoursSAS/sasbase/data';
00008
00009
00010
00011
```

SAS2: Log - Untitled

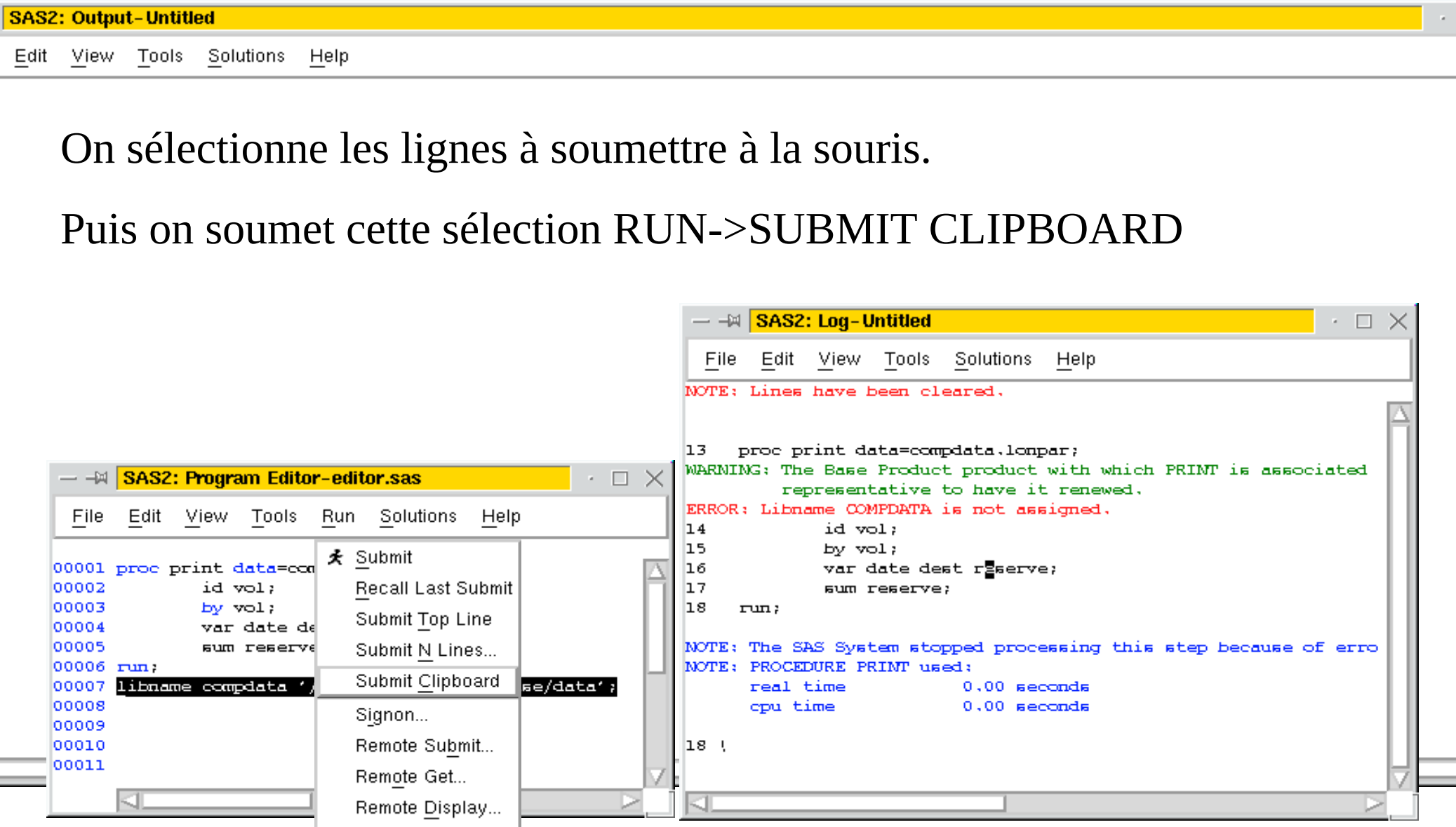
File Edit View Tools Solutions Help

```
NOTE: Lines have been cleared.


13   proc print data=compdata.lonpar;
WARNING: The Base Product product with which PRINT is associated
         representative to have it renewed.
ERROR: Libname COMPDATA is not assigned.
14           id vol;
15           by vol;
16           var date dest reserve;
17           sum reserve;
18   run;

NOTE: The SAS System stopped processing this step because of erro
NOTE: PROCEDURE PRINT used:
      real time             0.00 seconds
      cpu time              0.00 seconds

18 !
```

Editer et corriger un programme

On sélectionne les lignes à soumettre à la souris.

Puis on soumet cette sélection RUN->SUBMIT CLIPBOARD

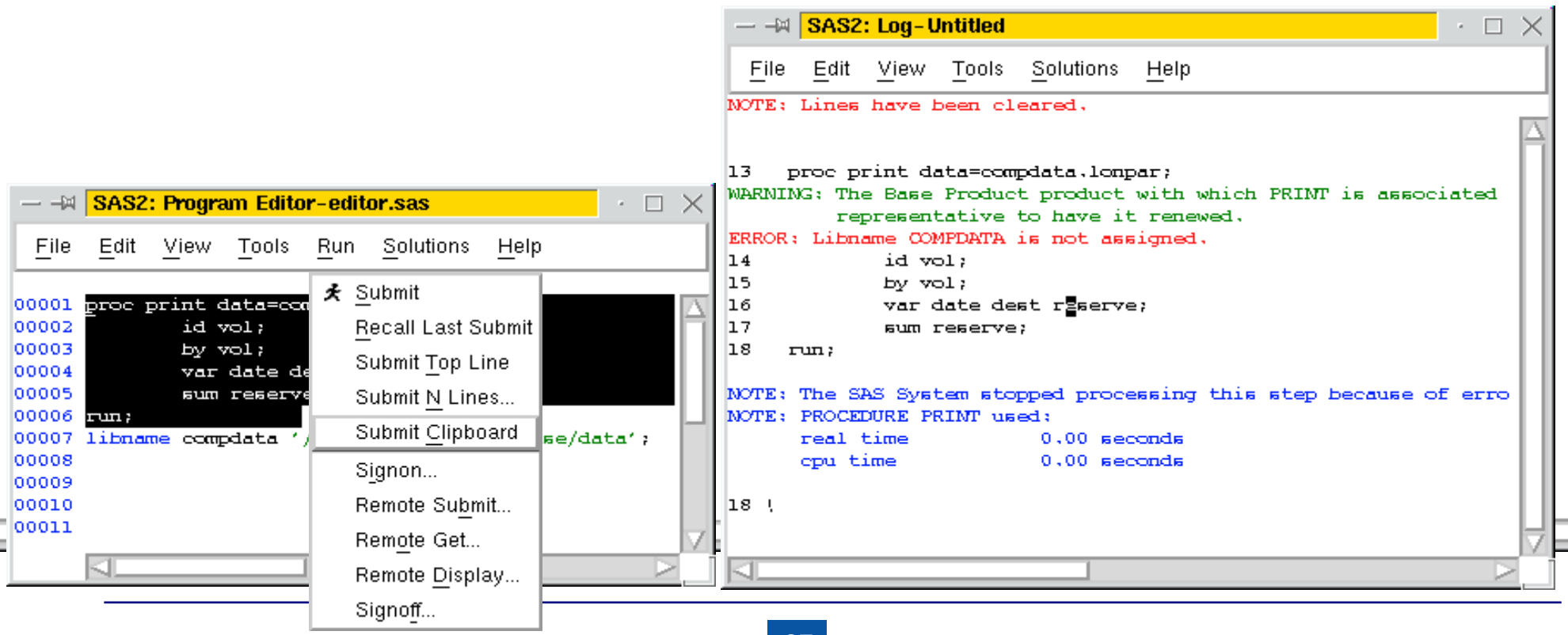


Editer et corriger un programme

SAS2: Output - Untitled

Edit View Tools Solutions Help

On sélectionne la proc print et on la soumet le presse papier.



Editer et corriger un programme

SAS2: Output-Untitled

Edit View Tools Solutions Help

Un erreur : l'énoncé BY dans la proc print nécessite que la table soit triée d'abord selon la variable.

On va utiliser une proc sort.

SAS2: Program Editor-editor.sas

File Edit View Tools Run Solutions Help

```
NOTE: 7 line(s) included.
00001 proc print data=compdata.lonpar;
00002          id vol;
00003          by vol;
00004          var date dest reserve;
00005          sum reserve;
00006 run;
00007 libname compdata '/root/CoursSAS/sasbase/data';
00008
00009
00010
00011
```

SAS2: Log-Untitled

File Edit View Tools Solutions Help

```
NOTE: Lines have been cleared.
26    proc print data=compdata.lonpar;
WARNING: The Base Product product with which PRINT is associated
         representative to have it renewed.
27            id vol;
28            by vol;
29            var date dest reserve;
30            sum reserve;
31    run;

ERROR: Data set COMPDATA.LONPAR is not sorted in ascending sequen
NOTE: The SAS System stopped processing this step because of erro
NOTE: There were 3 observations read from the data set COMPDATA.L
NOTE: PROCEDURE PRINT used:
      real time             0.96 seconds
      cpu time              0.03 seconds

31 !
WARNING: The Base Product product with which FSList Window is ass
         representative to have it renewed.
```

Editer et corriger un programme

SAS2: Output-Untitled

Edit View Tools Solutions Help

On sélectionne la proc sort et on soumet le presse papier.

On sélectionne la proc print et on soumet le presse papier.

L'erreur est persistante.

On va ordonner le code.

SAS2: Program Editor-editor.sas

File Edit View Tools Run Solutions Help

```
00001 proc print data=compdata.lonpar;
00002          id vol;
00003          by vol;
00004          var date dest reserve;
00005          sum reserve;
00006 run;
00007 libname compdata '/root/CoursSAS/sasbase/data';
00008 proc sort data=compdata.lonpar out=sorted;
00009  by vol;
00010 run;
00011 _
```

SAS2: Log-Untitled

File Edit View Tools Solutions Help

```
NOTE: Lines have been cleared.
26    proc print data=compdata.lonpar;
WARNING: The Base Product product with which PRINT is associated
         representative to have it renewed.
27            id vol;
28            by vol;
29            var date dest reserve;
30            sum reserve;
31    run;

ERROR: Data set COMPDATA.LONPAR is not sorted in ascending sequen
NOTE: The SAS System stopped processing this step because of erro
NOTE: There were 3 observations read from the data set COMPDATA.L
NOTE: PROCEDURE PRINT used:
      real time             0.96 seconds
      cpu time              0.03 seconds

31 !
WARNING: The Base Product product with which FSList Window is ass
         representative to have it renewed.
```

Editer et corriger un programme

SAS2: Output - Untitled

Edit View Tools Solutions Help

La proc print doit être à la fin du code.

On utilise la commande de numéro de ligne MM (déplacer un bloc).

On se place sur le numéro

de la ligne du début du bloc.

SAS2: Program Editor - editor.sas

File Edit View Tools Run Solutions Help

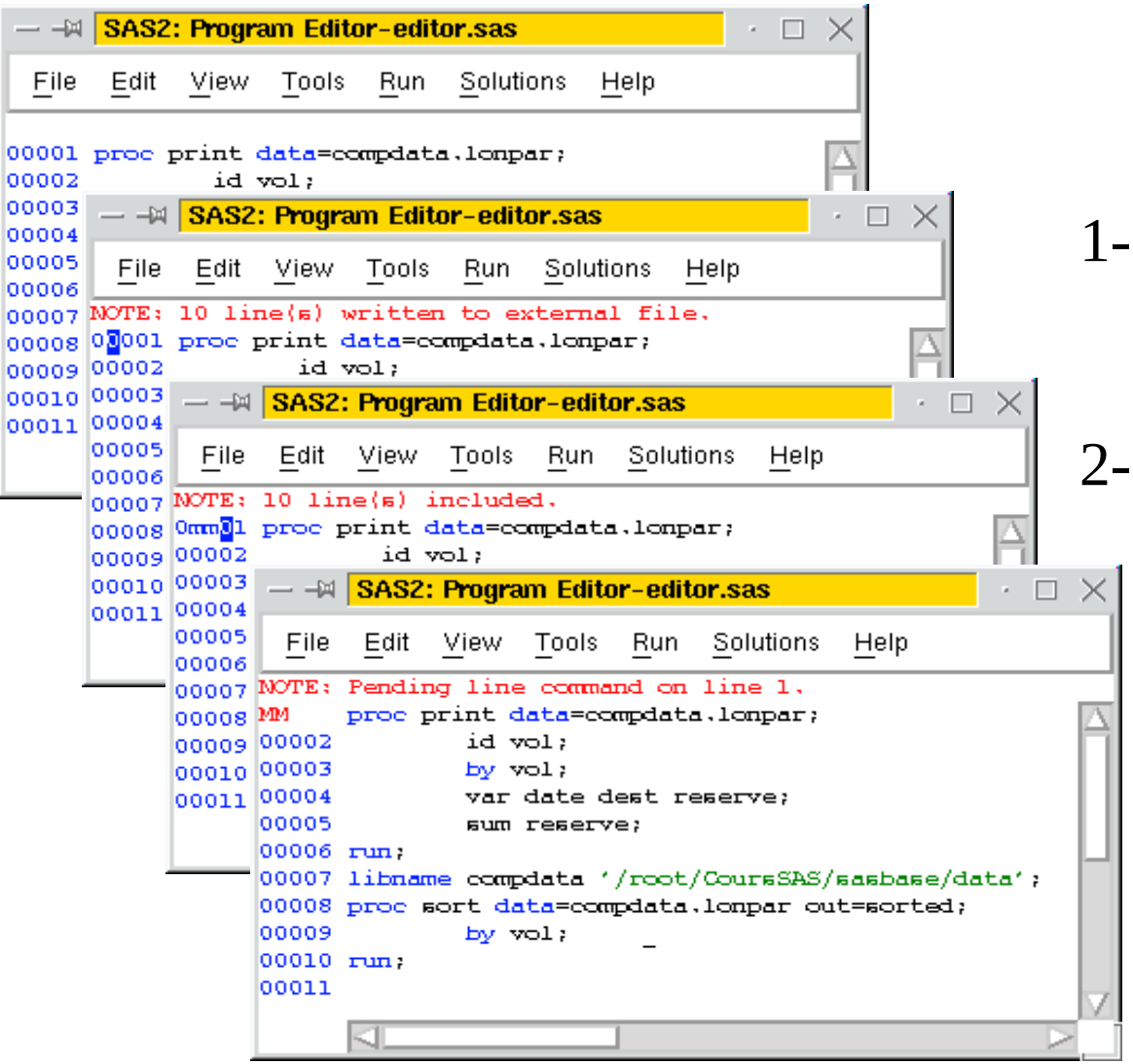
```
NOTE: 10 line(s) written to external file.
00001 proc print data=compdata.lonpar;
00002          id vol;
00003          by vol;
00004          var date dest reserve;
00005          sum reserve;
00006 run;
00007 libname compdata '/root/CoursSAS/sasbase/data';
00008 proc sort data=compdata.lonpar out=sorted;
00009  by vol;
00010 run;
00011
```

SAS2: Log - Untitled

File Edit View Tools Solutions Help

```
NOTE: Lines have been cleared.
26    proc print data=compdata.lonpar;
WARNING: The Base Product product with which PRINT is associated
         representative to have it renewed.
27            id vol;
28            by vol;
29            var date dest reserve;
30            sum reserve;
31    run;

ERROR: Data set COMPDATA.LONPAR is not sorted in ascending sequen
NOTE: The SAS System stopped processing this step because of erro
NOTE: There were 3 observations read from the data set COMPDATA.L
NOTE: PROCEDURE PRINT used:
      real time             0.96 seconds
      cpu time              0.03 seconds

31 !
WARNING: The Base Product product with which FSList Window is ass
         representative to have it renewed.
```

Commande de numéro de ligne MM

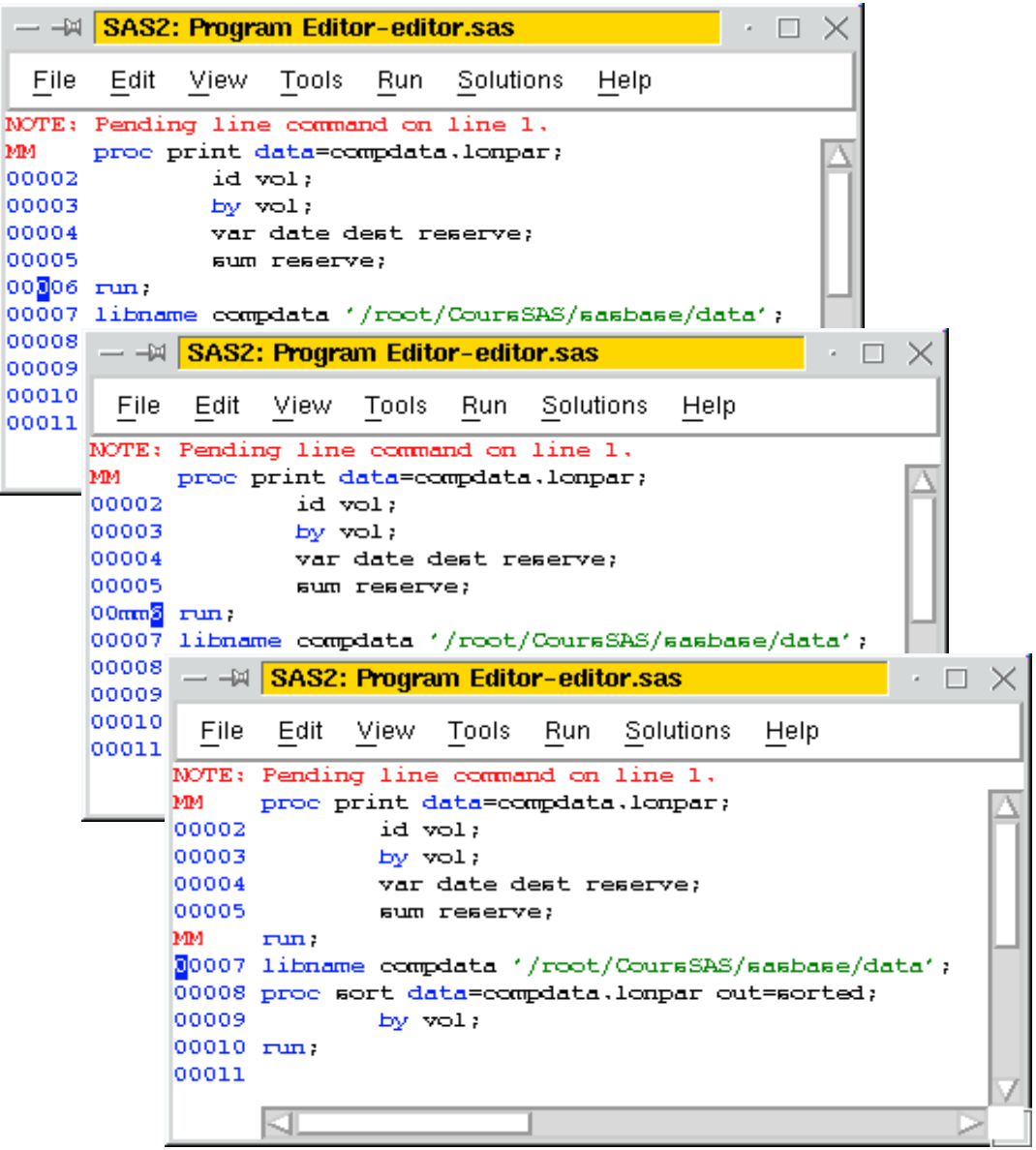


1- On se place sur le numéro de ligne du début du bloc.

2- On tape MM.

3- On valide avec ENTER.

Commande de numéro de ligne MM

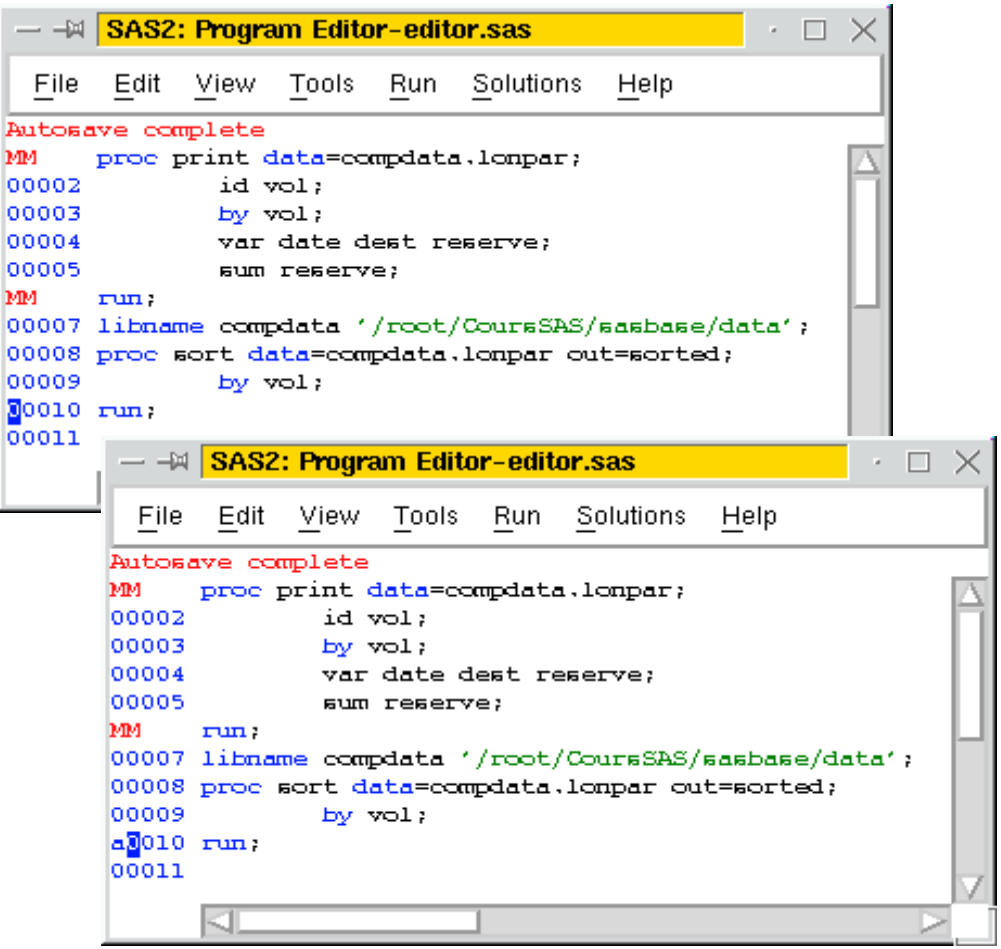


1- On se place sur le numéro de ligne de la fin du bloc.

2- On tape MM.

3- On valide avec ENTER.

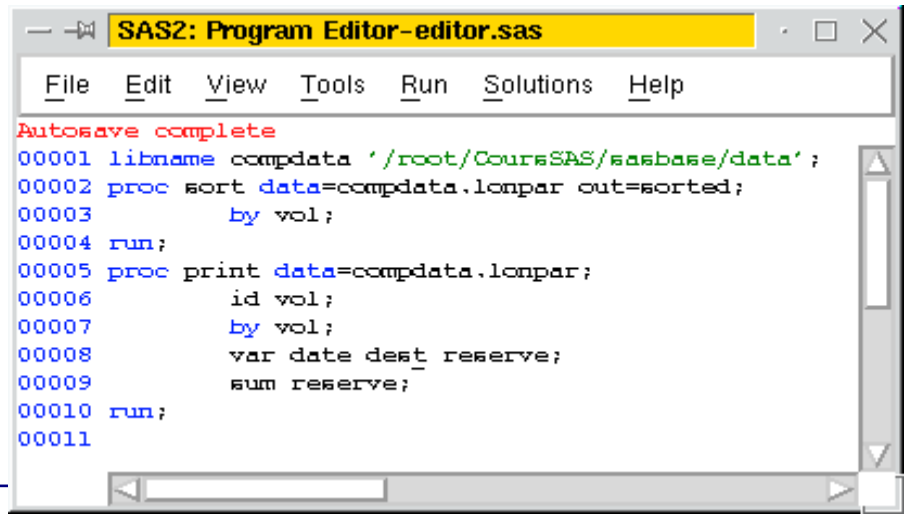
Commande de numéro de ligne MM



1- On se place sur le numéro de la ligne qui précède directement l'endroit de destination du déplacement du bloc.

2- On tape A pour AFTER.

B pour BEFORE.



3- On tape ENTER pour valider.

Le bloc est alors déplacé.

Editer et corriger un programme

SAS2: Output-Untitled

Edit View Tools Solutions Help

La proc print ne doit plus concerner la table SAS compdata.lonpar mais la table SAS work.sorted. On va effacer compdata grâce à la touche Del ou Suppr

SAS2: Program Editor-editor.sas

File Edit View Tools Run Solutions Help

```
Autosave complete
00001 libname compdata '/root/CoursSAS/sasbase/data';
00002 proc sort data=compdata.lonpar out=sorted;
00003          by vol;
00004 run;
00005 proc print data=compdata.lonpar;
00006          id vol;
00007          by vol;
00008          var date dest reserve;
00009          sum reserve;
00010 run;
00011
```

SAS2: Log-Untitled

File Edit View Tools Solutions Help

```
NOTE: Lines have been cleared.
26    proc print data=compdata.lonpar;
WARNING: The Base Product product with which PRINT is associated
         representative to have it renewed.
27            id vol;
28            by vol;
29            var date dest reserve;
30            sum reserve;
31    run;

ERROR: Data set COMPDATA.LONPAR is not sorted in ascending sequen
NOTE: The SAS System stopped processing this step because of erro
NOTE: There were 3 observations read from the data set COMPDATA.L
NOTE: PROCEDURE PRINT used:
      real time             0.96 seconds
      cpu time              0.03 seconds

31 !
WARNING: The Base Product product with which FSList Window is ass
         representative to have it renewed.
```

Editer et corriger un programme

SAS2: Output - Untitled

Edit View Tools Solutions Help

Il faut remplacer lonpar par sorted.

On bascule du mode insertion au mode remplacement grâce à la touche INSERT.

SAS2: Program Editor - editor.sas

File Edit View Tools Run Solutions Help

```
Autosave complete
00001 libname compdata '/root/CoursSAS/sasbase/data';
00002 proc sort data=compdata.lonpar out=sorted;
00003          by vol;
00004 run;
00005 proc print data=lonpar;
00006          id vol;
00007          by vol;
00008          var date dest reserve;
00009          sum reserve;
00010 run;
00011
```

SAS2: Log - Untitled

File Edit View Tools Solutions Help

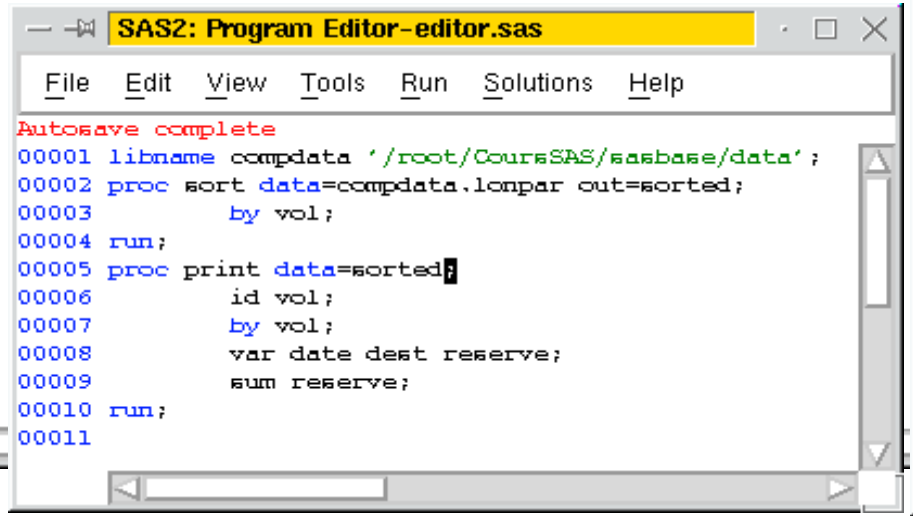
```
NOTE: Lines have been cleared.
26    proc print data=compdata.lonpar;
WARNING: The Base Product product with which PRINT is associated
         representative to have it renewed.
27            id vol;
28            by vol;
29            var date dest reserve;
30            sum reserve;
31    run;

ERROR: Data set COMPDATA.LONPAR is not sorted in ascending sequen
NOTE: The SAS System stopped processing this step because of erro
NOTE: There were 3 observations read from the data set COMPDATA.L
NOTE: PROCEDURE PRINT used:
      real time             0.96 seconds
      cpu time              0.03 seconds

31 !
WARNING: The Base Product product with which FSList Window is ass
         representative to have it renewed.
```

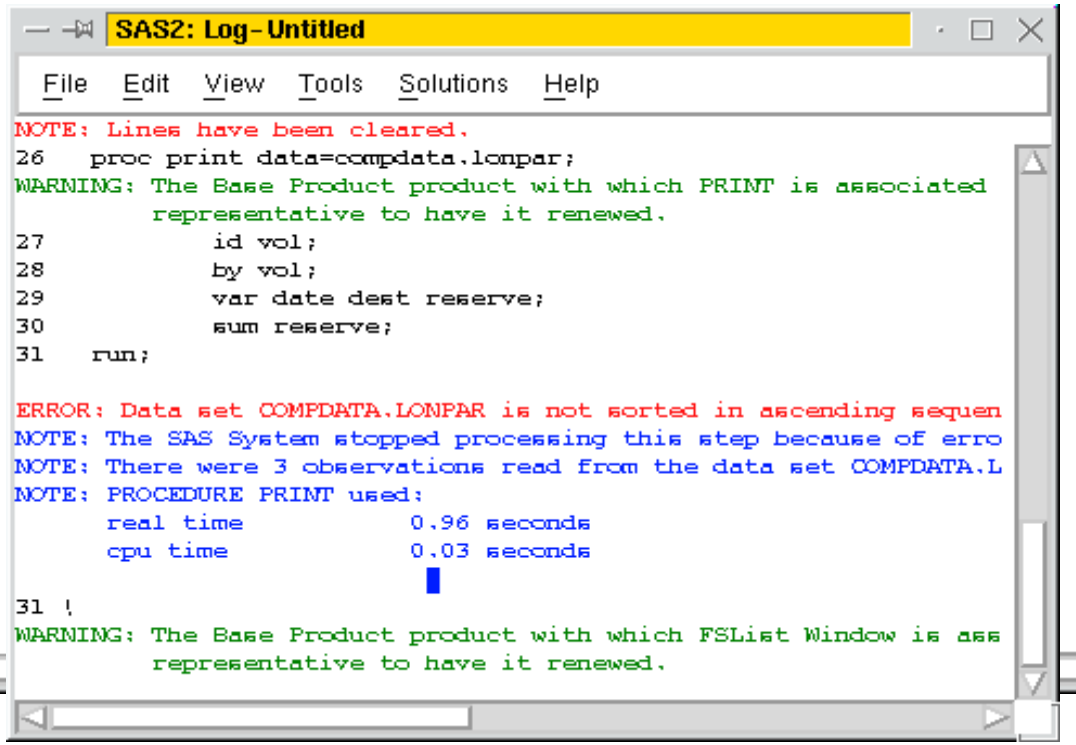
Editer et corriger un programme

SAS2: Output - Untitled

Edit View Tools Solutions Help

Il ne nous reste plus qu'à soumettre le programme entier.



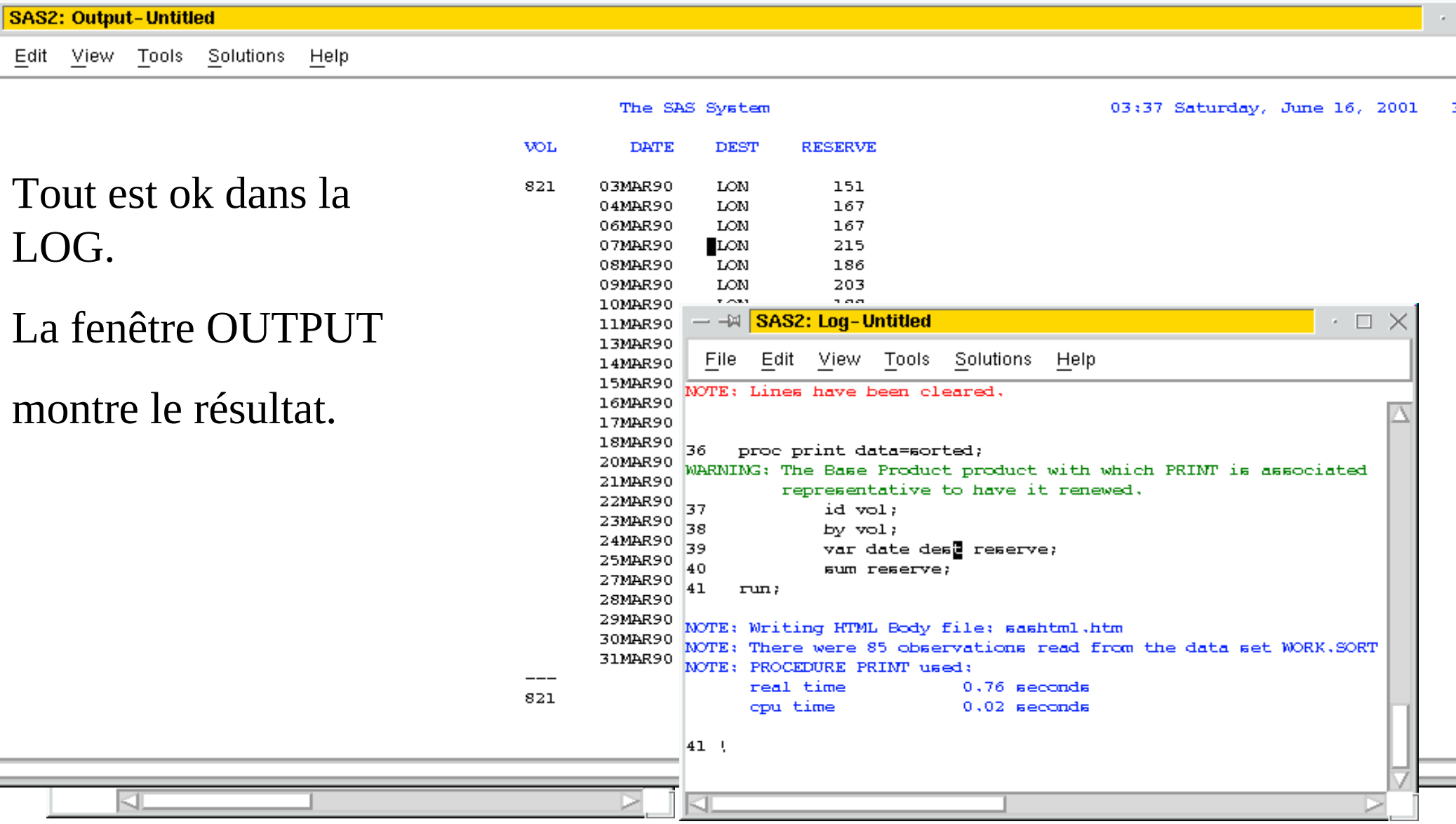


Editer et corriger un programme

Tout est ok dans la LOG.

La fenêtre OUTPUT

montre le résultat.



Commandes de numéro de ligne
Déplacement

M : Définition du bloc à déplacer comme étant la ligne

M3 : Définition du bloc à déplacer comme étant 3 lignes consécutives, la ligne courante étant le début.

MM : marque le début/la fin du bloc à déplacer

A : La destination est la ligne suivante. Le bloc sera après la ligne courante.

B : La destination est la ligne précédante. Le bloc sera avant la ligne courante.

Commandes de numéro de ligne
COPIE

C : Définition du bloc à copier comme étant la ligne

C3 : Définition du bloc à copier comme étant 3 lignes consécutives, la ligne courante étant le début.

CC : marque le début/la fin du bloc à copier

A : La destination est la ligne suivante. La copie sera après la ligne courante.

B : La destination est la ligne précédante. La copie sera avant la ligne courante.

Commandes de numéro de ligne
REPETITION de ligne

R ou R1 Répéter la ligne courante une fois.

R3 : Répéter la ligne courante trois fois.

RR et RR3 : Marque de début/fin du bloc à répéter.
Préciser le nombre d'espace soit en début, soit en fin.

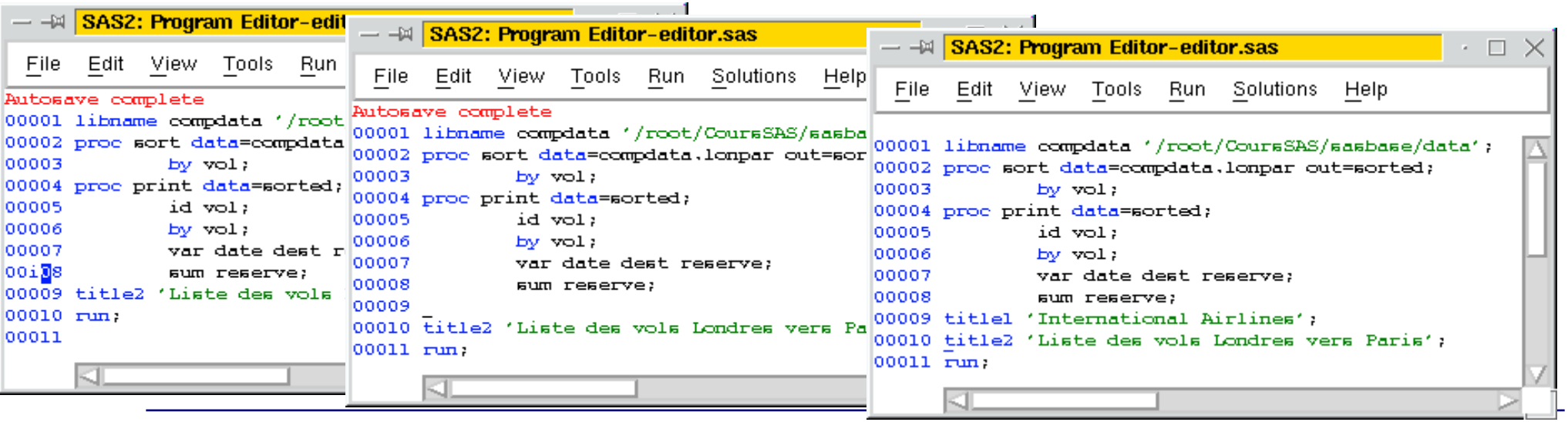
Commandes de numéro de ligne
INSERTION de ligne

I ou IA : Insertion d'une ligne vide après la ligne courante.

IB : Insertion d'une ligne vide avant la ligne courante.

I3 ou IA3 : Insertion de 3 lignes vides après la ligne courante.

IB3 : Insertion de 3 lignes vides avant la ligne courante.

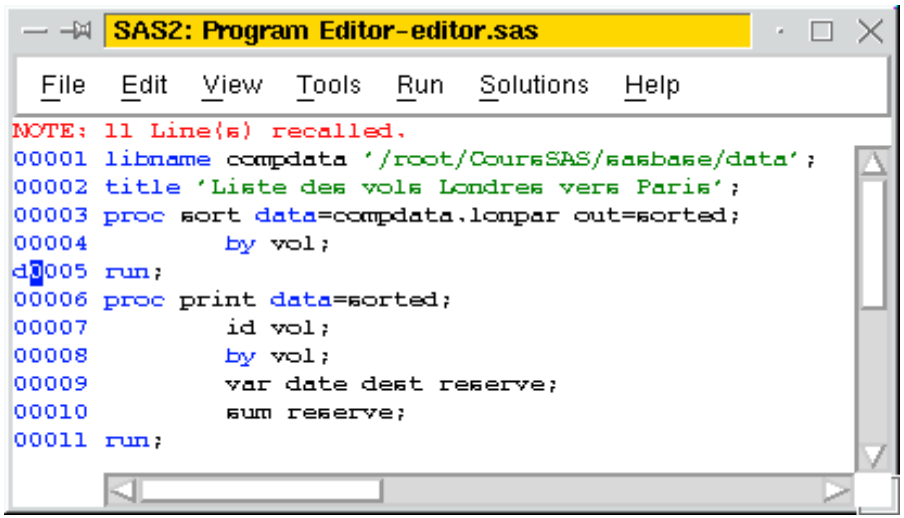


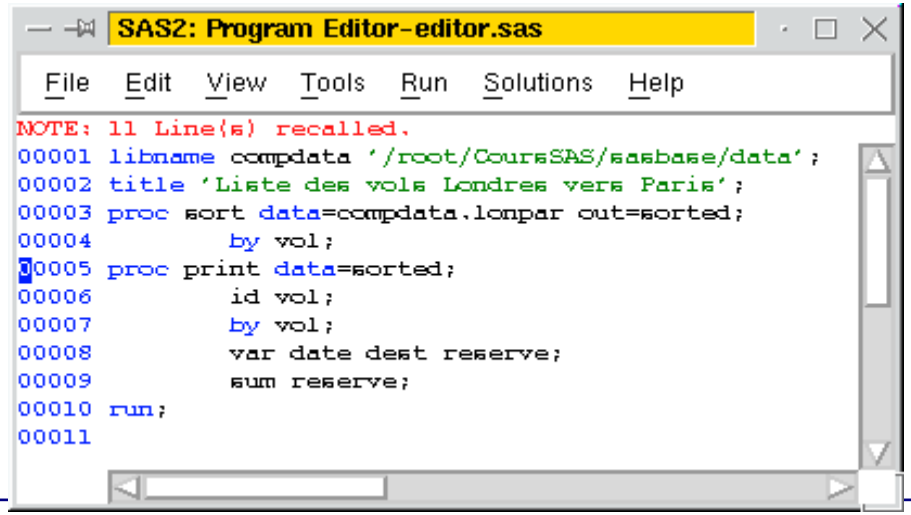
Commandes de numéro de ligne
SUPPRESSION de ligne

D : Suppression de la ligne courante.

D3 : Suppression de 3 lignes à partir de la ligne courante.

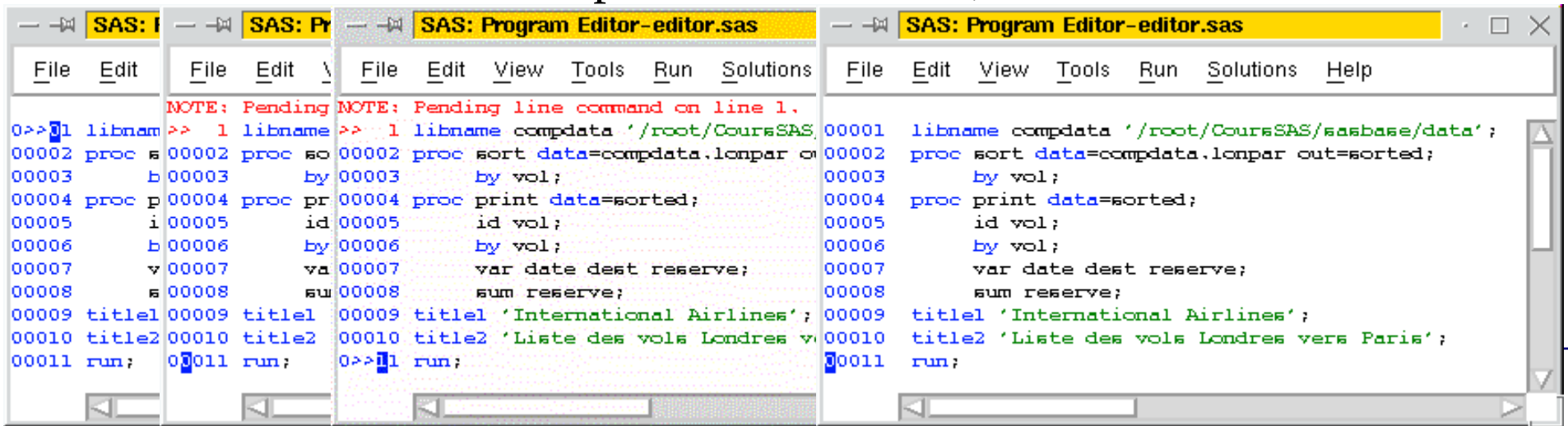
DD : Marque le début/la fin du bloc à détruire. La suppression intervient dès que le bloc a été défini.



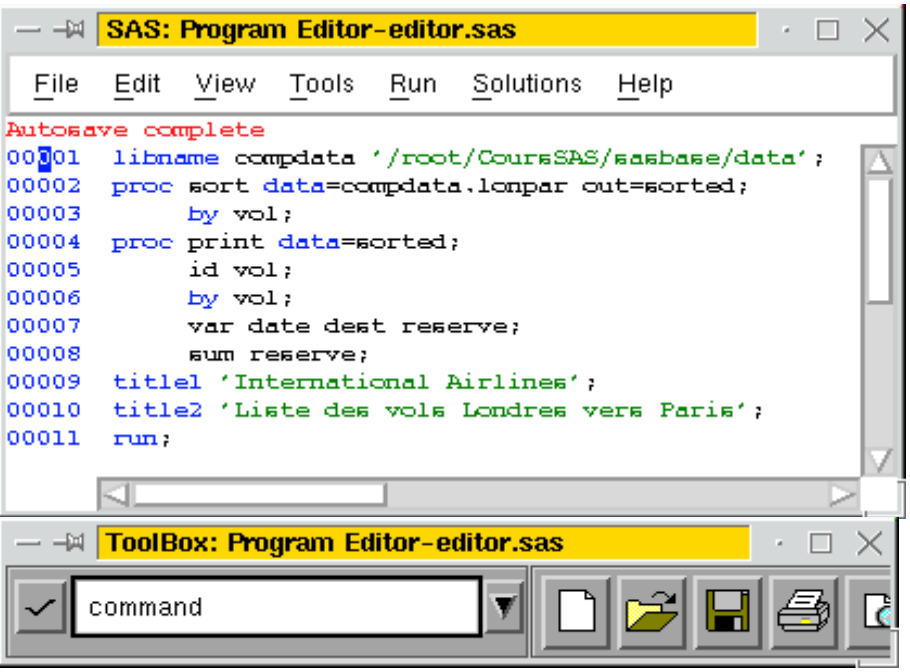


Commandes de numéro de ligne
INDENTATION de ligne

> ou >1 : Déplacer une ligne de texte vers la droite d'un espace.
>3 : Déplacer une ligne de texte vers la droite de 3 espaces.
>> et >>3 : Marque de début/fin du bloc à indenter vers la droite.
Préciser le nombre d'espace soit en début, soit en fin.
< ou <1 : Déplacer une ligne de texte vers la gauche d'un espace.
<3 : Déplacer une ligne de texte vers la gauche de 3 espaces.
<< et <<3 : Marque de début/fin du bloc à indenter vers la gauche.
Préciser le nombre d'espace soit en début, soit en fin.

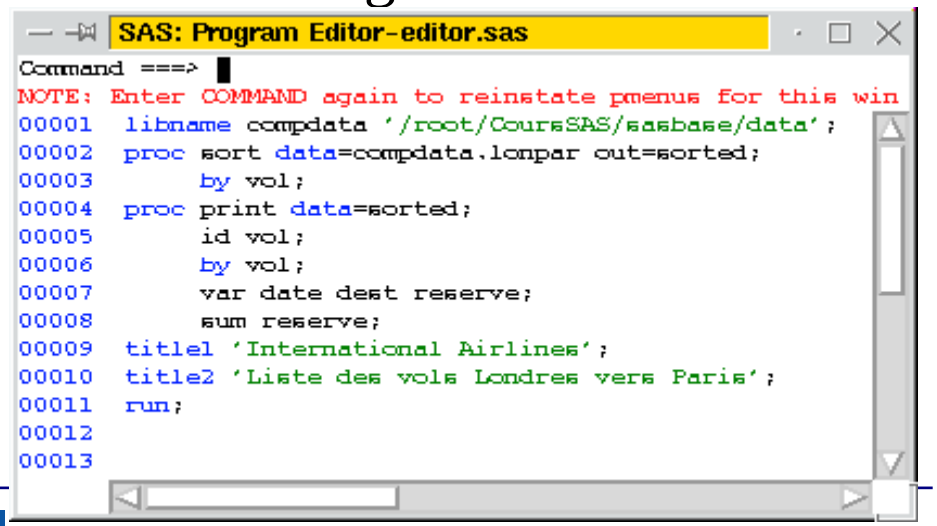


Commandes de navigation de l'éditeur
Utilisation de la ligne de commande



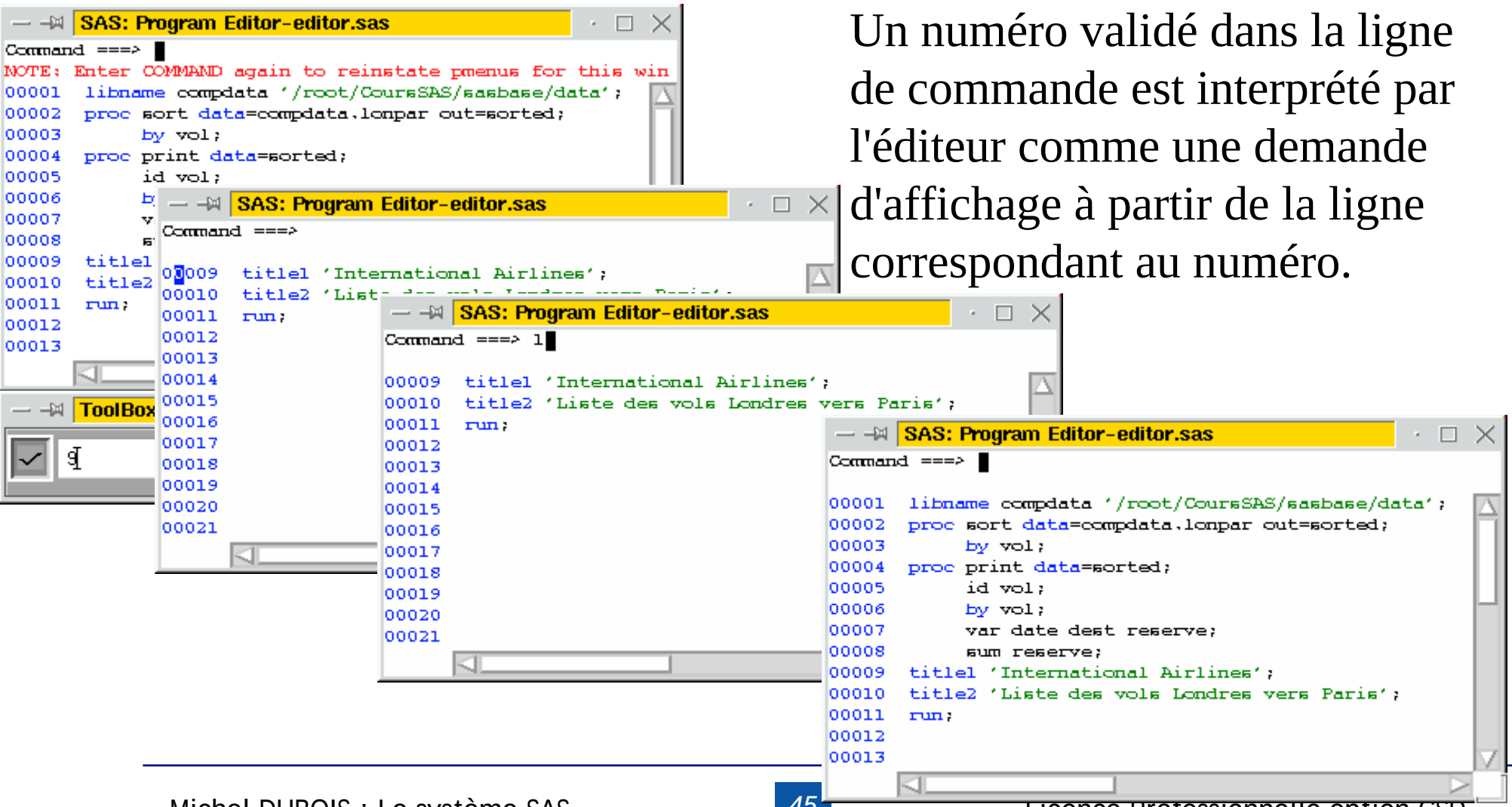
La boite à outil offre une
ligne de commande qui s'applique
à la fenêtre courante.

La commande COMMAND
sur une fenêtre permet de basculer
en mode menu ou ligne de commande



Toute commande entrée dans la ligne
de commande d'une fenêtre s'applique
à cette fenêtre

Commandes de navigation de l'éditeur
Editer à partir d'une ligne



Un numéro validé dans la ligne de commande est interprété par l'éditeur comme une demande d'affichage à partir de la ligne correspondant au numéro.

La ligne de commande (du PGM)

- **Pour exécuter les énoncés entrés, taper la commande SUBMIT (SUB)**
- **Pour faire afficher à l'écran les énoncés précédemment exécutés à l'aide de la commande RECALL (REC) pour effectuer les corrections nécessaires et exécuter de nouveau.**
- **La commande FILE 'nom_fichier permet de sauver le contenu d'une fenêtre sous le nom nom_fichier. La commande FILE permet de sauver des modifications effectuées sur un fichier déjà nommé.**
- **INC 'nom_fichier ramène en mémoire un programme déjà existant**

Touches particulières du PGM

- **La touche INSERT permet d'insérer des caractères dans un texte durant l'édition**
- **DELETE efface le caractère à la position courante du curseur**
- **HOME amène le curseur à la ligne de commande de la fenêtre courante**
- **BACKSPACE efface le caractère à la gauche du curseur**

Commandes de positionnement

- **Taper le nombre n sur la ligne de commande de la fenêtre PGM amène la ligne n au début de la fenêtre.**
- **On peut indiquer en option la quantité à déplacer. L'option PAGE précise de déplacer le contenu entier de la fenêtre; HALF, la moitié; MAX, permet d'aller au début (BACKWARD) ou à la fin (FORWARD) du texte et n, de se déplacer de n lignes ou espaces.**

  Les directions sont :
  - BACKWARD : bas
  - FORWARD : haut
  - LEFT : gauche
  - RIGHT : droite
- **TOP et BOTTOM permettent de se positionner à la première ligne ou à la dernière ligne.**

commandes sur les numéros de lignes de l'éditeur

- **C, CC: Pour copier une ou plusieurs lignes, taper C sur le numéro de la ligne à copier ou CC sur le numéro de la première et dernière ligne du bloc à copier.**
- **A, B: Pour indiquer l'endroit où les lignes déplacées ou copiées à l'aide des commandes C, M, CC, MM doivent être insérées, taper A ("after") sur le numéro de la ligne après laquelle on souhaite insérer le texte ou B ("before") pour l'insérer avant.**
- **D, DD: Pour détruire une ou plusieurs lignes, taper D sur le numéro de la ligne à éliminer ou DD sur le numéro de la première et dernière ligne du bloc à détruire.**
- **M, MM: Pour déplacer une ou plusieurs lignes, utiliser M ou MM (semblable à C ou CC).**

commandes sur les numéros de lignes de l'éditeur 2

- **R, RR[n]: Pour répéter une ligne ou plusieurs lignes, utiliser R ou RR. Pour répéter une ligne plus d'une fois, taper R suivi du nombre de fois n et d'un espace. Pour répéter un bloc plus d'une fois, taper RR suivi du nombre et d'un espace sur la première et dernière ligne du bloc à répéter.**
- **I[n]: Pour insérer une nouvelle ligne, taper I sur le numéro de la ligne après laquelle on souhaite faire l'insertion. Pour insérer n lignes, taper un I suivi du nombre n et d'un blanc.**
- **>[n], >>[n]: utiliser > suivi d'un nombre n et d'un espace pour déplacer une ligne de texte vers la droite de n espaces. Pour déplacer un bloc de lignes, utiliser >>.**
- **<[n], <<[n]: Pour déplacer une ou plusieurs lignes vers la gauche.**

Recherche et remplacement

- **On utilise la commande CHANGE dans la fenêtre PGM pour remplacer chaque occurrence d'une chaîne de caractères par une autre, et FIND dans n'importe laquelle des 3 fenêtres pour trouver chaque occurrence d'une chaîne de caractères. La forme générale de ces deux commandes est la suivante:**

  CHANGE chaîne1 chaîne2 [NEXT|FIRST|LAST|PREV|ALL]

  FIND chaîne_caractère [NEXT|FIRST|LAST|PREV|ALL]
- **Les commandes RCHANGE et RFIND permettent de poursuivre une recherche sans devoir entrer à nouveau les commandes CHANGE ou FIND.**

Autres fenêtres

- **KEYS**
- **HELP**
- **FOOTNOTES, TITLES**
- **QUERY**
- **ODSTEMPLATES**
- **La commande OPTIONS**
- **La commande FSVIEW, FSEDIT**
- **la commande NOTEPAD**
- **EFI (external file interface)**

Display Manager

```
dm " clear out; clear log;pgm;rec ";
```

Sommaire

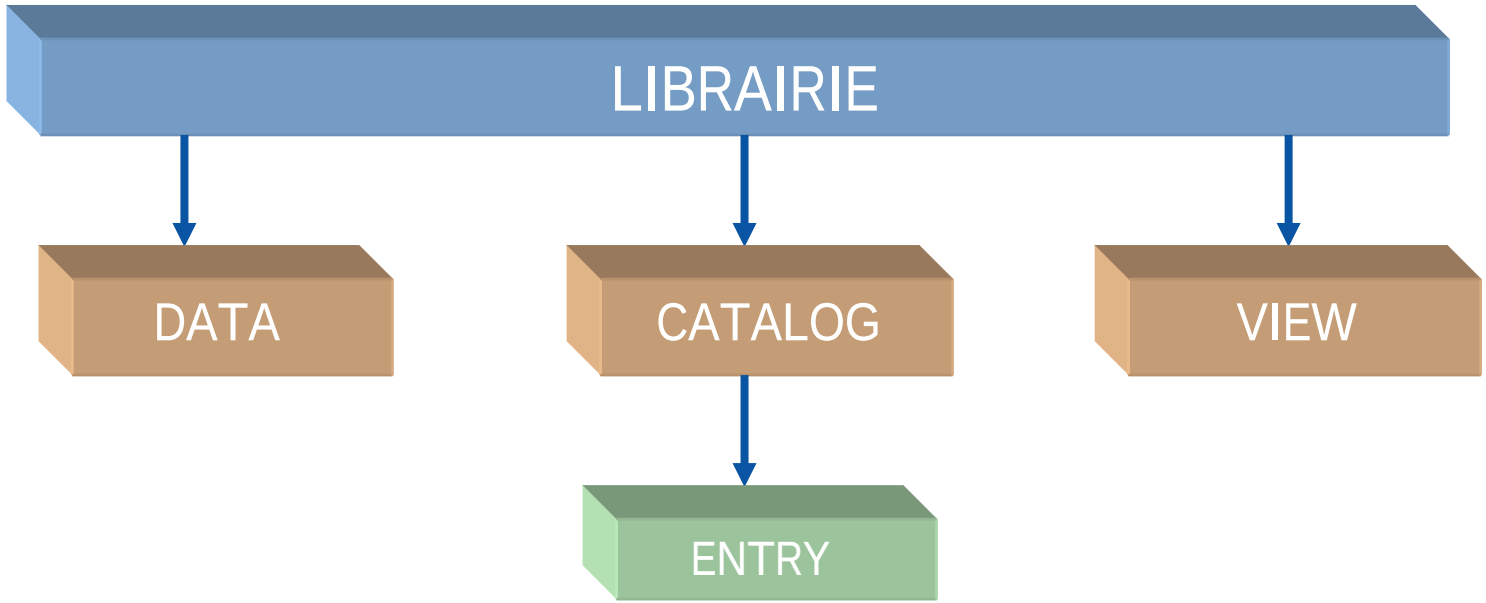
- Présentation de l'environnement
- **Les éléments SAS**
- Le langage SAS
- La proc SQL et le dictionnaire de données
- Introduction à l'ODS
- Méthodologie

Sommaire

- **Les éléments SAS**
  - Les librairies
  - Les tables
  - Les variables
  - Les vues
  - Les indexes
  - Les catalogues
  - Manipulation des éléments SAS.

Les éléments SAS - les librairies

- **Une librairie SAS contient des fichiers SAS de types différents :**
  - DATA : Table SAS
  - VIEW : Vues logiques SAS
  - CATALOG : Catalogues SAS



Les éléments SAS - les librairies

- **Une librairie est identifiée par son nom logique qui ne peut dépasser 8 caractères.**

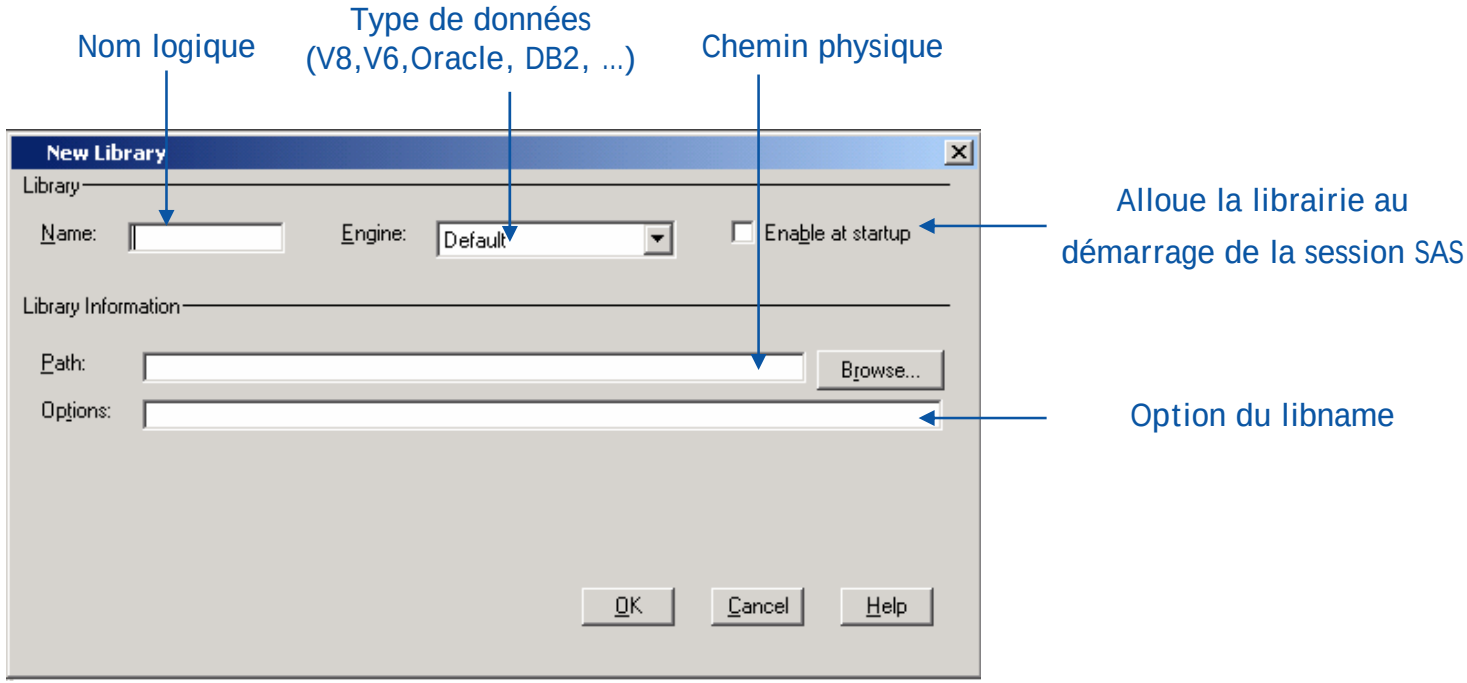
- **Elle pointe sur un ou plusieurs répertoires physiques.**

- **Les librairies réservées :**
  - WORK : Librairie de travail temporaire. Les fichiers sont détruits à la fermeture de la session SAS
  - SASHELP : contient des catalogues et des vues propres au système SAS. Cette librairie n'est accessible qu'en lecture seule. Elle pointe sur plusieurs répertoires.
  - SASUSER : librairie personnelle de l'utilisateur. Elle contient le catalogue PROFILE dans lequel sont sauvegardés différents paramètres de configuration de l'environnement SAS.

Les éléments SAS - les librairies

- **Pour allouer une nouvelle librairie :**
  - Clique droit sur l'icône librairies → New :

  - La boîte de dialogue ci-dessous apparaît :



  - Le code : syntaxe **libname <Nom logique> '<Nom physique de la librairie>' ;**

La version 8 donne la possibilité d'accéder directement à une ressource en précisant le chemin physique.

```
proc print data='c:\AircraftData\airplanes';
run;
proc print data='/users/AircraftData/airplanes';
run;
```

Ceci nuit grandement à la portabilité (multiplication dans le code de chemins physiques).

- **Un libname est associé à un moteur.**
- **On peut assigner un ensemble hétérogène (du point de vue du moteur) de bibliothèques à un nom logique afin de faciliter l'accès.**
- **L'ordre de déclaration est important pour la recherche des fichiers.**

```
libname local 'c:\mydata';
libname oradata oracle userid=ac password=xy;
libname server 'u:\public\prog3';
libname alldata (local oradata server);
```

- **L'ordre de déclaration est important pour la recherche des fichiers.**

- **Quand un fichier est ouvert pour …** — **Le fichier utilisé est …**

INPUT/UPDATE	**Premier fichier trouvé**
DELETE/RENAME	**Premier fichier trouvé**
OUTPUT	**Première bibliothèque trouvée**

- **Etats créés par** — **Le fichier décrit est …**

PROC CONTENTS	**Premier fichier trouvé**
PROC DATASETS	**Premier fichier trouvé**
Fenêtre EXPLORER	**Premier fichier trouvé**

Les éléments SAS - les tables

- **Une table SAS est un membre de type DATA d'une librairie SAS**

- **Son nom peut atteindre 32 caractères à partir de la version 8 (8 caractères en version 6.12)**

- **Exploitation des tables**
  - Les tables **permanentes** présentent dans les librairies exceptées la WORK

    Dans le code SAS, elles sont toujours identifiées par le nom de la librairie et leur nom propre séparés par un point.

    Exemple : SASUSER.NOMTABLE, la table NOMTABLE est dans la librairie SASUSER

  - Les tables **temporaires** présentent dans la librairie WORK

    Une table SAS temporaire ne peut appartenir qu'à la librairie WORK. Dans SAS, il n'est pas obligatoire de préfixer le nom de la table temporaire par « WORK. »

    Exemple : WORK.NOMTAB ⇔ NOMTAB

Les éléments SAS - les tables

- **Exemple de table SAS**

Identifie les lignes appelées observations en SAS

Identifie les colonnes appelées variables en SAS

VIEWTABLE: sashelp.class

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77

Les éléments SAS - les variables

- **Le nom d'une variable, comme pour celui des tables, peut atteindre 32 caractères à partir de la version 8.**

- **Une variable SAS peut être caractère ou numérique :**
  - Une variable numérique est stockée sur 8 octets maximum (valeur par défaut).

    La longueur minimale est 3.

    Une date est aussi une variable numérique correspondant au nombre de jours depuis le 1er janvier 1960. Un format spécial leur est appliqué selon le besoin.

  - Une variable caractère est une chaîne de caractères pouvant contenir des lettres, des nombres ou des caractères spéciaux (_,$,#,…).

    Sa longueur maximale est de 32767 octets.

Les éléments SAS - les variables

- **Chaque variable possède des caractéristiques qui lui sont propres**
  - **Type** Numérique ou caractère
  - **Length** Longueur
  - **Format** Masque en lecture
  - **Informat** Masque en écriture
  - **Label** Libellé

Les éléments SAS - les vues

- **Leurs caractéristiques**
  - Elles sont structurées de la même façon qu'une table, avec des observations et des colonnes.
  - Une vue n'est qu'une représentation logique d'une ou plusieurs tables, elle ne contient physiquement aucune observation propre.
  - Elles se définissent à l'aide de code SQL (instruction CREATE VIEW) ou d'une étape DATA.

  - **Avantage** : Elles ne consomment pas d'espace disque.
  - **Inconvénient** : Elles dégradent sensiblement les performances car il faut régénérer leur contenu à chaque utilisation.

Les éléments SAS - les indexes

- **Ils sont associés aux tables.**
- **Ils ne sont pas visibles par l'explorateur SAS, même s'il s'agit d'entité physique.**
- **Leur utilisation permet d'optimiser certains traitements et offre la possibilité d'accéder aux observations par clé.**
- **Comment les créer :**
  - Procédure SQL
  - Procédure DATASETS

Les éléments SAS - les catalogues

- **Un catalogue SAS est une entité physique**
  - Son nom peut atteindre 32 caractères
  - Il contient de 1 à N entrées (SAS Entry)
  - **Exemple** : le catalogue SASHELP.SCM
  - Lors de la programmation, un catalogue est toujours précédé de la librairie à laquelle il appartient et est séparé de cette dernière par un point (excepté pour la librairie WORK où ce n'est pas indispensable)

- **Une entrée d'un catalogue SAS**
  - Elle n'est visible que par SAS
  - Son nom peut atteindre 32 caractères
  - Différents types sont possibles
  - **Exemple** : l'entrée SASHELP.SCM.WINAPI.SOURCE

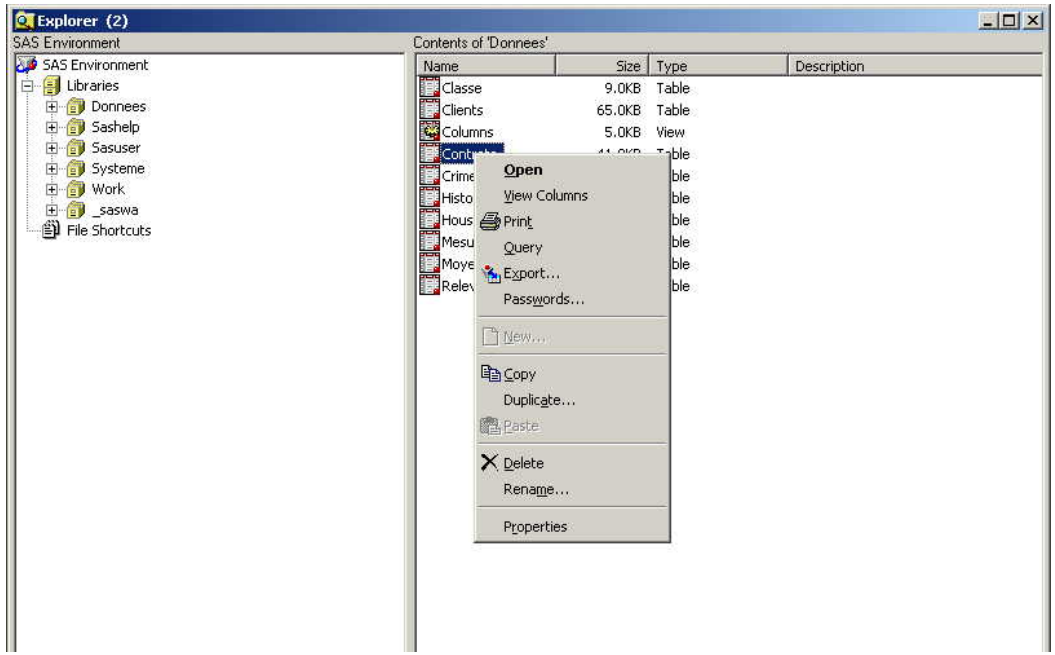
Les éléments SAS - Manipulation

- **Il est possible de manipuler les éléments SAS de 2 manières**
  - A l'aide du code
  - Par l'explorateur

- **On entend par manipuler**
  - Copier une table, un catalogue d'une librairie à une autre
  - Supprimer un élément SAS
  - Renommer un élément SAS
  - Visualiser les variables d'une table, consulter son contenu
  - Visualiser les propriétés d'une entrée, visualiser son contenu

Les éléments SAS - Manipulation

- **La manipulation par l'explorateur.**
  - Il suffit de se positionner sur l'élément SAS et de cliquer droit pour choisir l'action désirée.



  - Dans cet exemple, après la sélection, on peut choisir diverses actions.

Sommaire

Sommaire

- **Le langage SAS**
  - Généralités
  - Les procédures, principes et exemples
  - Utilisation des formats
  - L'étape DATA, principes et instructions majeures
  - La manipulation des fichiers externes
  - La manipulation de tables SAS
  - Eléments avancés

Les programmes SAS – généralités

- **Un programme SAS est composé d'une succession d'étapes DATA ou/et de procédures.**

- **Règles d'écriture :**
  - Toute instruction commence par un MOT CLE (DATA, SET, RUN, PROC, VAR,…)
  - Toute instruction se termine par un **;**
  - Une instruction peut tenir sur plusieurs lignes
  - Plusieurs instructions peuvent tenir sur une ligne (déconseillé)
  - Un nom de table ou de variable commence par une lettre ou un 'blanc souligné' (32 car max.)

Les programmes SAS - généralités

- **Les programmes SAS peuvent être sauvegardés en externe ou en interne**

  - Dans un fichier externe de type texte avec l'extension **.sas**
  - Dans un catalogue SAS. Les programmes seront de type SOURCE et ne seront lisibles que par SAS.

  Le programme aura pour nom :

  <Nom Librairie>.<Nom Catalogue>.<Nom programme>.source

Les programmes SAS - généralités

- **Les programmes SAS sont constitués :**

  - D'étapes DATA qui génèrent et manipulent des tables SAS ou des fichiers externes
  - De procédures qui effectuent des traitements particuliers sur les tables SAS comme des impressions, des tris, des graphiques, des statistiques …
  - D'instructions générales qui peuvent être placées à n'importe quel endroit du programme : FILENAME …; LIBNAME …; FOOTNOTE …; TITLE …; OPTIONS …;

Les programmes SAS - généralités

- **Les commentaires :**
  - /* …… */ permet de délimiter une zone commentaire, sur une ou plusieurs lignes

    /* Indique le début de la zone de commentaire

    */ Indique la fin de la zone de commentaire

  - * permet de mettre en commentaire une, et une seule, instruction.

    Le commentaire comprend donc la zone de code entre l'étoile et le premier point virgule, présent sur la même ligne que l'étoile ou sur une ligne différente.

blocage de SAS

- **Il se peut que SAS soit bloqué car le programme comporte une chaîne non terminée, un commentaire non formé ou il n'y a pas de RUN terminal**
- **Il faut alors soumettre la séquence suivante**

```
*';
*";
*/;
RUN;
```

Les programmes SAS – les procédures

- **La procédure effectue des traitements particuliers sur les tables SAS comme des impressions, des tris, des graphiques, des statistiques …**

- **Elle commence par une instruction « PROC …; »**

  Elle se termine toujours par l'instruction « RUN ; » (« QUIT; » pour les procédures SAS/GRAPH, la procédure SQL, …)

Les programmes SAS – les procédures

- **D'une façon générale, la syntaxe d'une procédure suit le schéma suivant**

```
Proc NomProcédure data = NomTableSas  option;
    Instruction1 Paramètre1 / option ;
    Instruction2 Paramètre2 / option;
run;
```

- **Exemple**

```
Proc print data = SORTIE ;
    var VAR1 VAR2 NEWVAR ;
    where VAR1 ne 0;
run;
```

→ Impression de la table SORTIE
→ Sélection des variables
→ Filtrage des données
→ Fin d'étape

Les programmes SAS – les procédures

- **La PROC CONTENTS édite les caractéristiques d'une ou plusieurs tables d'une librairie**

- **Syntaxe**
  - **Information sur une seule table d'une librairie**

```
Proc contents data = nom_librairie.nom_table ;
run;
```

  - **Information sur toutes les tables d'une librairie**

```
Proc contents data = nom_librairie._all_;
run;
```

Les programmes SAS – les procédures

- **Extrait de l'analyse de la table DONNEES.CLASSE avec la procédure CONTENTS**

-----Engine/Host Dependent Information-----	
Data Set Page Size:	8192
Number of Data Set Pages:	1
First Data Page:	1
Max Obs per Page:	203
Obs in First Data Page:	19
Number of Data Set Repairs:	0
File Name:	C:\Environnement SAS\Formation Travail\FormBD BASE\Donnees\classe.sas7bdat
Release Created:	8.0202M0
Host Created:	WIN_PRO

-----Alphabetic List of Variables and Attributes-----						
#	Variable	Type	Len	Pos	Format	Label
3	AGE	Num	8	0		Age
4	HAUTEUR	Num	8	8		Hauteur
1	NOM	Char	8	24		Prénom
5	POIDS	Num	8	16	Z8.	Poids
2	SEXE	Char	1	32		Sexe

Les programmes SAS – les procédures

- **La PROC SORT sert à trier les observations d'une table dans un ordre défini. Elle est nécessaire avant une procédure ou une étape DATA qui utilise l'instruction by.**

- **Syntaxe**

```
Proc sort  data = nom_entree  out=nom_sortie nodupkey ;
   By nom_var1 nom_var2 ;
run;
```

Out = Permet de créer une table qui sera triée.

Nodupkey = Élimine les doublons (supprime les observations pour lesquelles les variables de l'instruction by ont des valeurs égales).

Les programmes SAS – les procédures

- **La PROC PRINT est utilisée pour produire des éditions simples d'une table SAS.**

- **Syntaxe**

```
Proc print  data = nom_table options ;
   var nom_var1 nom_var2 ;
   id nom_var3;
   sum nom_var1 nom_var2 ;
   by nom_var3;
run;
```

- `var` → Variables à éditer
- `id` → Remplace la colonne OBS par le contenu d'une variable
- `sum` → Affiche les totaux sur une ou plusieurs variables numérique
- `by` → Permet d'effectuer des ruptures. Les sous-totaux sont alors calculés pour chaque niveau de rupture

Les programmes SAS – les procédures

- **Extrait de l'édition de la table DONNEES.CLASSE avec la procédure PRINT**

The SAS System

Obs	NOM	SEXE	AGE	HAUTEUR	POIDS
1	Guido	M	.	160	00000066
2	Sandy	F	11	123	00000025
3	Thomas	M	11	138	00000042
4	Karen	F	12	135	00000000
5	Kathy	F	12	143	00000042
6	James	M	12	137	00000041
7	John	M	12	141	00000049
8	Robert	M	12	155	00000064
9	Alice	F	13	135	00000042
10	Becka	F	13	156	00000049
11	Jeffrey	M	13	150	00000042
12	Gail	F	14	154	00000045
13	Tammy	F	14	150	00000051
14	Alfred	M	14	165	00000000
15	Duke	M	14	152	00000051
16	Mary	F	15	159	00000000

Les programmes SAS – les procédures

- **La PROC FREQ édite des tableaux de contingence à une ou deux entrées, avec les pourcentages associés.**

- **Syntaxe**

```
Proc freq  data = nom_table order=<FREQ> ;
   tables   nom_var1
                  ou
               nom_var2 * nom_var3;
   by nom_var3;
run;
```

Order : Afficher le résultat selon un certain ordre:
- FREQ = Trié par fréquence
- DATA = Ordre initial de la table en entrée
- INTERNAL = Ordre des valeurs des variables de l'instruction TABLES
- FORMATTED = Ordre des valeurs formatées des variables de l'instruction TABLES

Précise une ou plusieurs variables de rupture.

Les programmes SAS – les procédures

- **Extrait d'une procédure FREQ, exemple de la documentation SAS**

Hair Color				
Hair	Frequency	Percent	Cumulative Frequency	Cumulative Percent
black	22	2.89	22	2.89
dark	182	23.88	204	26.77
fair	228	29.92	432	56.69
medium	217	28.48	649	85.17
red	113	14.83	762	100.00

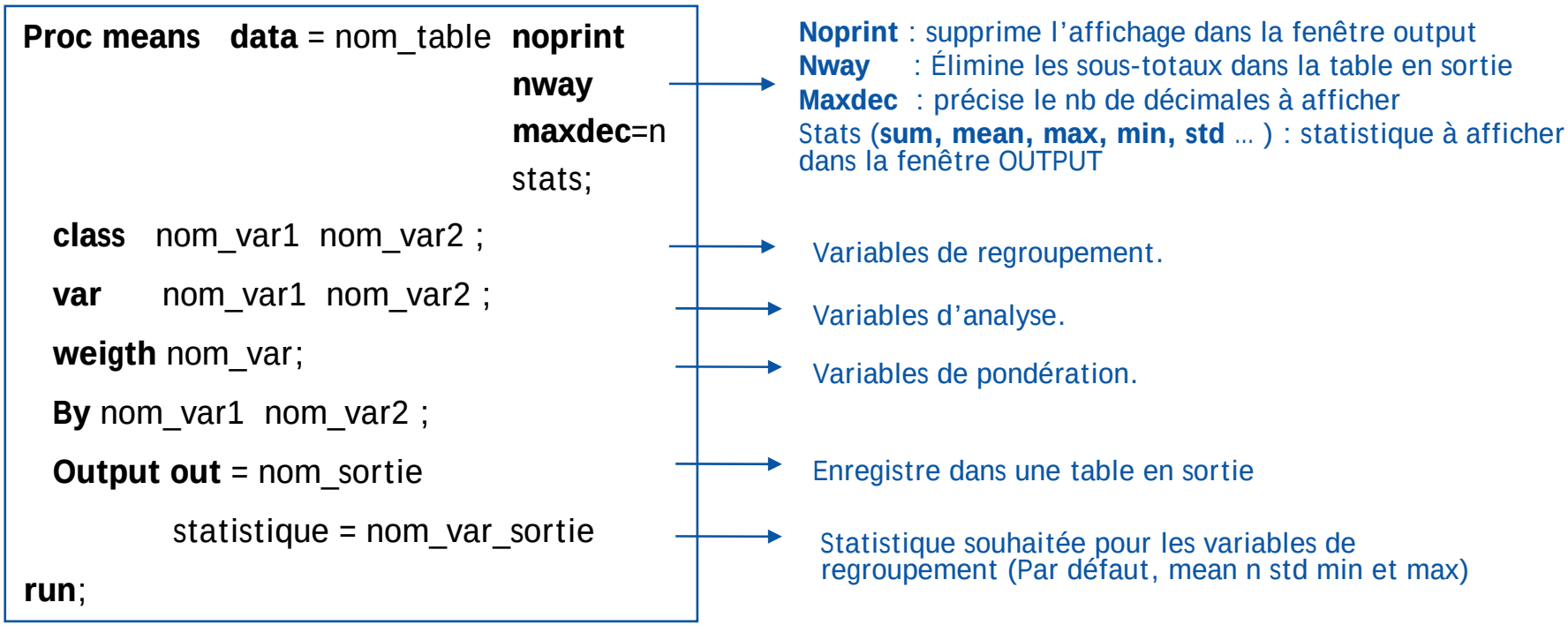
Frequency
Percent
Row Pct
Col Pct

Table of Eyes by Hair						
	Hair(Hair Color)					
Eyes(Eye Color)	black	dark	fair	medium	red	Total
blue	6 0.79 2.70 27.27	51 6.69 22.97 28.02	69 9.06 31.08 30.26	68 8.92 30.63 31.34	28 3.67 12.61 24.78	222 29.13
brown	16 2.10 4.69 72.73	94 12.34 27.57 51.65	90 11.81 26.39 39.47	94 12.34 27.57 43.32	47 6.17 13.78 41.59	341 44.75

Les programmes SAS – les procédures

- **La PROC MEANS est utilisée pour produire des statistiques descriptives à partir d'une table SAS**

- **Syntaxe**



Les programmes SAS – les procédures

- **Extrait d'une procédure MEANS**

Sexe=F

Analysis Variable : HAUTEUR Hauteur				
Age	N Obs	Minimum	Maximum	Median
11	1	123.0000000	123.0000000	123.0000000
12	2	135.0000000	143.0000000	139.0000000
13	2	135.0000000	156.0000000	145.5000000
14	2	150.0000000	154.0000000	152.0000000
15	2	150.0000000	159.0000000	154.5000000

Sexe=M

Analysis Variable : HAUTEUR Hauteur				
Age	N Obs	Minimum	Maximum	Median
11	1	138.0000000	138.0000000	138.0000000
12	3	137.0000000	155.0000000	141.0000000
13	1	150.0000000	150.0000000	150.0000000
14	2	152.0000000	165.0000000	158.5000000

Les programmes SAS – les procédures

- **La PROC TABULATE est utilisée pour produire des tableaux statistiques à partir d'une table SAS.**

- **Syntaxe**

```
Proc tabulate  data = nom_table ;
   class   nom_var1  nom_var2 ;
   var      nom_var1  nom_var2 ;
   table      dimension_page,
              dimension_ligne,
              dimension colonne / option;
run;
```

→ Variables de regroupement.

→ Variables d'analyse.

→ Chaque dimension correspond à une variable utilisée (Colonne obligatoire, page et ligne optionnelles). Elles peuvent être suivies de statistiques.

Les programmes SAS – les procédures

- **Extrait d'une procédure TABULATE, exemple de la documentation SAS**

Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$46,291	$37,555	$83,846

Les programmes SAS - les formats

- **Un FORMAT permet d'afficher une variable d'une façon pré-définie.**
  - Un format est utilisé pour
    - Remplacer un code par un libellé
    - Changer un mode de représentation
    - Effectuer des regroupements dynamiques
  - Un format s'applique sur
    - Des variables caractères
    - Des variables numériques

Les programmes SAS - les formats

- **Un INFORMAT permet de lire une variable de telle façon qu'elle soit reconnue par SAS.**
  - Un informat est utilisé pour que
    - SAS lise une variable selon un modèle donné
    - L'utilisateur écrive une valeur d'une variable selon un modèle donné
  - Un informat s'applique sur
    - Des variables caractères
    - Des variables numériques

Les programmes SAS - les formats

- **La gestion des Formats et Informats**
  - Ils sont
    - soient fournis par SAS,
    - soient créés par les utilisateurs
  - Les formats / informats fournis par SAS sont directement utilisables dans les sessions SAS.
  - Les formats / informats utilisateurs doivent être créés à l'aide de la procédure FORMAT.

Les programmes SAS - les formats

- **Les Formats et Informats fournis par SAS**
  - Leurs noms remplissent les règles suivantes

◆ Format / Informat caractère	Précédé par un $ Terminé par *w.*, *w* donnant la longueur	
◆ Format / Informat numérique	Terminé par *w.d*, *w.d* étant la représentation numérique 7.2 signifie 7 caractères en sortie dont 2 décimales	
◆ Exemples	Format caractère	$CHAR10.
	Format numérique	BEST7.2
	Format date	DDMMYY10.

Les programmes SAS - les formats

Les Formats et Informats fournis par SAS

- Quelques exemples de la bibliothèque SAS

Format/informat	Signification
$w.	Utilisé pour afficher des données **caractères** sur w positions
w.d	Utilisé pour afficher des données **numériques** sur w positions dont d décimales avec comme séparateur décimal le **point**.
NUMXw.d	Utilisé pour afficher des données **numériques** sur w positions dont d décimales avec comme séparateur décimal la **virgule**.
Zw.	Utilisé pour afficher des données **numériques** sur w positions avec des **zéros** devant.
COMMAw.d	Utilisé pour afficher des données **numériques** sur w positions dont d décimales avec un **point** comme séparateur décimal et une **virgule** comme séparateur de milliers.
COMMAXw.d	Utilisé pour afficher des données **numériques** sur w positions dont d décimales avec une **virgule** comme séparateur décimal et un **point** comme séparateur de milliers.

Les programmes SAS - les formats

Les Formats et Informats fournis par SAS

- Utilisation de formats français
  - Valable pour les formats de type date et datetime
  - Il ne faut plus utiliser les noms de format « par défaut » mais leur équivalent au format « européen »
  - Il faut positionner l'option dflang='FRENCH'
- Exemple
  - Remplacement du format MONNAME3. de la variable MONTH de la table DONNEES.PRDSALE par le format EURDFMN32.

Les programmes SAS - les formats

Les Formats et Informats personnels

- Leurs noms remplissent les règles suivantes
  - Format / Informat caractère
    - Précédé par un $
    - Terminé par un .
    - Longueur 8 maximum pour les formats, $ compris
    - Longueur 7 maximum pour les informats, $ compris
    - Ne se termine pas par un chiffre (avant le .)
  - Format / Informat numérique
    - Terminé par un .
    - Longueur 8 maximum pour les formats, 7 pour les informats
    - Ne se termine pas par un chiffre (avant le .)
  - Exemples
    - Format caractère $MONFORM.
    - Format numérique MONFORMAT.

Les programmes SAS - les formats

Les Formats et Informats personnels

- Les formats et informats personnels sont **automatiquement** enregistrés dans un catalogue FORMATS

- Ils peuvent être
  - Temporaires — le catalogue FORMATS est créé dans la librairie WORK
  - Permanents — le catalogue FORMATS est créé dans une librairie différente de la WORK

- Comment créer les formats et informats personnels ainsi que le catalogue FORMATS
  - A l'aide de la procédure **FORMAT**

Les programmes SAS - les formats

- **Syntaxe de la procédure FORMAT**

```
Proc format ;
Value $fmt_car  'val1' = 'libellé1'
                'val2' = 'libellé2';
Value  nom_fmt    val1  = 'libellé1'
                  val2  = 'libellé2'
                  other = 'libellé';
Value  nom_for    valinf1-valsup1  = 'libellé1'
                  valinf2-valsup2  = 'libellé2';
run;
```

Trois formats sont créés :

- $FMT_CAR. de type caractère

- NOM_FMT. de type numérique

- NOM_FOR. de type numérique, avec regroupement par tranche.

Les programmes SAS - les formats

- **Utilisation de formats et informats personnels de façon permanente**
  - Deux étapes sont nécessaires
  - **A la création des formats**, préciser l'option LIB

```
Proc format LIB = SYSTEME;
   Value $fmt_car  'val1' = 'libellé1'
                   'val2' = 'libellé2';
run;
```

L'option LIB permet de pointer sur la librairie qui va contenir le catalogue FORMATS

  - **A l'utilisation des formats**
    Pour pouvoir les utiliser il faut alimenter l'option **FMTSEARCH**

    Exemple : options fmtsearch=(SYSTEME) ;

Les programmes SAS - les formats

Utilisation de formats et informats personnels

- La procédure Format permet la création des formats à partir d'une table SAS grâce à l'option CNTLIN.

```
Proc format LIB = SYSTEME cntlin=SYSTEME.LISTEFMT;
run;
```

- A contrario, la procédure Format permet aussi la création d'une table SAS contenant le détail des formats présents dans le catalogue FORMATS avec l'option CNTLOUT.

```
Proc format LIB = SYSTEME cntlout=SYSTEME.LISTEFMT;
run;
```

Les programmes SAS - les formats

Gestion des dates avec les formats et informats SAS

- Le Système SAS mémorise les informations de type temps (DATE, HEURE) sous la forme d'un nombre de jours ou de secondes écoulés depuis une période de référence qui est :

Pour une date (jours)	01JAN60
Pour l'heure (secondes)	00:00:00.00
Pour une date/heure	01JAN60:00:00:00.00

- Pour manipuler ces valeurs, des fonctions de conversion permettent d 'extraire le jour, le mois, de calculer des écarts entre deux dates, ...

DATE()	retourne la date du jour
DATETIME()	retourne la date et l'heure du jour
DAY(date)	extrait le jour d'une date SAS
MONTH(date)	extrait le mois d'une date SAS
QTR(date)	extrait le trimestre d'une date SAS

Les programmes SAS - les formats

Utilisation des formats et informats

- Qu'ils soient personnels ou provenant de SAS
- Comportement lors de la lecture de variables SAS ayant un format d'affecté
- Il faut exécuter le traitement sur la valeur non formatée, autrement dit sur la valeur réelle.
- Exemple

```
data _null_ ;
 set donnees.prdsale ;

 if monyr = 12784 then put "Lecture" ;
run ;
```

Les programmes SAS - l'étape DATA

- **L'étape DATA génère et manipule des tables SAS ou des fichiers externes**

- **Elle commence par une instruction DATA …;**

  Elle se termine toujours par l'instruction RUN;

- **Exemple**

```
data SORTIE ;
   set ENTREE ;
   if VAR1 ne 0 then
   NEWVAR = VAR1 * VAR2 ;
run;
```

→ Création de la table SORTIE

→ Lecture séquentielle de la table ENTREE

→ Calcul conditionné

→ Création d'une nouvelle variable

→ Fin d'étape

Traitement d ’une étape DATA

- **Une étape DATA est traitée en deux phases :**
  - la phase de compilation
  - la phase d’exécution

Durant la compilation, le tampon d’entrée (INPUT BUFFER) et le PDV (Program Data Vector) sont créés.

Le tampon d ’entrée n ’est pas créé quand l’étape data lit des tables SAS

Des variables automatiques sont créées : _ERROR_ et _N_ dans le PDV.

ERROR signale l ’occurrence d ’une erreur dans une donnée durant l’exécution de l’étape data. La valeur 1 est prise lorsqu ’une erreur survient. Sinon _ERROR_ vaut 0.

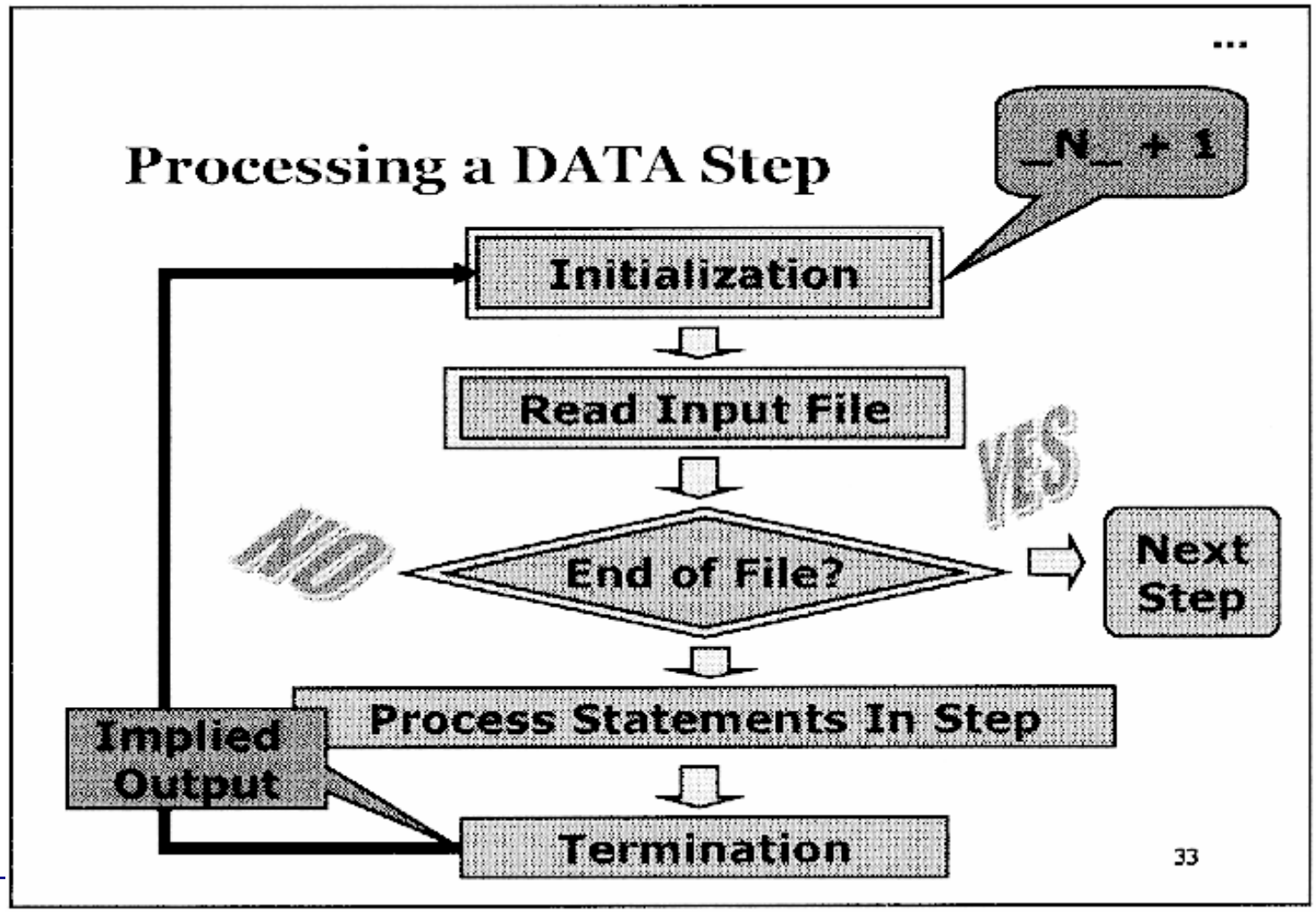
_N_ fait le comptage du nombre de passage dans l’étape DATA.

Le PDV peut contenir d’autres variables automatiques comme first.par-variable et last.par-variable.

La phase d ’exécution permet de lire les données des sources précisée par l ’énoncé INFILE ou SET, de modifier des données et d ’écrire de nouvelles tables sas.

A l ’éxécution :

Le traitement d'une étape DATA

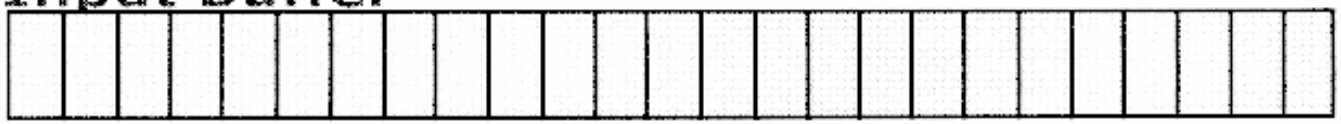


...

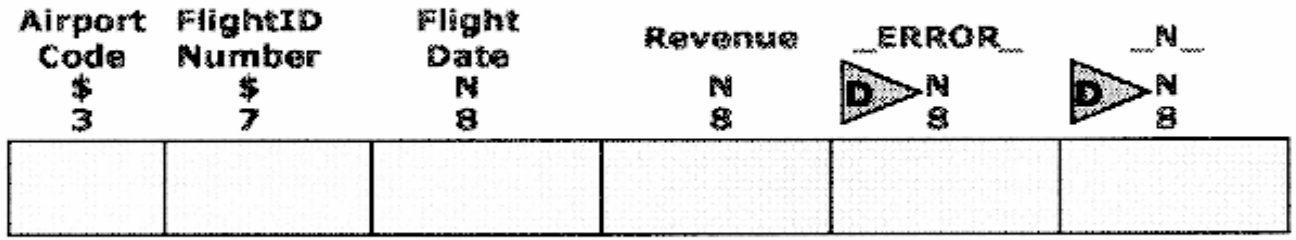
Processing a DATA Step

During compile time, buffers are created:

Input Buffer

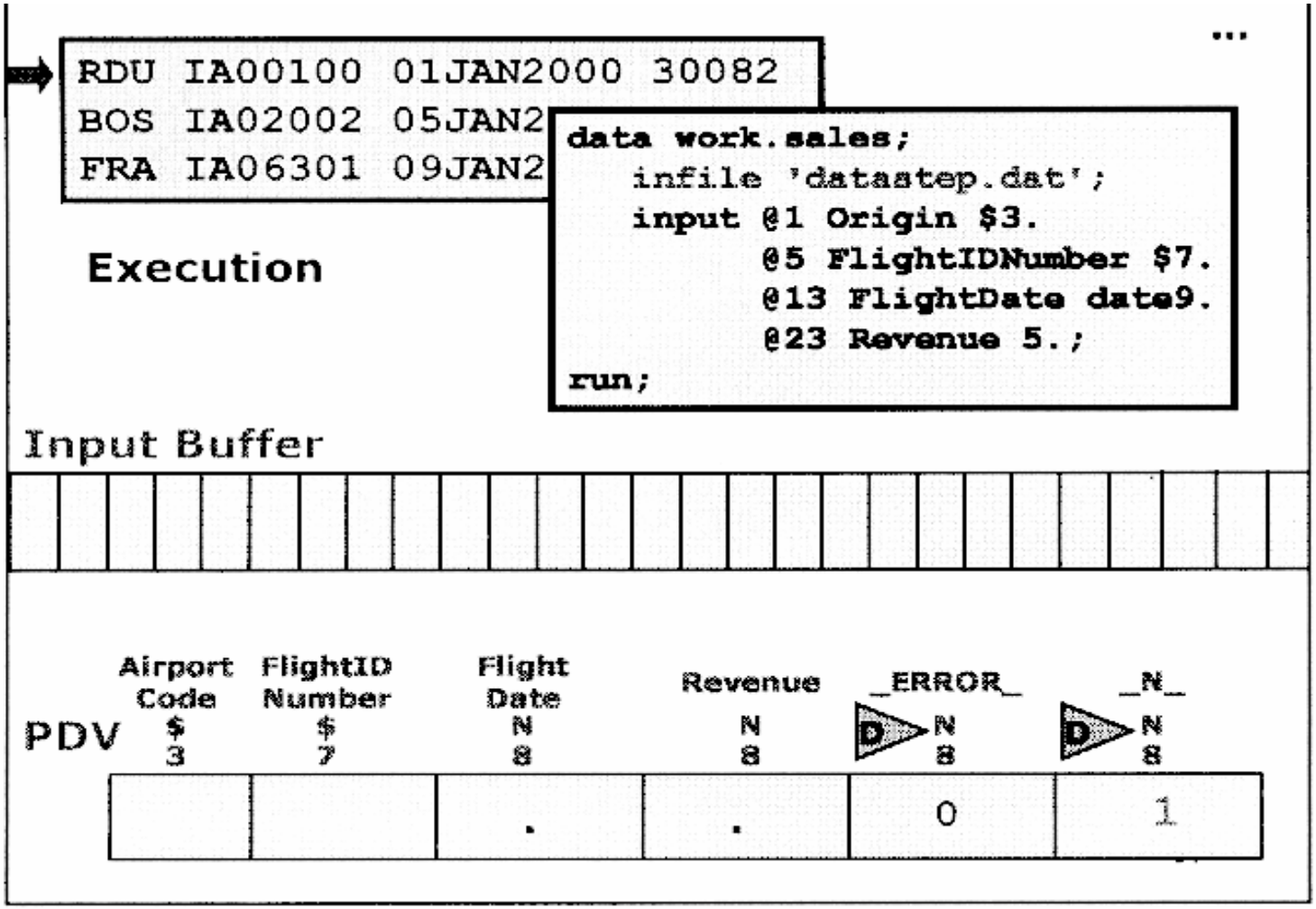


Program Data Vector (PDV)

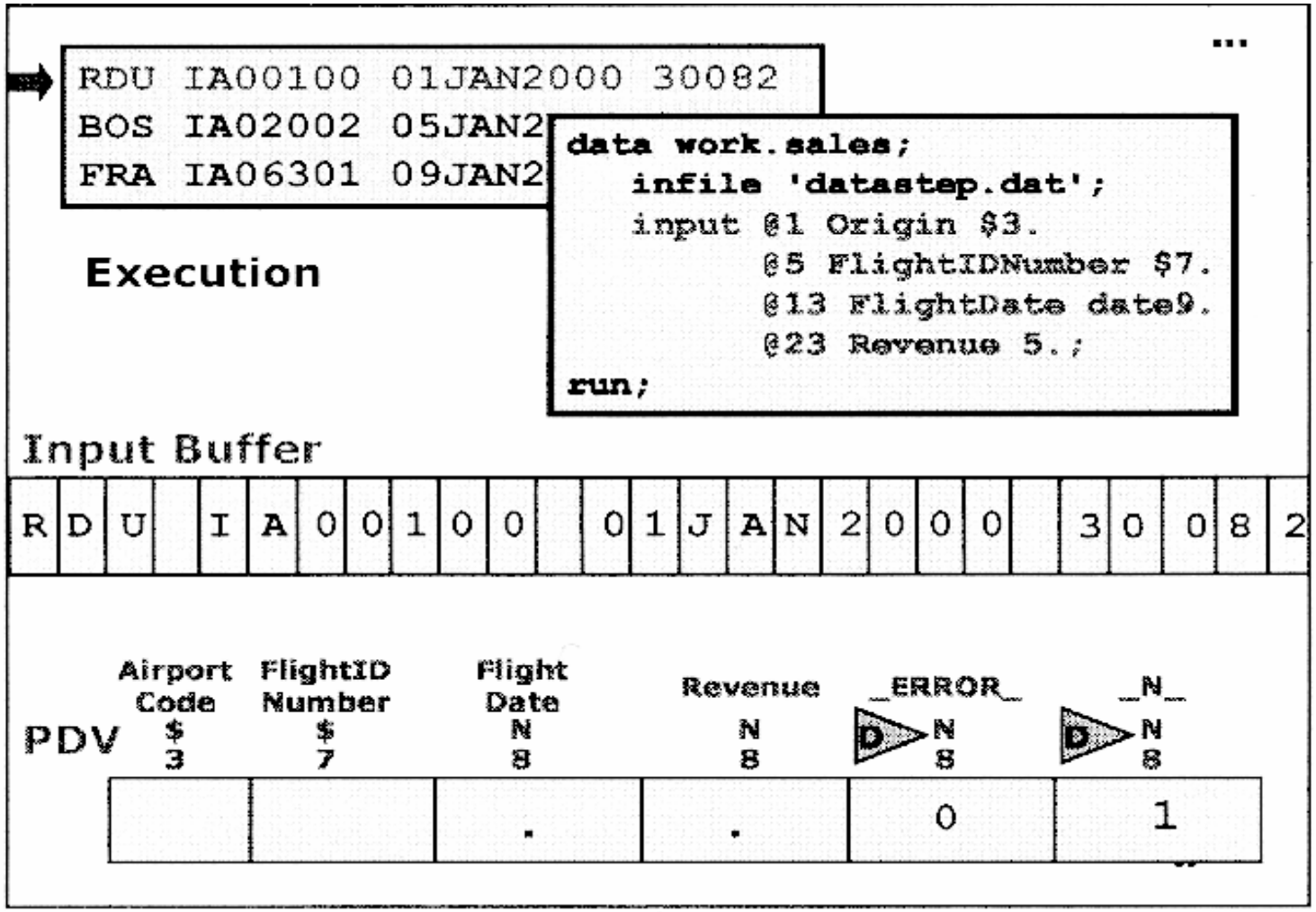


30

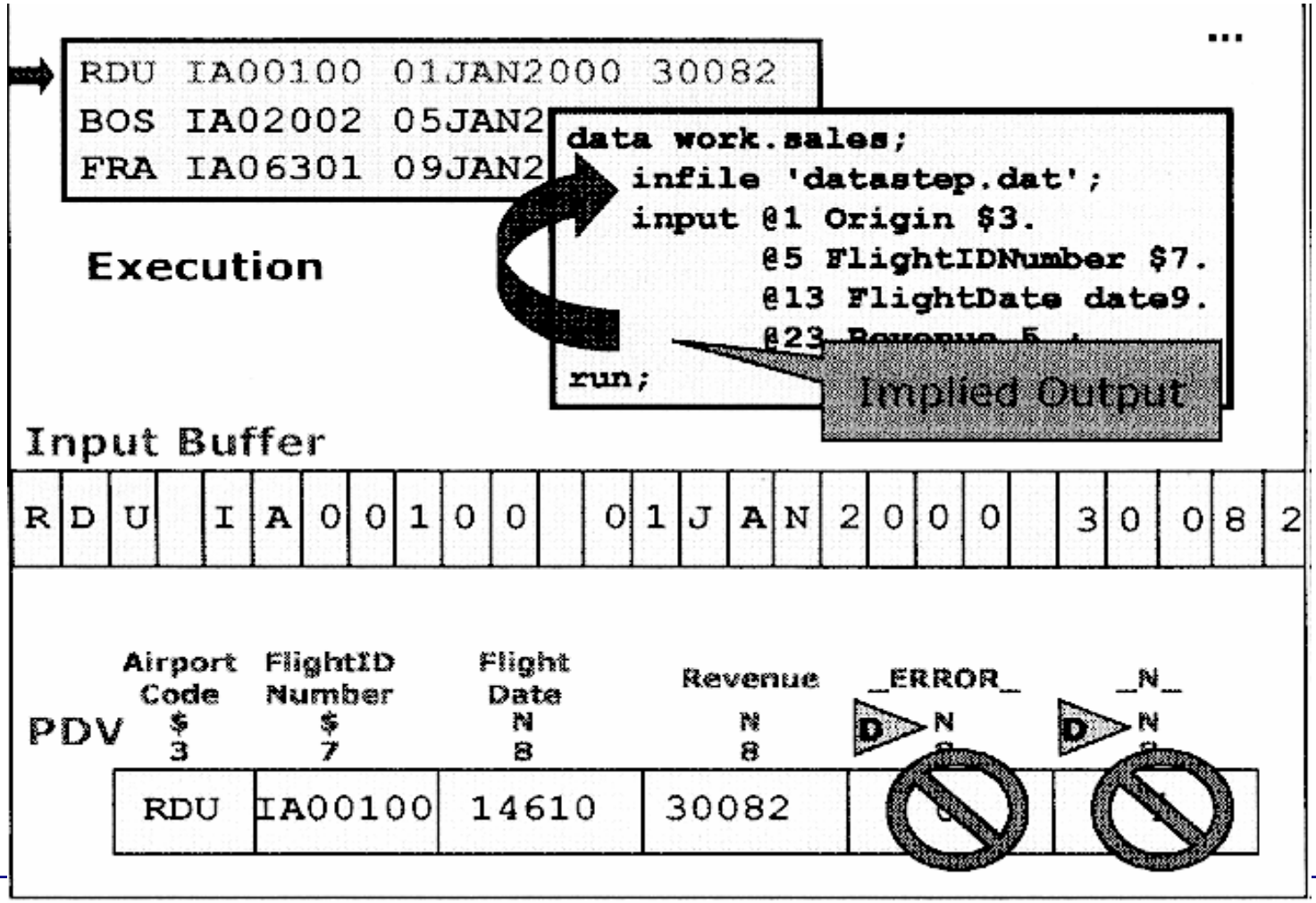
Exécution : phase initialisation des variables dans le PDV



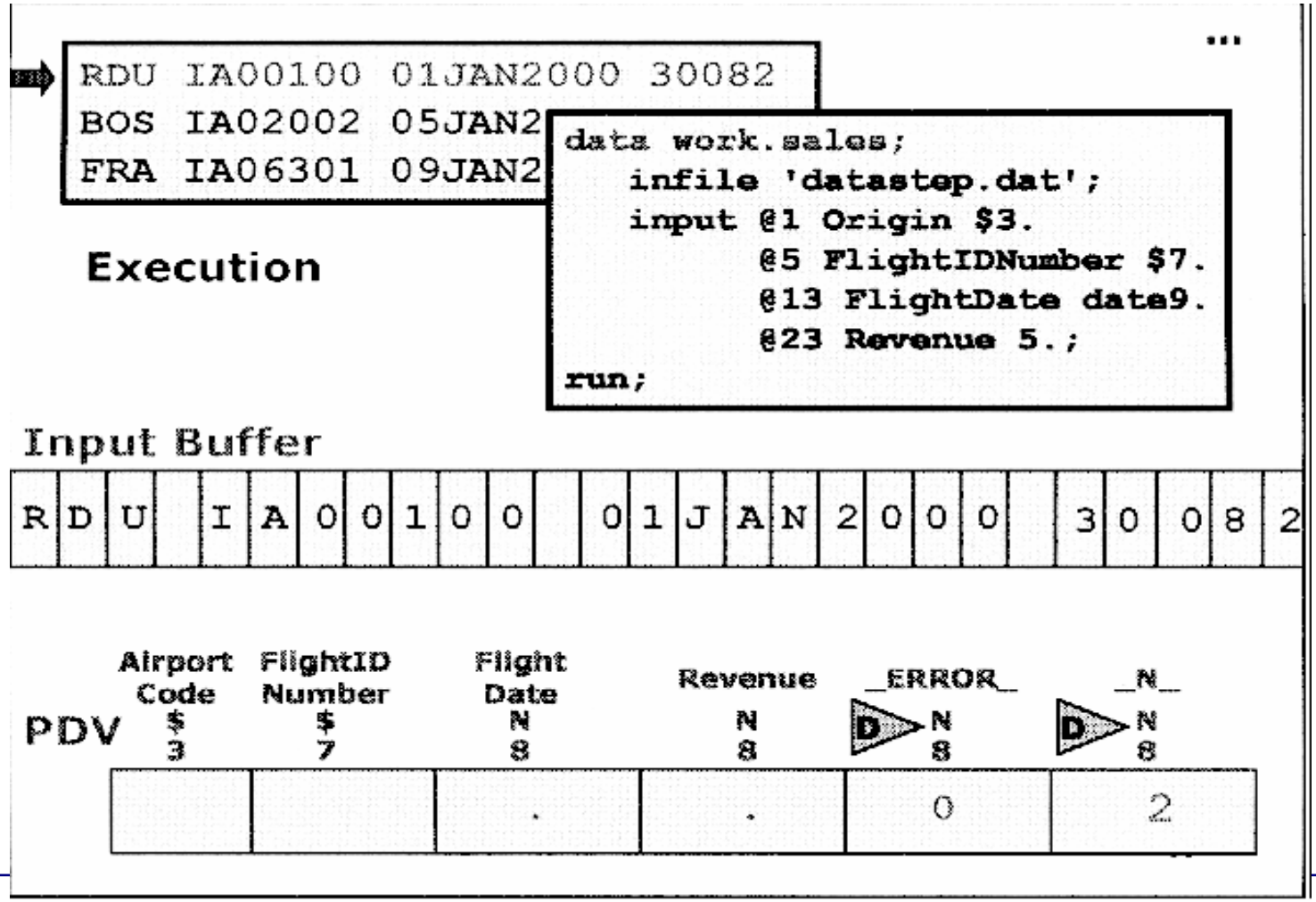
Exécution : Le tampon d'entrée contient les données à lire



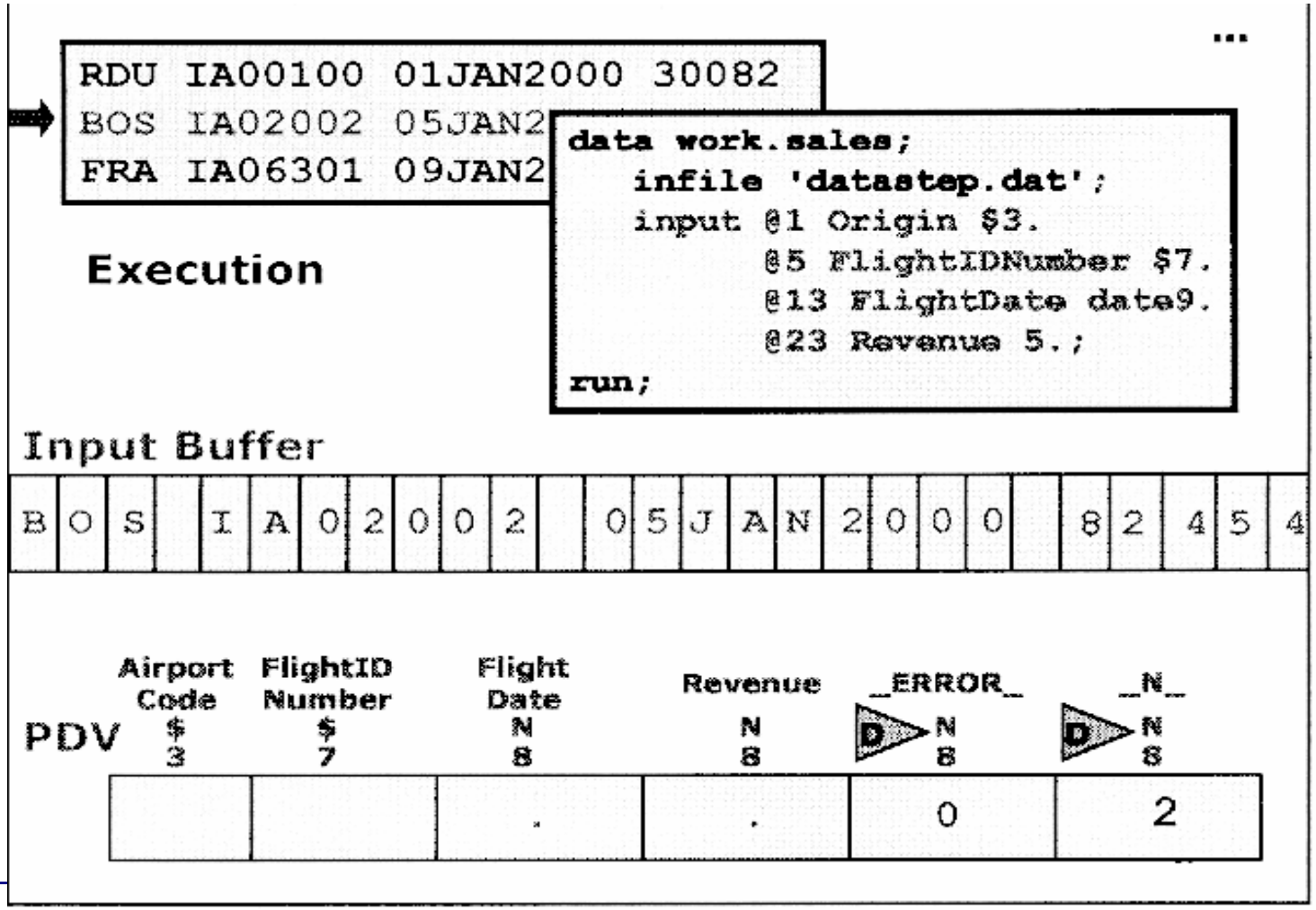
Exécution : Le PDV est rempli et l'observation est rajoutée à la table de sortie



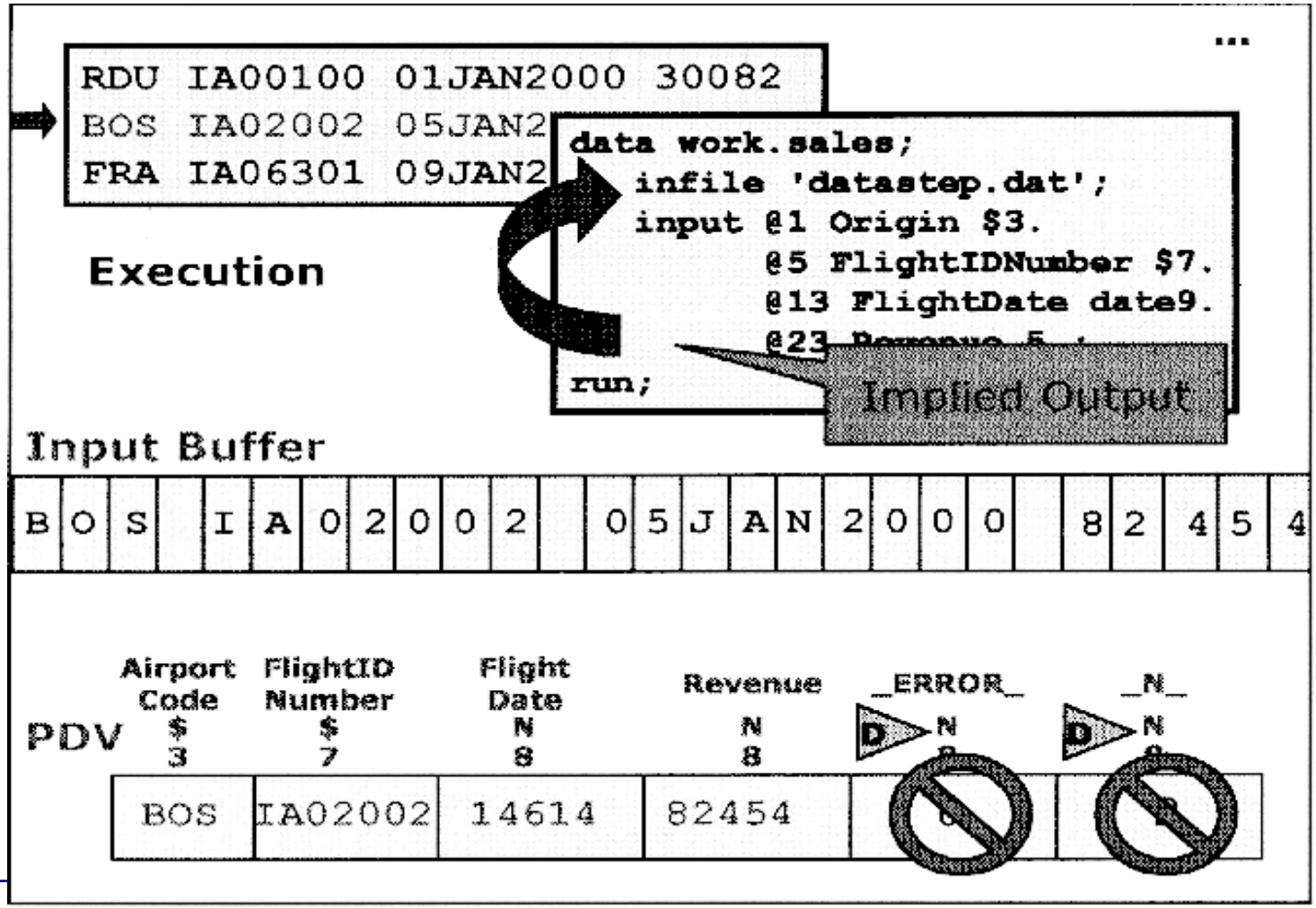
Exécution : retour à la phase d'initialisation des variables dans le PDV



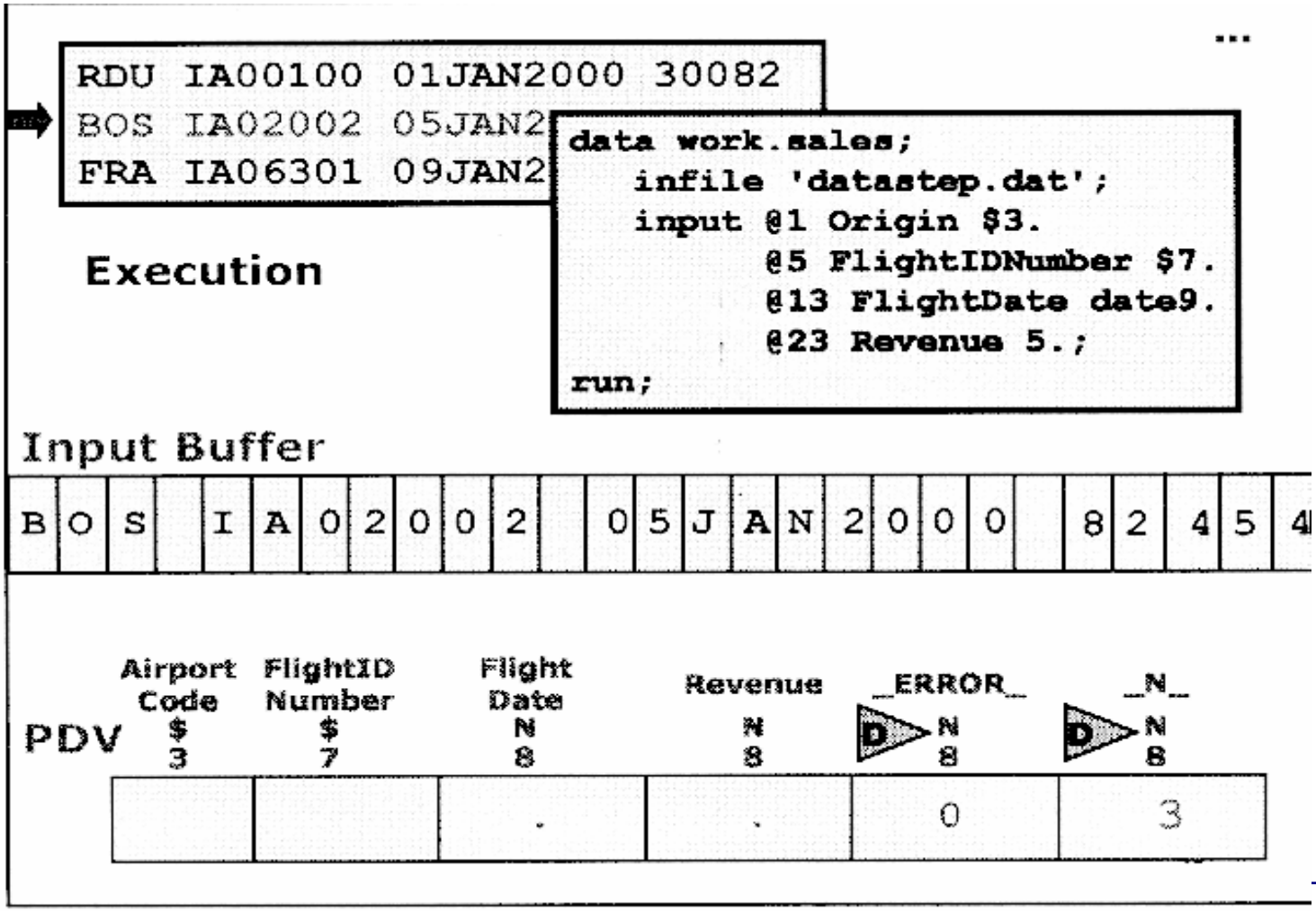
Lecture à nouveau dans le fichier. Le résultat est rangé dans le tampon d'entrée



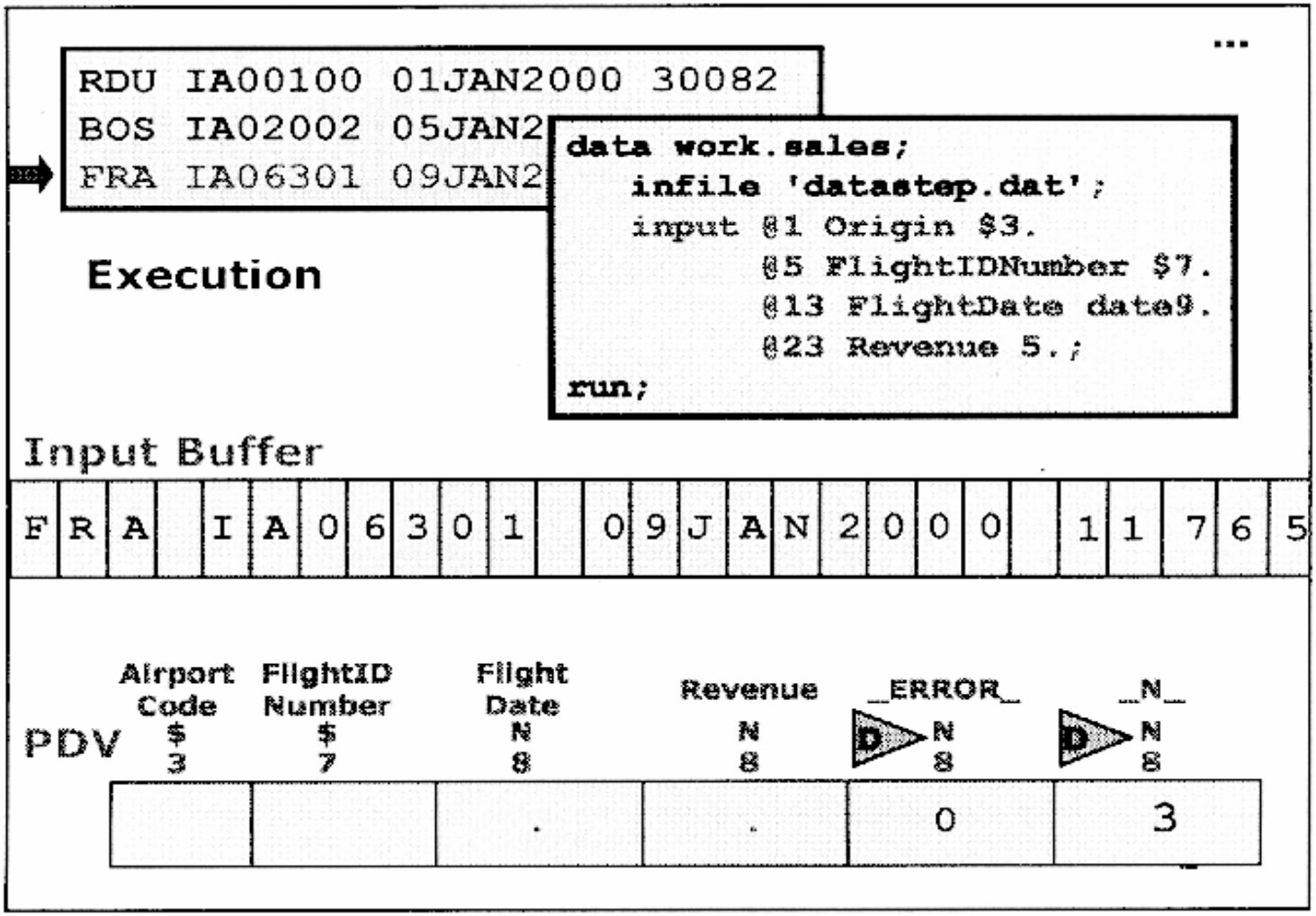
Le PDV est rempli et à la fin de l'étape data, l'observation est ajoutée à la table de sortie



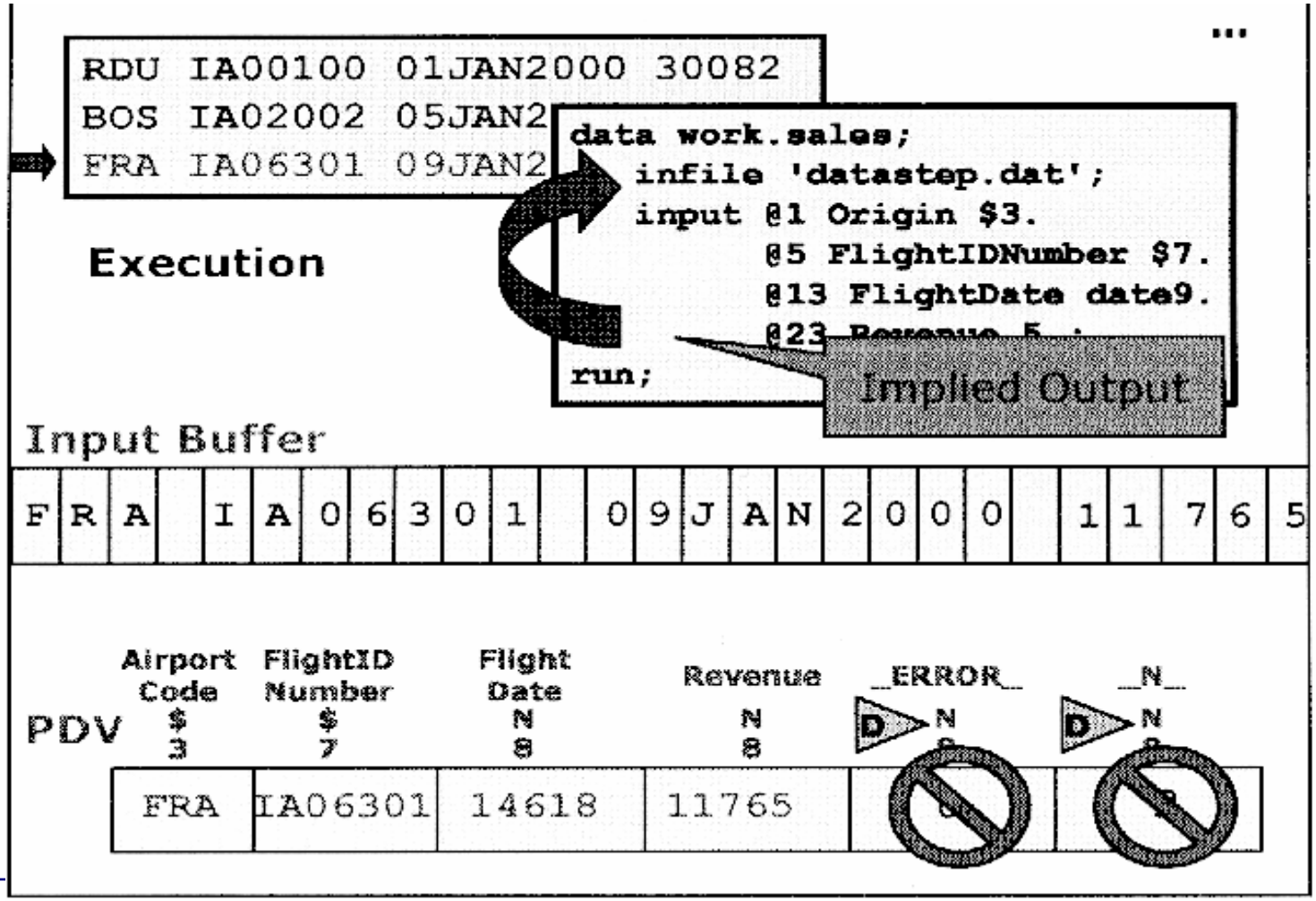
Initialisation des variables du PDV



Le tampon d'entrée reçoit les données du fichier



Le PDV est rempli à partir du tampon d'entrée. L'observation est ajoutée au dataset de sortie.



A la fin de l'exécution de l'étape DATA, le résultat est

Resulting Data Set

Airport Code	FlightID Number	Flight Date	Revenue
RDU	IA00100	01JAN2000	30082
BOS	IA02002	05JAN2000	82454
FRA	IA06301	09JAN2000	11765

44

Les boucles conditionnelles

- IF … THEN … et ELSE : permettent de faire des traitements différents selon qu'une condition est réalisée ou non

Syntaxe

```
Data nom_table;
  set nom_table;

  if condition then instruction ;
  else if condition then do;
         instruction1 ;
         instruction2 ;
  end;
  else instruction ;
Run ;
```

Les programmes SAS - l'étape DATA

Les boucles conditionnelles

- DO … END : permettent d'inclure plusieurs instruction dans une condition

Syntaxe

```
Data nom_table;
  set nom_table;

  VarTemp = 23 ;

  do Compteur1 = 1 to 10 ;
      instruction1 ;

      do Compteur2 = VarTemp to 8 by –1 ;
         instruction2 ;
      end;

      instruction3 ;
  end;
Run ;
```

Les programmes SAS - l'étape DATA

Les boucles conditionnelles

- SELECT … WHEN… OTHERWISE … permettent d'exécuter une instruction conditionnelle parmi une série d'autres.

  - **Syntaxe**

```
Data nom_table;
  set nom_table;

  select(variable);
       when(valeur1) instruction1 ;
       when(valeur2) do ;
         instruction2 ;
         …
                     end;
       otherwise instruction ;
  end;
run;
```

Les programmes SAS - l'étape DATA

Les opérateurs usuels

Opérateurs		Signification
=	Eq	Égal à
^=	Ne	Non égal à
>	Gt	Plus grand que
<	Lt	Plus petit que
>=	Ge	Plus grand que ou égal à
<=	Le	Plus petit que ou égal à
&	And	Et
!	Or	Ou
	In	Dans une liste in (element1,element2….)
	!!	Concaténation des caractères ou des chaînes de caractères

Les programmes SAS - l'étape DATA

Les fonctions

	Fonctions	Signification
Fonctions statistiques	N (arg1,arg2,…,argn)	Nombre de valeurs non manquantes
	NMISS (arg1,arg2,…,argn)	Nombre de valeurs manquantes
	SUM (arg1,arg2,…,argn)	Somme des valeurs non manquantes
	MEAN (arg1,arg2,…,argn)	Moyenne arithmétique
	MIN (arg1,arg2,…,argn)	Plus petite valeur non manquante
	MAX (arg1,arg2,…,argn)	Plus grande valeur non manquante
	RANGE (arg1,arg2,…,argn)	Différence entre MIN et MAX
	STD (arg1,arg2,…,argn)	Écart-type
Fonctions arithmétiques et mathématiques	ABS (arg)	Valeur absolue de l'argument
	INT (arg)	Partie entière de l'argument
	ROUND (arg1,arg2,…,argn)	Arrondit arg1 à l'unité d'ordre arg2 la plus proche
	SQRT (arg)	Racine carrée de l'argument
	EXP (arg)	Exponentiel
	CEIL (arg)	Retourne la plus petite valeur entière supérieure
	FLOOR (arg)	Retourne la plus grande valeur entière inférieure

Les programmes SAS - l'étape DATA

Les fonctions

	Fonctions	Signification
Fonctions sur les variables caractères	COMPRESS(arg<,'*caractère*'>)	Compacte l'argument en supprimant tous les *caractères* spécifiés en argument (blanc par défaut)
	UPCASE (arg)	Transforme les minuscules en majuscules
	LOWCASE (arg)	Transforme les majuscules en minuscules
	LEFT(arg)	Cadre la variable à gauche
	RIGTH(arg)	Cadre la variable à droite
	TRIM(arg)	Supprime les blancs à droite et à gauche
	LENGTH(arg)	Donne la longueur de la variable
	SCAN(arg,n <,'*séparateur*'>);	Donne le mot construit de rang n en découpant la variable selon les séparateurs (blanc par défaut)
	SUBSTR(arg,n,l);	Extraction d'une chaîne de caractères de longueur l à partir de la position n de la variable arg
Fonctions de Conversion	PUT(arg,format)	Retourne une valeur caractère, créée à partir d'une valeur numérique ou caractère dans un certain format.
	INPUT (arg,informat)	Retourne une valeur numérique ou caractère, créée à partir d'une valeur caractère dans un certain format.

La fonction INPUT

- Transformation d'une chaîne de caractères en valeur numérique
  - Syntaxe

VariableNumerique = input(VariableCaractere,Informat)

  - Exemple

```
data _null_ ;
   format Resultat ddmmyy10. ;

   ChaineATraiter = "251198" ;
   Resultat=input(ChaineATraiter,ddmmyy6.);
run ;
```

  - Résultat du traitement, Resultat = 25/11/1998

La fonction PUT

- Conversion des variables avec le format spécifié
  - Syntaxe

VariableResultat = put(VariableSource,format)

  - Exemple

```
data _null_ ;

   ValeurATraiter = 14208 ;
   VariableResultat=put(ValeurATraiter,ddmmyy10.);

run ;
```

  - Résultat du traitement, VariableResultat = 25/11/1998

Les programmes SAS - l'étape DATA

La fonction SUBSTR

- Deux modes d'utilisation sont possibles

- Remplacement d'une partie d'une chaîne de caractères

  - Syntaxe

    Substr(ChaineATraiter,PositionInitiale,NombreCaractère) = « ChaîneDeRemplacement »

  - Exemple

```
data _null_ ;
  ChaineATraiter = "Valeur Initiale" ;
  substr(ChaineATraiter,8,8) = "Finale" ;

  put ChaineATraiter= ;
run ;
```

  - Résultat du traitement, ChaineATraiter = « Valeur Finale »

La fonction SUBSTR

- Extraction d'une chaîne de caractères
  - Syntaxe

ChaineFinale = Substr(ChaineATraiter,PositionInitiale,NombreCaractère)

  - Exemple

```
data _null_ ;
  ChaineATraiter = "Valeur Initiale" ;
  ChaineFinale = substr(ChaineATraiter,8,8);

  put ChaineFinale= ;
run ;
```

  - Résultat du traitement, ChaineFinale = « Initiale »

Les programmes SAS - l'étape DATA

La fonction SCAN

- Extraction d'une zone délimitée par des séparateurs

  - Syntaxe

    ChaineFinale = Scan(ChaineATraiter, n <, *'séparateur'***)**

  - Exemple

```
data _null_ ;
  ChaineATraiter = "Valeur Initiale" ;
  ChaineFinale = scan(ChaineATraiter,2," ");

  put ChaineFinale= ;
run ;
```

  - Résultat du traitement, ChaineFinale = « Initiale »

Les programmes SAS - l'étape DATA

La fonction TRANSLATE

- Remplacement de caractères dans une chaîne

  - Syntaxe

ChaineFinale = Translate(ChaineATraiter, CaracteresResultat, ARemplacer)

  - Exemple

```
data _null_ ;
  ChaineATraiter = "Valeur Initiale" ;
  ChaineFinale = translate(ChaineATraiter,"p","l");

  put ChaineFinale= ;
run ;
```

  - Résultat du traitement, ChaineFinale = « Vapeur Initiape »

Les fonctions date et temps

Fonctions	Signification
DATE()	Retourne le nombre de jours écoulés entre le 01011960 et la date de jour
TIME ()	Retourne l'heure courante
DATETIME()	Retourne la date et l'heure courante
YEAR(date)	Retourne en valeur numérique l'année d'une variable contenant une date
MONTH(date)	Retourne en valeur numérique le mois d'une variable contenant une date
DAY(date)	Retourne en valeur numérique le jour d'une variable contenant une date
MDY(mois,jour,année)	Retourne une date à partir de 3 valeurs numériques (jour, mois, année)
INTNX(période, date, nombre)	Retourne une nouvelle date en fonction de la période et du nombre.
INTCK(intervalle,date1,date2)	Retourne l'intervalle entre les dates date1 et date2. Intervalle : 'year', 'month', 'week', …

La fonction INTCK

- Intervalle entre deux dates, selon l'unité voulue (year, quarter, day, …)

  - Syntaxe

```
Intervalle = Intck(Periode,dateDebut,dateFin)
```

  - Exemple

```
data _null_ ;

  Intervalle=intck('day','01JAN2002'd,'31DEC2002'd) ;
  put intervalle= ;
run ;
```

  - Résultat du traitement, Intervalle = 364

La fonction MONTH

- Extrait le mois d'une date SAS

  - Syntaxe

```
Mois = Month(date)
```

  - Exemple

```
data _null_ ;

  Mois = month('01AUG2002'd) ;
  put Mois= ;
run ;
```

  - Résultat du traitement, Mois = 8

La fonction DATEPART

- Extrait une date d'une variable de type datetime.

  - Syntaxe

```
VariableDate = datepart(VariableDatetime)
```

  - Exemple

```
data _null_ ;
  format date ddmmyy10. ;

  date = datepart('01AUG2002:9:45'dt) ;
  put date= ;
run ;
```

  - Résultat du traitement, Date = 01/08/2002

Les programmes SAS - les fichiers externes

- **La lecture / écriture de données stockées dans un fichier externe peut se faire de 3 façons**

  - Utilisation du Wizard, import ou export selon le cas
    - Import Transformation de fichier externe en table SAS
    - Export Transformation de table SAS en fichier externe

  - Utilisation des procédures IMPORT et EXPORT

  - Utilisation de l'étape DATA

- **Nous allons détailler la troisième méthode**

Les programmes SAS - les fichiers externes

- **Il faut distinguer la lecture de l'écriture.**

- **Les instructions à retenir**
  - Lecture — INFILE, FILENAME, INPUT
  - Ecriture — FILE, FILENAME, PUT

Les programmes SAS - les fichiers externes

- **Pour travailler sur un nom logique, il faut tout d'abord établir le lien entre le nom logique et le fichier physique.**

- **L'instruction FILENAME doit être utilisée. Elle se positionne en dehors d'une étape DATA**

- **Syntaxe**
  - Filename <Nom Logique> « <Nom physique> » ;

FILENAME fileref "nom_local_du_fichier_UNIX";

- **Associer un nom dans le programme au fichier UNIX de données. Pour une plus grande souplesse dans la localisation des fichiers et pour une plus grande probabilité**
- **Avant une étape DATA.**
- **Le nom local sera le nom utilise dans le programme pour designer le fichier de données. En particulier dans l'instruction INFILE.**
- **On peut utiliser les variables d'environnement (dont $HOME) dans les noms de fichier UNIX (guillemets plutôt que apostrophes pour permettre leur résolution).**

FILENAME (V8)

FILENAME libref "nom_local_du répertoire";

Il n'est plus obligatoire lors de la lecture ou création d'un fichier SAS de données. la référence directe d'un fichier système SAS permanent est maintenant possible :

DATA travail; **désigne le fichier temporaire travail,**

DATA 'travail'; **le fichier permanent travail dans le répertoire courant,**

DATA '/jmz/travail'; **représente le fichier permanent travail dans le répertoire UNIX /jmz.**

Ces nouveautés ne contribuent pas à la portabilité des programmes.

Simple quote versus double quote

"c'est une chaîne de caractère"

'c''est une chaîne de caractère'

Une chaîne de caractère doit être encadrée de guillemets (") ou d'apostrophe(')

Pour les libname ou filename, s'il y a des espaces ou de la ponctuation, il vaut mieux le double quote

Les programmes SAS - les fichiers externes

- **Pour lire ou écrire des données externes, il faut établir le lien entre les données et le fichier source ou destinataire.**

  - Pour lire — Instruction INFILE
  - Pour écrire — Instruction FILE

- **Syntaxe**
  - « INFILE <Chemin \ Nom réel fichier> options » ou « INFILE <Nom logique > options »
  - « FILE <Chemin \ Nom réel fichier> options » ou « INFILE <Nom logique > options »

  - Les options principales, dans les deux cas

    lrecl — spécifie la longueur maximum d'un enregistrement

    dlm — spécifie le délimiteur des variables

    dsd — lecture des délimiteurs consécutifs (ex ;;;)

    end=Fin — exécute l'étiquette fin lors de la lecture de la marque de fin de fichier

Les programmes SAS - les fichiers externes

- **Le lien établit, il faut écrire ou lire.**
  - Pour lire Instruction INPUT
  - Pour écrire Instruction PUT

- **Il existe 4 modes de lecture/écriture dans une instruction INPUT/PUT**
  - Colonne
  - Liste
  - Formaté
  - Nommé (peu utilisé)

Les programmes SAS - les fichiers externes

- **Le mode colonne permet de lire/écrire des variables en position fixe**

Syntaxe :

```
Input nom_variable position_début-position_fin;
```

```
Put nom_variable position_début-position_fin;
```

Exemple :

```
Dupont    Eric      29      1.82
Dupond    Patrice   31      1.62
Martin    Franck    24      1.72
```

Code :

```
Data WORK.PERSONNE ;
  length Nom $9 Prenom $6 ;

  infile 'c:\Fichiers\Personne.txt' ;
  input  Nom 1-9 Prenom 11-16 Age 21-23 Taille 30-33 ;
Run ;
```

Les programmes SAS - les fichiers externes

- **Le mode liste permet de lire des variables en position non fixe**

Syntaxe :

```
Input nom_variable ;
```

```
Put nom_variable ;
```

Exemple :

```
Dupont    Eric 22 1.82
Dupond        Patrice    31 1.62
Martin    Franck 24 1.72
```

Code :

```
Data WORK.PERSONNE ;
  length Nom $9 Prenom $6 ;

  infile 'c:\Fichiers\Personne.txt' ;
  input  Nom  Prenom  Age  Taille  ;
Run ;
```

Les programmes SAS - les fichiers externes

- **Le mode formaté permet de lire des variables en position non fixe**

Syntaxe :

Input nom_variable_num informat. nom_variable_car $informat. ;

Put nom_variable_num format. nom_variable_car $format. ;

Exemple :

```
Dupont   Eric   22  1.82
Dupond   Patrice 31  1.73
Martin   Franck  24  1.72
```

Code :

```
filename fic "C\temp\personne.txt" ;

Data WORK.PERSONNE ;
  infile fic ;
  input  Nom  $9. +1 Prenom $7. +1 Age 4.
Taille 4. ;
Run ;
```

Les programmes SAS - les fichiers externes

- **Les pointeurs à utiliser**
  - #n permet de se positionner sur une ligne
  - @n permet de se positionner sur une colonne
  - +n permet de déplacer le pointeur de n colonnes

  N représente le numéro de la ligne ou de la colonne, selon le cas.

  - @ permet de conserver les informations lues en mémoire durant une itération de l'étape DATA
  - @@ permet de conserver les informations lues en mémoire durant des itérations différentes de l'étape DATA
  - / permet de lire plusieurs lignes pour constituer une observation

Les programmes SAS - les fichiers externes

- **Les pointeurs à utiliser**
  - Exemple d'utilisation de @ en fin d'instruction input

```
0011DUPOND    BREST        15/01/97
0012PIERRE 1950 PAUL 1960
0013CC 50 CX 100 CP 100000
0021DURAND    PARIS        10/12/96
0023CD 200 CC 500000
0031TOTO      NICE         13/09/96
0032BRUNO  1945
0033CP 20000
```

  - Code

```
data table (drop = type );
  format date ddmmyy10. ;

  infile 'tordu.txt' lrecl=50 ;
  input @4 type $1. @ ;

  if type=1 then input @1 num $3. +1 nom $ ville $ date : ddmmyy8. ;
  else delete ;
run ;
```

Les programmes SAS - les fichiers externes

- **Les pointeurs à utiliser**
  - Exemple d'utilisation de @@ en fin d'instruction input

```
Dupont Eric 22
Fleche Thierry 35
Durant Yves 56
Dupond Patrice 31
Martin Franck 24
Durand Antoine 3
```

  - Code

```
data table;
   input Nom $ Prenom $ Age  @@;
   datalines;
Dupont Eric 22 Fleche Thierry 35 Durant Yves 56
Dupond Patrice 31
Martin Franck 24 Durand Antoine 3
;
Run ;
```

Les programmes SAS - les fichiers externes

- **Les pointeurs à utiliser**
  - Exemple d'utilisation de /

```
0011DUPOND    BREST        15/01/97
26 02.98.49.45.43
0021DURAND    PARIS        10/12/96
32 01.95.78.93.75
0031TOTO      NICE         13/09/96
43 06.15.43.21.78
```

  - Code

```
data table ;
    format date ddmmyy10. ;

    infile 'nLigneUneObs.txt' lrecl=50 ;
    input @1 num $3. +1 nom $ ville $ date :ddmmyy8. / age telephone $14.;

run ;
```

L'instruction SET

- Lecture séquentielle en deux étapes
  - Lors de la compilation, lecture du descripteur, placement des variables dans le PDV, Program Data Vector
  - Lors de l'exécution, lecture séquentielle des observations de la table
- Utilisation avec l'instruction BY
  - Lecture d'observations par groupe
  - Utilisation préalable de la procédure SORT
- Exemple

```
proc sort data = donnees.classe ;
  by sexe age ;
run ;

data _null_ ;
  set donnees.classe ;
  by sexe age ;
run ;
```

L'instruction SET

- Utilisation des variables temporaires FIRST.NOMVAR et LAST.NOMVAR
- Les instructions FIRST et LAST s'utilisent sur les variables triées d'une table. Le code testé est binaire (0 ou 1);
- Fonctions
  - **FIRST** permet de savoir si une modalité apparaît pour la première fois lors de la lecture d'une table
  - **LAST** permet de savoir si une modalité apparaît pour la dernière fois lors de la lecture d'une table

L'instruction SET

- Syntaxe : **first.***NomVariable* ou **last.***NomVariable*

- Exemple : Une table est triée sur l'année (YEAR nom de la variable)

Year	Quarter	First.Year	Last.Year
2000	A	1	0
2000	B	0	1
2001	A	1	1
2002	A	1	0
2002	B	0	0
2002	C	0	1

L'instruction SET

- Plusieurs tables par instruction

  - Lecture de chaque table de façon séquentielle de la gauche vers la droite

  - Exemple

```
data _null_ ;
  set donnees.classe donnees.classenew;
run ;
```

  - Cas de figure
    - Les structures sont identiques, les observations de la seconde table sont ajoutées après celles de la première , pour les variables correspondantes.

    - Les structures ne sont pas identiques, les observations de la seconde table sont ajoutées après celles de la première.

      Pour les variables non communes, les cellules contiennent des données manquantes.

L'instruction SET

- Plusieurs tables par instruction et instruction IN
  - Lecture de chaque table de façon séquentielle, connaissance de la provenance des observations
  - L'instruction IN permet de savoir d'où provient l'observation et de la traiter en conséquence
  - Exemple

```
data Test ;
  set donnees.classe(in=a) donnees.classenew(in=b);
  if b then Info = « Nouveau »  ;
  else Info = « Ancien » ;
run ;
```

Les observations sont classées selon leur origine, classe ou classenew.

L'instruction SET

- Plusieurs tables par instruction, instruction BY
  - Lecture de chaque table de façon séquentielle de la gauche vers la droite et classement des observations de la table en sortie selon les variables de l'instruction BY
  - Exemple

```
data classe2002 ;
  set donnees.classe donnees.classenew ;
  by nom prenom ;
run ;
```

  - Cas de figure
    - Les structures sont identiques, les observations sont classées par nom prenom dans la table classe2002.
    - Les structures ne sont pas identiques, les observations sont classées par nom et prénom dans la table classe2002.

      Pour les variables non communes, les cellules contiennent des données manquantes.

- **L'instruction SET**
  - Plusieurs instructions SET par étape DATA
    - Lecture de chaque table simultanément, observation par observation
    - Cas de figure pour la concaténation des observations ayant le même ordre
      - Les variables sont différentes et chacune est alimentée
      - Les variables sont communes, l'information de la table appelée dans la dernière instruction SET est retenue
    - La lecture s'arrête à la dernière observation de la plus petite table
      - Si donnees.classe contient 390 observations et donnees.classenew 300, la table en sortie aura
        - ✓ 300 observations si donnees.classenew est appelée en second
        - ✓ 390 observations si donnees.classe est appelée en second.

- **L'instruction SET**
  - Les options POINT et KEY

    Elles permettent un accès direct aux observations.

  - Syntaxe

    set NomTable point = VariableNumerique

    set NomTable key = NomIndex

  - Pré-requis
    - L'utilisation de POINT nécessite l'alimentation d'une variable numérique qui va contenir le numéro de l'observation à extraire.
    - L'utilisation de KEY nécessite la création d'un index et l'alimentation de la ou des variable(s) formant cet index

L'instruction MERGE

- Elle permet de lire et de grouper des observations provenant de plusieurs tables.

- Le comportement varie en fonction
  - De la présence de variables communes ou différentes dans les tables
  - De la présence d'une instruction BY

- **L'instruction MERGE**
  - Instruction BY absente
  - La lecture des tables se fait de la gauche vers la droite, observation par observation. La lecture ne s'arrête pas à la dernière observation de la table la plus petite
  - Pour les variables communes
    - L'information retenue est celle contenue dans l'observation correspondante de la table la plus à droite.
    - En cas de valeurs manquantes dans la table la plus à droite, cette valeur sera aussi retenue dans la table en sortie
  - Pour les variables non communes
    - L'information est inscrite dans la table finale.
    - Si aucune variable n'est commune, cela n'a pas d'importance

L'instruction MERGE

- Instruction BY présente

- La lecture des tables se fait de la gauche vers la droite, observation par observation.
  La lecture ne s'arrête pas à la dernière observation de la table la plus petite.
  Les tables doivent être triées sur les variables présentes dans le BY
  Les observations de la table en sortie sont triées par ces mêmes variables

- Pour les variables communes
  - L'information retenue est celle contenue dans l'observation correspondante de la table la plus à droite.
  - En cas de valeurs manquantes dans la table la plus à droite, cette valeur sera aussi retenue dans la table en sortie

- Pour les variables non communes
  - L'information est inscrite dans la table finale.

- Pour les observations qui n'existe pas dans une des tables
  - L'observation est créée dans la table en sortie

L'instruction MERGE

- Plusieurs tables par instruction, instruction BY et IN

  - Lecture de chaque table de façon séquentielle, connaissance de la provenance des observations et filtre des données

  - L'instruction IN permet de filtrer les doublons, les tables de l'instruction MERGE doivent être triées

  - Exemple

```
data donnees.classe2002 ;
  merge donnees.classe(in=a) donnees.classenew(in=b);
  by nom age ;
  if a and not b ;
run ;
```

Les observations sont classées par nom et age. Seules celles qui présentes dans la table A et pas dans la table B sont extraites.

- **L'instruction UPDATE**
  - Elle est utilisée pour mettre à jour une table MAITRESSE, celle à gauche dans l'instruction , à partir d'une table de mise à jour, celle à droite.
  - L'instruction BY est à positionner pour connaître la clé de mise à jour.
  - La table MAITRESSE ne doit pas contenir de doublons sur cette clé qui est présente dans les deux tables. S'il y a des doublons, seule la première observation est mise à jour.
  - La mise à jour de variables se fait pour les observations qui ont une clé commune.
  - L'ajout d'observations est fait lorsqu'une clé est présente dans la table de mise à jour et pas dans la table MAITRESSE.

- **L'instruction MODIFY**
  - Elle permet la mise à jour des observations d'une table
    - Grâce à un traitement fait dans l'étape DATA, sans ou avec les options **key** et **point**.
    - Grâce à une table de mise à jour, utilisation de l'instruction BY
  - Aucun tri préalable n'est nécessaire, ni d'index (excepté lors de l'utilisation de l'option **Key**). Il ne faut pas de doublon dans la table à mettre à jour sinon seule la première observation rencontrée est mise à jour
  - Lors de l'utilisation d'une table de mise à jour, les modifications ne se font que pour les observations communes (à clés égales) entre les deux tables.

Les programmes sas – éléments avancés

Les index

- Un index est associé à une table SAS.
- Il est stocké dans un fichier qui porte le même nom que la table SAS associée
- Il n'est pas visible à travers l'explorateur SAS
- Une table index peut contenir plusieurs index d'une même table

- Il existe deux types d'index
  - Simple — porte sur une variable et a le nom de celle-ci
  - Composé — porte sur plusieurs variables et a un nom choisi par l'utilisateur

- Il est créé pour améliorer les performances d'accès aux données de sa table de référence.

Les index

- Création d'un index simple

```
proc sql ;
  create index ident on donnees.clients
  ;
quit ;
```

- Création d'un index composé

```
proc sql ;
  create index compos1 on donnees.clients(type, ident)
  ;
quit ;
```

Les programmes sas – éléments avancés

Les index

- Ils sont utilisés dans plusieurs cas
  - Instructions SET et MODIFY avec l'option KEY= — Utilisation systématique
  - Instruction BY
  - Instruction WHERE
  - Jointure dans la procédure SQL

Les index

- Leur utilisation peut-être suivi grâce à l'option MSGLEVEL=I

```
102  data selection ;
103    set donnees.clients (where=( ident = "51001380")) ;
INFO: Index IDENT selected for WHERE clause optimization.
104  run ;

NOTE: There were 1 observations read from the data set DONNEES.CLIENTS.
      WHERE ident='51001380';
NOTE: The data set WORK.SELECTION has 1 observations and 8 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds
```

- Leur utilisation peut être contrôlée dans une étape DATA
  - IDXWHERE=YES/NO Utilisation ou non des index
  - IDXNAME=INDEXNAME Nom de l'index à utiliser

Les index

- La maintenance, extrait de la documentation SAS

Task	Result
delete a data set	index file is deleted
rename a data set	index file is renamed
rename key variable	simple index is renamed
delete key variable	simple index is deleted
add observation	index entries are added
delete observations	index entries are deleted and space is recovered for reuse
update observations	index entries are deleted and new ones are inserted

Les index

- Conseils
  - Positionner un index est avantageux sur une grosse table et sur une variable qui possède beaucoup de modalités.
  - Penser à trier la table avant la création de l'index afin de consommer moins de ressources.
  - Ne pas créer trop d'index sur une table pour ne pas occuper trop d'espace disque
  - Evaluer l'intérêt d'un index en fonction
    - De ce qu'il apporte comme performance lors des accès
    - Du coût de création et d'utilisation lors des accès en modification

La compression des tables SAS

- La première raison de la compression des tables SAS est le gain de place physique.
- La taille d'une table est égale à la taille d'une page multipliée par le nombre de pages.
- Une page de table est la donnée la plus petite que s'échange SAS et le système d'exploitation.

  Sa taille est définie à la création de la table.

La compression des tables SAS

- L'exécution d'une procédure CONTENTS permet d'analyser les caractéristiques d'une table

```
                          The CONTENTS Procedure

Data Set Name: DONNEES.PRDSALE                    Observations:          23040
Member Type:   DATA                               Variables:             11
Engine:        V8                                 Indexes:               0
Created:       10:10 Friday, October 18, 2002     Observation Length:    120
Last Modified: 10:10 Friday, October 18, 2002     Deleted Observations:  0
Protection:                                       Compressed:            NO
Data Set Type:                                    Sorted:                YES
Label:

                        -----Engine/Host Dependent Information-----
                        Data Set Page Size:         12288
                        Number of Data Set Pages:   227
                        First Data Page:            1
                        Max Obs per Page:           102
                        Obs in First Data Page:     83
                        Number of Data Set Repairs: 0
```

La compression des tables SAS

- La structure d'une table compressée est différente de celle d'une table non-compressée.
  - Les observations d'une table non compressée occupent une place identique, fonction de la longueur de chaque variable.

    Les observations d'une table compressée occupent une place différente, fonction de la place réelle occupée par le contenu des variables.
  - Dans les 2 cas, une zone système débute le contenu de chaque page mais leur taille est différente.
    - Table non compressée 16 octets
    - Table compressée 28 octets + 12 octets par observations
  - Chacune contient toutefois des zones correspondant à l'espace non utilisé dans une page.

La compression des tables SAS

- Utilisation de la compression des tables SAS : il faut positionner l'option COMPRESS à YES (NO par défaut)

- Exemple de compression

```
34   data prdsale ;
35     set donnees.prdsale ;
36   run ;

NOTE: There were 23040 observations read from the data set DONNEES.PRDSALE.
NOTE: The data set WORK.PRDSALE has 23040 observations and 11 variables.
NOTE: Compressing data set WORK.PRDSALE decreased size by 32.75 percent.
      Compressed is 232 pages; un-compressed would require 345 pages.
NOTE: DATA statement used:
      real time           0.54 seconds
      cpu time            0.17 seconds
```

La compression des tables SAS

- Caractéristiques de la nouvelle table

```
                     The CONTENTS Procedure

Data Set Name: WORK.PRDSALE                          Observations:          23040
Member Type:   DATA                                  Variables:             11
Engine:        V8                                    Indexes:               0
Created:       10:39 Friday, October 18, 2002        Observation Length:    120
Last Modified: 10:39 Friday, October 18, 2002        Deleted Observations:  0
Protection:                                          Compressed:            CHAR
Data Set Type:                                       Reuse Space:           NO
Label:                                               Point to Observations: YES
                                                     Sorted:                NO

                     -----Engine/Host Dependent Information-----
                     Data Set Page Size:         8192
                     Number of Data Set Pages:   232
                     Number of Data Set Repairs: 0
```

La différence se fait sur la taille de chaque page

La compression des tables SAS

- En plus de l'option COMPRESS, il est possible d'utiliser l'option REUSE.
- Cette option permet l'utilisation de l'espace non utilisé dans chaque page.
  - REUSE = No

    L'espace n'est utilisé que lorsque une observation est mise à jour. Dans ce cas, cette observation ne sera plus dans la même page

    Les nouvelles observations sont ajoutées en fin de fichier
  - REUSE = Yes

    L'espace de chaque page est utilisé en cas de modification d'une observation ou lors de l'ajout d'une observation.

    Une observation modifiée ne change pas de page et une observation peut-être ajoutée dans n'importe quelle page.

Les programmes sas – éléments avancés

- **La compression des tables SAS**
  - Elle apporte un gain de place important.
  - Elle coûte cependant davantage de CPU pour décompresser les données lors des traitements… et les compresser à nouveau…

Les tableaux ou Array

- Un Array est un tableau à une ou plusieurs dimensions. Il est constitué de cellules qui sont référencées par des numéros ou des noms de variable.

  Chaque cellule est utilisée pour stocker une information numérique ou caractère.

- Son utilisation sert à simplifier le code d'une étape DATA lors de
  - La lecture d'une table
  - La réalisation de calculs
  - La génération de variables aux profils identiques

- Exemple
  - Un Array appelé VENTE contient, pour chaque mois, le chiffre d'affaire.

    Ce tableau va contenir 12 cellules
    Chaque cellule est référencée par VENTE(i), i allant de 1 à 12

Les tableaux ou Array

- Leur définition

ARRAY Nom {Nombrecellule} <$> <longueur> <nomElement> <ValeurInitiale>

Nom	Nom du tableau
NombreCellule	Nombre de cellules par dimension
$	Lorsque les variables sont de type caractère
Longueur	Longueur des variables
nomElement	Nom des cellules
ValeurInitiale	Valeur initiale des cellules

Les tableaux ou Array

- Exemple

  Calcul du chiffre d'affaire par mois

```
data Quart ;
  array quart{4} q1-q4 (0 0 0 0);

  set donnees.prdsale ;

  quart(quarter) =  quart(quarter) + Actual ;
  Valeur = quart(quarter) ;

run ;
```

Au final, chaque cellule va contenir les ventes totales du trimestre, toutes années confondues.

Ce tableau est structuré sur une seule dimension.

Les tableaux ou Array

- Tableau à deux dimensions
  - La première représente les lignes, la seconde les colonnes.
  - Exemple
    On enregistre par mois, en ligne, et sur deux années, en colonnes, le chiffre des ventes

```
data MensuelParAn (keep = AnneeNom MoisNom Vente);
  array Mois{12,2}  ;

  set donnees.prdsale ;

  Pointeur = month(month) ;
  if Year = "1995" then Mois(Pointeur,1) + Actual ;
  else if Year = "1996" then Mois(Pointeur,2) + Actual ;
run ;
```

Les programmes sas – éléments avancés

- **Le tri d'une table SAS.**
  - Il est souvent utilisé
  - Le temps de traitement est important sur les grosses tables.
  - Utilisation de la procédure SORT
    - L'option FORCE supprime les index associés
    - L'option OUT crée une copie triée non indexée
    - L'option NODUPKEY élimine les doublons en fonction du contenu de l'instruction BY
  - Indicateur et caractéristiques de tri générés par la procédure CONTENTS
  - Manipulation des données grâce aux variables FIRST. et LAST.

Le tri d'une table SAS.

- Par défaut, le tri réalisé est effectué par le moteur SAS
- Possibilité de réaliser un tri système
  - Il faut positionner l'option SORTPGM
    - SORTPGM = SAS — Tri SAS
    - SORTPGM = HOST — Tri Système
    - SORTPGM = BEST — Choix de l'utilitaire de tri par SAS
  - Quelques options disponibles
    - SORTNAME — Nom de l'utilitaire de tri
    - SORTLIB — Chemin d'accès de l'utilitaire
    - SORTPARM — Paramètres d'appel

    Options disponibles sous Unix.

- **Le tri d'une table SAS.**

  - Gain de l'utilisation du tri système porte sur
    - La CPU
    - Les entrées/sorties

  - Le gain est fonction de la taille des tables : plus leur taille est importante, plus il est intéressant d'utiliser le tri système.

Sommaire

La proc SQL

- **Le langage SQL s'utilise dans SAS par l'intermédiaire de la procédure SQL.**

- **Syntaxe de base :**

```
PROC SQL  ;
   Instructions SQL
   ....
QUIT ;
```

- **Les instructions SQL sont exécutées immédiatement. L'instruction QUIT ferme l'environnement SQL.**

SAS SQL –Création de dataset

- **Le SQL (Structured Query Language) est le langage universel de gestion des données informatiques des SGBD relationnels.**

Syntaxe :

```
PROC SQL ;
CREATE TABLE nom_de_table (
variable1 type1(long1) <LABEL='label1' FORMAT=fmt1 INFORMAT=inf1>,
variable2 type2(long2) <LABEL='label2' FORMAT=fmt1 INFORMAT=inf1>,
variable3 type3(long3) <LABEL='label3' FORMAT=fmt1 INFORMAT=inf1>,
);
QUIT ;
```

SAS SQL –Création de dataset - Exemple

Exemple : créer la table PARC avec les variables :
- Zone (ZONE), entier sur 2 caractères
- Hauteur de l'herbe (HERB), caractère (longueur 6)
- Présence de bosquets (BOSQUET), caractère (longueur 3)
- Accès à l'eau (EAU), caractère (longueur 8)
- Présence d'hippopotames (HIPPO), caractère (longueur 8)

```
proc sql;
create table parc(
ZONE integer label='zone' format=2.,
HERB char(6) label='hauteur de l"herbe',
BOSQUET char(3) label='présence de bosquets',
EAU char(8) label='accès à l"eau',
HIPPO char(8) label='présence d"hippopotames');
```

SAS SQL – La commande SELECT

La commande SELECT permet de rechercher des informations par une sélection :
SELECT variable <,variable>…
FROM nom_de_table ou nom_de_vue <,nom_de_table ou nom_de_vue>…
<WHERE expression>
<GROUP BY variable <,variable…>…>
<HAVING expression>
<ORDER BY variable <,variable>…>;

```
proc sql;
select ZONE, HERB
from parc3
where ZONE>6;
quit;
```

Le résultat d'une requête SELECT est par défaut affiché dans l'OUTPUT

```
ZONE HERB
_____________________________
7 courte
9 courte
12 haute
13 courte
```

SAS SQL – La commande SELECT

On peut aussi stocker le résultat d'une requête SELECT dans une table SAS :
Exemple : on sélectionne toutes les variables pour les zones 7 et plus.

```
proc sql;
create table extract
as select *
from parc3
where ZONE>6;
quit;
```

SAS SQL – La commande SELECT – Les fonctions de groupement

- MEAN, AVG : moyenne
- COUNT, N : compte le nombre de valeurs non manquantes
- CV : coefficient de variation
- MAX : donne la plus grande valeur d'une variable
- MIN : donne la plus petite valeur d'une variable
- SUM : somme
- VAR : variance

Syntaxe :

SELECT fonction1(variable) <AS nomvar1 LABEL='label'>,

fonction2(variable) <AS nomvar2 LABEL='label'>,

FROM table

GROUP BY variable

AS permet de donner un nom à la variable résultat lorsque l'on crée une table en sortie (CREATE TABLE).

LABEL permet d'associer un label à la colonne.

SAS SQL – La commande SELECT – l'expression when

Cette expression permet de changer la représentation des données en donnant un code chiffré à une chaîne de caractères ou inversement, en donnant à un code un intitulé plus significatif.

Exemple : on recode en binaire la variable HIPPO : 1 si présence d'hippopotames, 0 sinon.

```
proc sql;
select *,
    case when HIPPO='aucun' THEN 0
    else 1
    end
    as BIN_HIP
from parc3;
run;
```

On obtient la sortie suivante :

```
ZONE    HERB    BOSQUET         EAU             HIPPO           BIN_HIP
-------------------------------------------------------------------------
1       courte  oui             proche          moyen           1
2       haute   oui             éloignée        aucun           0
3       courte  oui             proche          aucun           0
4       courte  oui             proche          important       1
```

SAS SQL – La commande SELECT – La jointure interne

On sélectionne uniquement les observations communes.

```
PROC SQL ;
SELECT a.*, b.*
FROM table1 AS a,
table2 AS b
WHERE a.var1=b.var1;
QUIT;
```

Cette jointure correspond au cas IF INA AND INB du merge en SAS de base.

SAS SQL – La commande SELECT – La jointure externe à gauche

```
PROC SQL ;
SELECT a.*, b.*
FROM table1 AS a
LEFT JOIN
table2 AS b
ON a.var1=b.var1;
QUIT;
```

Elle correspond au cas IF INA du merge.

SAS SQL – La commande SELECT – La jointure externe à droite

```
PROC SQL ;
SELECT a.*, b.*
FROM table1 AS a
RIGHT JOIN
table2 AS b
ON a.var1=b.var1;
QUIT;
```

Elle correspond au cas IF INB du merge.

SAS SQL – La commande SELECT – La jointure externe complète

```
PROC SQL ;
SELECT a.*, b.*
FROM table1 AS a
FULL JOIN
table2 AS b
ON a.var1=b.var1;
QUIT;
```

Elle correspond au merge by sans condition.

SAS SQL – La commande INSERT INTO

Cette commande est utilisée pour insérer des observations dans une table. Elle nécessite le nom de la table, le nom des variables à initialiser et la valeur à leur assigner.

```
INSERT INTO table SET variable1=valeur1, variable2=valeur2…
```

Exemple : insérer une observation dans la table parc

```
proc sql;
insert into parc3 set ZONE=14, HERB='haute', BOSQUET='non', EAU='éloignée', HIPPO='moyen' ;
quit;
```

On peut également insérer les valeurs de chaque observation avec l'instruction VALUES.

```
INSERT INTO table VALUES (valeur1, valeur2, valeur3…)
```

Exemple : insérer 2 observations dans la table parc3

```
proc sql;
insert into parc3
values(15,'haute','oui','proche','moyen')
values(16,'courte','non','proche','moyen')
quit;
```

Associée à une clause SELECT, cette commande permet de copier un sous-ensemble d'une table dans une autre. La table cible doit exister préalablement, et les types de données doivent correspondre de façon très précise.

Exemple : insérer des observations de la table parc4
(zone 14 à 20) dans la table parc3

```
proc sql;
insert into parc3
select *
from parc4
where ZONE>13;
quit;
```

SAS SQL – La commande UPDATE

La commande UPDATE spécifie le nom de la table qui va être modifiée, suivi d'une clause SET définissant les modifications à apporter, et d'une clause WHERE avec le critère précisant les observations à modifier.

```
UPDATE TABLE
SET variable1=valeur1, variable2=valeur2…
WHERE condition
```

Exemple : remplacer la modalité 'éloignée' de la variable EAU par 'loin' :

```
proc sql;
update parc3
set EAU='loin'
where EAU='éloignée';
quit;
```

SAS SQL – La commande DELETE

Cette commande consiste en une clause DELETE FROM spécifiant le nom de la table concernée, et d'une clause WHERE avec le critère précisant les observations à supprimer.

Exemple : détruire les observations des zones 14 et plus.

```
proc sql;
delete
from parc3
where ZONE>13;
quit;
```

Le dictionnaire de données

- **L ’environnement SAS (les librairies actives) est constitué de nombreux éléments ayant des caractéristiques spécifiques.**
- **L’ensemble de ces informations est regroupé dans le dictionnaire**
- **Il est automatiquement disponible au sein de la session SAS**

Le dictionnaire de données

- **Il est constitué d'un ensemble de tables.**

- **Chacune pointe sur des caractéristiques spécifiques**
  - CATALOGS
  - COLUMNS
  - EXTFILES
  - MACROS
  - MEMBERS
  - INDEXES
  - OPTIONS
  - STYLES
  - TABLES
  - TITLES
  - VIEWS

Le dictionnaire de données

- **Une liste de vues pointe sur le dictionnaire**
- **Elles sont disponibles dans la librairie SASHELP**
- **Elles sont manipulables dans le code**
  - Etape DATA
  - Procédure
- **Il est aussi possible de créer de nouvelles vues ou des tables directement à partir du dictionnaire à l'aide de la procédure SQL.**

Le dictionnaire de données

- **Pour connaître le mécanisme d'alimentation d'une vue**
  - Code

```
proc sql;
   describe view sashelp.vview;
quit ;
```

  - Résultat

```
131  proc sql;
132     describe view sashelp.vview;
NOTE: SQL view SASHELP.VVIEW is defined as:

        select *
          from DICTIONARY.VIEWS;

133  quit ;
```

Le dictionnaire de données

- **Pour connaître le contenu d'une table du dictionnaire**
  - Code

```
proc sql;
   describe table dictionary.extfiles ;
quit ;
```

  - Résultat

```
 proc sql;
141     describe table dictionary.extfiles;
NOTE: SQL table DICTIONARY.EXTFILES was created
like:
create table DICTIONARY.EXTFILES
  (
   fileref char(8) label='Fileref',
   xpath char(1024) label='Path Name',
   xengine char(8) label='Engine Name'
  );
142  quit ;
```

Le dictionnaire de données

- **Pour créer une vue qui pointe sur le dictionnaire**
  - Code

```
proc sql;
   create view Donnees.TabDon as
     select *
     from dictionary.Tables
          where libname = "DONNEES"
   ;
quit ;
```

  - Résultat

```
 proc sql;
144     create view Donnees.TabDon as
145       select *
146       from dictionary.Tables
147       where libname = "DONNEES"
148     ;
NOTE: SQL view DONNEES.TABDON has been defined.
149  quit ;
```

Le dictionnaire de données

- **Pour créer une table qui pointe sur le dictionnaire**
  - Code

```
proc sql;
   create table Donnees.TabDon as
     select *
     from dictionary.Tables
          where libname = "DONNEES"
   ;
quit ;
```

  - Résultat

```
157  proc sql;
158     create table Donnees.TabDon as
159       select *
160       from dictionary.Tables
161       where libname = "DONNEES"
162     ;
NOTE: Table DONNEES.TABDON created, with 10 rows
and 21 columns.
163  quit ;
```

Sommaire

Introduction à l'ODS

- **ODS ou Output Delivery System**
- **Disponible à partir de la version 7 du Système SAS**
- **Rôle**
  - Améliorer la qualité des éditions SAS en
    - Fournissant de nombreuses options
    - Diversifiant les formats d'éditions
  - Elargir les possibilités d'édition des procédures

Introduction à l'ODS

- **Les éditions disponibles**
  - Listing — Edition simple (valeur par défaut)
  - HTML — Edition au format HTML
  - RTF — Edition au format RTF
  - PRINTER — Impression indirecte

- **Les types d'édition**
  - Simple — A partir d 'une procédure PRINT par exemple
  - Complexe — En créant des FRAMEs
    En appliquant des styles personnels
    En liant les données entre elles
    En insérant des liens.

Introduction à l'ODS

Réaliser une édition simple

- Les actions à réaliser
  1. Ouvrir le fichier résultat au format voulu
  2. Exécuter le traitement qui génère les résultats
  3. Fermer le fichier résultat

- Ouvrir un fichier Résultat

  ODS NomFormat File = « chemin\FichierResultat.NomFormat » ;

- Ferme le fichier résultat

  ODS NomFormat close ;

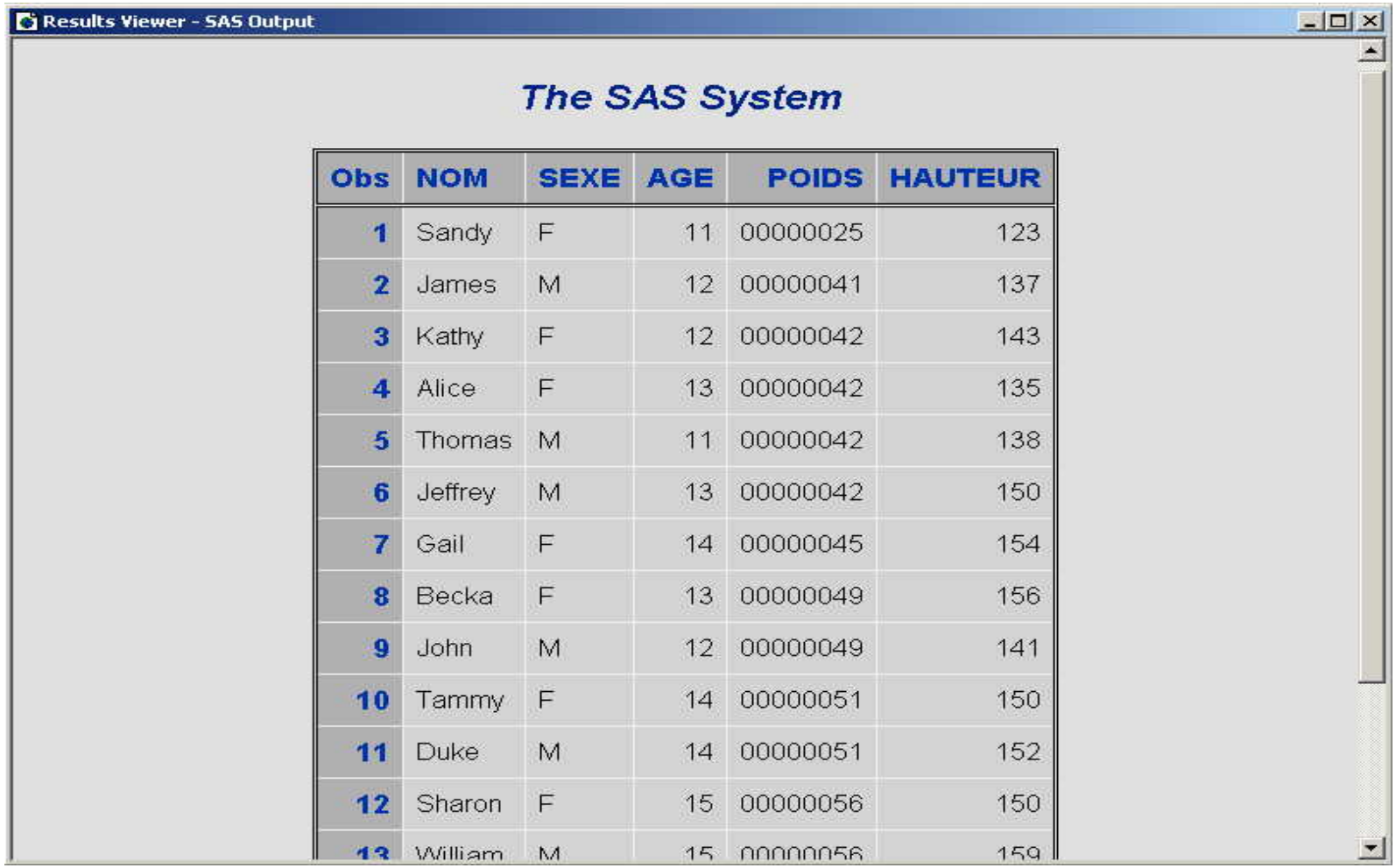
Introduction à l'ODS

- **Exemple d'édition simple**

```
ods html file="c:\temp\Print.html" ;

proc print ;
   var nom sexe age poids hauteur ;
run ;

ods html close ;
```

Introduction à l'ODS

- **Résultat d'édition simple**



Obs	NOM	SEXE	AGE	POIDS	HAUTEUR
1	Sandy	F	11	00000025	123
2	James	M	12	00000041	137
3	Kathy	F	12	00000042	143
4	Alice	F	13	00000042	135
5	Thomas	M	11	00000042	138
6	Jeffrey	M	13	00000042	150
7	Gail	F	14	00000045	154
8	Becka	F	13	00000049	156
9	John	M	12	00000049	141
10	Tammy	F	14	00000051	150
11	Duke	M	14	00000051	152
12	Sharon	F	15	00000056	150
13	William	M	15	00000056	159

Introduction à l'ODS

Présentation

- Il est possible de ventiler la présentation sur plusieurs fenêtres à l'aide d'options de l'instruction ODS HTML.

ODS HTML FILE=''	Sommaire
ODS HTML CONTENTS=''	Contenu
ODS HTML FRAME=''	Fenêtre d'appel

Introduction à l'ODS

- **Exemple d'édition avec plusieurs fenêtres**

```
ods html file="c:\temp\Printm.html"
         frame="c:\temp\Printf.html"
         contents="c:\temp\Printc.html"
          ;
title1 "Vente par année" ;

proc print noobs;
   var quarter month product actual predict ;
   by year ;
   pageby year ;
run ;

ods html close ;
```

Introduction à l'ODS

- **Résultat d'édition avec plusieurs fenêtres**

Sommaire

Méthodologie

- **Quelques optimisations**
- **L'environnement personnel**
- **Quelques principes de base**

Méthodologie – Quelques optimisations

- **La taille des variables**

  - Il faut penser à ajuster leur longueur pour économiser l'espace disque et améliorer les performances.

  - La gestion de la taille dépend de l'évolutivité du contenu des variables (nom de ville, adresse, nom de personne).

Méthodologie – Quelques optimisations

- **La taille des variables**
  - Pour une variable caractère, chaque lettre est enregistrée sur 1 octet.
  - La longueur devra toujours au moins correspondre à la taille de l'observation la plus grande.

Méthodologie – Quelques optimisations

La taille des variables

- Le stockage d'un chiffre ne correspond pas à un octet pour une variable numérique. La longueur de la variable correspond au nombre d'octets qu'elle occupe.

- Pour une variable numérique, nombre pur ou date, la longueur va dépendre du contenu.

  - Un petit nombre entier pourra être codé sur 3 octets tandis que les plus grands le seront sur 8.

  - Les variables contenant une date seront de longueur 4. Cependant, il faut faire attention à ne pas y enregistrer des données de type datetime.

- En cas d'optimisation, il faut utiliser la procédure COMPARE pour vérifier que certaines données n'ont pas été tronquées.

Méthodologie – Quelques optimisations

La taille des variables

- La modification ou l'assignation de la taille des variables se fait grâce aux instructions :
  - Length
  - Attrib
- Elles sont à positionner avant la lecture d'une table SAS.
- Exemple

```
data WORK.EXEMPLE ;
 attrib nom length=$24 label="Nom des sociétaires" ;
 length age 3 ;

 set DONNEES.PERSONNE ;

 ...
run ;
```

Méthodologie – Quelques optimisations

La modification des caractéristiques d'une variable

- **Problématique** : un développeur souhaite modifier la longueur et le libellé d'une variable d'une table contenant 2 millions d'observations.

- **A éviter**

```
data DONNEES.Personne / debug ;
 length nom $20 ;
 label  nom="Nom du sociétaire" ;

 set DONNEES.Personne ;
run ;
```

- **Solution** la procédure DATASETS

  Avantage modifier la structure de la table sans parcourir son contenu

Méthodologie – Quelques optimisations

- **La gestion des variables**

  - **Problématique** : diminuer les entrées/sorties et l'espace disque utilisé lors d'un traitement

  - **Solution :** ne garder que les variables nécessaires lors des traitements et dans les nouvelles tables

  - **Exemple**

```
data DONNEES.Personne2 (keep = nom prenom telephone) ;

  set DONNEES.Personne (drop = codePostal ville) ;
  …
run ;
```

Méthodologie – Quelques optimisations

La gestion des tests conditionnels

- **Problématique** : diminuer le temps de traitement lors des tests conditionnels.

- **A Eviter**

```
data DONNEES.Personne2 ;
  set DONNEES.Personne (drop = codePostal ville) ;
  if Nom = “Dupond” then … ;
  if Nom = “Dupont” then … ;
run ;
```

- **A Faire**

```
data DONNEES.Personne2 ;
  set DONNEES.Personne (drop = codePostal ville) ;
  if Nom = “Dupond” then … ;
  else if Nom = “Dupont” then … ;
run ;
```

Méthodologie – Quelques optimisations

La gestion des étapes DATA

- **Problématique** :

  Eviter les entrées/sorties inutiles lors des étapes DATA.

  Toutes les étapes DATA ne nécessitent pas la création d'une ou plusieurs tables.

  Utilisation de l'instruction _NULL_

- **A Eviter**

```
data DONNEES.Test ;
  set DONNEES.PERSONNE ;
  if Nom = "Dupond" then call symput('Prenom',Prenom) ;
run ;
```

- **A Faire**

```
data _null_ ;
  set DONNEES.PERSONNE ;
  if Nom = "Dupond" then call symput('Prenom',Prenom) ;
run ;
```

Méthodologie – Quelques optimisations

- **Le suivi des ressources**

  - **Générer un fichier de trace**

    Il contient l'ensemble des informations écrites dans la fenêtre LOG
    Utilisation du paramètre ALTLOG

    - Syntaxe –altlog « chemin\FichierTrace.log »
    - Positionnement ligne de commande de l'icône d'appel de SAS ou fichier SASV8.CFG

    Le fichier de trace d'une session écrase celui de la session précédente

    Par défaut, le fichier de trace n'est pas consultable en cours de session. Pour modifier ce comportement il faut positionner l'option $UNBUFLOG

Méthodologie – Quelques optimisations

- **Le suivi des ressources**

  - **Suivi de consommation mémoire et temps**

    Option Fullstimer à positionner en ligne de commande, dans le fichier de configuration ou l'instruction OPTIONS.

    Autres instructions disponibles selon l'environnement (MVS, …)

  - **Indicateurs à suivre**

real time	temps réel à la réalisation, en centième de seconde
user cpu time	temps nécessaire à la réalisation, en centième de seconde
system cpu time	
memory	taille mémoire utilisée, en ko

Méthodologie – Quelques optimisations

- **Le suivi des ressources**
  - **Exemple produit en utilisant l'option fullstimer**

```
235  option fullstimer;
236
237  data _null_ ;
238    format date ddmmyy10. ;
239
240    date = datepart('01AUG2002:9:45'dt) ;
241
242  run ;

NOTE: DATA statement used:
      real time           0.01 seconds
      user cpu time       0.01 seconds
      system cpu time     0.00 seconds
      Memory                           94k
```

Méthodologie – environnement personnel

- **L'organisation d'un environnement SAS Basique se base sur plusieurs répertoires présents sous une racine commune.**

- **Chacun d'entre eux gère des domaines de données bien spécifiques**

- **L'environnement doit contenir les librairies suivantes**
  - La librairie SYSTEME
    - catalogue FORMATS — formats personnels
    - catalogue SASMACR — macros compilées
    - catalogue MACROS — source des macros
    - catalogue SOURCES — autres sources
    - catalogue APPLI — application AF
  - La librairie DONNEES
    - tables de données personnelles
    - vues logiques
  - La librairie PARAM
    - tables de paramétrage des applications

Méthodologie - environnement personnel

- **Dans les fichiers de configuration, utilisation de chemin relatif**

- **Suivant la racine des répertoires, la valeur du LIBNAME peut changer (cet exemple est valable dans un environnement à répertoires).**

```
libname SYSTEME 'SYSTEME' ;
libname DONNEES 'DONNEES' ;

options fmtsearch=(SYSTEME) ;
```

Aucune dépendance vis à vis du chemin réel.

Méthodologie - quelques principes de base

- **Un bon développement doit permettre une maintenance plus rapide des programmes**

- **Ceci implique**
  - La mise en place d'une documentation
  - L'écriture de commentaires au sein des programmes
  - Le respect de certaines règles de développement

Méthodologie - quelques principes de base

Règles d'écriture de codes SAS Base

- Pour les noms de tables et de variables
  - Nom compréhensible ClientNom, ClientPrenom, …

    Pour les tables et variables, aller du général au particulier

    Pour leur casse, première lettre en majuscule, le reste en minuscule

- Faire précéder le point virgule par un blanc

```
proc print data=LIB.TABLE ;
    var A B C ;
run ;
```

Méthodologie - quelques principes de base

- **Règles d'écriture de codes SAS Base**
  - Instructions globales, de début et de fin d'étape en colonne 1
  - Autres instructions indentées
  - Exemple

```
data TEST ;
    do I = 1 to 10 ;
       output ;
    end ;
run ;
```

Méthodologie - quelques principes de base

- **Règles d'écriture de codes SAS Base**
  - Débuter par les instructions d'affectation : LENGTH, ATTRIB, FORMAT, INFORMAT, RETAIN.
  - Poursuivre par le code responsable des I/O.
  - Exemple

```
data TEST ;
   merge LIB.TABLE1
         LIB.TABLE2 ;
   by CLE ;

   ...
run ;
```

Méthodologie - quelques principes de base

- **Au cours du développement, ne pas hésiter à**

  - visualiser ses données à l'aide de la fenêtre FSVIEW (commande FSV) ou de la procédure PRINT (option obs=) .

  - Visualiser les caractéristiques des variables, commande VAR

  - Utiliser le debugguer

```
data Test / debug ;
   … ;
run ;
```

Méthodologie - quelques principes de base

- **Effectuer éventuellement quelques analyses simples afin de mieux connaître :**
  - les variables qualitatives (de classe)
  - les variables quantitatives (d'analyse)

- **Les moyens**
  - Procédure FREQ,
  - Procédure MEANS,
  - …

Formation

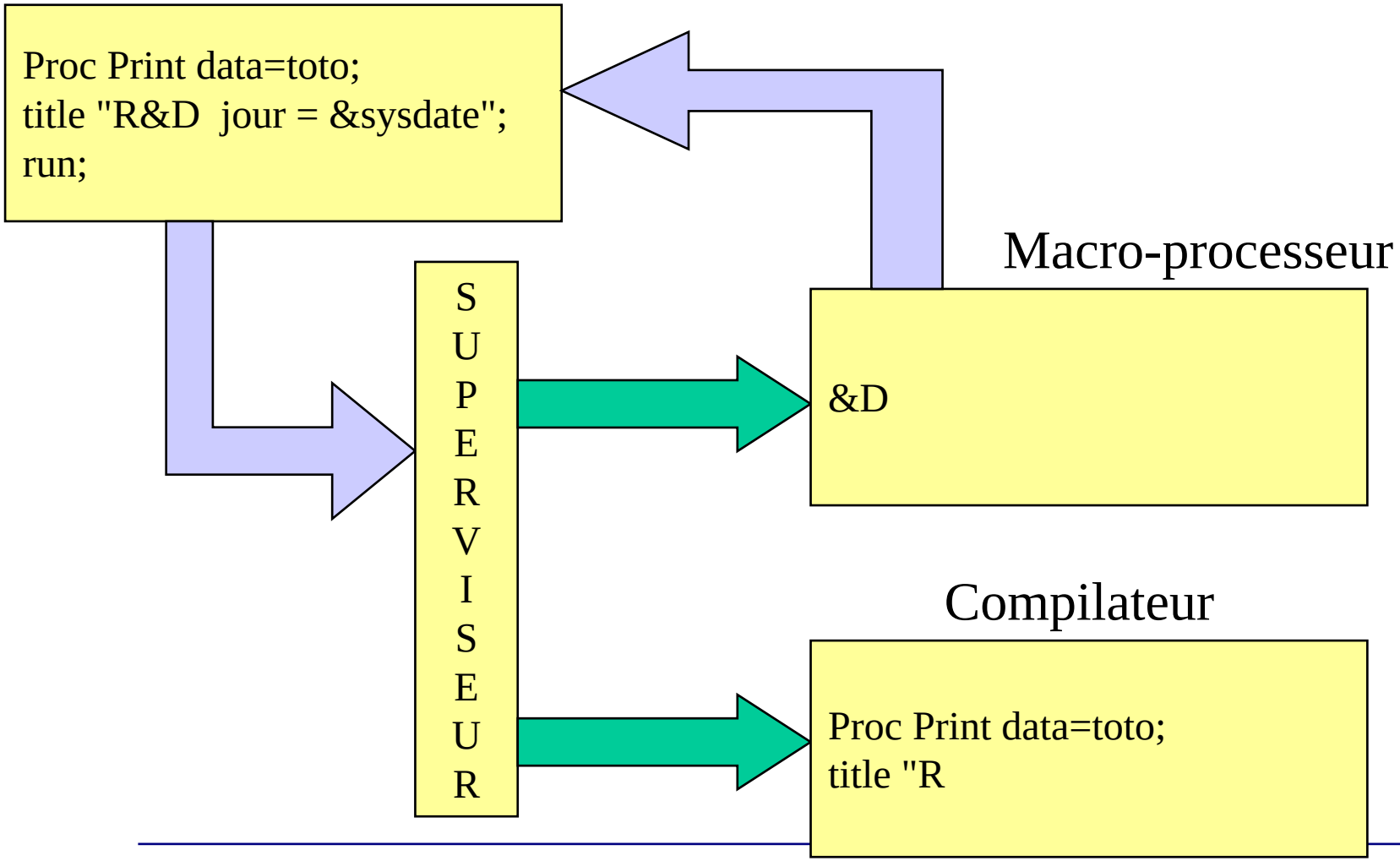
SAS MACRO

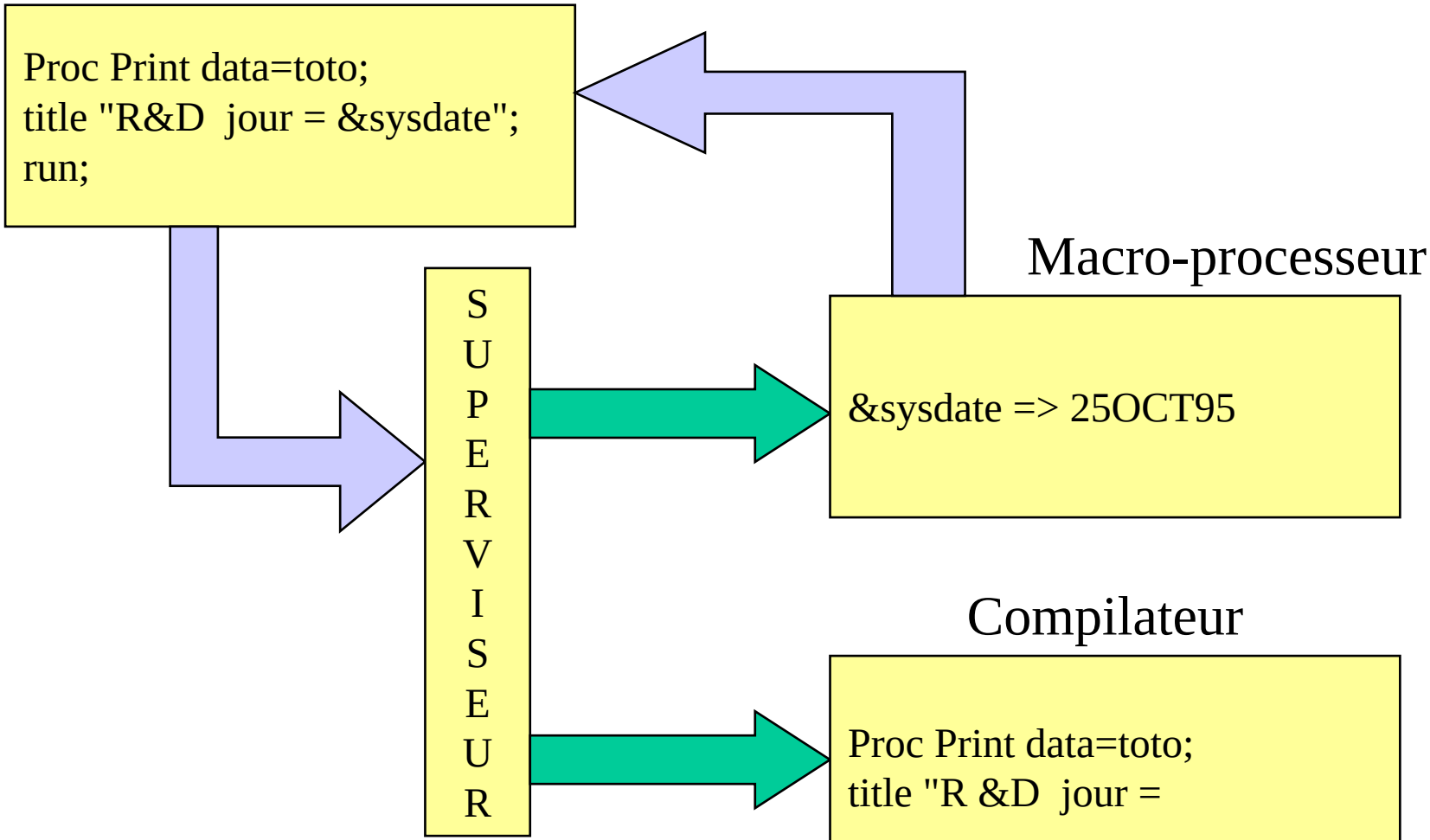
Plan du cours

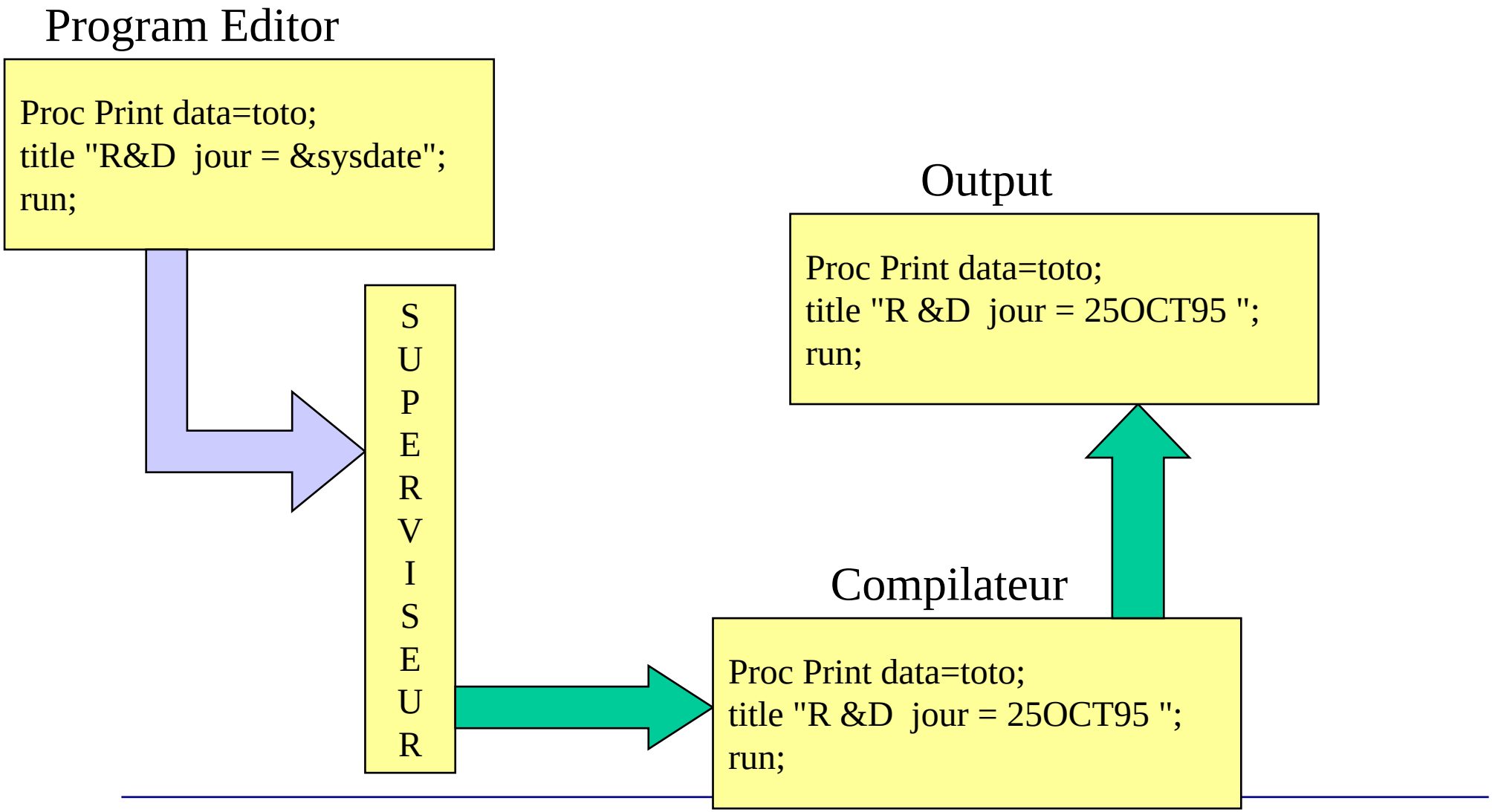
- Définition
- Applications pratique
- Mode de fonctionnement
- Les macros variables.
- Les macros procédure.

- **Moyen de stocker du texte :**
  - Chaînes de caractères, noms de table, noms de variables, instructions SAS….
  - Macros variables
  - Macros procédures
  - % et &

- **Extraire des infos => La date du jour**
- **Générer des instructions répétitives**
- **Communiquer entre étapes SAS**
- **Créer des exécutions conditionnelles**
- **Dialoguer avec l 'utilisateur**
- **Automatiser des traitements**


Proc Print data=toto;
title "R&D jour = &sysdate";
run;
S
U
P
E
R
V
I
S
E
U
R
Macro-processeur
&D
Compilateur
Proc Print data=toto;
title "R


Proc Print data=toto;
title "R&D jour = &sysdate";
run;
S
U
P
E
R
V
I
S
E
U
R
Macro-processeur
&sysdate => 25OCT95
Compilateur
Proc Print data=toto;
title "R &D jour =


Program Editor
Proc Print data=toto;
title "R&D jour = &sysdate";
run;
S
U
P
E
R
V
I
S
E
U
R
Output
Proc Print data=toto;
title "R &D jour = 25OCT95 ";
run;
Compilateur
Proc Print data=toto;
title "R &D jour = 25OCT95 ";
run;

Quelques options pour la mise au point des macros

- **MERROR : Avertissement si Macro non trouvée.**
- **SERROR : Avertissement si MacroVariable non définie.**
- **MLOGIC : L ’exécution de la macro.**
- **MPRINT : Le code généré par la macro.**
- **SYMBOLGEN : Les valeurs des macro-var.**

- **Macros-variables :**
  - Chaînes de caractères
- **Macros-procédures :**
  - Des constantes (caractères, nombres,nom de table SAS, instructions SAS)
  - Des macros variables
  - Des macros-instructions (%do…..%end, %input, %global)
  - Des Macros fonctions (%str, %substr)
  - Des macros-expressions

- **Deux types :**
  - Systèmes : définit au lancement de sas pour toutes la session (global)
  - Utilisateur :
- Globales ou locales à une procédure
  - 8 caractères au plus (commencent par une lettre ou _)
  - référencées par & placé devant la macro-variable
  - chaînes de caractères (0 à 1024)

- **Création :**
  - %let nomvar = valeur; pas d 'apostrophes
  - %Global ……;

- **Utilisation :**
  - &nomvar

  - title " fgsfgdsfgdf sfq gfg &nomvar ";

  - Le ; est compris comme un délimiteur

Macro variables

- **Création d'une macro variable**
  - %LET
    - Syntaxe : `%let nom_macro_variable = valeur ;`
  - CALL SYMPUT
    - Syntaxe : `CALL SYMPUT(« nom_macro_variable »,nom_var_SAS) ;`
- **Appel** `&nom_macro_variable`
- **Récupération dans une étape DATA :**

```
nom_var_SAS = SYMGET('nom_macro_variable') ;
```

MacroQuoting et fonctions %str( );

- **Rappel : une chaîne de caractère doit être encadrée de guillemets (") ou d'apostrophe(').**

  "c'est une chaîne de caractère"

  'c''est une chaîne de caractère'

- **L 'analyseur syntaxique traite les chaînes avec des doubles quotes mais pas celles avec les simples quotes qui ne peuvent pas être interprétées.**
- **La fonctions %STR();**
  - Permet d 'encapsuler les caractères spéciaux comme :

  ; + - * / , < > = blank LT EQ GT AND OR NOT LE GE NE
- **Les fonctions %NRSTR(), %NRSTR, %QUOTE, %NRQUOTE, %BQUOTE, %NRBQUOTE, and %SUPERQ.**

- **%put texte + &nomvar;**
- **Data _Null_;**

  Put texte + &nomvar;

  Run;

- **texte&nomvar => texte+contenu(nomvar)**
  - %let nom = 3;
  - keep var&nom; => keep var3;

- **&nomvartexte => pas résolu**
  - &nomvar.texte => le point indique la fin de la macro-variable
  - var &nomvar.1 - &nomvar.3 ;=> var age1-age3;

- **&nomvar1&nomvar2 =>**

- **Libname et nom de table**
  - &lib.&tab => libreftable
  - &lib..&tab => libref.table

Afficher des valeurs de macros-variables

- **%put texte + &nomvar;**

Résoudre des && successifs

- %let jour1 = tutu;
- %let jour2 = toto;
- %let jour3 = titi;
- %let n=3

- &&jour&n => &jour3 => titi

Macro variables

Leur contenu

- Il est possible de lister le contenu de toutes les macro-variables

  - Interne %put _automatic_ ;
  - Global %put _global_ ;
  - Local %put _local_ ;
  - Toute %put _all_ ;

- **Quelques remarques**
  - Le symbole ‘&’ n’est jamais utilisé lors de la création d’une macro variable, sinon il y a substitution.
  - Une macro variable n’a qu’une valeur.
  - Une macro variable est modifiable à tout moment par les macro instructions appropriées.
  - La valeur d’une macro variable est une chaîne de caractères, cette chaîne pouvant contenir n’importe quoi (lettres, chiffres, blancs, caractères spéciaux, …).
  - Une macro variable n’est pas une variable SAS (elle ne fait pas partie d’une table SAS).

%macro nomMac;

code macros ;
Etape Data;
Etape Proc;

%mend nomMac;

```
%IF expression %Then %Do;
        Instruction;
%End;
%else %if expression %Then %Do;
        Instruction;
%End;
%Else %do;
        Instruction;
%End;

            macros-expressions ;

%mend nomMac;
```

- **%DO** *macro-variable=expression* **%TO** *expression* **<%BY** *expression***>;**

  instructions;

  %END;

- **%DO %WHILE (expression);**

  instructions;

  %END;

- **%DO %UNTIL(***expression***);**

  instructions;

  %END;

- **Appel d'un macro programme via la fonction RESOLVE**
  - Le langage SAS de base propose la fonction RESOLVE( ) qui permet de faire appel à du code macro qui ne sera résolu que lors de l'exécution du programme appelant.
  - Syntaxe

```
Var = RESOLVE('%nom-macro[(Paramètres)]');
```

- **permet de copier la valeur d ’une variable dans une macro-variable**

  - Call symput (‘ Macro-var ’, variable);

    - variable=symget (‘ macro-var ’);

- **permet d ’executer des macros-procédures à partir d ’une étape data.**

  - Call execute (‘&NomMacro’);

- **Définition : « into :NomMacro**
  separed by ‘ ’

- **Utilisation : «&NomMacro » ou symget(‘ NomMacro ’)**

%scan => retourne un mot dans une chaîne de carac

%str => ignore les caratères spéciaux

%Substr => extraire une chaîne de caractère

%Upcase => Passage en majuscule

%index => comparer des chaînes de caractères

%eval => évaluer des expressions arithmétiques

%length =>longueur d 'une macrovariable

sysscp => système d 'exploitation

sysver => version de sas

syserr => erreur d 'execution

syslast => dernière table traitée

sysdate => date système

sysday => jour de la semaine

sysenv => mode d 'execution de la session

%put _automatic_;

%input NomMacro => récupère le texte entré au clavier

```
%put Entrez votre nom:;
%input nom;
```

%window => écran de saisie

- **Définition suivi de l 'invocation dans le même programme.**
- **Macro définie dans un fichier incluse par l 'instruction**

```
%INCLUDE fileref /SOURCE2;
```

- **Macro définie dans un catalogue incluse par l 'instruction**

```
filename fileref
catalog  'libref.catname';  %INCLUDE fileref(entree-
catalogue) /SOURCE2;
```

- **Macro dans une bibliothèque autocall par les options `mrecall mautosource sasautos`.Un catalogue peut faire office de bibliothèque.**
- **Macro compilée stoquée dans un catalogue sasmacr.**

```
%MACRO macro () /STORE DES='Descript';
 OPTIONS MSTORED SASMSTORE=lib;
```

Sauvegarde des macro programmes

- Il existe deux possibilités
  - dans un fichier externe : enregistrer le macro programme dans un fichier ayant le même nom, ou pas, que la macro.
  - Dans une entrée SAS de type source portant le même nom que la macro.

Stockage : %INCLUDE

- %INCLUDE insert le texte d'un macro programme.
  - Syntaxe

    %include « MonChemin\Mamacro.sas » ;

- Pour exécuter la macro présente dans ce fichier, il suffit alors de l'appeler.

Stockage : Autocall Facility

- Permet d'accéder directement à des macro programmes stockés dans un ou plusieurs répertoires ou une ou plusieurs bibliothèques.

- Une seule macro par programme sauvegardée.

- Au premier appel, le macro programme est compilé puis exécuté; aux appels suivants, il est exécuté sans être recompilé.

Stockage : Autocall Facility

- Mise en œuvre pour les répertoires
  - Sauver le macro programme dans le répertoire. Le nom du fichier devant être identique au nom du macro programme.
  - Utiliser l'option SASAUTOS= pour identifier la librairie dans laquelle SAS trouvera les membres contenant les macro programmes et spécifier l'option MAUTOSOURCE.
  - Exemple

```
options sasautos='c:\Temp' mautosource ;

%etiquette ;
```

Stockage : Autocall Facility

- Mise en œuvre pour les librairies
  - Sauver le macro programme dans la bibliothèque. Le nom de l'entrée devant être identique au nom du macro programme.
  - Utiliser l'option SASAUTOS= pour identifier la librairie dans laquelle SAS trouvera les membres contenant les macro programmes.

    Spécifier l'option MAUTOSOURCE.
  - Exemple

```
libname malib "c:\temp" ;
filename moncat CATALOG "malib.moncatal" ;
options sasautos = moncat mautosource ;

%mamacro ;
```

- **Stockage : Stored Compiled Facility**
  - Offre la possibilité de stocker dans une librairie SAS des macro programmes précompilés et directement utilisables par simple appel.
  - Mise en œuvre :
    - Sauver la version compilée du macro programme dans un membre de bibliothèque en précisant l'option / STORE.
    - Utiliser l'option SASMSTORE= pour identifier la librairie dans laquelle SAS trouvera le catalogue SASMACR contenant les macro programmes compilés.
    - Spécifier l'option MSTORED pour indiquer que l'option STORE va être utilisée pour stocker et compiler le macro programme.

Stockage : Stored Compiled Facility

- Exemple

```
LIBNAME malib "c:\temp" ;
OPTIONS MSTORED SASMSTORE = malib ;

%MACRO mamacro / STORE des = "Libellé" ;

%MEND mamacro ;

%mamacro ;
```

Stockage – Ordre d'appel

- Lors de l'appel des macro programmes par les utilisateurs afin de les exécuter, SAS effectue sa recherche dans un ordre bien précis

1. Recherche parmi les macro programmes compilés dans la session
2. Recherche dans le catalogue SASMACR si MSTORED activée
3. Recherche dans la librairie AUTOCALL si MAUTOSOURCE activée

- Une fois trouvé, le macro programme est compilé et exécuté

  Il est enregistré dans le catalogue WORK.SASMACR pour une utilisation directe ultérieure (sans re-compilation).

Formation

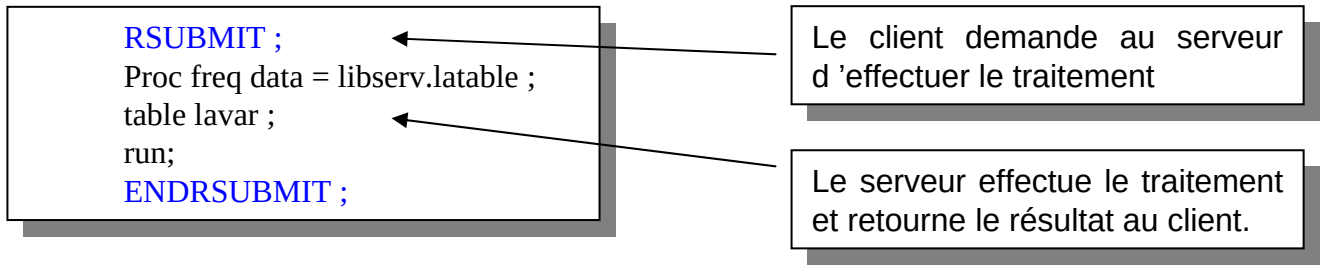
SAS CONNECT

Présentation du Client / Serveur

Coté client il est tout aussi facile de faire travailler le serveur que le poste local. Toute la partie qui consiste à empreinter le réseau, se connecter au serveur, faire exécuter le traitement et enfin récupérer le résultat, est quasi transparente pour l 'utilisateur.

Il vous suffit juste d 'ajouter RSUBMIT au début de votre code puis ENDRSUBMIT à la fin.

Par exemple pour faire exécuter une PROC FREQ au serveur NT :



nb : La librairie libserv doit bien sûr être connue du serveur !!! Et votre administrateur doit avoir fait le nécessaire sur votre poste pour que vous puissiez vous connecter au serveur !!!

Un exemple type d ’architecture Client/ serveur

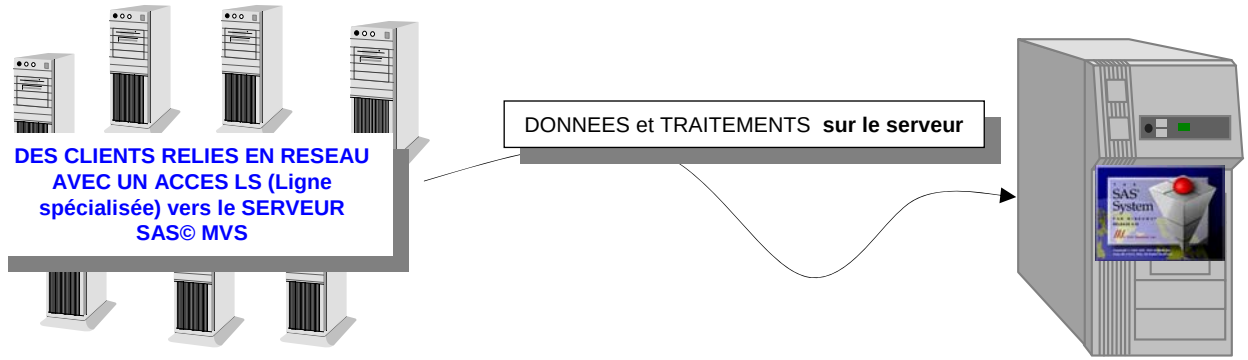
Travailler en mode C/S permet à un client dont la puissance de calcul est limitée de demander à un SERVEUR, beaucoup plus puissant, d ’effectuer un traitement à sa place.

A noter que cette puissance n’est pas seulement relative à la puissance de calcul brute (le CPU), mais aussi à la capacité de stockage (espace disque).

A la lumière de ceci , on comprend donc très vite pourquoi il peut être utile de s ’appuyer sur un SERVEUR.

Les clients comme les serveurs peuvent être de nature très différente.

Par exemple, il est courant de trouver sur les clients le système d ’exploitation Windows et sur les serveurs des systèmes plus robustes et capables de gérer plus de ressources comme Windows NT, Linux ou encore MVS (Mainframe).



Connexion : Définition de la méthode de communication

La définition de la méthode de communication permet de déclarer le protocole que vous souhaitez utiliser pour converser avec le serveur.

Dans l 'exemple suivant, nous utilisons le protocole TCP/IP (protocole utiliser sur Internet et protocole de base des prochaines versions de Windows NT)

option comamid = TCP ;

Connexion : définition du serveur

Une fois le protocole définit il faut maintenant indiquer l 'identité du serveur qui va effectuer les traitements.

Plus exactement, nous allons indiquer **l 'adresse ip** de la machine ainsi que le **port de communication** auquel doit s 'adresser la session SAS© du client !!!!

En effet, il y a sur le serveur un « spawner » (petit programme tournant en continu) qui intercepte sur un port prédéfini les demandes des clients SAS© et qui démarre, sur le serveur, des sessions SAS© à la demande.

Dans l 'exemple suivant, nous souhaitons nous appuyer sur un serveur dont l 'adresse (IP) est 127.0.0.1, nous indiquons ensuite que sur ce serveur nous allons nous adresser au port 2323. Cela implique que le service « écoute » sur ce port.

```
%LET serveur = 127.0.0.1 2323
option remote = serveur; /*notez l 'utilisation d 'une macro sans le &*/
```

Connexion : Définition du script de connexion

Comme lorsque vous vous connectez à internet, vous allez devoir utiliser un script de connexion pour vous connecter au serveur.

Il existe des scripts de communication pour chaque plate-forme. Vous pouvez les trouvez dans :

le_disque:\sas\connect\saslink\ *.scr

Nous allons utiliser le script adapté à windows :

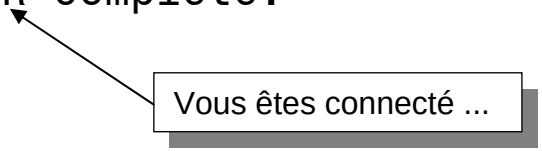
filename rlink ‘ c:\sas\connect\saslink\tcpwin.scr ’ ;

Connexion !

Maintenant que toutes les options sont définies, il nous reste à nous connecter : **signon ;**

La log …

```
5    %let serveur = 127.0.0.1 2323 ;
6    filename rlink "C:\AppDevStudio\SAS\connect\saslink\tcpwin.scr" ;
7    option comamid = tcp remote=serveur;
8    signon ;
NOTE: Remote signon to SERVEUR commencing (SAS Release 6.12.0055P090998).
NOTE: Script file 'tcpwin.scr' entered.
NOTE: Logged onto Windows... Starting remote SAS now.
NOTE: SAS/CONNECT conversation established.
...
NOTE: Remote signon to SERVEUR complete.
```



Déconnexion

signoff ;

La log …

```
9    signoff ;
NOTE: Remote signoff from SERVEUR commencing.
NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: Script file 'tcpwin.scr' entered.
NOTE: SAS/CONNECT conversation terminated.
NOTE: Remote signoff from SERVEUR complete.
```

Faire exécuter des traitements au serveur : REMOTE COMPUTE SERVICES

si vous voulez effectuer un traitement sur le serveur il faut qu'une librairie pointant vers les données soit allouée.

```
rsubmit ;
    /*allocation de la librairie RDATA */
    /* SUR LE SERVEUR                   */
        libname RDATA 'c:\inter\sasbase\cas6\bdonnees' ;
        proc means data = RDATA.TCA1999;
        var NBR: ;
        class SOC PROD ;
        run;
endrsubmit ;
```

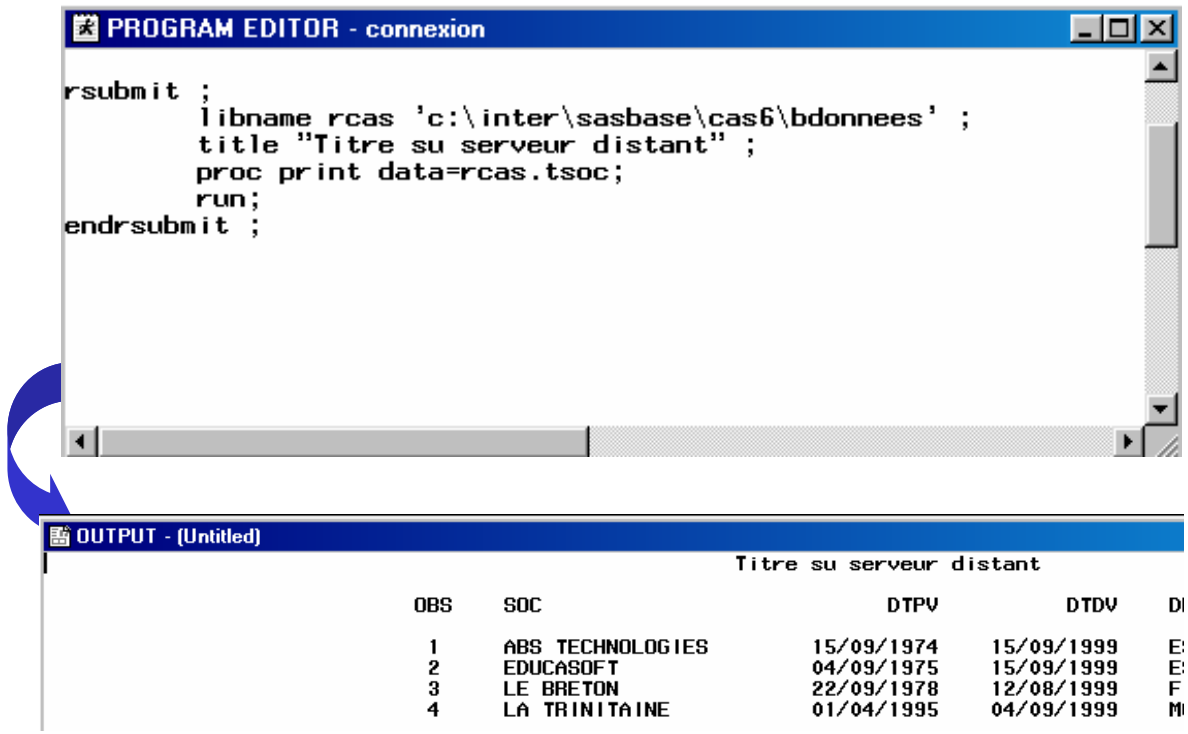
Inutile donc de soumettre ensuite le programme suivant sur le client (sans rsubmit) car la librairie est allouée UNIQUEMENT sur le serveur :

```
    proc means data = RDATA.TCA1999;
        var NBR: ;
        class SOC PROD ;
        run;
```

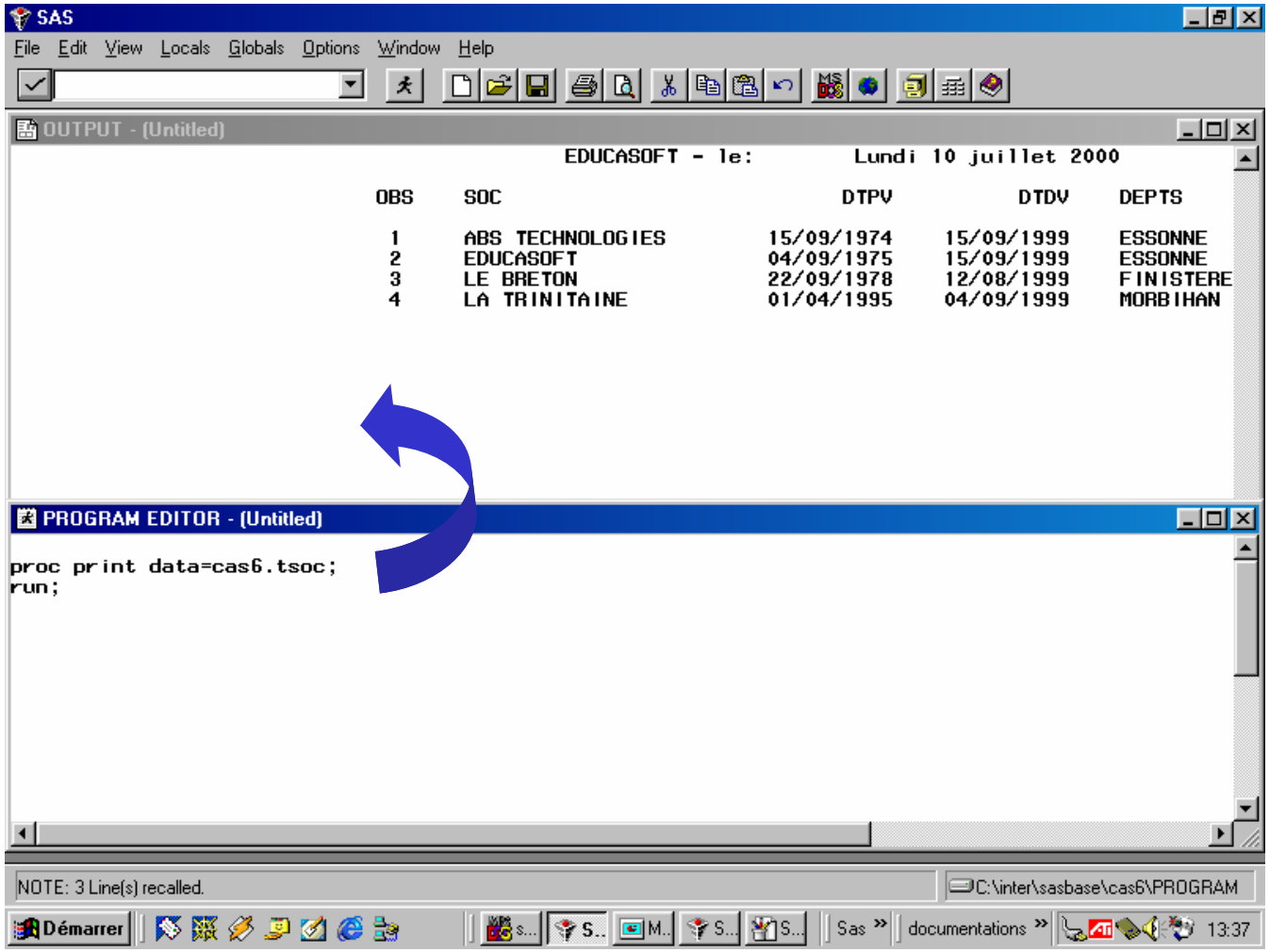
Exemple

Ce qu'il faut bien comprendre lorsque vous travaillez en mode client serveur c 'est que vous travaillez « comme si vous étiez sur le serveur ».

Par exemple dans l 'exemple suivant, nous allons définir un titre pour la session distante …

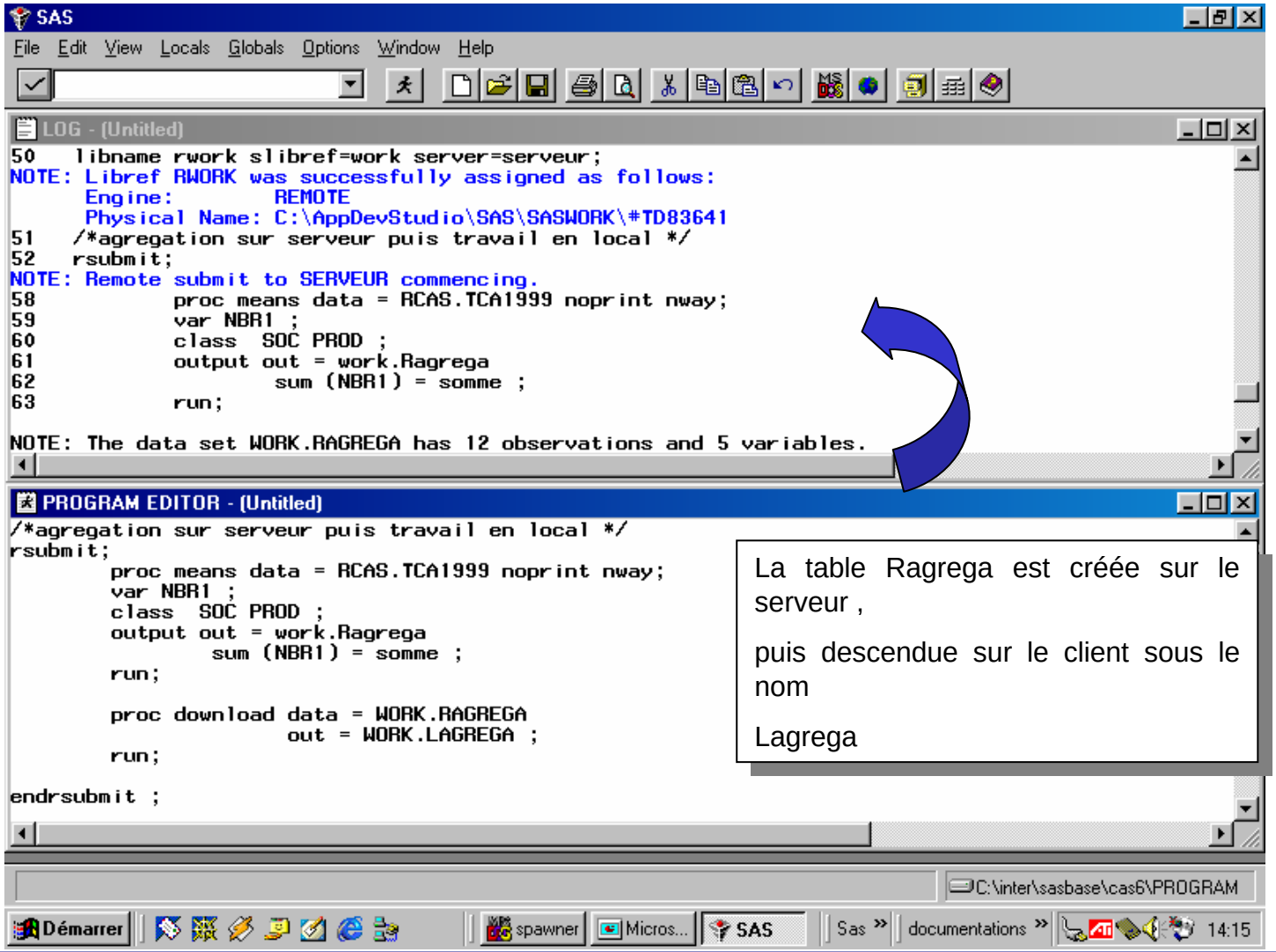


qui n 'est bien sûr pas actif pour le client SAS© …


SAS
File Edit View Locals Globals Options Window Help
OUTPUT - (Untitled)
EDUCASOFT - le: Lundi 10 juillet 2000
OBS SOC DTPV DTDV DEPTS
1 ABS TECHNOLOGIES 15/09/1974 15/09/1999 ESSONNE
2 EDUCASOFT 04/09/1975 15/09/1999 ESSONNE
3 LE BRETON 22/09/1978 12/08/1999 FINISTERE
4 LA TRINITAINE 01/04/1995 04/09/1999 MORBIHAN
PROGRAM EDITOR - (Untitled)
proc print data=cas6.tsoc;
run;
NOTE: 3 Line(s) recalled.
C:\inter\sasbase\cas6\PROGRAM
Démarrer
Sas
documentations
13:37

PROC DOWNLOAD : DATA TRANSFERT SERVICES

- **Pour descendre des données sur le client**



La table Ragrega est créée sur le serveur ,

puis descendue sur le client sous le nom

Lagrega

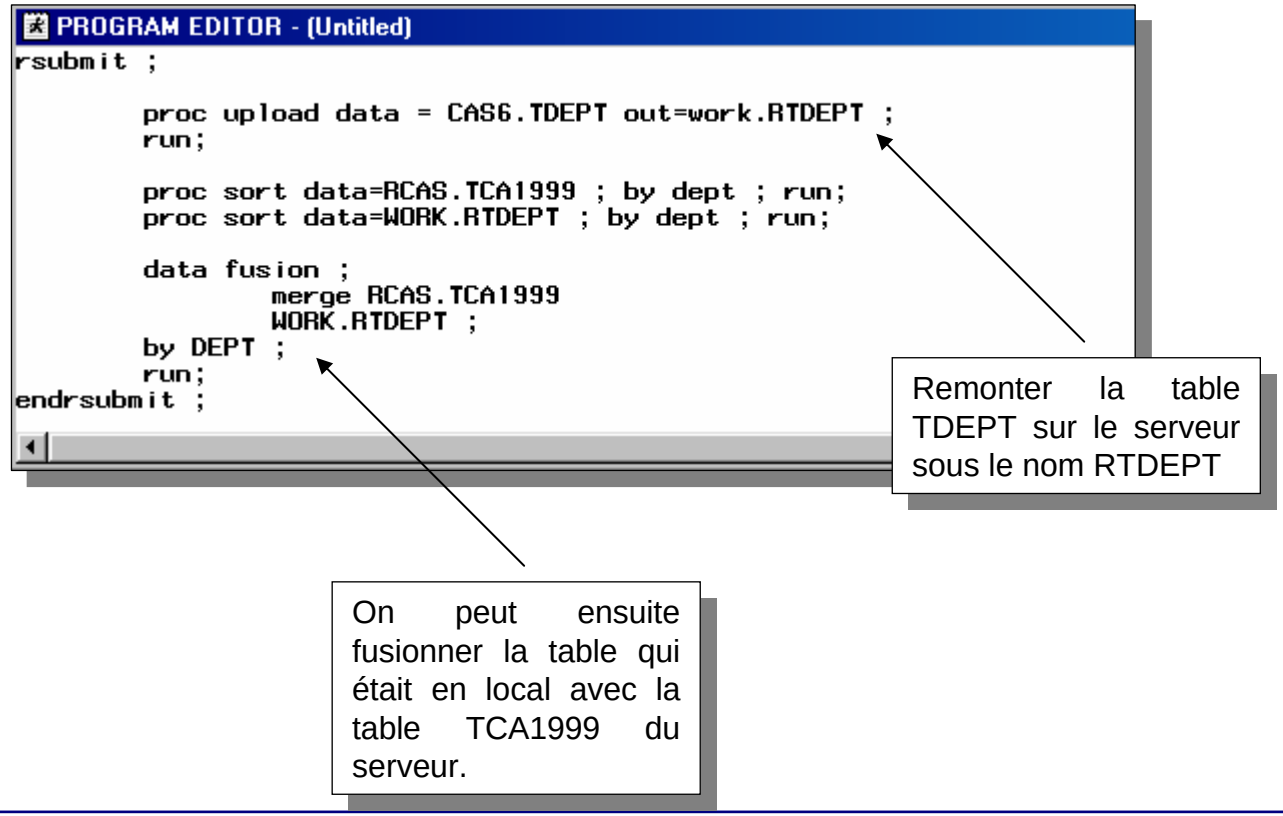
PROC UPLOAD : DATA TRANSFERT SERVICES

- **Pour remonter des données sur le serveur**

  Il arrive très fréquemment que vos données soient sur le serveur SAUF une table qui se trouve sur le client.

  Comme il est possible de descendre une table du serveur vers le client , nous allons remonter cette table sur le serveur afin de la fusionner avec d 'autres tables.

  Pour effectuer une telle opération, nous allons utiliser la procédure **UPLOAD.**



Remote Library Service

- **Manipuler une bibliothèque distante de manière transparente.**

libname rhost '~/sas/connect/rhost/epicerie' server=ulysse;

Remote SQL PASS-THROUGH

- **Pour réduire les transferts de données**

```
proc sql;
connect to remote as rmt(server=ulysse);
* pas tres bon : un trafic important ...  ;
select libname, path from connection to rmt(select * from SASHELP.VSLIB) where
     libname='rhost' or libname ='RHOST' ;
 * mieux ;
select * from connection to rmt(select libname, path from SASHELP.VSLIB where
     libname='rhost' or libname='RHOST');
quit;
```

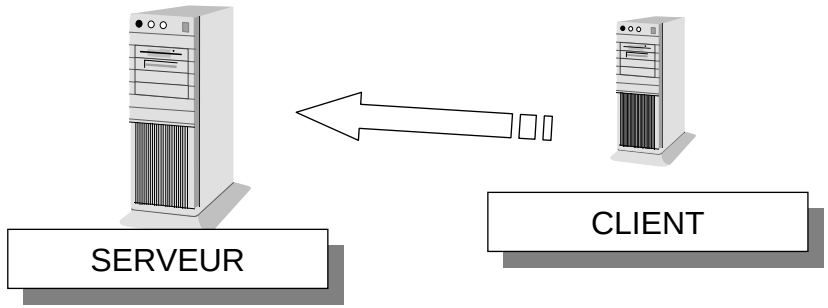
```
proc sql;
     connect to remote (server=ulysse dbms=oracle DBMSARG=(user='ops$duboism'
      password=xxxx path='BASE'));
     execute (delete from mafourniture where qte is null or qte=0) by remote;
quit;
```

Remonter une macro variable du client vers le serveur

Comme les librairies ou les catalogues de formats, les macros variables sont locales au client ou au serveur.

C ’est à dire que si vous créez une macro variable sur le client vous devez la remonter sur le serveur pour pouvoir l ’utiliser.

Pour effectuer cette opération il faut utiliser la macro %LPUT (qui remplace %SYSLPUT). Mais au préalable vous devez compilez le code de cette macro car elle n ’est pas présente en standard.



CODE LPUT%

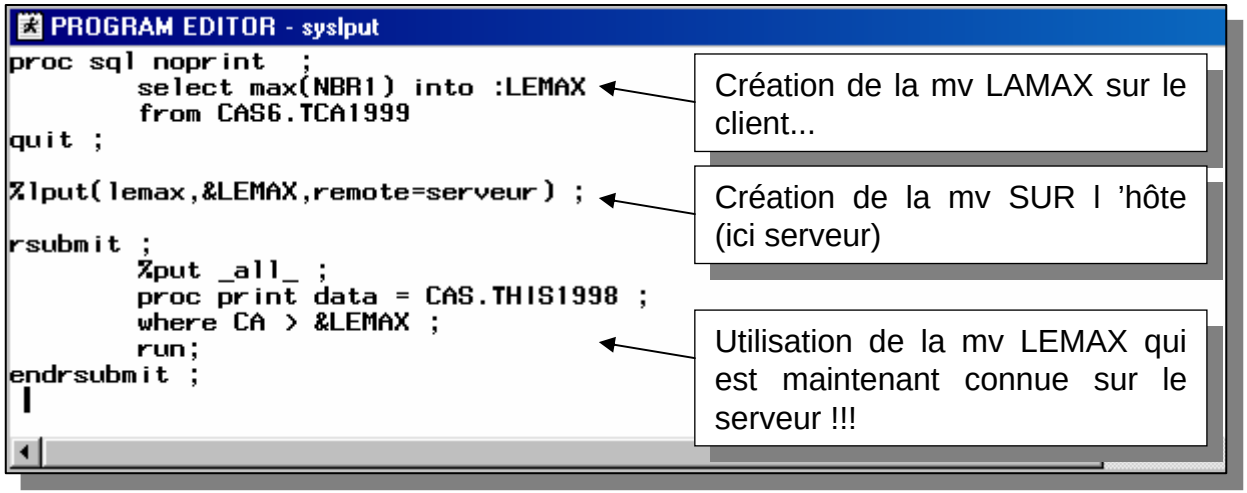
Voici donc le code de %LPUT :

PROGRAM EDITOR - syslput

```
%macro lput(macvar,macval,remote=);
   options nosource nonotes;
   %let str=%str(rsubmit &remote;options nosource;)
    %nrstr(%let) %str(&macvar = &macval;options source;endrsubmit;);
   &str; options notes source;
%mend lput;
```

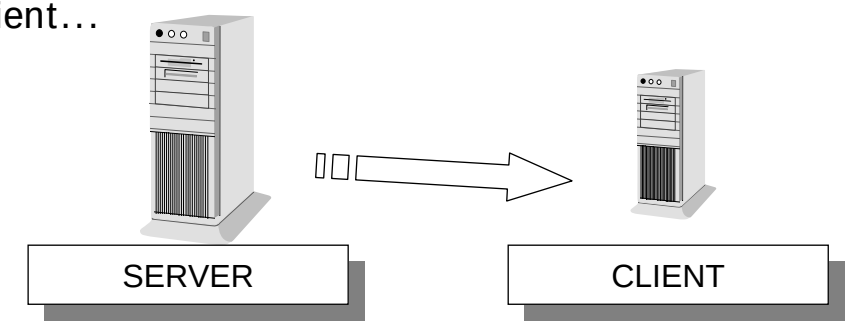
Une fois ce code tapez, soumettez le … il est compilé dans la work du client par défaut.

Nous allons donc maintenant créer une macro variable (lemax) sur le serveur qui contient la valeur de la macro variable LEMAX (mv locale).



Descendre une macro variable du serveur vers le client.

La chose est beaucoup plus simple dans cette situation. La macro %SYSRPUT est disponible par défaut dans le système SAS©. Il suffit de l 'utiliser dans un bloc RSUBMIT. Dans l 'exemple suivant nous allons récupérer la valeur de la macro variable du serveur SYSJOBID pour la transférer sur le client…



PROGRAM EDITOR - sysrput

```
rsubmit ;
%SYSRPUT lamacr = &SYSJOBID ;
endrsubmit ;


%put la valeur de SYSJOBID du serveur :  &lamacr ;
%put la valeur de SYSJOBID du client :  &sysjobid ;
```

LOG - (Untitled)

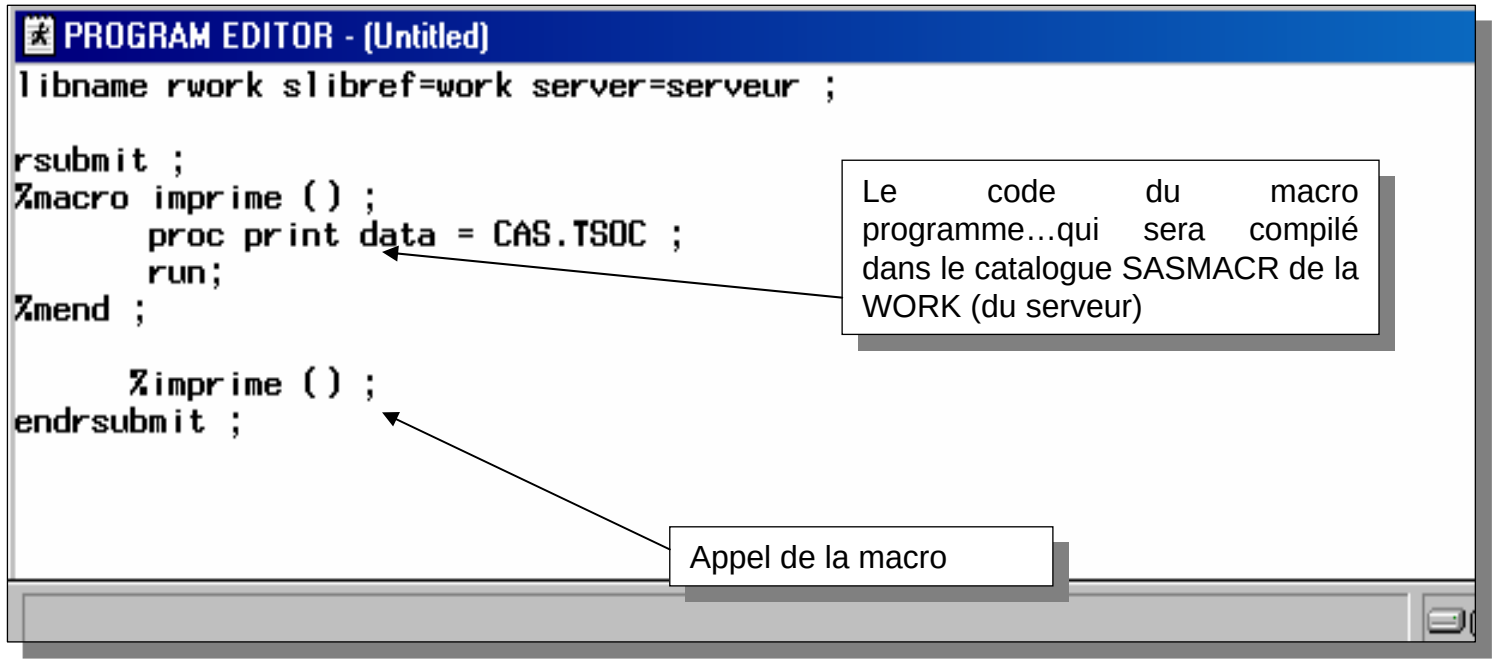
```
69   rsubmit ;
NOTE: Remote submit to SERVEUR commencing.
16   %SYSRPUT lamacr = &SYSJOBID ;
NOTE: Remote submit to SERVEUR complete.
70
71
72   %put la valeur de SYSJOBID du serveur :  &lamacr ;
la valeur de SYSJOBID du serveur :  4293333175
73   %put la valeur de SYSJOBID du client :  &sysjobid ;
la valeur de SYSJOBID du client :  4293210815
```

Travailler avec les macros programmes en mode Client/Serveur

Contrairement aux formats, les macros programmes ont besoin d 'être recompilés sur chaque système avant de pouvoir être utilisés.

Deux techniques s 'offrent à nous :

- Soumettre le code de la macro dans un bloc RSUBMIT pour le compiler,
- remonter les macros sous forme de fichiers textes avec la proc upload et utiliser les options mautosource et sasautos



Connexion à l'IUT

Obligation de rentrer le mot de passe

```
options remote=ulysse comamid=tcp;
filename rlink 'H:\tcpulysse.scr';
signon;
…
Signoff;
```

Sans rentrer le mot de passe mais de manière assez sure :

Une fois pour toute :

```
DATA test/PGM=sasuser.uidpass;
   CALL SYMPUT('USERID','votre login');
   CALL SYMPUT('PASSWORD','votre mot de passe');
RUN;
```

Puis utilisation d'un script de connexion modifié

```
DATA PGM=sasuser.uidpass;
RUN;
options remote=ulysse comamid=tcp;
filename rlink 'H:\autoulysse.scr';
signon;
…
Signoff;
```