JCL
Job Control Language

Introduction

- JCL: langage de traitement batch.
- JCL permet de décrire le travail que le système (MVS Z/OS) doit exécuter.

- 3 types d'instructions (appelées aussi cartes)
  - JOB: identifie le début d'un travail (première carte d'un job)
  - EXEC: définit le début d'une étape du travail (un job peut comprendre jusqu'à 255 étapes), le paramètre essentiel d'une étape est le nom du programme à appeler.
  - DD (Data Definition): définit les data sets à utiliser par le travail.
- Syntaxe d'une instruction:

L'instruction JOB

Syntaxe

//NOM JOB

Autres paramètres optionnels:

- Une information comptable pour savoir pour quel utilisateur facturer la charge du travail
- REGION= : ressources mémoire à allouer au travail.
- NOTIFY= : nom d'utilisateur à notifier une fois le travail terminé.
- USER= : Compte utilisateur avec lequel le travail sera exécuté.
- CLASS=: définit la file d'attente dans laquelle le job pourra être exécuté, une installation peut associer à une classe de nombreux paramètres par défaut (temps d'exécution, etc.) qui s'appliquent à tout travail s'exécutant dans cette classe.
- TYPRUN=: la valeur « SCAN » permet de vérifier la syntaxe JCL sans exécution MSGCLASS=: dirige les sorties du job vers une file d'attente de sortie particulière.
- MSGLEVEL= : contrôle le nombre de message système à recevoir.

- Exemple : //monJob JOB 1,NOTIFY=&SYSUID, REGION=10M.

L'Instruction EXEC

Syntaxe:

//nomEtape EXEC PGM=(nom programme ou procédure JCL)

Exemple

//etape1 EXEC PGM=SORT

Autres paramètres

PARM= : paramètres à passer au programme

COND= : une condition pour contrôler l'exécution du programme en fonction des codes retour des étapes précédentes.

TIME= : Impose une limite de temps d'exécution (en minutes et secondes)

L'instruction DD (Data Definition)

L'instruction DD décrit un fichier utilisé par l'étape, elle possède un nombre important de paramètres

Principaux paramètres

- DSN=: nom du dataset
- DISP
- DCB=: pour préciser les caractéristiques DCB du dataset, sous paramètres:
  - LRECL: nombre d'octets/caractères par enregistrement
  - BLKSIZE: généralement un multipe de LRECL (0 pour laisser le système choisir une taille optimale)
  - DSORG: sequentiel (PS), partitioned (PO)
  - RECFM: (F, FB, V, U, VB)
  - BUFNO.
- UNIT=:définit le type de l'unité de stockage.
- Dummy: fichier ficitif, on peut tester des programmes en indiquant dummy pour les les fichiers qu'on veut pas créer physiquement.
- *: données en entrée pour le programme
- LABEL=: étiquette d'une bande magnétique.

Le paramètre DCB

- DCB=dsnom (utilise les caractéristiques d'un Dataset existant).
- DCB=*. nomDD : caractéristiques d'un DD déjà défini de la même étape.
- DCB=*. nomEtape.nomDD: caractéristiques d'un DD d'un étape précédente.

Le paramètre disp

- DISP= (Disposition): définit l'état du dataset et de quelle manière le système doint en disposer, ce paramètre possède 3 sous paramètres:
  - État du fichier:indique s'il faut le créer (NEW), s'il existe déjà et s'il faut l'allouer de façon exclusive (OLD) ou en permettant le partage avec les autres jobs (SHR).
    Le sous paramètre MOD permet d'ajouter des enregistrement à la fin du fichier, s'il existe , ou il équivaut à NEW dans le cas contraire.
  - La façon dont le système doit disposer du fichier en cas de fin normale de l'étape (pas en cas d'ABEND): DELETE, KEEP, CATLG, UNCATLG, PASS (remet la décision à plus tard en conservant le fichier pour les étapes suivantes) .
  - De même la disposition à prendre en cas d'ABEND: DELETE, KEEP, CATLG, UNCATLG.

  Syntaxe DISP

  DISP=(etat,normal action,abnormal action)

  DISP=(etat,normal action)

  DISP=etat

Actions par défaut:

- Si DISP n'est pas défini alors la valeur par défaut est : DISP=(NEW,DELETE,DELETE)
- Si uniquement le premier paramètre est spécifié, alors la valeur par défaut pour le 2e est le retour au statut du Dataset avant l'exécution du JOB (S'il n'existe pas, il sera supprimé, s'il existe la valeur KEEP sera utilisée par défaut.
- La valeur par défaut du 3e paramètre est celle du 2e paramètre.

Exemples:

- DISP=(NEW,CATLG,DELETE)
- DISP=SHR
- DISP(OLD,DELETE): supprime un dataset existant.
- DISP=MOD

Création d'un nouveau dataset DISP=NEW

Pour créer un nouveau dataset il faut aussi fournir les informations suivantes:

- Nom du dataset (DSNAME) :
  - Nom d'un dataset séquentiel
  - Nom d'un membre (exemple: user1.ds.membre1)

- Type de l'unité
- volser pour un disque ou une bande magnétique étiquetée.
  - Format VOL=SER=xxxxxx,
- Pour un disque:
  - Taille de l'espace de stockage du premier extent.
  - Pour un PDS, il faut aussi préciser la taille du répertoire.
  - Optionnellement les paramètres DCB (le programme qui va écrire les données dans le dataset peut aussi fournir ces paramètres)
- UNIT
  - UNIT= 3390 (pour un disque IBM)
  - UNIT=300 (l'adresse d'un volume)
  - UNIT=SYSDA (nom défini au moment de l'installation, le système choisit l'unité ou les unités de stockage à utiliser).
- SPACE: espace requis sur l'unité de stockage, exemples:
  - SPACE=(TRK,20): 20 pistes sans extent secondaire
  - SPACE=(TRK,(20,5)): 5 pistes pour chaque extent secondaire
  - SPACE=(TRK,(20,5,16)):PDS avec 16 blocs pour le répertoire.
  - SPACE=(500,(1000,500)) : 1000 enregistrements pour l'extent primaire, chaque enregistrement occupe 500 octets, et 500 enregistrements pour chaque extent secondaire .

  L'unité CYL peut être utilisée à la place de TRK.

Paramètres nécessaires pour l’instruction DD

- Dataset existant
  - DSN
  - DISP
- Nouveau DataSet
  - DSN
  - DISP
  - UNIT (type unité)
  - SPACE
  - DCB

Les paramètres SYSOUT et SYSIN

SYSOUT

- //SYSPRINT DD SYSOUT= classe

Classe: une classe de sortie (A..Z,0..9) qui désigne une file d'attente de sortie définie dans le système, si la valeur * est utilisée alors la classe sera celle définie dans le paramètre MSGCLASS de l'instruction JOB

SYSIN

- Désigne l'entrée standard
- Données instream

//SYSIN DD * * indique que les données en entrée commencent à partir de la ligne qui suit l'instruction DD, et se terminent avant /* ou //

Le paramètre DATA

DATA est utilisé Si les données instream contiennent // ou /* dans les deux premières colonnes, par exemple le contenu d'un script JCL

```
 DD DATA, DLM=##
//
//
##
```

Les noms de fichiers symboliques

Dans l'instruction dd:

//ddNom DD dsn=nomDataSet

ddNom est un nom de fichier symbolique qui redirige vers le dataset dont le nom est défini par le paramètre.

Les programmes sous z/os n'utilisent pas les noms réels des datasets mais des noms symboliques.

Un nom symbolique ne doit pas dépasser 8 caractères, et doit être différent des noms réservés suivants:

JOBLIB, STEPLIB, JOBCAT, STEPCAT, SYSABEND, SYSUDUMP, SYSMDUMP, CEEDUMP.

Continuation et concaténation

- Une ligne est limitée à 72 caractères (+ 8 caractères réservés à un numéro de séquence de la carte)

  Pour continuer une instruction sur une nouvelle ligne, la ligne doit se terminer par « , » et la ligne suivante doit commencer par // + au moins un espace

  Exemple:

  L'instruction:

  ```
  //job1 JOB 1,REGION=10M,NOTIFY=user1
  ```

  Est identique à

  ```
  //job1 JOB 1,
  //  REGION=10M,
  //  NOTIFY=user1
  ```

- un même nom symbolique peut être associé à plusieurs instructions dd, ce qui produit la concaténation de plusieurs datasets

  Exemple:

  ```
  //DATAIN DD DISP=OLD,DSN=ibmuser.input1
  // DD DISP=OLD,DSN=ibmuser.input2
  // DD DISP=SHR,DSN=ibmuser.input3
  ```

le programme de l'étape reçoit un seul data set en entrée.

Utilitaires

- SORT :tri et fusion de datasets séquentiels
- IEBCOPY : copie et fusion de datasets partitionnés
- IEBCOMP: comparaison de datasets séquentiels ou partitionnés
- IEBGENER: Copie de datasets séquentiels ou membres d'un dataset partitionné
- IEBEDIT: permet de copier des blocs de JCL
- IEFBR14: programme null
- IDCCAMS: destiné aux datasets VSAM et à la gestion des catalogues

SORT

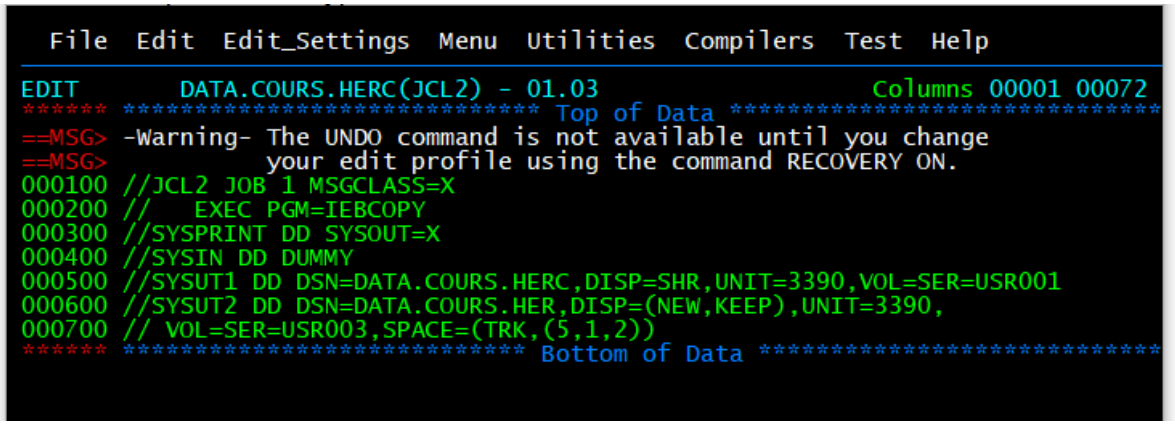
- SORTIN: données à Trier
- SORTINxx: Pour fusionner et trier plusieurs datasets
- SORTOUT: dataset de sortie.
- SYSOUT: sortie des messages.
- SYSIN: Dataset contenant les instructions de contrôles
- Instructions de contrôles
  - SORT FIELDS=([position, nombre, format, ordre]): l'ordre SORT définit les champs qui seront utilisées pour le tri . Informations sur les champs
  - Position: colonne de début
  - Nombre: nombre de colonnes
  - Format: CH pour le mode texte ou BIN pour le mode binaire
  - Ordre: ordre de tri A ou D

IEBCOPY

- Fonctions
  - Copier des datasets
  - Fusionner des PDS
  - Copie de memb res
  - Compresser des PDS
- Instructions DD requises:
  - SYSPRINT
  - SYSIN
  - SYSUT1
  - SYSUT2
  - SYSUT3 (optionnel): utilisé pour stocker les données du répertoire en entrée dans le cas où la mémoire est insuffisante
  - SYSUT4 (optionnel):utilisé pour stocker les données du répertoire en sortie dans le cas où la mémoire est insuffisante

Exemple IEBCOPY: Copie de tous les membres d'un PDS dans un autre PDS.

- Cet exemple copie le contenu du PDS DATA.COURS.HERC dans un nouveau PDS nommé DATA.COURS.HER appartenant à un autre volume USR003



IEBGENER

- Fonctions
  - Créer des copies de datasets séquentiels
  - Créer des PDS .
  - Changer LRECL
  - Ajouter des membres à un PDS
- Instructions DD requises:
  - SYSIN: instructions de contrôle
  - SYSPRINT: messages du programme.
  - SYSUT1: Données en entrée
  - SYSUT2: Données en sortie

SORT: Exemple

```
EDIT       EXEMP.JCL.COURS(SORT1) - 01.02                Columns 00001 00072
Command ===> _                                            Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //SORT1 JOB
000200 //SR1 EXEC PGM=SORT
000300 //SYSOUT DD SYSOUT=*
000400 //SORTIN DD *
000500    MARRAKECH
000600    CASA
000700    RABAT
000800    AGADIR
000900    FES
001000    SAFI
001100    TETOUAN
001200    TANGER
001300 //SORTOUT DD DSN=VILLES.DATA,DISP=(NEW,CATLG),
001400 //    UNIT=3390,
001401 //    SPACE=(TRK,1),
001410 //    DCB=(RECFM=FB,LRECL=20,BLKSIZE=0,DSORG=PS),
001430 //    VOL=SER=USR003
001440 //SYSIN DD *
001450      SORT FIELDS=(1,20,CH,A)
****** ***************************** Bottom of Data ***************************
```

Dataset produit

```
EDIT       VILLES.DATA                                  Columns 00001 00020
Command ===> _                                            Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000001    AGADIR
000002    CASA
000003    FES
000004    MARRAKECH
000005    RABAT
000006    SAFI
000007    TANGER
000008    TETOUAN
****** ***************************** Bottom of Data ***************************
```

Procédures JCL

Structure d'une procédure

- Une procédure commence par une instruction PROC, et se termine par PEND.
- Comme un JOB, une procédure peut contenir des instructions EXEC et DD.
- Le programme appelant peut redéfinir une instruction dans la procédure, syntaxe //nomEtape.NomDD
- Appel d'une procédure
  //etape exec nomProcedure, tridsn=…
  Ou
  //etape exec PROC=nomProcedure, tridsn=…
- Une procédure peut être instream (définie dans le script JCL appelant) ou cataloguée (définie dans un fichier séparé)

Exemple d'une procédure

```
//nomProcedure PROC
//tri EXEC PGM=SORT
//SORTIN DD DISP=SHR,DSN=&TRIDSN
//SORTOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
// PEND
```

Une procédure qui affiche le contenu d'un dataset

Procédure instream

```
VIEW       EXEMP.JCL.COURS(PROC1) - 01.00                Columns 00001 00072
Command ===> 000200_//GEN3_JOB_____________________________ Scroll ===> PAGE
****** ******************************** Top of Data ********************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PROC1 PROC
000200 //ET1  EXEC PGM=IEBGENER
000300 //SYSIN DD DUMMY
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=COPIE.JCL.COURS,DISP=SHR
000600 //SYSUT2 DD SYSOUT=*
000700 // PEND
000800 //PRINT EXEC PROC1
****** ******************************* Bottom of Data ******************************
```

Procédure cataloguée enregistrée dans SYS1.PROCLIB

```
EDIT       SYS1.PROCLIB(PROC2) - 01.01                Columns 00001 00072
Command ===> _______________________________________________ Scroll ===> PAGE
****** ******************************** Top of Data ********************************
000001 //PROC2 PROC
000002 //ET1  EXEC PGM=IEBGENER
000003 //SYSIN DD DUMMY
000004 //SYSPRINT DD SYSOUT=*
000005 //SYSUT1 DD DSN=&DATA1,DISP=SHR
000006 //SYSUT2 DD SYSOUT=*
000007 // PEND
****** ******************************* Bottom of Data ******************************
```

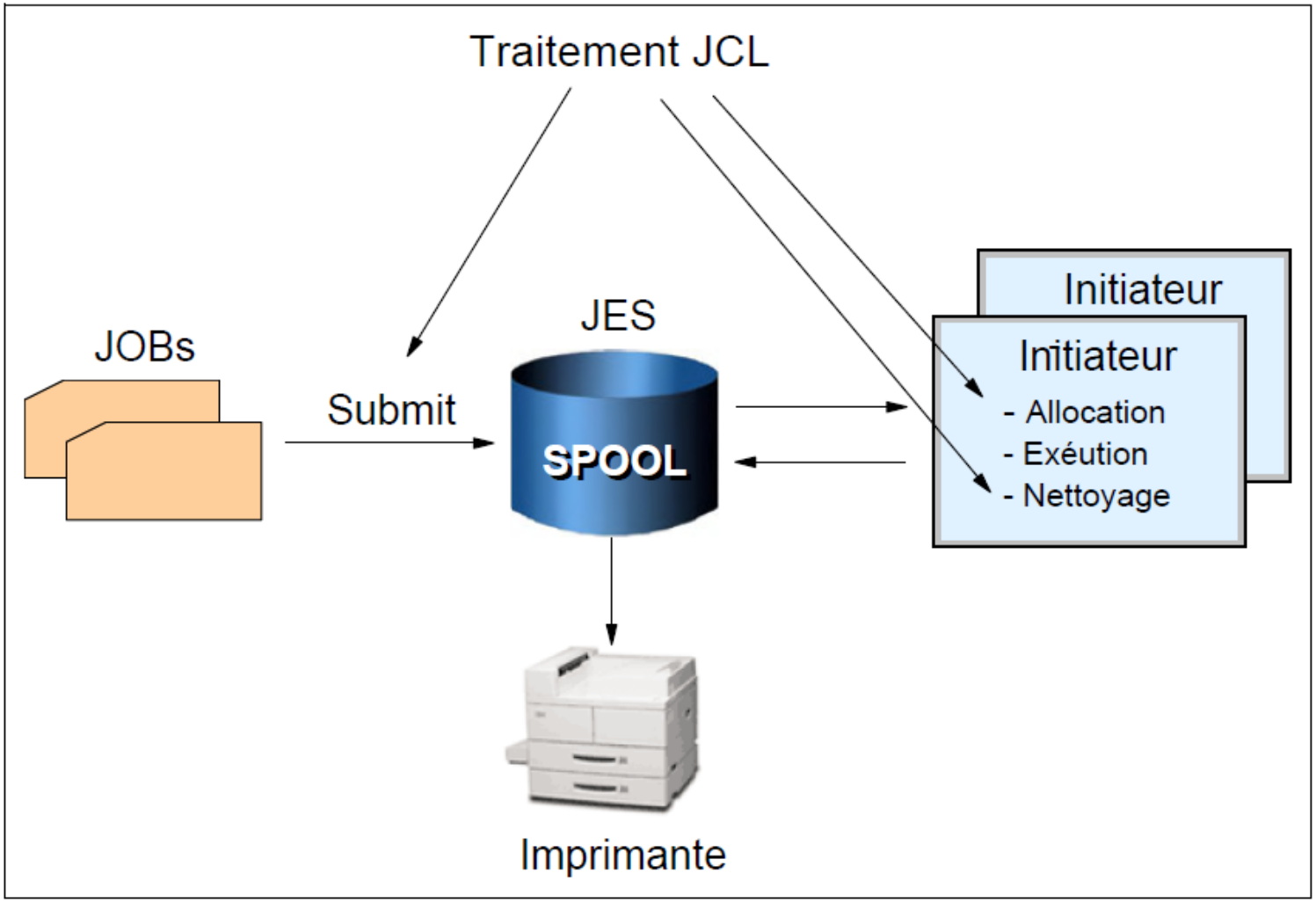
Appel de la procédure cataloguée

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EXEMP.JCL.COURS(GEN4) - 01.01                Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ******************************** Top of Data ********************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //GEN4 JOB
000200 //ET EXEC PROC=PROC2,DATA1=COPIE.JCL.COURS
****** ******************************* Bottom of Data ******************************
```

JES (Job Entry System)

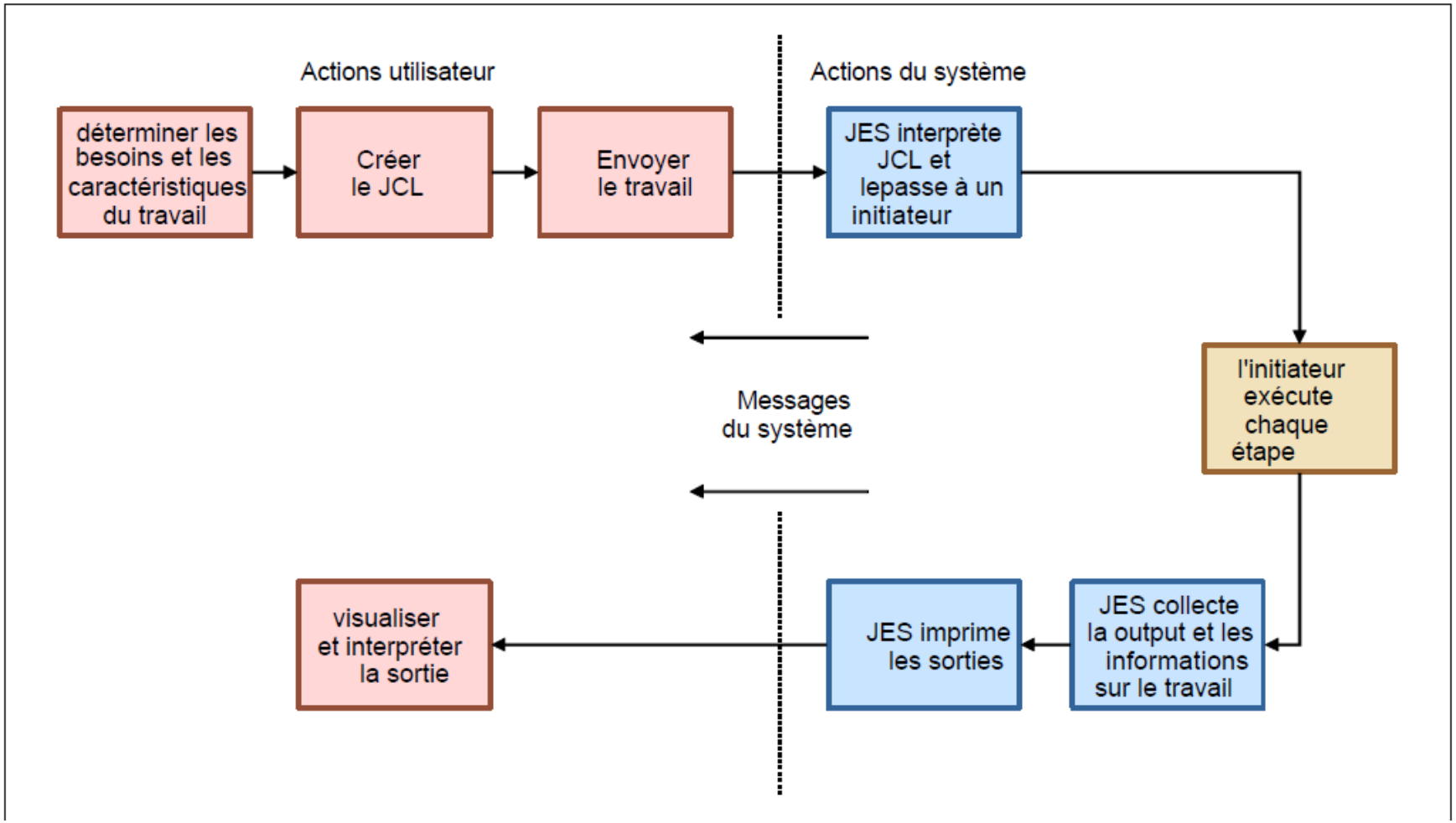
- JES est un sous système de Z/OS dont le rôle est la planification des travaux (jobs), Il traite les travaux à leurs entrée dans le système et à leurs sortie (fin d'exécution, impressions).
- JES utilise un SPOOL pour stocker les travaux, leurs paramètres (SYSIN), et leurs sorties ou impressions (SYSOUT) .
- Le SPOOL est constitué d'un ou plusieurs data sets stockés sur plusieurs volumes.

Traitement JCL

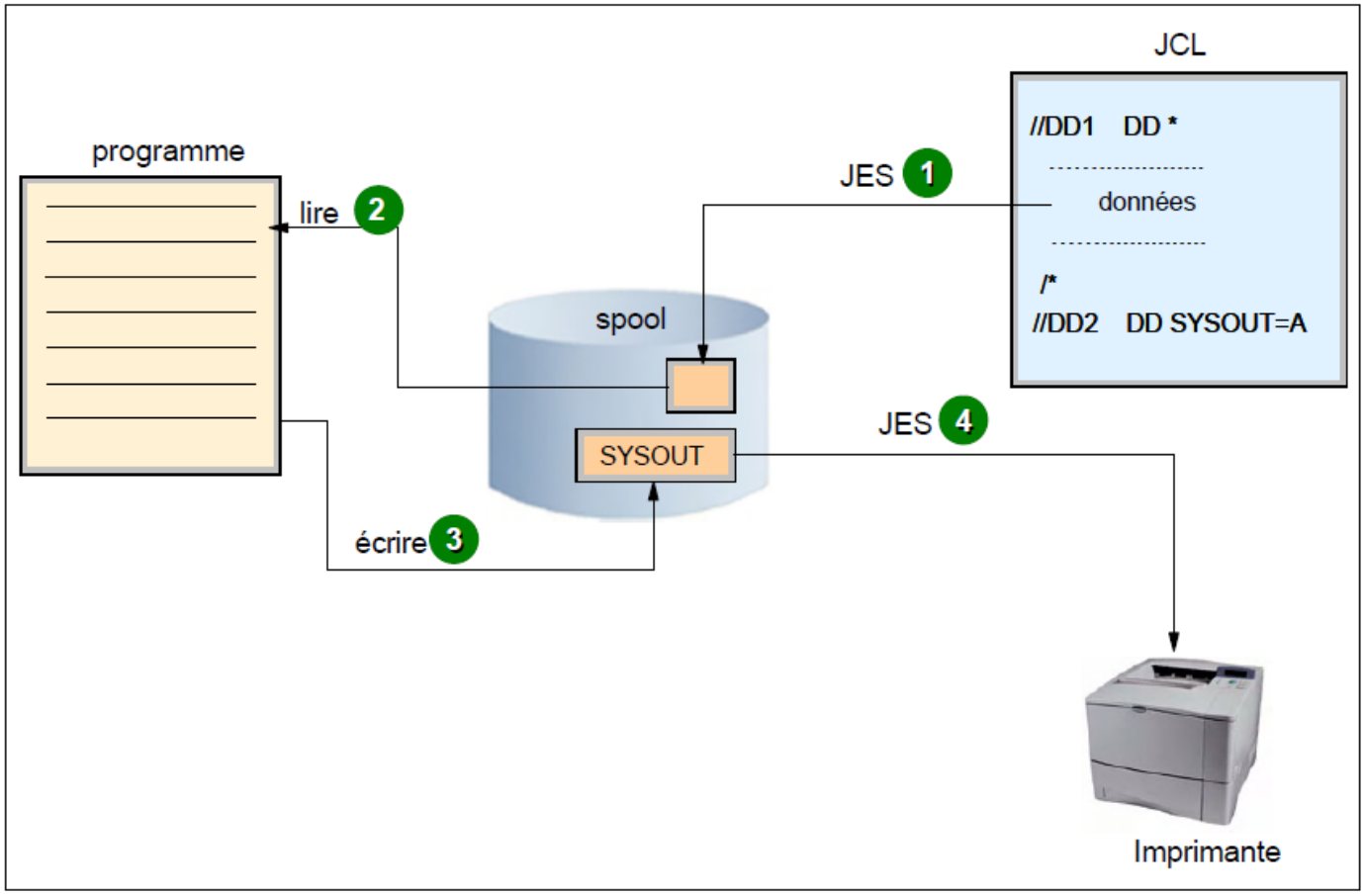


- L'initiateur est un composant du système Z/OS qui lit, interprète et exécute les instructions JCL.
- Un initiateur possède son propre espace adresse et ne peut traiter qu'un travail à la fois.
- L'opérateur peut contrôler le nombre d'initiateurs, en démarrer ou en arrêter plusieurs de manière à décider du nombre maximum de travaux batch et des classes de travaux à traiter. Les initiateurs sont rattachés à un certain nombre de classes de travaux qu'ils traitent exclusivement.

Traitement d'un travail par JES et les initiateurs.

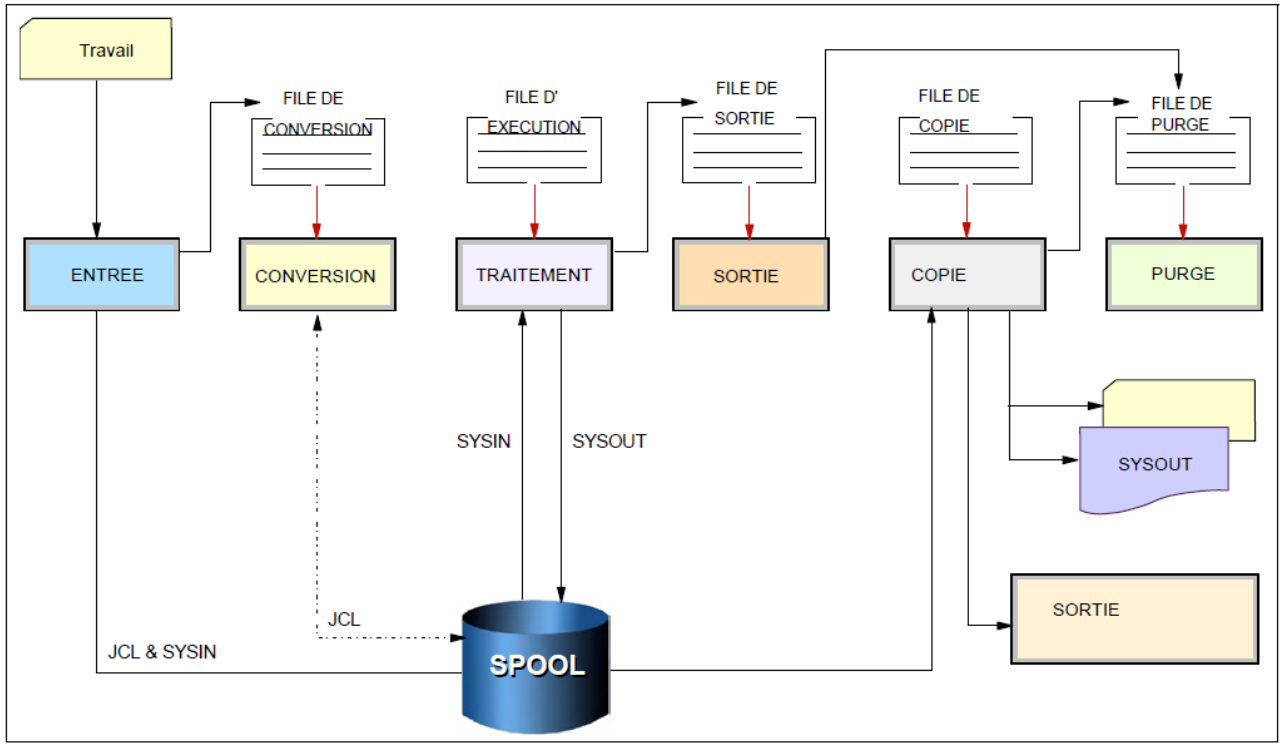


Spooling Entrée/Sortie



- Pour envoyer des données dans le fichier JCL à un programme, il faut les placer entre l'instruction DD * et /*
- Pour imprimer la sortie d'un programme SYSOUT est utilisé

Phases de traitement d'un travail



- **La purge**: le travail et toutes ses données sont purgées du SPOOL, soit que l'opérateur ou le propriétaire du travail en ait décidé ainsi (par exemple par la commande CANCEL PURGE de TSO ou par une commande JES qui nettoie le SPOOL en fonction de l'âge des SYSOUTs), soit que ce travail soit purgé automatiquement .

- **La réception** (job entry): un travail est soumis à JES depuis un périphérique (disque le plus souvent) ou d'une station de travail éloignée (remote) reliée par des lignes de Télécommunication (remote job entry). Le travail et ses données sont copiés sur le SPOOL; le travail reçoit un numéro (job id).
- **La conversion**: le JCL du travail est analysé et les procédures appelées sont éventuellement incorporées, puis il est converti par le "converteur" en texte interne (s'il ne comporte pas d'erreur). Ce texte interne, est compris par JES et l'initiateur.
- **L'interprétation**: le texte interne est utilisé pour construire en mémoire divers blocs de contrôle. JES2 n'effectue cette opération que lorsque le travail sort de la file d'attente pour être exécuté (job select); JES3 effectue l'interprétation à l'avance et stocke les blocs sur le SPOOL.
- **Le traitement**: JES passe à un initiateur le travail de plus haute priorité dans une classe donnée; en fait JES répond à l'initiateur qui lui a requis un travail. L'initiateur lance le programme de l'instruction EXEC;
- l'allocation des fichiers: avant de démarrer une étape, l'initiateur alloue tous les fichiers de l'étape et crée les blocs correspondants: les fichiers nouveaux sont créés; en fin d'étape les fichiers sont désalloués.
- **La sortie** (output): les messages et impressions créés par le travail sont recueillis une fois celui-ci terminé pour être éventuellement dirigés sur les imprimantes.

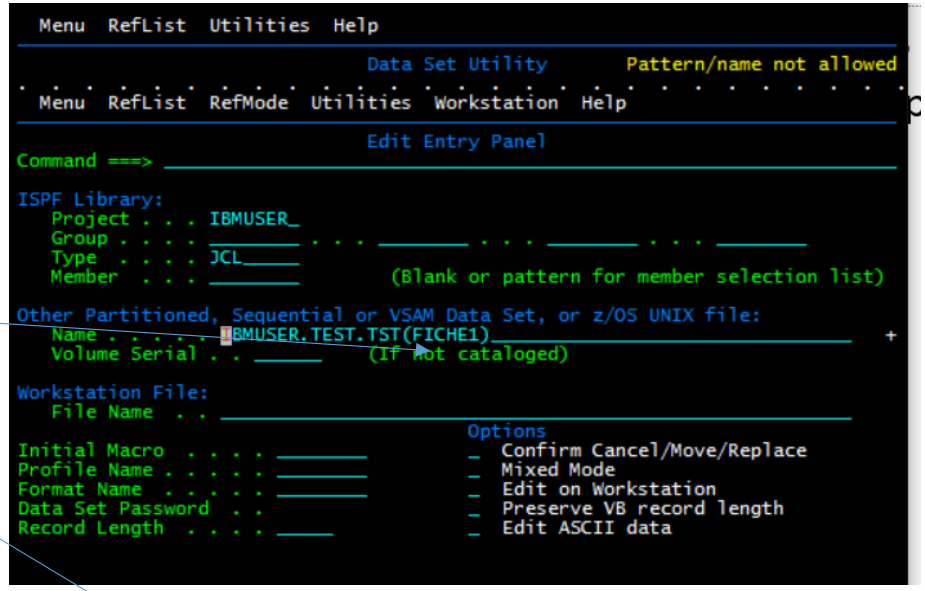
SDSF
System Display and Search Facility

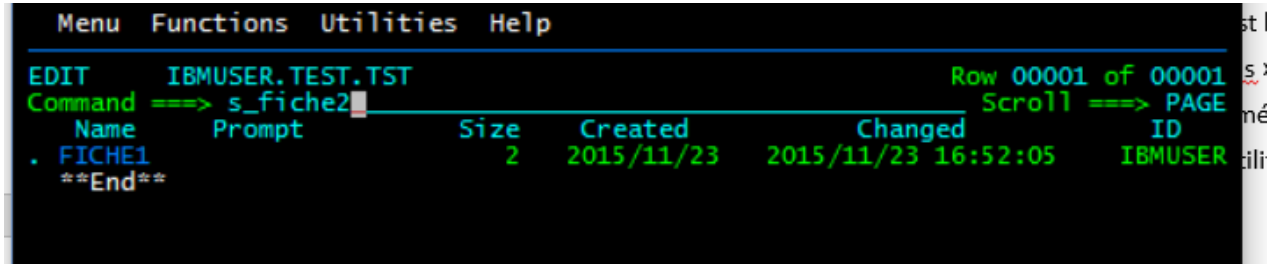
- Après avoir soumis un job, nous pourrons visualiser l'état de son d'exécution à l'aide du système du système SDSF
- SDSF fournit les fonctions suivantes:
  - Visualiser et rechercher dans les logs systèmes.
  - Contrôler l'exécution des jobs (annuler, mettre en pause et purger les jobs)
  - Superviser les jobs en cours de traitement.
  - Contrôler l'ordre d'exécution des jobs.
  - Contrôler l'ordre d'impression des sorties de jobs.
  - Contrôler les imprimantes et les initiateurs.

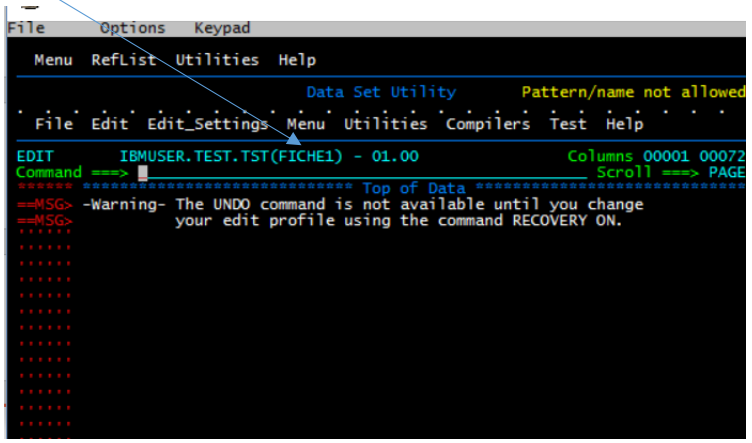
Ajouter un membre dans un dataset

- Ajouter un membre dans un dataset
  1. A partir du menu principal ISPF, choisir le panneau 2 (Edit)
  2. dans la zone « name » saisir le nom du nouveau dataset suivi du nom du nouveau membre entre parenthèses, puis taper « Entree », l'éditeur ISPF est affiché avec le nom du nouveau membre.
- Autres méthodes (pour un data set contenant au moins un membre)

  Un nouveau membre peut être ajouté lors de l'affichage du contenu d'un dataset en mode édition.
  - Saisir dans la zone name du panneau 2 (Edit) ou bien 3.4, le nom du dataset, et taper « Entree »
  - dans la ligne de commande saisir: s nom_membre , et taper « Entree »



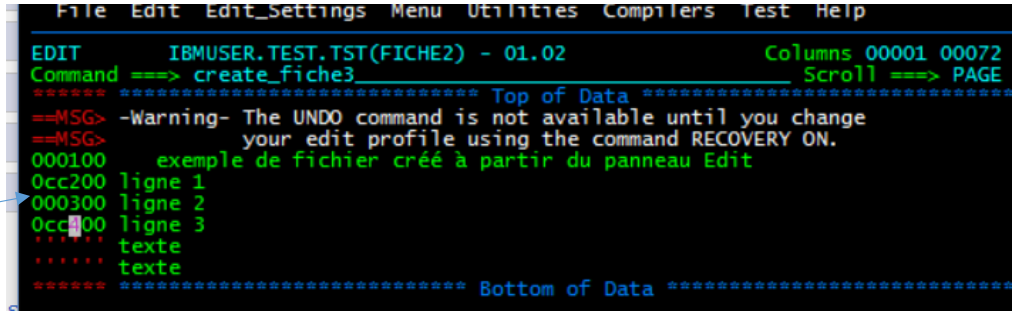


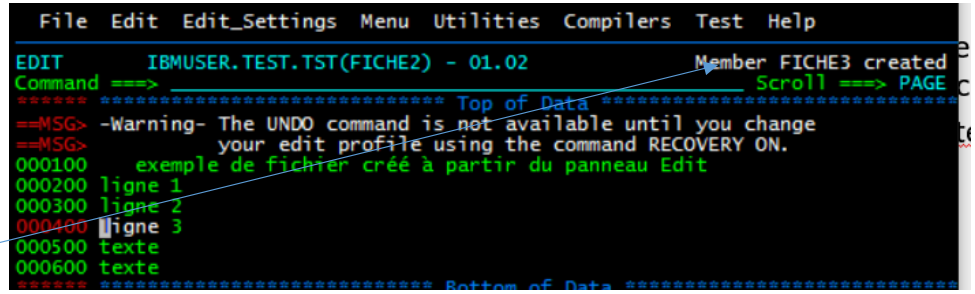


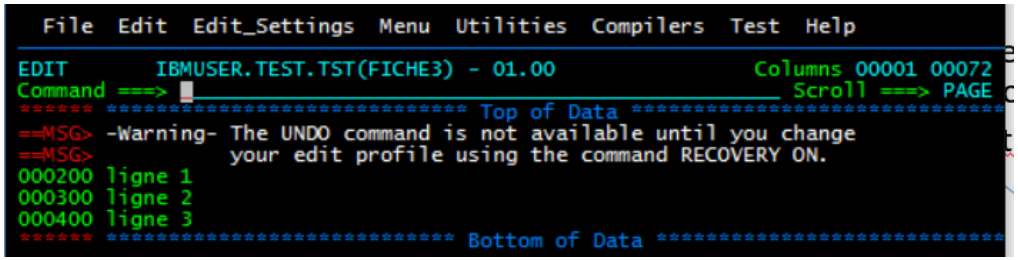
Nous pouvons aussi ajouter un nouveau membre lors de l'édition d'un autre membre:

Exemple :

- Marquer d'abord le bloc de lignes à copier dans le nouveau fichier, placer « cc » devant la ligne 'ligne 1' et cc devant la ligne 'ligne 3'.
- Saisir la commande « create fiche3 », et taper « Entree »
- Le membre « fiche3 » est créé avec le bloc de lignes marqué par « cc » comme contenu.

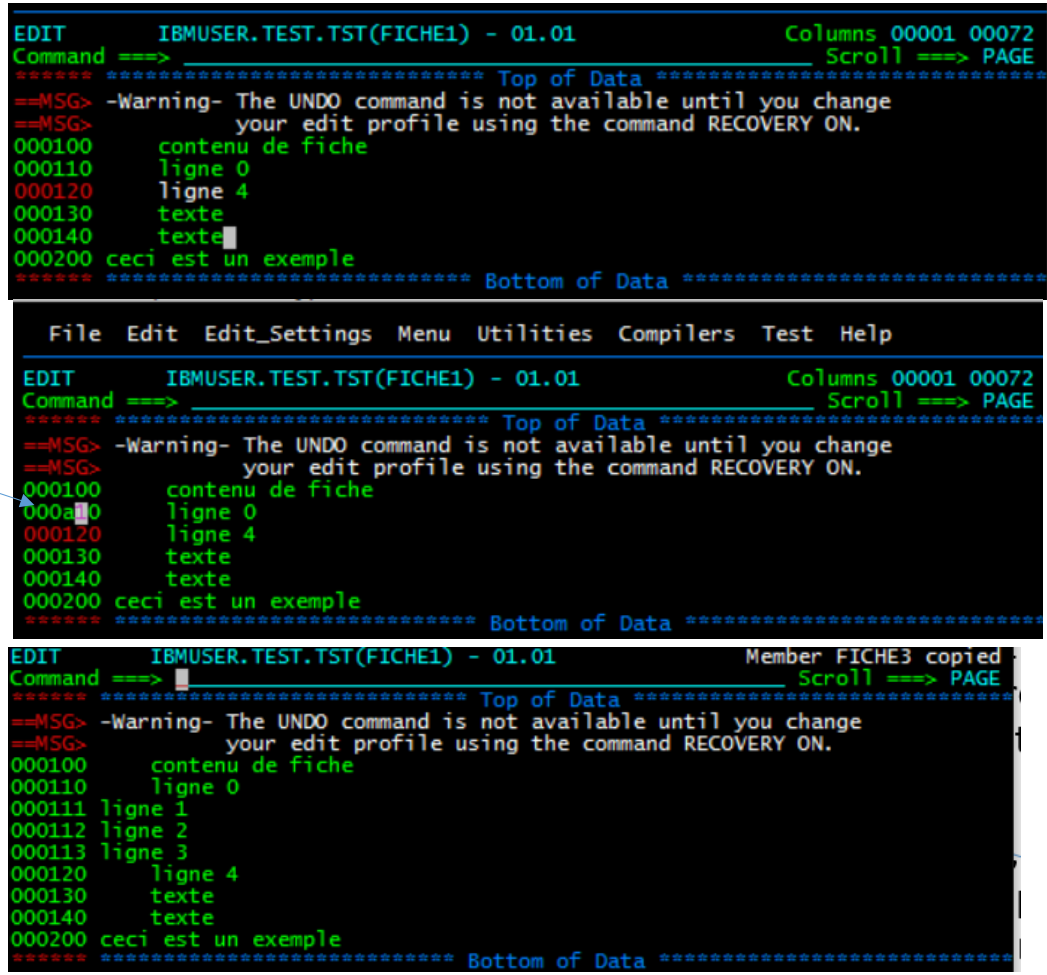






Copier le contenu d'un membre

- Soit le membre fiche1 suivant, nous souhaitons insérer le contenu de fiche3, entre les lignes ligne0 et ligne4.
- Le marqueur a (after), permet d'insérer le contenu après une ligne et b (before) avant une ligne
- Saisir la commande « copy fiche3 »



Atelier

Créer un job

1. Créer un nouveau membre dans le dataset ibmuser.jcl nommé job1
2. Écrire l'instruction JOB
3. Écrire l'instruction EXEC qui appelle le programme null IEFBR14
4. Envoyer le travail à l'aide de la commande submit (ou sub)
5. Taper « Entrée »
6. Découper l'écran en deux fenêtres

Utiliser SDSF pour afficher des informations sur le job exécuté

1. Pour afficher simultanément deux panneaux, diviser l'écran en deux fenêtres (deux sessions ISPF) à l'aide de la commande split (avant d'exécuter la commande split, placer le d'abord le curseur au milieu de l'écran)
2. Dans la première fenêtre afficher le panneau SDSF (M.5)
3. Exécuter la commande ST job*
4. Placer « s » devant le nom du job recherché (job1), puis taper « entree »
5. Modifier le job pour créer un nouveau dataset nommé ibmuser.dds1 dans le volume usr001
6. Envoyer le job et quitter avec PF3
7. Visualiser dans SDSF l'état d'exécution de ce job (la dernière exécution possède le jobid le plus élevé)
8. Quel est le code retourné par le job (condition code), si la valeur est supérieur à 0, localiser l'erreur, puis recommencer depuis l'étape 5, en corrigeant l'erreur
9. Naviguer vers le panneau DSList (=3.4) et visualiser le dataset créé
10. Supprimer le dataset

Copier un membre dans un nouveau Dataset séquentiel, sans spécifié explicitement le volume

```
EDIT       EXEMP.JCL.COURS(GENE2) - 01.01                Columns 00001 000
Command ===> _                                            Scroll ===> PA
****** ***************************** Top of Data ****************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //EXEMP2 JOB
000200 //ETAPE1 EXEC PGM=IEBGENER
000300 //SYSPRINT DD SYSOUT=*
000400 //SYSIN DD DUMMY
000500 //SYSUT1 DD DSN=IBMUSER.JCL1.JCL(JOB1),DISP=SHR
000600 //SYSUT2 DD DSN=COPI2.JCL.COURS,DISP=(NEW,CATLG),
000700 //        UNIT=SYSDA,DCB=(LRECL=80,RECFM=FB),
000800 //        SPACE=(CYL,(1,1))
****** **************************** Bottom of Data ***************************
```

JCL qui crée un dataset séquentiel à partir d'un membre, l'unité SYSDA signifie Disque, le volume est sélectionné automatiquement (JASYS1)

Informations sur le dataset créé

```
Data Set Name . . . . : COPI2.JCL.COURS

General Data                            Current Allocation
 Management class . . : **None**         Allocated cylinders : 1
 Storage class  . . . : **None**         Allocated extents . : 1
  Volume serial . . . : JASYS1
  Device type . . . . : 3390
 Data class . . . . . : **None**        Current Utilization
  Organization  . . . : PS               Used cylinders  . . : 0
  Record format . . . : FB               Used extents  . . . : 0
  Record length . . . : 80
  Block size  . . . . : 27920
  1st extent cylinders: 1
  Secondary cylinders : 1
  Data set name type  :                  SMS Compressible  :   NO

  Creation date . . . : 2015/12/06       Referenced date . . : 2015/12/06
  Expiration date . . : ***None***
```

Copie d’un dataset séquentiel dans un nouveau dataset

```
EDIT       EXEMP.JCL.COURS(GEN3) - 01.01                Columns 00001 00072
Command ===> _                                           Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000200 //GEN3 JOB
000300 //ET1 EXEC PGM=IEBGENER
000400 //SYSIN DD DUMMY
000500 //SYSPRINT DD SYSOUT=*
000600 //SYSUT1 DD DSN=COPIE.JCL.COURS,DISP=SHR
000700 //SYSUT2 DD DSN=COPIE2.JCL.COURS,DISP=(,CATLG),
000800 //   DCB=*.SYSUT1,
000900 //   UNIT=3390,
000910 //   SPACE=(CYL,(1,1)),
001000 //   VOL=SER=USR003
****** **************************** Bottom of Data ****************************
```

- Ligne 700: la disposition NEW est la valeur par défaut.
- Ligne 800: le nouveau dataset aura les mêmes paramètres DCB que le ddname sysut1.

Une procédure qui affiche le contenu d'un dataset

Procédure instream

```
VIEW       EXEMP.JCL.COURS(PROC1) - 01.00                Columns 00001 00072
Command ===> 000200_//GEN3_JOB_____________________________ Scroll ===> PAGE
****** ***************************** Top of Data ********************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PROC1 PROC
000200 //ET1  EXEC PGM=IEBGENER
000300 //SYSIN DD DUMMY
000400 //SYSPRINT DD SYSOUT=*
000500 //SYSUT1 DD DSN=COPIE.JCL.COURS,DISP=SHR
000600 //SYSUT2 DD SYSOUT=*
000700 // PEND
000800 //PRINT EXEC PROC1
****** ***************************** Bottom of Data *****************************
```

Procédure cataloguée enregistrée dans SYS1.PROCLIB

```
EDIT       SYS1.PROCLIB(PROC2) - 01.01                 Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ********************************
000001 //PROC2 PROC
000002 //ET1  EXEC PGM=IEBGENER
000003 //SYSIN DD DUMMY
000004 //SYSPRINT DD SYSOUT=*
000005 //SYSUT1 DD DSN=&DATA1,DISP=SHR
000006 //SYSUT2 DD SYSOUT=*
000007 // PEND
****** ***************************** Bottom of Data *****************************
```

Appel de la procédure cataloguée

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EXEMP.JCL.COURS(GEN4) - 01.01               Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ********************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //GEN4 JOB
000200 //ET EXEC PROC=PROC2,DATA1=COPIE.JCL.COURS
****** ***************************** Bottom of Data *****************************
```