



# PRESENTATION

---

## JCL

10/12/2020

# INTRODUCTION

# OBJECTIFS

---

Comprendre les notions suivantes :

- JCL
- ressource
- partageabilité
- travail
- Catégories de jobs
- spooling
- classe
- priorité
- Fonctions de JES

# DÉFINITION DU JCL

---

- Le JCL est un langage qui permet de décrire un travail (JOB) à exécuter, et de faire appel à certaines fonctions du système pour cette exécution
- **Le JCL est un langage codifié**
- Il permet de décrire :
  - les conditions générales d'exécution du JOB
  - le nom de chacune des étapes et ses conditions d'exécution
  - les ressources physiques de ces étapes et leur utilisation

# NOTION DE RESSOURCE

---

Pour qu'une tâche puisse s'exécuter il lui faut des **ressources** tels que : tâche.

- de la mémoire
- de la CPU
- des périphériques
- des blocs de contrôle
- des données
- du code de programme
- etc...

Lorsqu'une ressource est indisponible pour une tâche, celle-ci se retrouve en "WAIT"

# LA NOTION DE RESSOURCE

---

## Caractéristiques des ressources

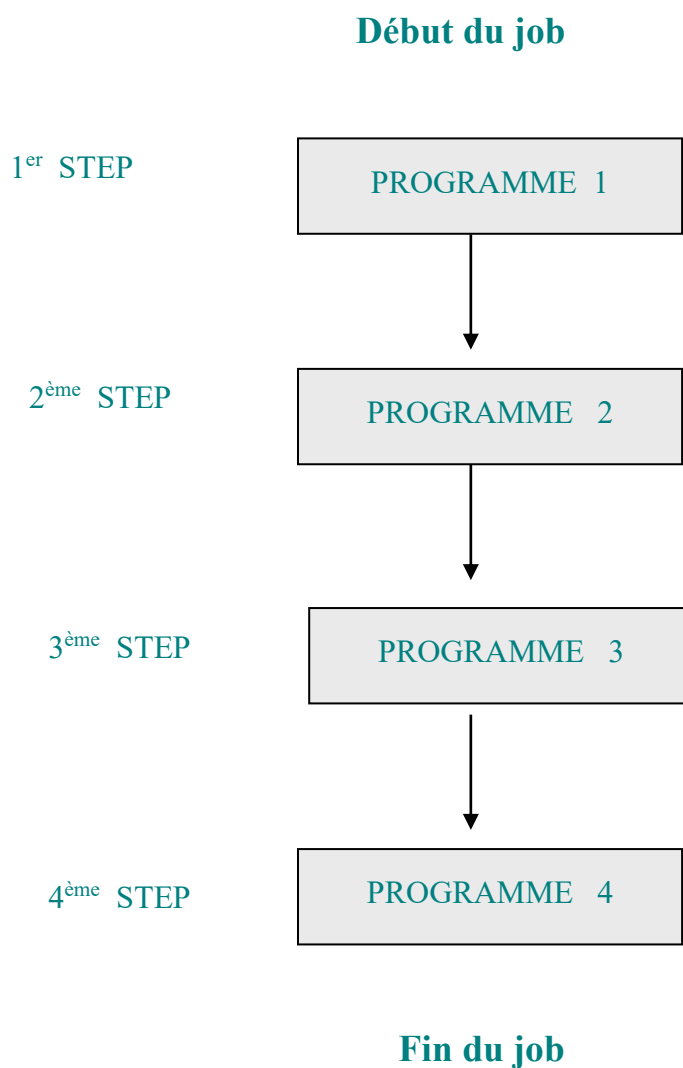
Une ressource est dite :

- **locale** : lorsqu'elle est utilisée par une seule tâche
- **globale** : lorsqu'elle est utilisée par plusieurs tâches
  - **critique** : n'est attribuée qu'à une tâche à un instant "t"
  - **partageable** : peut être attribuée à plusieurs tâches

# NOTION DE JOB

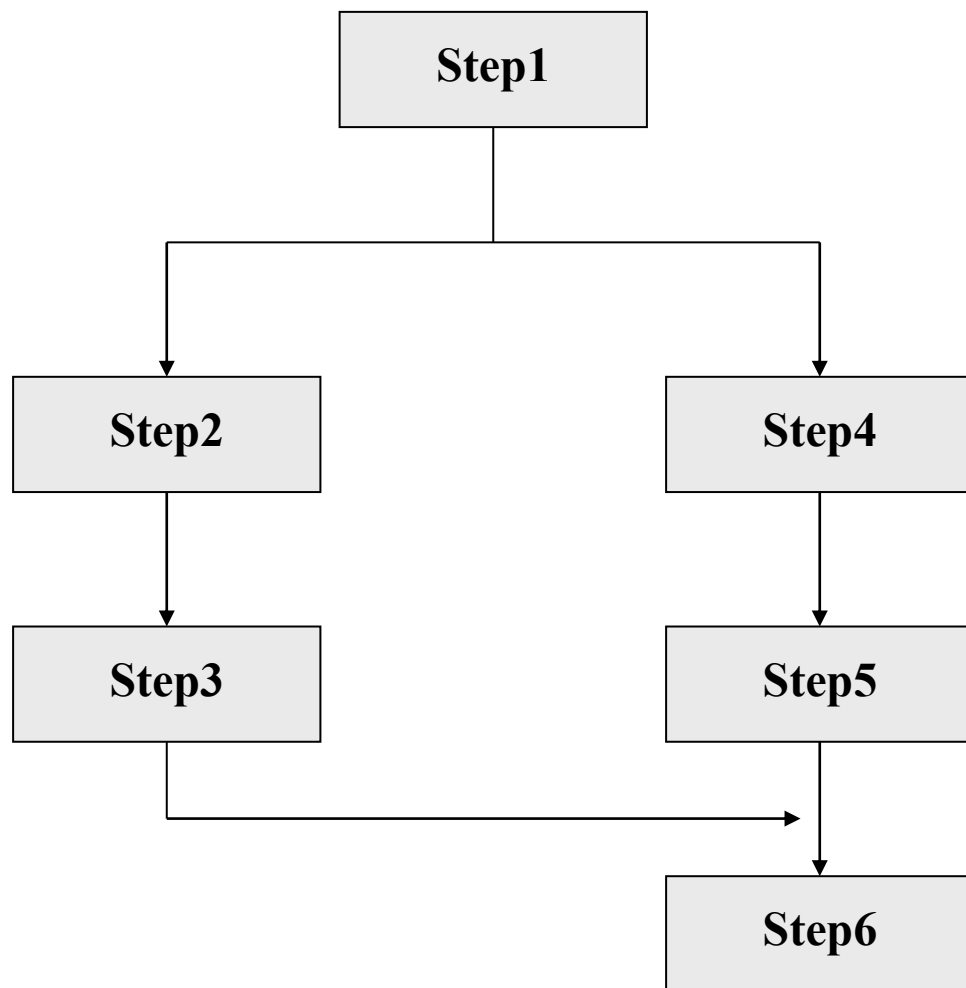
---

- STEP : exécution d'un programme
- JOB : ensemble de STEPs exécutés



# EXÉCUTION CONDITIONNELLE DE STEPS

---

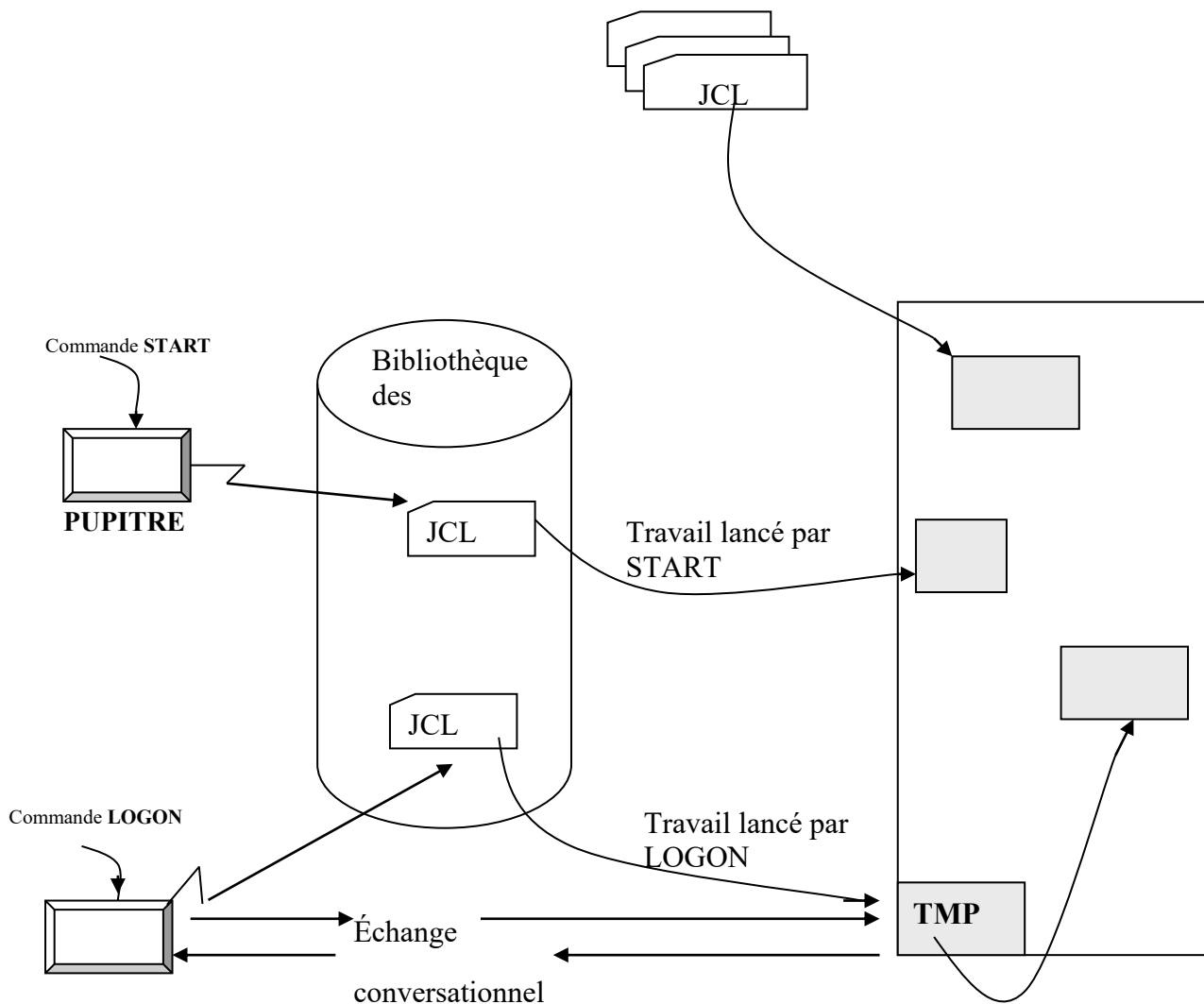




# LES CATÉGORIES DE JOBS

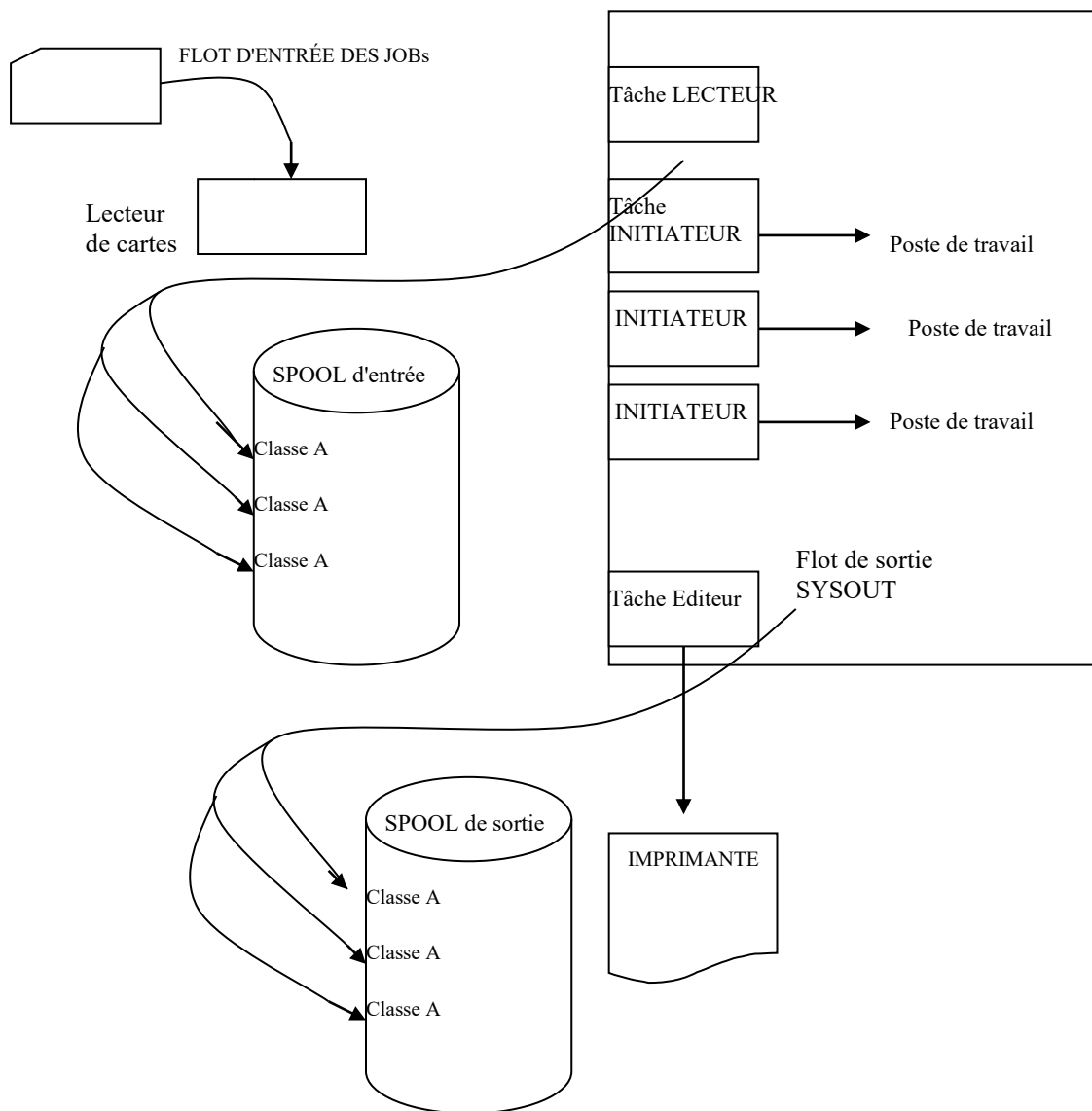
---

- Soumis par lots (BATCH)
- Lancés au pupitre (START)
- Exécutés dans une région TSO (LOGON)



# SPOOLING

## Simultaneous Peripheral Operation Online



# NOTION DE CLASSE

---

- Classe d'entrée
  - par JOB
  - paramètre CLASS (ordre JOB)
  - par exit d'installation
- Classe de sortie
  - par fichier / JOB
  - paramètre SYSOUT (ordre DD)
  - paramètre MSGCLASS (ordre JOB) pour les sorties "Système"

# LA PRIORITÉ

---

de scheduling :

- pour un JOB,  
sa place dans la file d'attente de JES
  - paramètre PRTY (ordre JOB)
  - /\*PRIORITY (JECL)
  - exit d'installation

de dispatching :

- liée à un espace-adresse,  
au moment de l'exécution d'une étape
  - paramètre DPRTY (ordre EXEC)
  - exit d'installation

# FONCTIONS D'UN JES

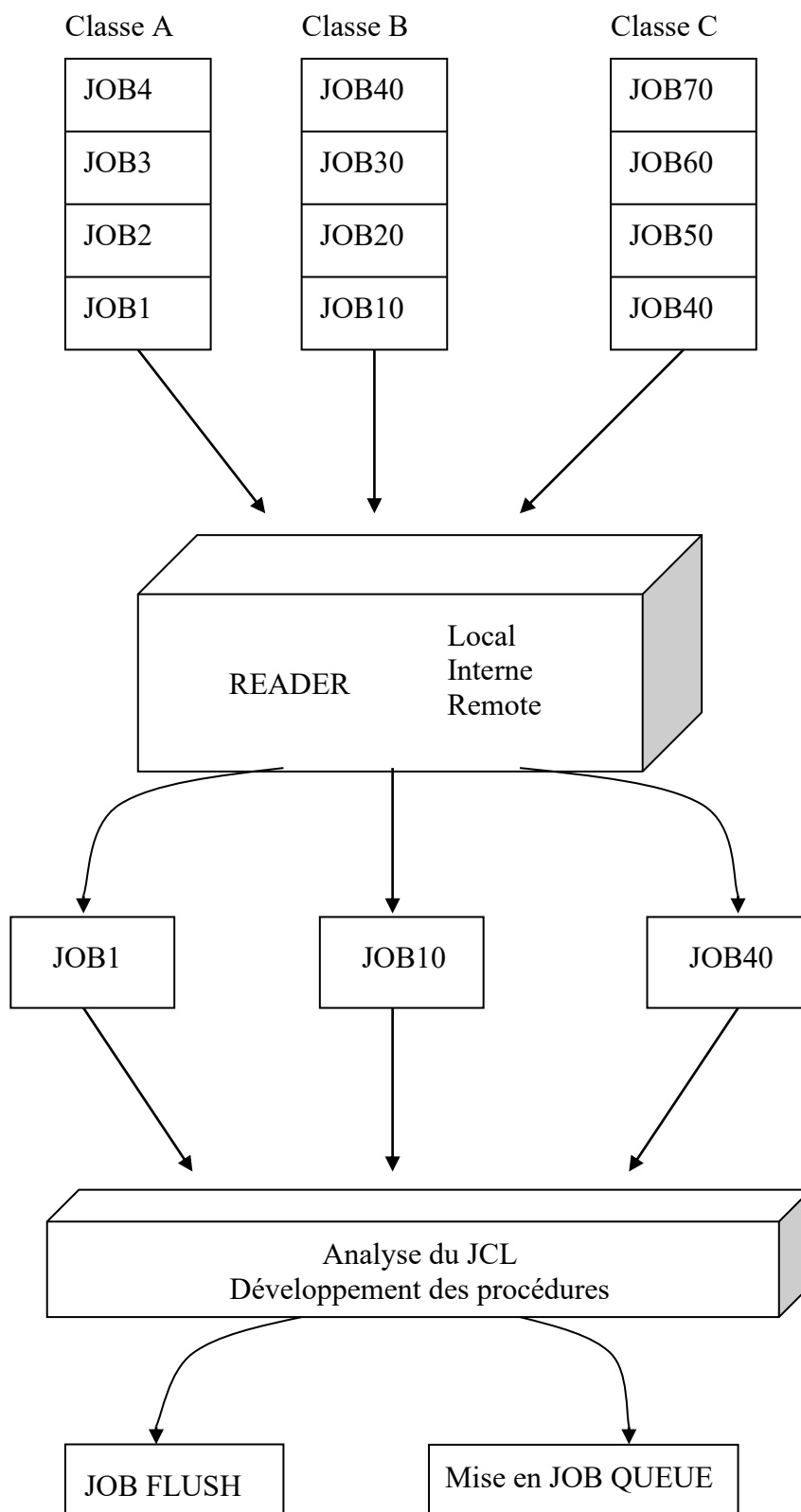
---

## **Job Entry Subsystem**

- Réception des travaux et mise en file d'attente
- Lancement des travaux et des étapes
  - sélection des JOBS
  - interprétation des ordres JCL  
lecture des fichiers "in-stream" (en ligne)
  - allocation des ressources
  - appel à l'initiateur OS/MVS
  - désallocation des ressources
- Sortie des fichiers SYSOUT
- Echange avec le pupitre

# FONCTIONS D'UN JES

---



# TYPES D'ORDRES JCL

---

- Ordres de base

```
// JOB  
// EXEC  
// DD
```

- Ordres de procédure

```
// PROC  
// PEND
```

- Autres

```
// OUTPUT  
// CNTL, ENDCNTL  
// XMIT
```

# DÉFINITIONS

---

- Flot d'entrée (input stream) :
  - description d'un certain nombre d'étapes
- JOB :
  - unité de travail (sens externe)
  - notion comptable pour le système
  - Un job = plusieurs steps
- STEP :
  - étape
  - utilise des ressources :
    - programmes
    - fichiers
    - exécution conditionnelle possible



# TRAVAUX ET ÉTAPES

---

```
//JOBNAME JOB          1 JOB
//STEP1  EXEC          3 STEPs
//DDNAME DD
//DDNAME DD
//STEP2  EXEC
//DDNAME DD
//DDNAME DD *
.....
.....
//STEP3  EXEC
//DDNAME DD
```

```
//JOBNAME JOB          1 JOB
//STEP1  EXEC          1 STEP
//DDNAME DD
//DDNAME DD *
```

.....  
.....

# SYNTAXE DU JCL

---

- Longueur maximum d'une ligne : 71 caractères
- Si insuffisant : continuation sur une autre ligne en
  - mettant une virgule au bout de la commande à continuer
  - codant la suite ainsi :

*// suite des paramètres*

**suite débutant entre les colonnes 4 et 16**

# SYNTAXE DU JCL

---

Un ordre peut débuter

- en JES2, par

```
//  
//*  
/*
```

# SYNTAXE DU JCL

---

Le format général d'une commande JCL est

//nom ORDRE paramètres commentaires

- nom : 1 à 8 caractères alpha (le premier alphabétique)
- ordre : JOB, EXEC, DD, ...
- Paramètres : séparés par des virgules
  - paramètres positionnels
  - paramètres à mot-clés:

NOMPARAM=parm  
ou  
NOMPARAM=(liste de sous-paramètres)

- Commentaires : séparés par un blanc du reste de la commande

# SYNTAXE ET PARAMÈTRES

---

## Syntaxe

```
//nomjob JOB paramètres [commentaires]
```

### Paramètres à mots-clés :

- ADDRSPC
- BYTES
- CLASS
- CARD
- COND
- GROUP
- LINES
- MSGCLASS
- MSGLEVEL
- NOTIFY
- PAGES
- PASSWORD
- PERFORM
- PRTY
- REGION
- RESTART
- SECLABEL
- TIME
- TYPRUN
- USER

# PARAMÈTRES POSITIONNELS

---

- informations comptables
- chaîne de caractères (20 car. max.)

## PARAMÈTRES DE L'ORDRE JOB

---

- CLASS= A-Z, 0-9
- MSGCLASS= A-Z,0-9
- MSGLEVEL=(contenu, qualité)
  - contenu analysé
    - 0 uniquement ordre JOB
    - 1 tous ordres exécutés
    - 2 tous ordres soumis
  - qualité de message
    - 0 aucun message
    - 1 messages d'allocation, d'exécution
- NOTIFY=nom de logon
- REGION=nnnK
- TIME=(minutes,secondes)

## PARAMÈTRES DE L'ORDRE JOB

---

- PRTY=1-15
- TYPRUN=HOLD/SCAN



# ADDRSPC

---

## Mot-clé optionnel

Sert à indiquer au système que le job requiert de la mémoire virtuelle ou réelle Syntaxe

$\text{ADDRSPC} = \begin{cases} \text{VIRT} \\ \text{REAL} \end{cases}$
---

VIRT : le système peut paginer le job

REAL : le système **ne peut pas** paginer le job, qui doit être en mémoire réelle

# BYTES

---

Mot-clé optionnel

Indique le nombre maximum de caractères imprimables en sortie de job

Indique les dispositions à prendre en cas de dépassement Syntaxe

```
BYTES={nnnnnn      }  
      {([nnnnnn][,CANCEL]) }  
      {([nnnnnn][,DUMP])   }  
      {([nnnnnn][,WARNING])}
```

nnnnnnnn : nombre de milliers d'octets de 0 à 999999

# CARDS

---

## Mot-clé optionnel

Indique le montant maximum de cartes à perforer en sortie de job

Indique les dispositions à prendre en cas de dépassement Syntaxe

```
CARDS={nnnnnnnnn }  
      {([nnnnnnnnn][,CANCEL]) }  
      {([nnnnnnnnn][,DUMP]) }  
      {([nnnnnnnnn][,WARNING])}
```

nnnnnnnnnn : nombre de cartes (de 0 à 99999999)

# CLASS

---

Mot-clé optionnel

Affecte une classe à un job Syntaxe

```
CLASSE=classe
```

classe : un caractère, de A à Z ou de 0 à 9,  
représentant une classe valide Exemple

```
//JSDJFO1 JOB 123456,SDJ,CLASS=X
```

# COND

---

Mot-clé optionnel

Spécifie le mode de test du code retour

COND est exécuté avant et après chaque step

Syntaxe

```
COND=(code,opérateur)  
COND=((code,opérateur)[,(code,opérateur)]...)
```

code : nombre de 0 à 4095 représentant le code retour à tester pour chaque step

opérateur : opérateur de comparaison utilisé : si le test est vrai, toutes les étapes suivantes sont ignorées

# COND

---

## Opérateur et signification

**LT** : plus petit que  
**LE** : plus petit ou égal  
**GT** : plus grand que  
**GE** : plus grand ou égal  
**NE** : non-égal

## Exemple

```
//STEP1 EXEC PGM=PROG1  
  
//STEP5 EXEC PGM=PROG2,COND=(20,LT,STEP1)
```

Ne pas exécuter STEP5 si le code retour du STEP1 est supérieur à 20

# GROUP

---

## Mot-clé optionnel

Spécifie le groupe RACF auquel l'utilisateur doit être relié Syntaxe

`GROUP=nom-de-groupe`

nom-de-groupe : 1 à 8 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

## Exemple

```
//USERIDC JOB 'XYZ,12345',GROUP=SDJ,USER=SDJFO1,PASSWORD=SDJWB
```

# LINES

---

## Mot-clé optionnel

Indique le montant maximum, en milliers d'octets, de lignes imprimables en sortie de job

Indique les dispositions à prendre en cas de dépassement Syntaxe

```
LINES={nnnnnn }  
      {([nnnnnn][,CANCEL]) }  
      {([nnnnnn][,DUMP]) }  
      {([nnnnnn][,WARNING])}
```

nnnnnnn : nombre de milliers de lignes (de 0 à 999999)



# MSGCLASS

---

## Mot-clé optionnel

Spécifie la classe de sortie de la trace d'exécution (log), dont le contenu dépend du paramètre MSGLEVEL Syntaxe

`MSGCLASS=classe`

classe : un caractère de A à Z ou 0 à 9 représentant une classe de sortie valide

# MSGLEVEL

---

Mot-clé optionnel

Spécifie le contenu de la trace d'exécution (log)

Syntaxe

`MSGLEVEL=([instructions][,messages])`

## Instructions :

- 0 : impression de l'instruction JOB uniquement
- 1 : impression de tous les ordres JCL, JES2 ou JES3 du contenu des procédures et des valeurs affectées aux paramètres symboliques
- 2 : impression des ordres JCL ,JES2 ou JES3 uniquement

## Messages :

- 0 : impression des messages JCL uniquement
- 1 : impression des messages JCL, JES, opérateur et SMS

# NOTIFY

---

## Mot-clé optionnel

Spécifie le nom d'un userid TSO devant être informé de la fin du traitement Syntaxe

**NOTIFY=userid**

userid : nom d'utilisateur TSO valide  
(1 à 7 caractères alphanumériques)

# PAGES

---

## Mot-clé optionnel

Indique le montant maximum, en milliers d'octets, de pages imprimables en sortie de job

Indique les dispositions à prendre en cas de dépassement Syntaxe

```
PAGES={nnnnnnnn      }  
      {([nnnnnnnn][,CANCEL]) }  
      {([nnnnnnnn][,DUMP])  }  
      {([nnnnnnnn][,WARNING])}
```

nnnnnnnnnn : nombre de milliers de lignes (de 0 à 99999999)

# PASSWORD

---

## Mot-clé optionnel

Spécifie le mot de passe RACF courant ou affecte un nouveau mot de passe

## Syntaxe

```
PASSWORD=(mot-de-passe[,nouveau-mot])
```

mot-de-passe (courant et nouveau) : 1 à 8 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

## Exemple

```
//SDJFO1 JOB 'FORMATION',GROUP=SDJ,USER=SDJI,PASSWORD=SDJ
```

# PERFORM

---

Mot-clé optionnel

Spécifie le groupe de performances du job.

Syntaxe

PERFORM=n

n : groupe de performances de 1 à 999

## Exemple

```
//SDJFO1 JOB ,SDJ,CLASS=X,PERFORM=20
```

# PRTY

---

Mot-clé optionnel

Spécifie la priorité de sélection du job.

Syntaxe

PRTY=priorité

priorité : nombre de 0 à 15 pour JES2, 0 à 14 pour JES3 (0 est la plus basse priorité)

Exemple

```
//SDJFO1 JOB , SDJ FORMATION',PRTY=15
```

# RD

---

## Mot-clé optionnel

Permet d'utiliser le paramètre de redémarrage (RD : Restart Definition) pour demander d'effectuer automatiquement un redémarrage si le travail échoue.

## Syntaxe

```
RD= {R }  
    {RNC}  
    {NR }  
    {NC }
```

R : Restart, checkpoints autorisés

RNC : Restart, No Checkpoints

NR : No Restart, checkpoints autorisés

NC : No Restart, No checkpoints



# REGION

---

## Mot-clé optionnel

Permet de spécifier la taille de mémoire réelle ou virtuelle allouée au job et à ses étapes

## Syntaxe

```
REGION= {ValeurK}  
        {ValeurM}
```

ValeurK : taille de mémoire en Kilo-octets, avec 1 à 7 chiffres. La valeur doit être multiple de 4 et ne pas dépasser 2096128

ValeurM : taille de mémoire en Méga-octets, avec 1 à 4 chiffres. La valeur ne doit pas dépasser 2047

# RESTART

---

## Mot-clé optionnel

Permet d'indiquer l'étape ou le point de contrôle où le job doit redémarrer

## Syntaxe

```
RESTART= ( { *           } [,checkid] )  
          ( { step        }           )  
          ( { step.procstep }           )
```

\* : redémarrage à la première étape

step : redémarrage à l'étape spécifiée

step.procstep : redémarrage à l'étape procstep de la procédure cataloguée appelée dans l'ordre EXEC de nom étape.

checkid : nom de point de contrôle

# SECLABEL

---

## Mot-clé optionnel

Précise le niveau de sécurité auquel le travail sera exécuté

### Syntaxe

`SECLABEL=niveau-de-sécurité`

niveau-de-sécurité : niveau défini par l'installation 1 à 8 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

# TIME

---

Mot-clé optionnel

Précise le temps maximum alloué au travail

Syntaxe

```
TIME= {[minutes][,secondes]}  
      {1440          }  
      {NOLIMIT       }  
      {MAXIMUM       }
```

minutes : nombre maximum de minutes allouées, entre 1 et 357912

secondes : nombre de secondes, entre 1 et 59, à ajouter au nombre de minutes

1440 : signifie un temps illimité (24 heures)

NOLIMIT : signifie un temps illimité

MAXIMUM : signifie le temps maximum de 357912 minutes

# TYPRUN

---

Mot-clé optionnel

Demande un traitement particulier

Syntaxe

```
TYPRUN= {COPY }  
         {HOLD }  
         {JCLHOLD}  
         {SCAN }
```

COPY : en JES2, copie le jcl sur la sysout en sortie

HOLD : bloque le job, qui ne sera exécuté que si l'opérateur le relâche

JCLHOLD : en JES2 seulement, demande que JES2 bloque le job tant que le traitement du JCL n'est pas effectué

SCAN : demande l'analyse de syntaxe du JCL, sans traitement

# USER

---

## Mot-clé optionnel

Permet d'identifier un utilisateur soumettant le travail.

## Syntaxe

`USER=userid`

userid : nom d'un utilisateur du système 1 à 7 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

# INFORMATIONS COMPTABLES

---

Paramètre positionnel, requis (dépend de l'installation)

## Syntaxe

([numéro-compte][,infos-compte]...)

## Exemples

```
//JOBSDJ JOB SDJ-1234
```

```
//JOBSDJ2 JOB (SDJ-1234,'20/07/97',PROG1)
```

# CHAÎNE DE CARACTÈRES

---

Paramètre positionnel, requis (dépend de l'installation)

20 caractères maximum (utilisation : nom du programmeur)

Suit immédiatement les informations comptables, et précède les mots clés

Apostrophes si caractères spéciaux

## Exemple

```
//JOBSDJ JOB (325345,SDJFO1)
```



# CARTE EXEC

# GÉNÉRALITÉS

---

- Début d'une Step de traitement et marque la fin du précédent
- Demande l'exécution :
  - d'un load module
  - d'une procédure

## Syntaxe

```
//[étape] EXEC paramètres [commentaires]
```

- étape : nom facultatif mais conseillé
- Paramètres positionnels :
  - PGM
  - PROC
  - Nom de procédure

# GÉNÉRALITÉS

---

Paramètres à mots-clés :

- ACCT
- ADDRSPC
- COND
- DPRTY
- DYNAMNBR
- PARM
- PERFORM
- RD
- REGION
- TIME

# ACCT

---

## Mot-clé optionnel

Spécifie un ou plusieurs sous-paramètres comptables s'appliquant à cette étape

### Syntaxe

```
ACCT[.procstep]=(infos-comptables)
```

procstep : nom d'étape de procédure

infos-comptables : dépendent du site

# ADDRSPC

---

## Mot-clé optionnel

Sert à indiquer au système que l'étape requiert de la mémoire virtuelle ou réelle

## Syntaxe

$\text{ADDRSPC} = \begin{cases} \text{VIRT} \\ \text{REAL} \end{cases}$
---

VIRT : le système peut paginer le step

REAL : le système **ne peut pas** paginer le step, qui doit être en mémoire réelle

Ce paramètre n'est pris en compte que s'il n'a pas été indiqué au niveau du JOB

## COND

---

### Mot-clé optionnel

Spécifie le test du code retour, pour savoir si le step doit être exécuté

Utilisé sur l'ordre EXEC si on veut :

- spécifier des tests différents pour chaque étape
- définir un conditionnement pour une étape spécifique
- définir des conditions d'exécution spéciales pour une étape
- ne court-circuiter qu'une étape

Note : Utiliser plutôt IF/THEN/ELSE/ENDIF

# COND

---

## Syntaxe

```
COND[.procstep]=(code,opérateur)
COND[.procstep]=((code,opérateur[,étape][.procstep])
                 [(code,opérateur[,étape][.procstep])...[,EVEN])
                 [,ONLY]

COND=EVEN
COND=ONLY
```

Code : nombre de 0 à 4095 représentant le code retour à tester pour toutes les étapes précédentes ou pour des étapes spécifiques

opérateur ::

GT , GE , EQ , NE, LT, LE
---------------------------

Etape : identifie une étape précédente ayant émis le code retour que l'on veut tester (si on l'omet, on compare avec le code émis par toutes les étapes précédentes)

Etape.procstep : identifie une étape de procédure cataloguée

EVEN : l'étape est exécutée **même si** une étape précédente s'est terminée anormalement

ONLY : l'étape est exécutée **seulement si** une étape précédente s'est terminée anormalement

# COND

---

## Opérateur et signification

<b>LT :</b>	plus petit que
<b>LE :</b>	plus petit ou égal
<b>GT :</b>	plus grand que
<b>GE :</b>	plus grand ou égal
<b>NE :</b>	non-égal

## Syntaxe

<code>DPRTY[.procstep]=([valeur1][,valeur2])</code>
---

**valeur1:** nombre de 0 à 15 indiquant si le step a une priorité égale à celle du job ou si elle est différente

**valeur2 :** nombre de 0 à 15 à ajouter à valeur1 pour former la priorité de dispatching Exemple



# DYNAMNBR

---

## Mot-clé optionnel

Indique au système de réserver un nombre de ressources pour anticiper leur réutilisation Syntaxe

`DYNAMNBR[.procstep]=n`

**n** : valeur de 0 à (3273 - nombre d'ordres DD de l'étape) spécifiant le nombre de maximum de fichiers alloués par anticipation que le système doit conserver

Note : utiliser ce paramètre plutôt que de coder DYNAM sur plusieurs ordres DD

# PARM

---

## Mot-clé optionnel

Permet de passer une information variable d'une exécution à l'autre à un programme qui a prévu cette possibilité :

## Syntaxe

```
PARM[.procstep]=paramètre  
PARM[.procstep]=(paramètre,paramètre)  
PARM[.procstep]='paramètre',paramètre)  
PARM[.procstep]='paramètre',paramètre'
```

paramètre : jusqu'à 100 caractères. Ces paramètres n'ont aucune signification pour le système.

## Exemples

```
//STEP1 EXEC PGM=PROG1,PARM='A,B,C=5"  
  
//RUN1 EXEC PGM=SDJP1,PARM=(NOOBJECT,'LINECNT=60',DECK)  
  
//STP2 EXEC PROC=ASMFLG ,PARM.LKED=(MAP,LET)
```

# PERFORM

---

Mot-clé optionnel

Spécifie le groupe de performances de l'étape

Syntaxe

```
PERFORM=n  
PERFORM[.procstep]=n
```

n : groupe de performances de 1 à 999

Ce paramètre n'est pris en compte que si le paramètre PERFORM n'a pas été codé sur l'ordre JOB, et il n'est valable que pour cette étape

# PGM

---

Paramètre positionnel optionnel

Indique le nom du programme à exécuter

Doit être le premier paramètre de l'ordre EXEC

Syntaxe

}

```
PGM= {nom-de-programme      }  
     {*.étape.ddname        }  
     {*.étape.procstep.ddname }  
     {JCLTEST                }  
     {JSTTEST
```

nom-de-programme : 1 à 8 caractères

alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

Le programme doit être un membre d'une bibliothèque système, privée ou temporaire.

**\*.étape.ddname** : se réfère à un ordre DD d'une étape précédente définissant le programme à exécuter comme membre d'un fichier partitionné

**JCLTEST** ou **JSTTEST** : en JES3, demande le contrôle de syntaxe des ordres de l'étape sans l'exécuter

# PGM

---

## Exemples

```
//LKD      EXEC PGM=IEWL
//SYSLMOD  DD DSN=*&PARDS(PROG),
//          DISP=(MOD,PASS),...
//EXC      EXEC PGM=*.LKD.SYSLMOD
//STEP1    EXEC PGM=MAJ
//DD1      DD DSN=SYS1.LINKIB(P10),
//          DISP=OLD
//STEP2    EXEC PGM=*.STEP1.DD1
```

# PROC

---

## Paramètre positionnel optionnel

Indique que le système doit exécuter une procédure cataloguée ou en ligne

Doit être le premier paramètre de l'ordre EXEC (exclusif de PGM)

## Syntaxe

<pre>{PROC=nom-de-procédure } {nom-de-procédure }</pre>
---

nom-de-programme : 1 à 8 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

Identifie la procédure à exécuter, membre d'une bibliothèque ou nom de l'instruction PROC d'une procédure en ligne apparaissant plus loin.

## Exemples

```
//STEP1 EXEC PROC=PAIE  
//STEP2 EXEC OPERATION
```

# RD

---

## Mot-clé optionnel

Permet d'utiliser le paramètre de redémarrage

(RD : Restart Definition) pour demander d'effectuer automatiquement un redémarrage de l'étape si le travail échoue.

## Syntaxe

```
RD= {R }  
    {RNC}  
    {NR }  
    {NC }
```

R : Restart, checkpoints autorisés

RNC : Restart, No Checkpoints

NR : No Restart, checkpoints autorisés

NC : No Restart, No checkpoints

Note : valable seulement si RD n'est pas codé sur l'ordre JOB, et seulement pour cette étape

# REGION

---

## Mot-clé optionnel

Permet de spécifier la taille de mémoire réelle ou virtuelle allouée à l'étape

## Syntaxe

```
REGION[.procstep]={ValeurK}  
                  {ValeurM}
```

ValeurK : taille de mémoire en Kilo-octets, avec 1 à 7 chiffres. La valeur doit être multiple de 4 et ne pas dépasser 2096128

ValeurM : taille de mémoire en Méga-octets, avec 1 à 4 chiffres. La valeur ne doit pas dépasser 2047

Note : valable seulement si REGION n'est pas codé sur l'ordre JOB, et seulement pour cette étape



# TIME

---

Mot-clé optionnel

Précise le temps maximum alloué au travail

Syntaxe

```
TIME[.procstep]={([minutes][secondes])}  
                {1440           }  
                {NOLIMIT        }  
                {MAXIMUM        }
```

minutes : nombre maximum de minutes allouées,  
entre 1 et 357912

secondes : nombre de secondes, entre 1 et 59, à  
ajouter au nombre de minutes

1440 : signifie un temps illimité (24 heures)

NOLIMIT : signifie un temps illimité

MAXIMUM : signifie le temps maximum de  
357912 minutes

## TIME

---

Note : Si le paramètre TIME codé sur l'ordre JOB n'est ni NOLIMIT ni 1440, le système prend

- la plus petite valeur entre le temps codé pour l'étape et le temps restant pour le job,
- le temps restant pour le job, si le paramètre TIME n'est pas codé pour l'étape

# CARTE DD

# GÉNÉRALITÉS

---

- Établit le lien entre le fichier logique et le fichier physique
- Fichier logique déclaré par le programme :

**SELECT F1-INT ASSIGN TO F1-EXT**

- Fichier physique : disque, bande, etc.

**//DDNAME DD identification du fichier  
(DDNAME : Nom externe)**

- Alloue la ressource constituée par le fichier au JOB en cours d'exécution

Le lien, concrétisé par un bloc de contrôle, le DCB, est réalisé au moment de l'OPEN du fichier

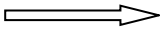
- Identification d'un fichier :
  - nom du fichier (DSNAME)
  - identification du volume (VOLUME)
  - identification de l'unité (UNIT)
- Intérêt de l'emploi d'un catalogue:

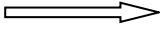
**DSNAME <=== VOLUME + UNIT**

# LES FICHIERS MVS

---

- Chaînage des informations permettant d'accéder à un fichier:

- DSNNAME  Catalogue

- Catalogue  Unité et volume

- Volume disque :

La VTOC (**V**olume **T**able **O**f **C**ontents) contient la liste des DSCB (**D**ata**S**et **C**ontrol **B**lock):

- nom fichier
- adresse(s) sur le disque
- format
- Volume bande :

Le catalogue contient le numéro du fichier sur la bande

## CHOIX DES PARAMÈTRES

---

### En création, préciser :

- Statut du fichier            DISP
- Nom du fichier            DSN
- Nom du volume            VOL
- Nature de l'unité            UNIT
- Taille du fichier            SPACE
- Format du fichier            DCB
- En lecture, préciser:

### Si catalogué:

- Statut du fichier            DISP
- Nom du fichier            DSN

### Si non catalogué, rajouter :

- Volume du fichier            VOL
- Nature de l'unité            UNIT

# FORMAT DE LA CARTE DD

---

## Syntaxe

```
//[ddname      ]  DD paramètres  
[comment.]  
//[procstep.ddname]  
  
//[ddname      ]  DD  
//[procstep.ddname]
```

# LE DDNAME

---

ddname : 1 à 8 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

Représente un nom de fichier logique

Omis si

- concaténation au fichier précédent
- deuxième ou troisième ordre DD d'un fichier séquentiel indexé

ddnames à usage réservé

JOBCAT	SYSABEND	SYSIN
JOBLIB	SYSMDUMP	SYSCHK
STEPLIB	SYSUDUMP	SYSCHKEOV
STPCAT	SYSCHK	

ddnames réservés au système et ne devant pas être utilisés

JCBIN	JESJCL	JS3CATLG
JCBLOCK	JESMSG LG	J3JBINFO
JCBTAB	JOURNAL	J3SCINFO
JESJCLIN	JST	
JESInnnn	JESYSMSG	



# PARAMÈTRES

---

## Paramètres à blanc

- pour "override" un ordre DD concaténé dans une procédure cataloguée
- quand SMS fournit les options nécessaires

## Paramètres positionnels:

- \*
- DATA
- DUMMY
- DYNAM

# PARAMÈTRES

---

## Paramètres positionnels:

### Exemples

- Insertion d'un fichier en ligne :

```
//DD1 DD *  
donnee1  
donnee2  
/*
```

- Suppression de l'accès à un fichier

```
//DD2 DD DUMMY
```

# PARAMÈTRES

---

## Paramètres à mot-clé :

Après les paramètres positionnels

ACCODE	DSNTYPE	QNAME
AMP	EXPDT	RECFM
AVGREC	FCB	REORG
BURST	FLASH	REFDD
CHARS	FREE	RETPD
CHKPT	HOLD	SECMODEL
CNTL	KEYLEN	SEGMENT
COPIES	KEYOFF	SPACE
DATACLAS	LABEL	SPIN
DCB	LIKE	STORCLASS
DDNAME	LRECL	SUBSYS
DEST	MGMTCLAS	SYSOUT
DISP	MODIFY	TERM
DLM	OUTLIM	UCS
DSID	OUTPUT	UNIT
DSNAME	PROTECT	VOLUME

# CONCATÉNATION DE FICHIERS

---

- Un seul DDNAME mais un ordre DD par fichier physique
- Seul le premier ordre DD contient le DDNAME
- Uniquement en lecture

## Exemple

```
//FICH1    DD DSN=SDJ.ENTREE1,DISP=SHR  
//         DD DSN=SDJ.ENTREE2,DISP=SHR
```

# ACCODE

---

## Mot-clé optionnel

Définit ou modifie le code d'accessibilité pour des bandes en sortie de format ISO/ANSI/FIPS

### Syntaxe

`ACCODE=code-accès`

code-accès : de 1 à 8 caractères commençant par une lettre majuscule de A à Z

Note : ACCODE ignoré pour les bandes de format standard (SL)

## Mot-clé optionnel

Avec SMS, permet de créer un fichier VSAM avec un ordre DD

## Syntaxe

```
AMP=(sous-paramètre)
AMP=('sous-paramètre[,sous-paramètre]...')
AMP='sous-paramètre[,sous-paramètre]...'
```

## Sous-paramètre

```
AMORG
BUFND=nombre
BUFNI=nombre
BUFSP=nombre
CROPS={NCK}
      {NRC}
      {NRE}
      {RCK}
OPTCD={I }
      {L }
      {IL}
RECFM={F }
      {FB}
      {V }
      {VB}
STRNO=nombre
SYNAD=module
TRACE=(sous-
paramètre[,sous-paramètre]...)
```

# AVGREC

---

## **Mot-clé optionnel,**

utilisable uniquement avec SMS

Quand on définit un nouveau fichier, permet de spécifier l'unité d'allocation

### Syntaxe

```
AVGREC={U}  
        {K}  
        {M}
```

U : l'unité utilisée dans le paramètre SPACE est le nombre d'enregistrements (unité)

K : l'unité utilisée dans le paramètre SPACE est le millier d'enregistrements (multiplier par 1024)

M : l'unité utilisée dans le paramètre SPACE est le million d'enregistrements (multiplier par 1048576)

# BLKSIZE

---

## Mot-clé optionnel

Permet de définir la longueur en octets d'un bloc

## Syntaxe

BLKSIZE=octets

**octets** : nombre dépendant de l'unité (maximum 32760)



# BURST

---

## Mot-clé optionnel

Sur une imprimante 3800, permet de définir si la sortie doit être produite sur papier continu ou sur feuilles séparées

### Syntaxe

```
BURST= {YES}  
        {Y }  
        {NO }  
        {N }
```

YES : sortie sur feuilles séparées

NO : sortie en continu

# CHARS

---

## Mot-clé optionnel

Sur une imprimante 3800, permet de définir une ou plusieurs tables de caractères utilisées pour imprimer cet état

## Syntaxe

```
CHARS={nom-de-table      }  
      {(nom-de-table[,nom-de-table]...)}  
      {DUMP              }  
      {DUMP[,nom-de-table]...} }
```

nom-de-table : nom d'une table de caractères de 1 à 4 caractères alphanumériques ou \$, #, @.

On peut indiquer de une à quatre tables

DUMP : demande un format DUMP, de 204 caractères par ligne, sur SYSABEND ou SYSUDUMP

# CHKPT

---

## Mot-clé optionnel

Demande la prise de points de contrôle à la fin de chaque volume (pour un fichier multivolumes QSAM ou BSAM).

## Syntaxe

CHKPT=EOV

EOV : demande la prise de point de contrôle à la fin de chaque volume

# CNTL

---

## Mot-clé optionnel

Permet de référencer un ordre CNTL défini précédemment dans le job. Les instructions entre CNTL et ENDCNTL sont alors exécutées.

## Syntaxe

```
CNTL={*.label      }  
CNTL={*.step.label  }  
CNTL={*.step.procstep.label}
```

\*.label : référence un ordre CNTL précédent, plus haut dans l'étape ou avant le premier ordre EXEC

\*.step.label : référence un ordre CNTL précédent, plus haut dans l'étape step

\*.step.procstep.label : idem, procstep est le nom d'étape de procédure cataloguée contenant l'ordre CNTL

# COPIES

---

## Mot-clé optionnel

Demande la production d'un nombre de copies spécifiées d'un fichier d'impression.

Sur une imprimante 3800, permet également d'indiquer que l'on veut plusieurs copies de chaque page

## Syntaxe

```
COPIES={nnn  
COPIES={(nnn,(val-groupe[,val-groupe]...))}  
COPIES={((val-groupe[,val-groupe]...)) }
```

nnn : spécifie le nombre de copies, de 1 à 255 pour JES2, et de 1 à 254 pour JES3. Ignoré pour une 3800 si les paramètres suivants sont indiqués

Val-groupe : pour une 3800, spécifie le nombre copies de chaque page à imprimer (1 à 8 groupes). Le nombre de copies est la somme des nombres.

# DATA

---

## Paramètre positionnel optionnel

Permet de spécifier que les données suivantes sont un fichier de données en images-cartes pouvant comporter les caractères // en colonnes 1 et 2.

Le fichier se termine

par les caractères /\* en colonnes 1 et 2

- par les deux caractères spécifiés par le paramètre DLM
- par le fin du flot d'entrée

## Syntaxe

```
//ddname DD DATA[,paramètres]... [comment.]
```

# DATACLAS

---

## Mot-clé optionnel

utilisable uniquement avec SMS

Permet de spécifier une classe de données  
pour un nouveau fichier

## Syntaxe

```
DATACLAS=nom-de-classe
```

nom-de-classe : nom de classe utilisée pour  
allouer les fichiers

# DCB

---

## Mot-clé optionnel

Permet de spécifier les attributs d'un fichier, permettant de construire le DCB (Data Control Block)

## Syntaxe

```
[      DCB=(sous-paramètre[,sous-paramètre]...)]  
  
[DCB=({dsname          }[,param.]...)]  
[      ({*.ddname       }          )]  
[      ({*.step.ddname  }          )]  
[      ({*.step.procstep.ddname}    )]
```

Sous-paramètres du DCB : voir liste

dsname : le système copie les caractéristiques du DCB du fichier de nom dsname

\*.ref : définit une référence à un ordre DD défini précédemment, dont on copie les caractéristiques de DCB



# DCB

---

## Sous-paramètres

BFALN	INTVL
BFTREK	IPLTXID
<b>BLKSIZE</b>	<b>KEYLEN</b>
<b>BUFNI</b>	LIMCT
BUFL	<b>LRECL</b>
BUFMAX	MODE
<b>BUFNO</b>	NCP
BUFOFF	NTM
BUFOUT	OPTCD
BUFSIZE	PCI
CPRI	PRSTP
CYLOFL	RECFM
DEN	RESERVE
DIAGNS	RKP
<b>DSORG</b>	STACK
EROPT	THRESH
FUNC	TRTCH
GNCP	

# DDNAME

---

## Mot-clé optionnel

Permet de différer la création d'un fichier à un ordre défini plus loin dans l'étape, ou défini dans une procédure appelée

### Syntaxe

`DDNAME=ddname`

ddname : référence un ddname défini plus loin

Cinq ordres DD avec le paramètre DDNAME au maximum par étape

# DEST

---

## Mot-clé optionnel

Permet de définir la destination d'un fichier d'impression en sortie (valable uniquement avec SYSOUT).

## Syntaxe

DEST=destination

## destination JES2

LOCAL  
name  
Nnnnn  
NnnRmmmm  
NnnnRmmm  
NnnnnRmm  
Rnnnn  
RMnnnn  
RMTnnnn  
Unnnn  
(node,userid)

# DEST

---

## Destination JES3

ANYLOCAL  
device-name  
device-number  
group-name  
nodename  
(node,userid)

# DISP

---

## Mot-clé optionnel

Indique l'état du fichier et les dispositions à prendre après la fin de l'étape ou du travail

### Syntaxe

```
{DISP=état }  
{DISP=( [état] [,fin-normale] [,fin-anormale] ) }
```

```
DISP=( [NEW] [,DELETE ] [,DELETE ] )  
      [OLD] [,KEEP ] [,KEEP ] )  
      [SHR] [,PASS ] [,CATLG ] )  
      [MOD] [,CATLG ] [,UNCATLG] )  
      [, ] [,UNCATLG]  
      [, ]
```

### Sous-paramètre état

NEW : le fichier doit être créé

OLD : le fichier existe et cette étape requiert un accès exclusif

SHR : le fichier existe, et d'autres travaux peuvent y accéder

MOD : des enregistrements vont être ajoutés en fin de fichier

## DISP

---

### **Sous-paramètre fin-normale**

DELETE : fichier annulé

KEEP : fichier sauvegardé sur le volume

CATLG : fichier catalogué

UNCATLG : fichier décatalogué

PASS : fichier passé à une étape suivante

### **sous-paramètre fin-anormale**

DELETE : fichier annulé

KEEP : fichier sauvegardé sur le volume

CATLG : fichier catalogué

UNCATLG : fichier décatalogué

# DISP

---

## Options par défaut

Etat initial : **NEW**

### Fin normale :

- Si état initial NEW : DELETE
- Si état initial OLD/SHR : KEEP

### Fin anormale :

- même valeur que pour fin normale
- Si aucun DISP : **NEW, DELETE, DELETE**
- Si DISP=OLD: OLD, KEEP, KEEP

### Exemple:

```
//STEP1    EXEC PGM=PROG1
//DD1      DD UNIT=VIO,DISP=(,PASS),
//          SPACE=(TRK,12)
//STEP2    EXEC PGM=PROGN
//DDN      DD DSN=*.STEP1.DD1,DISP=(OLD,DELETE)
```

# DLM

---

## Mot-clé optionnel

Indique le délimiteur d'un flot d'entrée en ligne si le défaut /\* n'est pas utilisé (à employer uniquement avec DD DATA ou DD \*)

## Syntaxe

DLM=délimiteur

délimiteur : deux caractères indiquant la fin du flot d'entrée

## Exemple

```
//DD1 DD *,DLM=ZZ  
..AAAAAA.  
..BBBBBB..  
ZZ
```



# DSID

---

## Mot-clé optionnel

Indique l'identificateur de fichier pour les unités de diskettes 3540

### Syntaxe

DSID= {id    }  
          {(id,[V])}

id : identificateur de fichier (8 caractères alphabétiques, \$, #, @, -, [.

V : le fichier doit avoir été vérifié

# DSNAME

---

## Mot-clé optionnel

Spécifie le nom d'un fichier

## Syntaxe

```
{DSNAME} = nom  
{DSN }
```

## Nom de fichier permanent

```
dsname  
dsname(membre)  
dsname(génération)  
dsname(area)
```

## Nom de fichier temporaire

```
&&dsname  
&&dsname(membre)  
&&dsname(area)
```

## Nom de fichier en ligne ou sysout

```
&&dsname
```

# DSNAME

---

Nom copié d'un step précédent

```
*.ddname  
*.stepname.ddname  
*.stepname.procstepname.ddname
```

nom de fichier inexistant

```
NULLFILE
```

Nom : il peut être

- simple : 1 à 8 caractères alphanumériques, œ, #, @, - ou x'C0'
- qualifié : noms simples reliés par des points (44 caractères au maximum)

# DSNAME

---

## Exemples

```
DSN=entrée  
DSN=compta.sortie  
DSN=biblio(module)  
DSN=&&goset(go)  
DSN=*.DD1  
DSN=*.STEP1.DD2
```

# DSNTYPE

---

## Mot-clé optionnel

utilisable uniquement avec SMS

Permet de définir un nouveau fichier partitionné (PDS) ou partitionné étendu (PDSE).

## Syntaxe

```
DSNTYPE={LIBRARY}  
        {PDS  }
```

LIBRARY : définit un PDSE géré par SMS

PDS : définit un PDS

# DUMMY

---

## Paramètre positionnel optionnel

Permet de spécifier que

- aucune unité de stockage ne doit être allouée au fichier
- aucun traitement ne doit être réalisé sur le fichier
- pour un fichier BSAM ou QSAM, aucune opération d'entrée ou de sortie ne doit être effectuée sur le fichier

## Syntaxe

```
//ddname DD DUMMY[,paramètres]...
```

# DYNAM

---

## Paramètre positionnel optionnel

Permet d'augmenter de 1 le nombre de ressources allouées dynamiquement pour réutilisation

Utilisé pour compatibilité avec des systèmes plus anciens

### Syntaxe

```
//ddname DD DYNAM [commentaires]
```

Ne pas coder d'autres paramètres sur l'ordre DD

# EXPDT

---

## Mot-clé optionnel

Permet de préciser une date d'expiration pour les nouveaux fichiers

### Syntaxe

```
EXPDT= {aa jjj }  
        {aaaa/ jjj }
```

aa jjj : année sur 2 chiffres (99 = 1999) et jour sur trois chiffres (001 à 366)

aaaa/ jjj : année sur 4 chiffres (jusqu'à 2155) et jour sur trois chiffres (001 à 366)

**Note** : les dates 99365, 99366, 1999/365 et 1999/366 sont considérées comme ineffaçables



## Mot-clé optionnel

Permet de spécifier un buffer de contrôle d'état (FCB : Forms Control Block) simulant l'ex bande pilote et donnant aux imprimantes les informations nombres de lignes par pouce et pour l'état

## Syntaxe

```
FCB= {nom-fcb      }  
      {(nom-fcb[,ALIGN|VERIFY])}
```

nom-fcb : nom de membre de la  
SYS1.IMAGELIB

ALIGN : demande à l'opérateur de vérifier  
l'alignement avant d'imprimer

VERIFY : demande à l'opérateur de vérifier  
que l'image imprimée correspond au bon FCB

# HOLD

---

## Mot-clé optionnel

Demande au système de conserver un fichier sysout jusqu'à ce qu'il soit libéré par l'opérateur

## Syntaxe

```
HOLD= {YES}  
      {Y }  
      {NO }  
      {N }
```

YES : demande que le système conserve le fichier sysout jusqu'à sa destruction par l'opérateur

NO : pas de conservation (option par défaut)

# KEYLEN

---

## Mot-clé optionnel

Précise la longueur de la clé d'un nouveau fichier

### Syntaxe

KEYLEN=octets

octets : nombre d'octets de la clé (0 à 255 pour les fichiers non VSAM, 1 à 255 pour les VSAM KSDS)

S'applique aux organisations BDAM, BPAM, BSAM, EXCP, QISAM, TCAM et, avec SMS, à VSAM.

# KEYOFF

---

## Mot-clé optionnel

utilisable uniquement avec SMS

Permet de définir, pour un fichier VSAM, la position de départ de la clé par rapport au début de l'enregistrement

## Syntaxe

KEYOFF=position

position : position de départ de la clé (le 1er octet de l'enregistrement est en **position 0**)

# LABEL

---

## Mot-clé optionnel

Précise le type et le contenu du ou des labels sur les fichiers bande ou à accès direct, ainsi que d'autres informations comme la protection par mot de passe, la période de rétention ou le mode d'ouverture

## Syntaxe

```
LABEL=( [seq][,label][,PASSWORD][,IN ][,RETPD=nnnn ] )  
        [, ][,NOPWREAD][,OUT][,EXPDT={aa jjj }]  
        [, ] [ {aaaa/jjj} ]
```

seq : position relative du fichier sur la bande (1 par défaut)

label :

- SL (par défaut) : standard
- SUL : standard et user
- AL : ISO/ANSI version 1 ou ISO/ANSI/FIPS version 3

## LABEL

---

- AUL : idem
- NSL : non standard label
- NL : no label
- BLP : bypass label processing
- LTM : leading tapemark

PASSWORD : le mot de passe doit être fourni pour toute opération via TSO/E

NOPWREAD : mot de passe sauf pour la lecture

IN : lecture seule (BSAM, BDAM)

OUT : écriture seule (BSAM)

RETPD et EXPDT : période de rétention ou date d'expiration

# LIKE

---

## Mot-clé optionnel

utilisable uniquement avec SMS

Permet de copier la définition d'un fichier à partir d'un autre servant de modèle  
(analogue à DCB=dsname sans SMS)

## Syntaxe

LIKE=nom-de-fichier
---------------------

nom-de-fichier : fichier servant de modèle

# LRECL

---

## Mot-clé optionnel

Permet de définir la longueur de l'enregistrement d'un nouveau fichier

## Syntaxe

LRECL=octets

octets : longueur en octets, d'un enregistrement de longueur fixe, ou longueur maximum d'un enregistrement de longueur variable (de 1 à 32760 pour les fichiers non VSAM, et de 1 à 32761 pour les VSAM KSDS, ESDS ou RRDS)

LRECL=nnnnnK : longueur en kilo-octets d'enregistrements "spanned" dans les bandes ISO/ANSI/FIPS version 3

LRECL=X : pour QSAM, enregistrement 32760



# MGMTCLAS

---

## **Mot-clé optionnel**

utilisable uniquement avec SMS pour des fichiers gérés par SMS

Permet de spécifier une classe de gestion pour un nouveau fichier géré par SMS

La classe de gestion contrôle :

- la migration du fichier
- les copies de sécurité du fichier (nombre, fréquence)

## Syntaxe

`MGMTCLAS=nom-de-classe`

nom-de-classe : nom de classe utilisée pour gérer le fichier après son allocation

# OUTLIM

---

## Mot-clé optionnel

Permet de limiter le nombre d'enregistrements d'impression en sortie (valable uniquement avec SYSOUT).

## Syntaxe

OUTLIM=nombre

nombre : nombre de 1 à 6 chiffres spécifiant le nombre maximum d'enregistrements à écrire (de 1 à 16777215)

## Exemple

```
//SYSPRINT DD SYSOUT=*,OUTLIM=2000
```

# OUTPUT

---

## Mot-clé optionnel

Associe à une sortie SYSOUT les paramètres d'un ordre OUTPUT

## Syntaxe

OUTPUT={référence }

{(référence[,référence]...)}  
référence :

**\*.name**

**\*.stepname.name**

**\*.stepname.procstepname.name**

name : nom figurant dans le champ nom d'un ordre OUTPUT Exemple

# PROTECT

---

## Mot-clé optionnel

utilisable seulement si RACF est installé et actif (avec SMS, utiliser SECMODEL)

Permet de demander à RACF de protéger un fichier

## Syntaxe

```
PROTECT={YES}  
        {Y }
```

YES : demande à RACF de protéger le fichier

# RECFM

---

## Mot-clé optionnel

Définit le format et les caractéristiques des enregistrements d'un nouveau fichier

Sa syntaxe dépend de l'organisation

## Syntaxe avec SMS

```
RECFM={F } [A]  
      {FB } [M]  
      {FBS}  
      {FS }  
      {V }  
      {VB }  
      {VBS}  
      {VS }  
      {U }
```

## Syntaxe BDAM

```
RECFM={U }  
      {V }  
      {VS }  
      {VBS}  
      {F }
```

# RECORD

---

## Mot-clé optionnel

utilisable uniquement avec SMS

Permet de définir l'organisation d'un nouveau fichier VSAM avec SMS

## Syntaxe

```
RECORD= {KS}  
         {ES}  
         {RR}  
         {LS}
```

**KS** : spécifie un fichier VSAM KSDS

**ES** : spécifie un fichier VSAM ESDS

**RR** : spécifie un fichier VSAM RRDS

**LS** : spécifie un fichier VSAM LDS

# REFDD

---

## Mot-clé optionnel

utilisable uniquement avec SMS (sans SMS, utiliser la forme DCB=\*.ddname)

Permet de copier les attributs d'un fichier à partir de ceux d'un autre fichier défini auparavant dans le JCL

## Syntaxe

```
REFDD={*.ddname  
       {*.stepname.ddname  
       {*.stepname.procstepname.ddname}  
       {LS}}
```

# RETPD

---

## Mot-clé optionnel

Permet de préciser une période de rétention pour les nouveaux fichiers

### Syntaxe

RETPD=nnnn

nnnn : de 1 à 4 chiffres spécifiant un nombre de jours de rétention. Le système ajoute nnnn à la date courante pour définir une date d'expiration



# SPACE

---

## Mot-clé optionnel

Permet l'allocation d'espace à un nouveau fichier sur volume à accès direct

### Syntaxe pour allocation d'espace

```
SPACE=({TRK,}(pri-qty[,sec-  
qty][,dir.])[RLSE][CONTIG][ROUND])  
      ({CYL,}    [,    ][,indx] [,    ][,MXIG ]  
      ({blkng,}      [,ALX ]  
      ({reclng,}      [,    ]
```

### Syntaxe pour demande de pistes spécifiques

```
SPACE=(ABSTR,(pri-qty[,adresse[,dir.]  
              [,indx]
```

### Syntaxe pour directory

```
SPACE=(,dir.)
```

# SPACE

---

## unités

TRK : allocation en pistes

CYL : allocation en cylindres

blkng : longueur moyenne, en octets, du bloc

reclng : longueur moyenne, en octets, de l'enregistrement

## quantités

pri-qty : quantité primaire d'unités à allouer

sec-qty : quantité secondaire d'unités à allouer  
Autres paramètres

dir. : nombre de blocs de directory de PDS à allouer

index : nombre de pistes ou de cylindres d'index pour un fichier séquentiel indexé

RLSE : l'espace non utilisé est restitué à la fermeture du fichier

## SPACE

---

CONTIG : l'espace demandé doit être contigu

MXIG : l'espace alloué doit être la plus grande plage d'espace contigu disponible sur le volume et être plus grand ou égal à la quantité primaire (ce paramètre n'est valable que pour la quantité primaire)

ALX : demande d'allouer jusqu'à cinq zones d'espace contigu de taille supérieure ou égale à la quantité primaire

ROUND : quand le premier paramètre spécifie une longueur de bloc moyenne, permet d'obtenir un nombre entier de cylindres

ABSTR : demande une allocation à un emplacement spécifique sur le volume

# STORCLAS

---

Mot-clé optionnel, utilisable uniquement avec SMS pour les fichiers gérés par SMS

Permet de spécifier une classe de stockage pour un nouveau fichier géré par SMS (remplace les attributs UNIT et VOLUME des fichiers non SMS)

## Syntaxe

STORCLAS=nom-de-classe

nom-de-classe : nom de classe utilisée pour le stockage des fichiers

# SUBSYS

---

## Mot-clé optionnel

Permet de demander qu'un sous-système traite ce fichier, et peut fournir des paramètres à ce sous-système

## Syntaxe

```
SUBSYS= {nom-sous-système      }  
        {(nom-sous-système[,paramètres]...)}
```

nom-sous-système : 1 à 4 caractères  
alphanumériques ou \$, #, @. Le premier  
caractère doit être alphabétique ou \$, # ou @.  
Ce sous-système doit être disponible

paramètres : jusqu'à 254 sous-paramètres  
requis par le sous-système

# SYSOUT

---

## Mot-clé optionnel

Indique qu'un fichier est une sortie sur terminal ou imprimante, et lui fournit des paramètres comme la classe, l'affectation à un programme (writer), le type d'état

## Syntaxe

```
SYSOUT= { classe          }  
        { *                }  
        { ([classe][,nom-pgm][,nom-form.]) }  
        [,INTRDR ][,nom-code ] }
```

classe : assigne la classe de sortie pour les impressions (valeurs : A-Z, 0-9)

\* : la classe de sortie est celle spécifiée par MSGCLASS de l'ordre JOB

(,) : classe nulle, devant être codée pour utiliser le paramètre CLASS de l'ordre OUTPUT

# SYSOUT

---

nom-pgm : programme à utiliser à la place du Writer de JES

INTRDR : le fichier sysout doit être envoyé au reader interne comme flot d'entrée de JCL

nom-form. : identifie un formulaire d'impression

nom-code : identifie un ordre /\*OUTPUT de JES2 comportant les caractéristiques d'impression

# TERM

---

## Mot-clé optionnel

Permet de spécifier qu'un fichier provient, ou est à destination, d'un utilisateur TSO/E

### Syntaxe

TERM=TS

TS : indique, dans un travail d'avant-plan soumis par un utilisateur TSO/E, que le fichier provient de cet utilisateur ou est à sa destination



## Mot-clé optionnel

Identifie le jeu de caractères universel (Universal Character Set) pour imprimer un fichier sysout, ou un jeu de caractères spécifique

## Syntaxe

```
UCS= {jeu-de-caractères }  
      {(jeu-de-  
caractères[FOLD][,VERIFY])}  
      [, ]
```

jeu-de-caractères : 1 à 4 caractères représentant un jeu de caractères valide pour le modèle d'imprimante

FOLD : le jeu de caractères est chargé en mode FOLD (unité 2821)

VERIFY : demande une vérification visuelle par l'opérateur

# UNIT

---

## Mot-clé optionnel

Demande au système de placer le fichier sur une unité spécifique, un type ou un groupe d'unité, ou la même unité qu'un autre fichier

## Syntaxe

```
{UNIT=([numéro-unité][,compte-unité][,DEFER])}  
[type-unité ][,P    ]  
[nom-groupe ][,    ]
```

numéro-unité : adresse physique en 3 caractères hexadécimaux (cuu)

type d'unité : nom générique d'unité (exemple 3330, 3350, 3380, 3390)

nom-groupe : nom symbolique de groupe d'unité (exemple SYSALLDA)

compte-unité : spécifie le nombre d'unités (1 à 59) pour ce fichier

## UNIT

---

P : demande au système d'allouer le même nombre d'unités que celui demandé dans le paramètre VOLUME

DEFER : demande au système de ne monter le volume qu'au moment de l'ouverture du fichier

AFF=ddname : affinité : demande au système d'allouer différents fichiers, résidant sur différents volumes, à la même unité durant l'exécution du step

# VOLUME

---

## Mot-clé optionnel

Identifie le ou les volumes sur lesquels résidera le fichier, ainsi que des paramètres comme la rétention, etc.

## Syntaxe

```
{VOLUME}=( [PRIVATE] [ , RETAIN ] [ , num-seq ] [ , vol-compte ]  
{VOL }      [ ,      ]  
  
[SER=num-série      ]  
[SER=(num-série [ , num-série ] ...) ]  
[ , ] [REF=dsname      ]  
[REF=* .ddname      ]  
[REF=* .stepname .ddname      ]  
[REF=* .stepname .procstepname .ddname ]
```

**PRIVATE** : demande un volume privé  
(nécessite VOL=SER, ou un démontage de bande)

**RETAIN** : indique que le volume sera utilisé dans une étape ultérieure; le système ne le démonte donc pas

## VOLUME

---

num-seq : numéro de séquence de volume pour un fichier multi-volumes (1 à 255)

vol-compte : nombre maximum de volumes pour un fichier en sortie

SER= : identifie le ou les numéros de série sur lesquels le fichier réside. 1 à 6 caractères alphanumériques. 255 numéros au maximum

REF= : demande au système de prendre le numéro de série sur un autre fichier (dsname) ou un autre ordre DD

# ORDRES DD PARTICULIERS

---

## Format

```
//ddname DD mot-clé[,mot-clé]... [comment.]
```

## Liste

- JOBCAT
- JOBLIB
- STEPCAT
- STEPLIB
- SYSABEND
- SYSCHK
- SYSCHKEOV
- SYSIN
- SYSMDUMP
- SYSUDUMP

# JOBCAT

---

Définit un catalogue VSAM privé ou un catalogue ICF utilisateur pour le job

## Syntaxe

```
//JOBCAT DD  
DISP={OLD},DSNAME=catalogue[,paramètre]...[comment.]  
{SHR}
```

Ce catalogue est prioritaire pour toute recherche

Concaténation possible

1. Après JOB
2. Après JOBLIB
3. Avant EXEC

## Exemple

```
//JOBCAT DD DSNAME=A.UCAT,DISP=SHR
```

# STEPCAT

---

Identique à JOBCAT au niveau d'une étape

## Syntaxe

```
//STEPCAT DD  
DISP={OLD},DSNAME=catalogue[,paramètre]...[comment.]  
{SHR}
```

Position quelconque dans ordres DD

Remplace la JOBCAT dans cette étape

## Exemple :

```
//STEPCAT DD DSN=A.UCAT,DISP=SHR
```

En travail normal, l'usage des ordres JOBCAT et STEPCAT est fortement déconseillé.

Avec **SMS**, sauf autorisation spéciale, la rencontre d'un de ces ordres provoque l'**arrêt du traitement**.



# JOBLIB

---

Indique la bibliothèque des load modules utilisés dans le JOB

## Syntaxe

```
//JOBLIB DD paramètre[,paramètre]... [comment.]
```

- Disponible pour toutes les étapes de travail
- Derrière l'ordre JOB
- Ne peut figurer dans une procédure
- Concaténation possible
- Attention au DISP

## Exemple

```
//JOBLIB DD DSN=jjjjj,DISP=SHR
```

# STEPLIB

---

Indique la bibliothèque du load module utilisé dans l'étape

## Syntaxe

```
//STEPLIB DD paramètre[,paramètre]... [comment.]
```

- Disponible pour l'étape uniquement
- Peut figurer dans une procédure
- Concaténation possible
- JOBLIB ignorée

## Exemple

```
//STEPLIB DD DSN=sssss,DISP=SHR
```

# SYSABEND SYSUDUMP SYSMDUMP

---

Ces ddnames définissent le fichier dans lequel sera écrit le dump en cas de fin anormale

SYSABEND et SYSUDUMP: Dump formaté du programme en cours (contenu variable suivant site)

SYSMDUMP : Dump non formaté du noyau système et du programme

Exemple :

```
//SYSABEND DD SYSOUT=*
```

# LES PROCÉDURES

# GÉNÉRALITÉS

---

Une procédure est un **JCL** utilisé fréquemment et pouvant être appelé par d'autres JCLs

Deux formes

1. Procédure **cataloguée**
2. Procédure **in-stream** en ligne (inclus dans le JCL)

Une procédure est **appelée par un ordre EXEC**

Une procédure est **modifiable** et **paramétrable**

# PROCÉDURE "IN STREAM"

---

Placée dans le jeu d'ordres en entrée

Peut comporter les ordres :

SET, EXEC, DD, OUTPUT JCL, INCLUDE IF/THEN/ELSE/ENDIF, CNTL, ENDCNTL
---

**Commence** par un ordre **PROC**

**Fini** par un ordre **PEND**

Ne peut pas comporter de données (DD DATA ou DD \*)

15 au maximum par job

Une procédure en ligne ne peut pas contenir une autre procédure en ligne (pas d'imbrication)

# PROCÉDURE CATALOGUÉE

---

Placée dans une bibliothèque (PDS)

- système (exemple : **SYS1.PROCLIB**)
- publique
- privée

Le nom de la procédure est **celui du membre** où elle se trouve

Peut comporter les ordres :

SET, EXEC, DD, OUTPUT JCL, INCLUDE IF/THEN/ELSE/ENDIF, CNTL, ENDCNTL
---

**Peut commencer** par un ordre **PROC**

**Peut finir** par un ordre **PEND**

# ORDRE PROC

---

L'ordre PROC marque le début d'une procédure

Il est obligatoire pour une procédure en ligne,  
facultatif pour une procédure cataloguée

Syntaxe : pour procédure cataloguée

```
//[nom] PROC [paramètres [comment.]]  
//[nom] PROC
```

Syntaxe : pour une procédure en ligne

```
//nom PROC [paramètres [comment.]]  
//nom PROC
```

nom : obligatoire pour une procédure en ligne. 1 à 8 caractères alphanumériques ou \$, #, @. Le premier caractère doit être alphabétique ou \$, # ou @.

paramètres : permet d'affecter des valeurs par défaut aux paramètres symboliques de la procédure



## ORDRE PEND

---

L'ordre **PEND** marque la **fin d'une procédure**

Il est **obligatoire** pour une **procédure en ligne**,  
facultatif pour une procédure cataloguée

### Syntaxe

```
//[nom] PEND [comment.]
```

nom : facultatif. 1 à 8 caractères alphanumériques  
ou \$, #, @. Le premier caractère doit être  
alphabétique ou \$, # ou @.

# PARAMÈTRES SYMBOLIQUES

---

Permettent, à l'exécution, de modifier, sur un ordre JCL

- un paramètre
- un sous-paramètre
- une valeur

Syntaxe :

- un paramètre symbolique commence par le caractère & suivi du nom du paramètre (1 à 8 caractères alphanumériques ou \$, #, @, le premier caractère doit être alphabétique ou \$, # ou @).
- le nom ne doit pas être un paramètre ou un sous-paramètre de l'ordre EXEC
- nom réservé : **SYSUID** (remplacé par l'userid du job)

# PARAMÈTRES SYMBOLIQUES

---

## Règles de codification

Un paramètre symbolique peut être placé entre quotes dans les cas suivants:

- paramètre AMP de l'ordre DD
- paramètre SUBSYS de l'ordre DD
- paramètre PARM de l'ordre EXEC

Dans les autres cas, le paramètre symbolique placé entre quotes ne sera correctement résolu que s'il est immédiatement précédé d'un paramètre symbolique non placé entre quotes

## Exemple

```
//DD1 DD &A'&B',DISP=OLD
```

# PARAMÈTRES SYMBOLIQUES

---

## Règles de codification

Un point est nécessaire après un paramètre symbolique quand ce qui suit ce paramètre

- est une partie qui ne varie pas
- commence par une lettre, \$, # ou @
- est un point

Ce point appartient alors au paramètre symbolique, et n'apparaît pas si le paramètre est remplacé par une valeur quelconque ou nulle

Par contre, si un point doit suivre le paramètre symbolique, il faut coder deux points à la suite

## Exemples

DSNAME=&JOUR.DATA

donnera XYDATA, ZTDATA, etc.

DSNAME=&JOUR..DATA

donnera XY.DATA, ZT.DATA, etc.

# PARAMÈTRES SYMBOLIQUES

---

## Affectation d'une valeur

Une valeur peut être affectée à un paramètre symbolique

- dans un ordre EXEC appelant une procédure (substitution)

### Exemple

```
//STEP1 EXEC PROC=CHARGT,FICHER='CLIENT'
```

- dans un ordre PROC (valeur par défaut)

### Exemple

```
//CHARGT PROC FICHER=VIDE,BANDE=158722
```

- dans un ordre SET (affectation)

### Exemple

```
//SET1 SET FICHER=PLEIN,BANDE=246810
```

# PARAMÈTRES SYMBOLIQUES

---

## Affectation d'une valeur nulle

On affecte une valeur nulle en faisant immédiatement suivre le nom du paramètre du signe =

Les éventuelles valeurs par défaut sont annulées

Attention aux virgules pouvant provoquer des erreurs de syntaxe

## Exemples

```
UNIT=(3380,&NOMBRE,DEFER)
```

donne une valeur correcte avec NOMBRE=

```
UNIT=(3380,,DEFER)
```

Mais coder

```
VOL=SER=(&PREM&SEC)
```

et appeler par

```
//EXE1 EXEC PROC=APPEL,PREM=123,SEC=',456'
```

# APPEL D'UNE PROCÉDURE

---

On appelle une procédure en codant son nom sur un ordre EXEC, et en attribuant éventuellement une valeur aux paramètres symboliques

**//STEP1 EXEC PROC=nom-de-procédure**

**//STEPN EXEC nom-de-procédure,XYZ=222**

Ordre de recherche :

4.procédure en ligne

5.bibliothèque privée spécifiée en JCLLIB

6.bibliothèque système

# MODIFICATION D'UNE PROCÉDURE

---

Au moment de l'appel d'une procédure, on peut

- Modifier, annuler ou ajouter des paramètres à l'ordre EXEC
- Modifier, annuler ou remplacer des paramètres des ordres DD ou OUTPUT JCL
- ajouter des ordres DD ou OUTPUT JCL



# MODIFICATION D'UNE PROCÉDURE

---

## Substitution de paramètres symboliques

- les règles générales s'appliquent
- on ne peut pas modifier un nom de programme
  - param.nomstep=valeur

## Modification carte DD

- //nomstep.ddname DD paramètres nouveaux
- Les paramètres qui deviennent incompatibles sont annulés  
Exemple : DISP et SYSOUT
- Les cartes de modification doivent apparaître dans l'ordre des cartes de la procédure
- Les cartes ajoutées doivent apparaître après les cartes modifiées
- Cas de fichiers concaténés

# EXEMPLE

---

## Contenu de la procédure PLKEDT

```
//PLKEDT PROC SOUT='*',LMOD=,MBR=nullfile,  
//          NOCAL=,REUS=,UNIT=SYSDA  
//LKED    EXEC  PGM=IEWL,  
//          PARM='MAP,LIST,XREF,&NOCAL,&REUS'  
//SYSLIB   DD    DSN=SYS1.LINKLIB,DISP=SHR  
//SYSLIN   DD    DSN=&&LOADSET, DISP=(OLD,DELETE)  
//          DD    DDNAME=SYSIN  
//SYSLMOD   DD DSN=&LMOD(&MBR),DISP=SHR  
//SYSPRINT   DD SYSOUT=&SOUT  
//SYSUT1    DD SPACE=(1024,(50,20)),UNIT=&UNIT
```

# EXEMPLE

---

## Exemples d'appel de PLKEDT

### 1. Premier format d'appel

```
//STEP1 EXEC PLKEDT,LMOD=SDJ.LIB,  
//          MBR=PROG1,NOCAL=NOCALL  
//SYSIN DD *  
ENTRY PROG1  
/*
```

### 2. Deuxième format d'appel

```
//STEP1 EXEC PROC=PLKEDT,LMOD=SDJ.LIB,  
//          PARM.LKED='MAP,LIST,XREF,NOCALL'  
//SYSLIB DD  
//          DD DSN=USER.LOADLIB,DISP=SHR  
//SYSIN DD *  
ENTRY PROG1  
/*
```

## Procédure cataloguée

- XX : pas de modification

## Appel d'une procédure

- considérée XX\* carte comme commentaire
  - \*\*\* carte commentaire

## Procédure IN STREAM

- ++ pas de modification
- +/- modification
- ++\* carte considérée comme commentaire
- \*\*\* carte commentaire

# PROCÉDURES IMBRIQUÉES

---

Une procédure (en ligne ou cataloguée) peut appeler une autre procédure (jusqu'à 15 niveaux d'imbrication)

Une procédure en ligne ne peut pas contenir la définition d'une autre procédure en ligne (la séquence PROC PROC PEND PEND est invalide)

# PROCÉDURES IMBRIQUÉES

---

## Exemple :

```
//G  PROC
//CS1 EXEC PGM=PGMC
...
//PEND 3

//B  PROC
//BS1 EXEC PROC=C 1
...
//BS2 EXEC PGM=PGMB
...
//PEND

//A  PROC
//AS1 EXEC PROC=B
...
//AS2 EXEC PGM=PGMA
...
//PEND
```

## Appel:

```
//JOB1 JOB
//STEP1 EXEC PROC=A
...
//STEP2 EXEC PGM=PGM2
```

## Exécution

```
//JOB1 JOB
//CS1 EXEC PGM=PGMC
...
//BS2 EXEC PGM=PGMB
...
//AS2 EXEC PGM=PGMA
...
//STEP2 EXEC PGM=PGM2
```

# LES UTILITAIRES

# UTILITAIRES SYSTÈME

---

- **IEHINITT** : Écriture des labels standards sur bandes
- **IEHLIST** : Listage de fichiers systèmes :
  - VTOC
  - répertoire des partitionnés
  - IEHMOVE : Copie ou déplacement des données
  - IEHPROGM : Construction et maintenance



- **IEBCOPY** : Copie, compression, fusion de partitionnés en totalité ou partiellement
- **IEBDG** : Constitution de fichiers de test
- **IEBGENER** : Copie de fichier séquentiel
- **IEBUPDTE** : Mise à jour de fichiers séquentiels ou partitionnés
- **IEBCOMPR** : Comparaison d'enregistrements dans des fichiers partitionnés ou séquentiels
- **IEBISAM** : Mise de sources séquentielles indexées dans un fichier séquentiel
- **IEBPTPCH** : Impression d'enregistrements de fichier partitionné ou séquentiel
- **IEFBR14** : Allocation et désallocation de fichiers

# IEBCOPY

---

- Sauvegarde/restauration
- Copie en totalité ou partielle
- Remplacement de membres
- Compression sur place
- Rename de membres
- Fusion de partitionnés

<b>//SDJFO1</b>	<b>JOB</b>
<b>//STEP</b>	<b>EXEC PGM=IEBCOPY</b>
<b>//SYSPRINT</b>	<b>DD Sortie des messages</b>
<b>//DD1</b>	<b>DD Fichier en entrée</b>
<b>//DD2</b>	<b>DD Fichier en sortie</b>
<b>//SYSUT3/4</b>	<b>DD Fichiers d'overflow (optionnel)</b>
<b>//SYSIN</b>	<b>DD Fichier de commandes</b>

# IEBGENER

---

- Sauvegarde d'un fichier séquentiel
- Impression d'un fichier séquentiel ou partitionné
- Changement de facteur de blocage
- Changement de taille d'enregistrement logique

<b>//SDJFO1</b>	<b>JOB</b>
<b>//STEP</b>	<b>EXEC PGM=IEBGENER</b>
<b>//SYSPRINT</b>	<b>DD Sortie des messages</b>
<b>//SYSUT1</b>	<b>DD Fichier en entrée</b>
<b>//SYSUT2</b>	<b>DD Fichier en sortie</b>
	Si le DCB n'est pas précisé, ce sera le même que celui du fichier en entrée
<b>//SYSIN</b>	<b>DD Fichier de commandes</b>

# IEHLIST

---

- Liste du répertoire d'un partitionné
  - Noms de membres
  - Début du membre
  - Point d'entrée
  - Informations diverses pour les Load Modules
- Liste d'une VTOC
  - Contenu des DSCBs
  - Formatté complet
  - Formatté réduit
  - Dump

//SDJFO1	JOB
//STEP	EXEC PGM=IEHLIST
//SYSPRINT	DD Sortie des messages
//DD1	DD UNIT=3390,VOL=SER=VOL1,
//	DISP=OLD
//SYSIN	DD Fichier de contrôle

## UTILITAIRES DE TRI ET FUSION

---

Produit IBM ou non : DFSORT, CASORT ...  
mis en oeuvre par le programme SORT

Cartes obligatoires :

- SORTIN
- SORTOUT
- SYSOUT
- SYSIN
- SORTWK0n

## UTILITAIRES D'EXPLOITATION

---

DFDSS : principal gestionnaire des mouvements de données : sauvegarde, ...  
mis en oeuvre par le programme ADRDSSU

DSF : produit destiné à l'initialisation des disques, au contrôle de leur surface ...  
mis en oeuvre par le programme ICKDSF

Cartes nécessaires à cette mise en oeuvre :

- SYSPRINT
- SYSIN
- cartes décrivant les objets des traitements

Dépendent du contenu du fichier SYSIN

- commandes d'ADRDSSU :  
DUMP, RESTORE, COPY, RELEASE,  
DEFRAG , etc...  
éventuellement combinées avec IF, ELSE,  
SET ...
- commandes d'ICKDSF :  
INIT, ANALYSE, BUILDX, etc...

# **System Managed Storage**



# OBJECTIFS

---

L'objectif de ce chapitre est de présenter :

- la nécessité d'un système centralisé de gestion des mémoires externes
- la réponse DFSMS

# HISTORIQUE

---

- Groupe de travail IBM 1979
  - Groupes de travail SHARE et GUIDE
- ⇒ une solution coordonnée pour :
- amélioration des performances I/O
  - plus grande disponibilité des ressources (espace, temps)
  - meilleur partage des ressources
  - usage efficace de nouvelles unités
  - meilleure convivialité

## DEMANDES UTILISATEURS

---

- migration automatique en fonction :
  - des caractéristiques des unités
  - de l'utilisation des données
- backup automatique géré par le système
  - en fonction d'attributs définis par l'utilisateur
  - quand les données ont changé
- recovery automatique
  - avec évaluation des dommages

## DEMANDES UTILISATEURS

---

- transfert facile vers les nouvelles unités
    - avec coexistence possible entre unités
  - régulation et contrôle dynamiques
    - gestion automatique et contrôle humain
  - sécurité et audit des accès
- interface convivial avec valeurs par défaut

### Storage Management Subsystem

- Storage group
  - ensemble de volumes gérés par DFSMS
- Storage class
  - niveau de service requis (performances, disponibilité)
- Management class
  - contrôle des backups et des migrations
- Data class
  - *grille* de définition

## COMPOSANTS DE DFSMS

---

- **Data Facility Product**
  - gestion des données actives
- **Data Facility Hierarchical Storage Manager**
  - gestion des données inactives
- **Data Facility Data Set Services**
  - utilitaire de copie
- **Resource Access Control Facility**
  - utilitaire de contrôle d'accès
- **Interactive System Productivity Facility**
  - interface applications interactives
- **Interactive Storage Management Facility**
  - interface utilisateur

## STORAGE GROUP

---

- Défini par l'administrateur des données
- Groupe de volumes
- Modifiable dynamiquement (sans IPL)
- Invisible pour l'utilisateur
- Géré par :
  - ISMF
  - DFDSS
  - DFHSM

# STORAGE CLASS

---

- 
- Performances
  - temps de réponse direct
  - temps de réponse séquentiel
- Disponibilité
  - dual-copy du 3990-3
- lien avec SG via **Automatic Class Selection** routines

Un fichier géré par SMS **appartient** à une SC

Un fichier non géré par SMS **n'appartient** à aucune SC



# MANAGEMENT CLASS

---

- Migration
- Backup
- Retention
- Libération de l'espace inutilisé

DFHSM est l'outil de mise en oeuvre

## DATA CLASS

---

- Concerne les fichiers gérés et non-gérés par SMS
- Valeurs par défaut des attributs
  - organisation physique
  - taille
  - etc.
- Modifiables par l'utilisateur

**NOTE : les ACS routines utilisent les noms des fichiers pour leur attribuer des classes, qui éventuellement priment sur celle précisées par l'utilisateur.**

### **Nouveaux paramètres :**

- DATACLAS
- STORCLAS
- MGMTCLAS
- RECORG : type de fichier VSAM
- KEYOFF : pour un KSDS, position de la clé
- AVGREC : pour préciser le SPACE

### **Sous-paramètres devenus paramètres :**

- LRECL, RECFM, KEYLEN
- RETPD, EXPDT

## EXEMPLE de JCL avec SMS

---

### Créer un fichier séquentiel :

```
//  
//      //FCLIENT DD DSN=SDJFO1.GROUP1.SEQ,  
//              DISP=(NEW,CATLG,DELETE),  
//              DATACLAS=fiqs,STORCLAS=BASE
```

### Créer un fichier VSAM KSDS :

```
//  
//CLIENTK      DD DSN=SDJFO1.GROUP1.KSDS  
//              DISP=(NEW,CATLG,DELETE),  
//              DATACLAS=SDJCL,STORCLAS=SDJ,  
//              RECORG=KS,KEYOFF=1,KEYLEN=5  
//
```

# APPORTS MVS

# CONDITIONNEMENTS

---

- **IF - THEN - ELSE**
- Conditions
  - nom-étape.**RC**=valeur
  - nom-étape.**ABEND**=TRUE/FALSE
  - nom-étape.**RUN**=TRUE/FALSE
- Possibilités : OR, AND, imbrications

# CONDITIONNEMENTS

---

## Exemple

```
//SDJFO1      JOB ...
//STEP1       EXEC PGM=pgm1
//DD1         DD fichier.A
//DD2         DD fichier.B
//...
//STEP2       EXEC PGM=pgm2
//DD3         DD fichier.C
//...
//test        IF etp1.RC=0 AND etp2.RC=0 THEN
//STEP3       EXEC PGM=pgm3
//DD4         DD fichier.D
//...
//suite       ELSE
//STEP4       EXEC PGM=pgm4
//DD5         DD .SDJ.CLIENT
//...
//fintest     ENDIF
```

## Ordre DD

---

Paramètres pour impression :

- **SEGMENT**=n
- **SPIN**=UNALLOC/NO



# NOUVEAUX ORDRES

---

- **SET**
  - Affectation de valeur à un paramètre
- **JCLLIB**
  - Choix de bibliothèques de procédures
- **INCLUDE**
  - Inclusion d'ordres JCL

# **SDJ INFORMATIQUE**

## **EXEMPLES DE JCL**

```

//*****
//*                EXEMPLE1                DE                JCL
//*****
//*
//SDJJOB  JOB  NOTIFY=&SYSUID,TIME=(,10),MSGLEVEL=(1,1),
//          MSGCLASS=H,CLASS=A,TYPRUN=SCAN
//*
//*****
//*                SUPPRESSION D'UN FICHIER SEQUENTIEL
//*****
//*
//DESTRUCT EXEC PGM=IEFBR14
//VIEUFIC  DD DSN=SDJ.EXO.PDF.SEQ1,
//          DISP=(OLD,DELETE,DELETE)
//*
//*****
//*                CREATION D'UN FICHIER SEQUENTIEL
//*****
//*
//CREATION EXEC PGM=IEFBR14
//NOUVFIC  DD DSN=SDJ.EXO.PDF.SEQ1,
//          DCB=(LRECL=80,RECFM=FB,DSORG=PS),
//          DISP=(NEW,CATLG,DELETE)
//*
//*****
//*                COPIE D'UN FICHIER SEQUENTIEL VERS UN AUTRE FICHIER
//*****
//*                SEQUENTIEL
//*****
//*
//COPIE    EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SDJ.EXO.PDF.SEQ,DISP=SHR
//SYSUT2   DD DSN=SDJ.EXO.PDF.SEQ1,DISP=SHR
//SYSIN    DD DUMMY
//*

```

```

//*****
//*          TRI  D'UN  FICHIER  SEQUENTIEL
//*****
//*
//TRI          EXEC PGM=SORT
//SYSOUT      DD SYSOUT=*
//SORTIN      DD *
ZERMEKRMKEZKRMZKRLZERKOZEIROPEZIRERER
EREZRRRRZMELKRME
ERFSDFGGFG
BVBNBNBV
YJUIKUIOL
QDSD
TYU-RYU-(Ô_
UYIIUYIYI
FDGGDGFDFDGFDF
IUOUIOUIOUIOUIO
RETTTRET
MPOOUILHJB,K
GBF
HBGFH
TFH
SDF
FQDQ
SCFDZEFEZTERT
/*
//SORTOUT     DD DSN=SDJ.EXO.PDF.SEQ1,DISP=SHR
//SYSIN       DD *
              SORT FIELDS=(1,80,A)

/*
//*****
//*          REDIRECTION (COPIE)  DU FICHIER TRIÉ VERS LA SORTIE STANDARD
//*****
//*
//SORTIE      EXEC PGM=IEBGENER
//SYSPRINT    DD    SYSOUT=*
//SYSUT1      DD    DSN=SDJ.EXO.PDF.SEQ1,DISP=SHR
//SYSUT2      DD    SYSOUT=*
//SYSIN       DD    DUMMY

```

```

//*****
//*                EXEMPLE2                DE                JCL
//*****
//SDJJOB  JOB  NOTIFY=&SYSUID,TIME=(,10),MSGLEVEL=(1,1), ,
//          MSGCLASS=A,CLASS=A,RESTART=S10
//*
//*****
//*  CRÉATION D'UN FICHIER SÉQUENTIEL VIA L'UTILITAIRE IEFBR14
//*****
//S01      EXEC PGM=IEFBR14
//FIC1      DD DSN=SDJ.FORM2005.UTIL,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=23200,DSORG=PS)
//SYSPRINT DD SYSOUT=*
//*
//*
//*****
//*  REDIRECTION DE L'ENTRÉE STANDARD SYSUT1 DE L'UTILITAIRE
//*  IEBGENER
//*  VERS LA SORTIE STANDARD SYSUT2 ELLE MÊME REDIRIGÉE VERS L'ÉCRAN.
//*****
//S02      EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SDJ.FORM2005.SEQ,DISP=SHR
//SYSUT2   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//*
//*****
//*  COPY DE L'ENTRÉE STANDARD SYSUT1 SUR UNE SORTIE STANDARD SYSUT2
//*  VIA L'UTILITAIRE IEBGENER
//*****
//S03      EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SDJ.FORM2005.SEQ,DISP=SHR
//SYSUT2   DD DSN=SDJ.FORM2005.UTIL,DISP=SHR
//SYSIN    DD DUMMY
//*

```

```

//*****
//*  SUPPRESSION D'UN FICHIER VIA L'UTILITAIRE IEFBR14
//*****
//S04      EXEC PGM=IEFBR14
//SUPR      DD DSN=SDJ.FORM2005.SEQ,
//          DISP=(OLD,DELETE,DELETE)
//*
//*
//*****
//*  EDITION D'UN MEMBRE DE PDS VIA L'UTILITAIRE IEBGENER,
//  EN SORTIE STANDARD SYSOUT=*
//*****
//S05      EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=COBOL.JCL.SOURCE(EXOJCL1),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*
//*
//*****
//*  CRÉATION D'UN FICHIER VSAM VIA L'UTILITAIRE APPROPRIÉ IDCAMS
//*  VEILLEZ A L'EFFACER AVANT LA CRÉATION.
//*****
//S06      EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
//          DELETE SDJ.FORM2005.SALARIES
//          DEFINE CLUSTER(NAME(SDJ.FORM2005.SALARIES) -
//              KEYS(6 28) -
//              RECSZ(80 80)) -
//              DATA(NAME(SDJ.FORM2005.SALARIES.DATA)) -
//              INDEX(NAME(SDJ.FORM2005.SALARIES.INDEX))
//*
//*

```

```

//*****
//*      TRI DU FICHIER SDJ. ... .UTIL VIA L'UTILITAIRE DFSORT
//*      ATTENTION SORT UTILISE UNE SYSOUT ET NON UNE SYSPRINT.
//*****
//S07      EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=SDJ.FORM2005.UTIL,DISP=SHR
//SORTOUT  DD DSN=SDJ.FORM2005.UTIL,DISP=SHR
//SYSIN    DD *
          SORT FIELDS=(2,4,CH,A,22,2,CH,D)
/*
/*
/*
//*****
//*      EDITION D'UN FICHIER DE COMPTE RENDU SUR UN MEMBRE DES PDS
//*      ( C'EST EN FAIT UNE REDIRECTION )
//*      ICI LA CARTE EST SYSOUT CAR ON A L'UTILITAIRE SORT
//*****
//S08      EXEC PGM=SORT
//SYSOUT   DD DSN=COBOL.JCL.SOURCE(MEMBRE),DISP=SHR
//SORTIN   DD DSN=SDJ.FORM2005.UTIL,DISP=SHR
//SORTOUT  DD DSN=SDJ.FORM9806.UTIL,DISP=SHR
//SYSIN DD *
          SORT FIELDS=(2,4,CH,A,22,2,CH,D)
/*
//*****
//*      COPIE VIA IDCAMS DU CONTENU D'UN FICHIER1 (NON VSAM)
//*      EN ENTREE SUR UN AUTRE FICHIER2 (VSAM) EN SORTIE.
//*****
//S09      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
          REPRO      INFILE(FENTREE) -
                   OUTFILE(FSORTIE)
/*
//FENTREE  DD DSN=SDJ.FORM2005.UTIL,DISP=SHR
//FSORTIE  DD DSN=SDJ.FORM2005.SALARIES,DISP=SHR

```

```

//*****
//*      EDITION DU CONTENU DU FICHIER VSAM VIA L'UTILITAIRE IDCAMS,
//*      SUR LA SORTIE STANDARD SYSOUT.
//*****
//*
//S10      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//EDITION  DD DSN=SDJ.FORM2005.SALARIES,DISP=SHR
//SYSIN    DD *
          PRINT INFILE(EDITION) CHAR

```