Atos Origin

MAINFRAMES IBM

Introdução

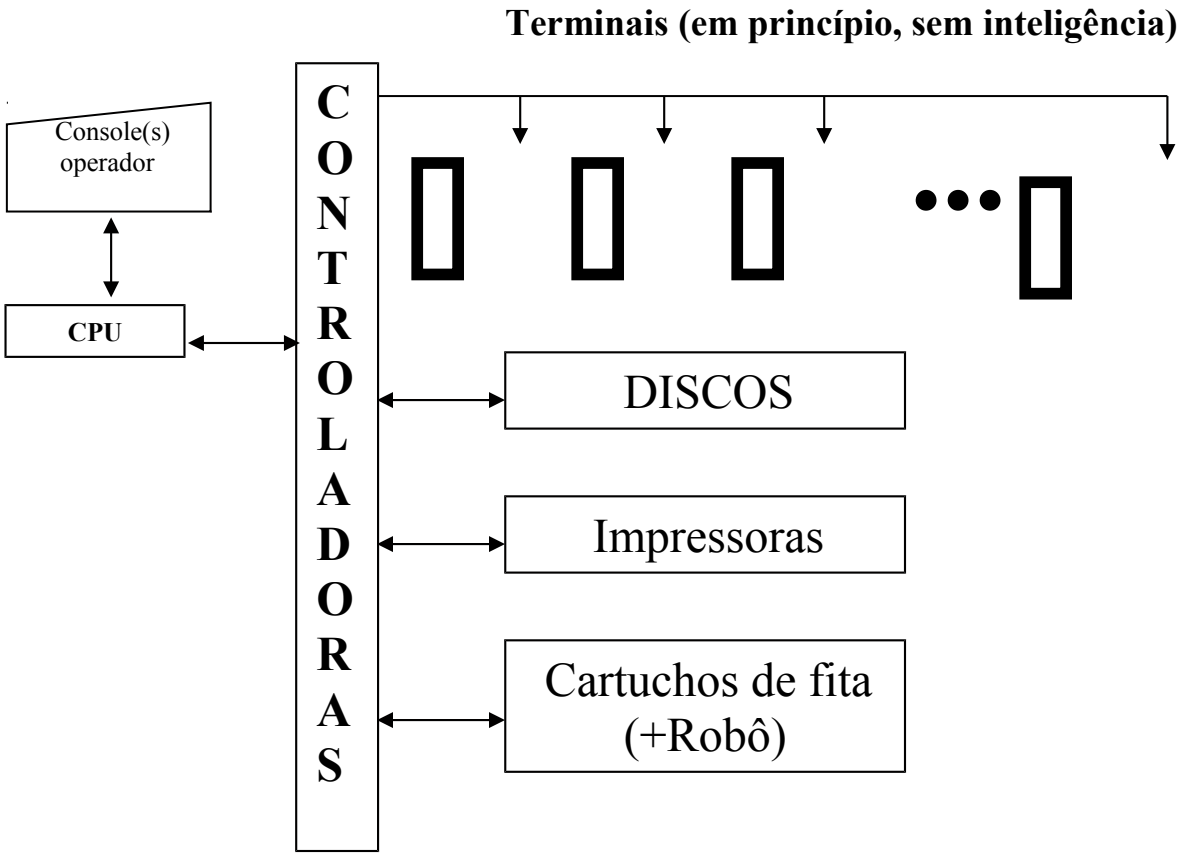
JCL

Utilitários

Set/2004

1. Mainframes IBM - Hardware

1.1 Geral



1.2 Processadores

Processadores : S/390 Generation 5 and Generation 6

1 Frame (mín.)
peso : 612 kg (1346 lbs)
área : 1m^2 (10.2 pés^2)
área para manutenção : 2.5 m^2 (27.4 pés^2)
2 Frame (máx.)
peso : 938 kg (2057 lbs)
área : 1.8m^2 (19.7 pés^2)
área para manutenção : 4.8 m^2 (51.9 pés^2)

Refrigeração : ar; comparativamente aos 9021-9x2 (refrigerados a água), consomem menos energia e ocupam menos espaço.
CoProcessador - Criptográfico : 2 - standard ; PCI Criptográfico : até 8
Memória : Mínima : 1 GB; Máxima : 32 GB
Sistemas Operacionais : OS/390, MVS, VM e TPF

Processadores : S/390 Multiprise 3000

Frame básico (máx.)
peso : 236 kg (520 lbs)
área : 0.54m^2 (5.69 pés^2)
área manut. : 4.1m^2 (43.6 pés^2)

Frame expansão (máx.)
peso : 232 kg (510 lbs)
área : 0.54m^2 (5.69 pés^2)
área manut : 4.1m^2 (43.6 pés^2)

Canais
Máximo : 56
Velocidade : 17 B=MB/seg (Escon)
Conexões : RS-232, X.21, V.35; Adaptador PCI Ethernet , Adaptador PCI Token Ring

Memória : Mínima : 1 GB; Máxima : 4 GB
Sistemas Operacionais : OS/390, MVS, VM e VSE
Disco Interno : 0 a 792 GB

1.3 Discos

(Enterprise storage server)

15K rpm
Capacidade usável (RAID-5) : 420 GB a 22.4 TB
Capacidade total física : 582 GB a 27.9 TB
Cache size : 8, 16, 24, 32 ou 64GB

Características físicas
Dimensões : 75.25" alt x 54.50" larg x 35.75" prof *
(1913 mm x 1383 mm x 909 mm) ;
Peso : 2200 lb. (998kg)

Ambiente Operacional
Temperatura : 60 a 90 F (16 a 32 C)
Umidade relativa do ar : 20 a 80%
Dissipação calor : 16,000 BTU/h
Consumo : 6.4 kVA

Sistemas suportados
S/390 and zSeries; AS/400; iSeries ; Compaq ; Data General; DEC; Hewlett-Packard (9000 and 8000); Intel™-based PC servers; (Novell NetWare; Linux ; Windows NT; Windows 2000, Linux); RS/6000®; RS/6000 SP; Sun™;

1.4 Fitas (cartuchos)

IBM TotalStorage™ Enterprise Tape System 3590 Model E11s /
IBM TotalStorage**™** Enterprise Tape Controller Model A60

Drives : máximo 12

Canais ESCON :
Velocidade : 17MB/seg
Distância : 3 km

Canais FICON :
Velocidade : 100MB/seg
Distância : 20km

Cartuchos :
Compressão LZ1 :padrão
Qtdade trilhas : 256
Capacidade cartucho (nativa) : 20 GB
Capacidade cartucho (comprimido) : 60 GB
Capacidade cartucho estendido (nativa) : 40 GB
Capacidade cartucho estendido (comprimido) : 120 GB
Capacidade total, comprimido : 1.2 TB
Velocidade : 14 MB/sec

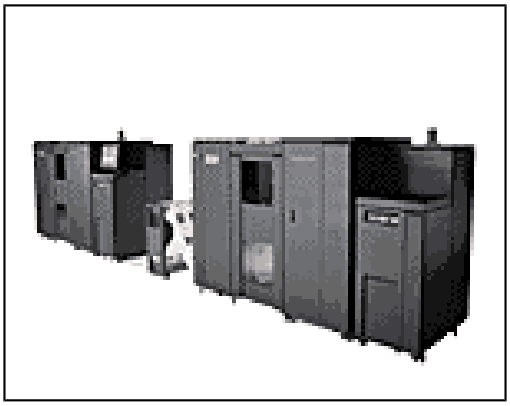
1.5 Impressoras

Impressoras (desde $49,995)

- High speed production printers improving productivity for distributed print, data center, and inplant/reprographics printing environments.
- Seamless, integrated printing solutions for IBM eserver and non-IBM printing platforms
- Output print speeds from up to 70 impressions per minute to 110 ipm
- Large monthly print volumes from up to 600,000 imp to 2M ipm
- Flexible paper handling including large input and output media capacities with output finishing support e.g. collation, stapling, insertion, and booklet making

Impressoras (desde $200,000)

- Ideal for high-volume statements printing and on-demand publishing.
- Cutsheet production print quality at up to four times the speed
- Industry-leading 600-dpi image quality
- Speeds from 172 ipm up to 1002 ipm



Impressoras (desde $520,000**)**

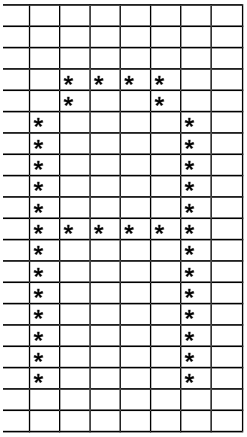
- Excellent for on-demand and personalized printing.
- Off-set quality color printing at high speed and digital direct-to-paper capability
- Models with advanced technology that automates production workflow, increasing efficiency by minimizing the potential for human error

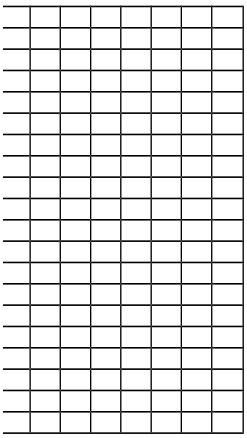
1.6 Terminais (3270)

Comparativo teclado terminal X teclado micro

PC	Ítem	Terminal 33270
Ex.: se 640 horiz x 480 vert = 307.200	Qtd Pixels (*picture elements*) Endereçáveis / Configuráveis	80 linhas x 24 colunas = 1920 (*)
Qualquer conteúdo (cor, intensidade, etc)	Cada posição endereçável Pode receber	Caracteres do padrão EBCDIC (190 dos 256 possíveis)
Sim	Processador	Não
Sim	Software	Não
Sim / grande	Memória	*Buffer* de 1920 bytes

(*) Matriz de 8 x 20 onde os caracteres são formatados

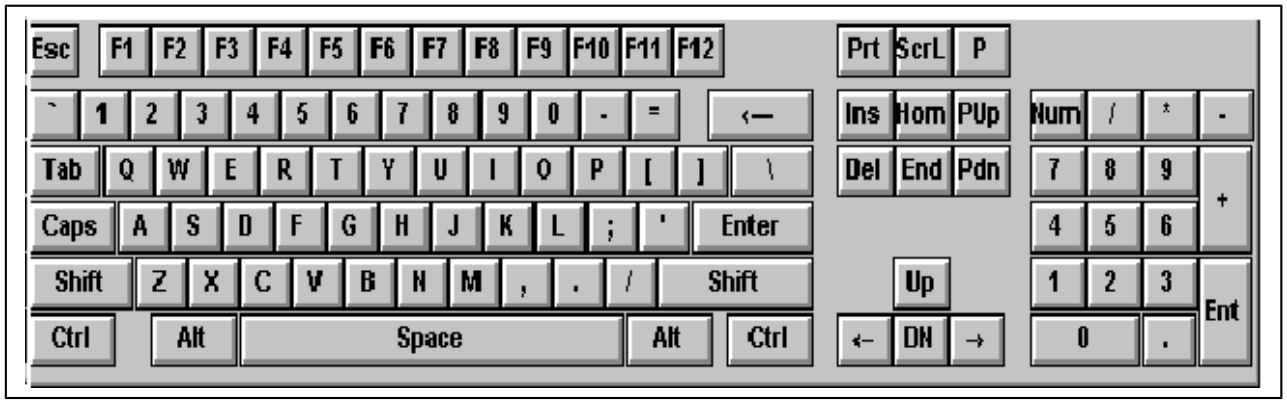




Obs.: A exata (ou não) utilização dos recursos 3270 no PC depende do emulador e do teclado utilizado.

Teclado 3270

Teclado PC



1.7 Representação de dados na memória

Tipo texto (caráter) : padrão EBCDIC
(Extended Binary-Coded Decimal Interchange Code)

Tipo numérico : padrões :
- **Decimal (Zonado ou Compactado)**
- **Binário (Ponto Fixo ou Ponto Flutuante)**

1.7.1 Tipo texto (caráter) : padrão EBCDIC

Alguns caracteres :

```
A = C1h = X'C1' = 1100 0001 = 193d (em Ascii = 41h = 65d)
B = C2h = X'C2' = 1100 0010 = 194d (em Ascii = 42h = 66d)
0 = F0h = X'F0' = 1111 0000 = 240d (em Ascii = 30h = 48d)
1 = F1h = X'F1' = 1111 0001 = 241d (em Ascii = 31h = 49d)
$ = 5Bh = X'5B' = 0101 1011 = 091d (em Ascii = 24h = 36d)
```

Exemplo :

```
'ICH LIEBE' = X'C9C3C840D3C9C5C2C5'
```

1.7.2 Tipo numérico decimal Zonado :
- cada algarismo do número ocupa um byte;
- configuração equivalente ao EBCDIC de cada algarismo, exceto no último byte à direita :
- se positivo, sinal (configuração hexa) C ou F
  (em geral : dados inputados = F
  após op. Aritmét. = C)
- se negativo, sinal (configuração hexa) D

- Exemplos
  +187 em 4 bytes = X'F0F1F8F7' ou X'F0F1F8C7'
  cuidado na análise : X'C7' também equivale ao carater 'G' !!!
  -187 em 4 bytes = X'F0F1F8D7'
  cuidado na análise : X'D7' também equivale ao carater 'P' !!!

1.7.3 Tipo numérico Compactado :

- cada algarismo do número ocupa meio byte;
- configuração equivalente ao meio byte à direita da configuração EBCDIC de cada algarismo, exceto no último meio byte à direita:
- se positivo, sinal (configuração hexa) C ou F
- se negativo, sinal (configuração hexa) D

- Exemplos

+187 em 4 bytes = X'0000187C' ou X' 0000187F'
cuidado na análise : X'7C' é a configuração do caracter arroba ...

-187 em 4 bytes = X' 0000187D'
cuidado na análise : X'7D' é a configuração do caracter apóstrofe ...

1.7.4 Tipo numérico Binário Ponto Fixo

- em geral ocupa tamanho fixo (2, 4 ou 8 bytes)
- número (representado em binário) ocupa todos os bits do campo MENOS o primeiro bit à esquerda, reservado para o sinal
- se positivo, bit em 0
- se negativo, bit em 1. Mas... o número binário é o complemento para a "próxima potência" de 2

- Exemplos

+187 em 4 bytes
= X'000000BB'
= B'0000 0000 0000 0000 0000 0000 1011 1011'
cuidado na análise : algumas configurações podem equivaler a outros caracteres

-187 em 4 bytes
= X' FFFFFF45' (X'100' - X'BB' = X'45')
= B' 1111 1111 1111 1111 1111 1111 0100 0101'
cuidado na análise : algumas configurações podem equivaler a outros caracteres

1.7.5 Tipo numérico Binário Ponto Flutuante :

- em geral ocupa tamanho fixo (4 ou 8 bytes)
- número (representado em binário) ocupa 4 ou 8 ou 16 bytes

- Ex. em 4 bytes :

Conteúdo (expresso em decimal)	Tamanho	Conteúdo efetivo (expresso em hexadecimal)	Conteúdo efetivo (expresso em binário)
+187	4	42BB0000	0100 0010 1011 1011 0000 0000 0000 0000
+18.7	4	4212B333	0100 0010 0001 0010 1011 0011 0011 0011
+1.87	4	411DEB85	0100 0001 0001 1101 1110 1011 1000 0101
+0.187	4	402FDF3B	0100 0000 0010 1111 1101 1111 0011 1011
+0.0187	4	3F4C985F	0011 1111 0100 1100 1001 1000 0101 1111
+0.00187	4	3E7A8D65	0011 1110 0111 1010 1000 1101 0110 0101
+0.000187	4	3DC4156E	0011 1101 1100 0100 0001 0101 0110 1110
-187	4	C2BB0000	1100 0010 1011 1011 0000 0000 0000 0000
-18.7	4	C212B333	1100 0010 0001 0010 1011 0011 0011 0011
-1.87	4	C11DEB85	1100 0001 0001 1101 1110 1011 1000 0101
-0.187	4	C02FDF3B	1100 0000 0010 1111 1101 1111 0011 1011
-0.0187	4	BF4C985F	1011 1111 0100 1100 1001 1000 0101 1111
-0.00187	4	BE7A8D65	1011 1110 0111 1010 1000 1101 0110 0101
-0.000187	4	BDC4156E	1011 1101 1100 0100 0001 0101 0110 1110

+187 = X' 4 2 B B 0 0 0 0'
= B'0100 0010 1011 1011 0000 0000 0000 0000'

- 187 = X' C 2 B B 0 0 0 0'
= B'1100 0010 1011 1011 0000 0000 0000 0000'

- Ex. em 8 bytes :

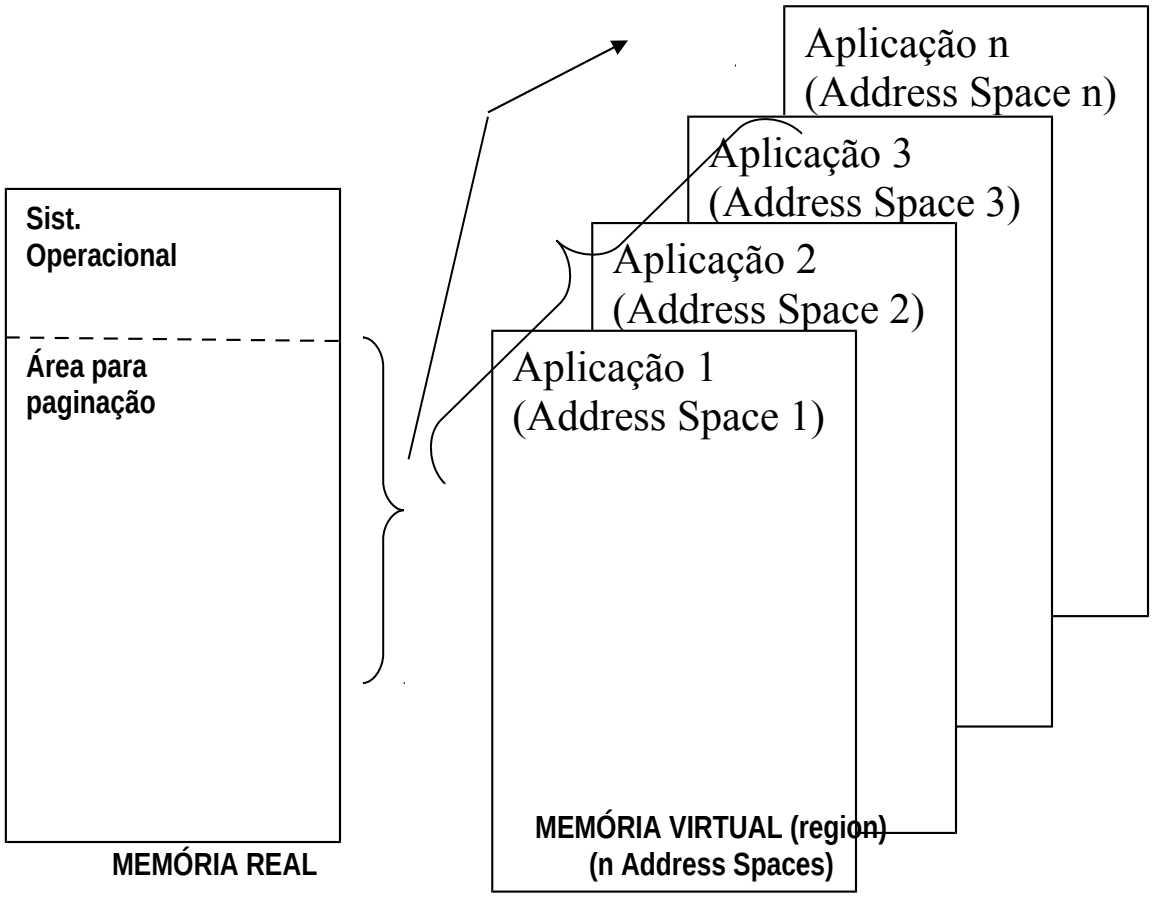
```
42BB000000000000     +187.0
C2BB000000000000     -187.0
4212B33333333333     +18.7
C212B33333333333     -18.7
411DEB851EB851EC     +1.87
C11DEB851EB851EC     -1.87
402FDF3B645A1CAC     +0.187
C02FDF3B645A1CAC     -0.187
3F4C985F06F69446     +0.0187
BF4C985F06F69446     -0.0187
3E7A8D64D7F0ED3E     +0.00187
BE7A8D64D7F0ED3E     -0.00187
3DC4156E264E4862     +0.000187
BDC4156E264E4862     -0.000187
```

2. Mainframes IBM - Software

2.1 Geral

Principais Sistemas Operacionais : OS/390 (MVS) , (DOS/)VSE, VM

Gerenciamento de Memória



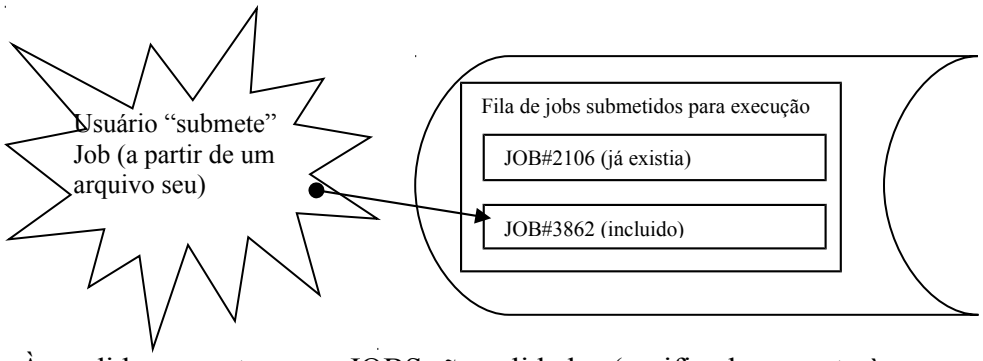
Softwares :

CICS	Gerenciador de aplicações on-line
CSP	Desenvolvimento aplicativos on-line
DB2	Gerenciador de base de dados relacional (usa SQL)
EASYTRIEVE	Linguagem para desenvolvimento que pode ser usada de forma interpretada
ENDEVOR	Controle de objetos fonte
FILE-AID	Manipulação on-line de arquivos
IDCAMS	Manipulação batch de arquivos, principalmente VSAM
IEFBR14	No-operation; não efetua nenhuma função; usado para que se possa efetuar funções de alocação e desalocação do sistema operacional (especificadas via JCL)
JCL	(Job Control Language) Linguagem para especificação de “conjuntos” de serviços a executar
RACF	Controle de acesso aos recursos
ROSCOE	(Remote Operating System Conversation Operating Environment) Funções utilitárias de interface com S.O.
TSO	(Time Sharing Option) Interface para processsamento On-line; funções utilitárias de interface com S.O.
VISUAL AGE	Desenvolvimento de aplicativos on-line

2.2 Entrada, Processamento e Saída de serviços

- Serviços submetidos são enfileirados em spool de entrada
- Spool de serviços (jobs) de entrada são selecionados para execução por *initiators* com base na classe de seleção (parâmetro CLASS do statement JOB)
- Durante a execução, "impressões" dos jobs são direcionadas para o spool de saída do sistema
- Uma vez no spool de saída do sistema, as "impressões" podem ser vistas e/ou efetivamente impressas
- A opção de como será tratado (disponibilizado para visualização ou impresso efetivamente) o "relatório" é baseada na sua classe (parâmetro SYSOUT=*classe* do statement DD)

2.2.1 Entrada de JOBS



À medida que entram, os JOBS são validados (verificados quanto à sua correção) pela Reader / Interpreter.

Se não houver erro, o JOB é incluido na fila para aguardar seleção para ser executado.

Se houver erro, eles são exibidos através de uma sysout gerada para indicar os erros de JCL. Ex.: se for submetido este JOB

```
sub
> APPLID(ABNROSCD)   USER(OB3,TORI243)                    L PENDING
> AWS(OB3.AMSVERIF)  SCRL FULL  COLS 00001 00072                  A<TMP1>2
>       <...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
 ****** ******************************** T O P ********************************
000001 //EPC##ARS JOB ('VERIFY'),'VERIFY',CLASS=1,MSGCLASS=X
000002 //*-------------------------------------------------------------
000003 //STEP1    EXEC PGN=IDCAMS
000004 //ARQUIVO  DD  DSN=DSVAABPS.FIN.A999,
000005 //             DISP=(OLD,KEEP)
000006 //SYSIN DD *
000007  VERIFY FILE(ARQUIVO)
000008 //SYSPRINT DD SYSOUT=*
 ****** ***************************** B O T T O M ******************************
```

Observe que o statement EXEC está errado (PGN ao invés de PGM).
Será gerado em spool um conjunto de 3 sysouts referentes ao JOB :

```
> APPLID(ABNROSCD)   USER(OB3,TORI243)                    J,L PENDING
> STA(EPC##ARS,283)  SCRL FULL  COLS 00001 00079
>....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
                         ROSCOE ALTER/STATUS PROCESSOR

 JOB NAME     NO    FILE      LINE    PAGE/   LINE  FIND LIMIT  I/O COUNT
 EPC##ARS    283       1         1      1        1      64000           6


 (1)         (2) (3)              (4)      (5)
  A  FILE STA C  DEST       LINES FORM     CPY NOTES
  _      1 NOP X LOCAL           11 STD      1   JES2.JESMSGLG
  _      2 NOP X LOCAL            6 STD      1   JES2.JESJCL
  _      3 NOP X LOCAL            2 STD      1   JES2.JESYSMSG
 ========================== END OF OUTPUT FILES ==========================
```

JES2.JESMSGLG

É o registro dos eventos ocorridos no processo de leitura / interpretação pelo sistema.

```
J E S 2  J O B  L O G  --  S Y S T E M  A B 7 3  -- N O D E  A B N M V S 1

08.18.58 JOB00322 ---- FRIDAY,    16 APR 2004 ----
08.18.58 JOB00322  IRR010I  USERID TORI243  IS ASSIGNED TO THIS JOB.
08.18.58 JOB00322  IEFC452I EPC##ARS - JOB NOT RUN - JCL ERROR  045
------ JES2 JOB STATISTICS ------
           8 CARDS READ
          21 SYSOUT PRINT RECORDS
           0 SYSOUT PUNCH RECORDS
           1 SYSOUT SPOOL KBYTES
        0.00 MINUTES EXECUTION TIME
```

JES2.JESJCL

É o registro de COMO o sistema entendeu (“enxergou”) o JCL recebido. Observe que a numeração que ele atribui não corresponde á quantidade de linhas e sim à quantidade de statements.

```
1 //EPC##ARS JOB ('VERIFY'),'VERIFY',CLASS=1,MSGCLASS=X
  //*-------------------------------------------------------
2 //STEP1    EXEC PGN=IDCAMS
3 //ARQUIVO  DD  DSN=DSVAABPS.FIN.A999,
  //             DISP=(OLD,KEEP)
4 //SYSIN DD *
5 //SYSPRINT DD SYSOUT=*
```

JES2.JESYSMSG

São as mensagens relativas aos eventos ocorridos no processamento do JOB.
Neste caso, é composta da indicação do(s) erro(s) detectados.

```
STMT NO. MESSAGE
    2 IEFC032I REQUIRED PARAMETER PROC OR PGM MUST PRECEDE ALL OTHER PARAMETERS ON THE EXEC STATEMENT
```

No caso de haver erro de JCL, o JOB não é incluido na fila de JOBS para execução.

Se o JOB não tivesse dado erro de JCL, ele teria :

- sido incluido na fila de jobs para execução
- sido executado
- sido incluido na fila de saída (suas sysouts)

Dados editados submetidos :

```
sub
> APPLID(ABNROSCD)   USER(OB3,TORI243)                    L PENDING
> AWS(OB3.AMSVERIF)  SCRL FULL  COLS 00001 00072                  A<TMP1>2
>       <...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
 ...... ================================ T O P ================================
000001 //EPC##ARS JOB ('VERIFY'),'VERIFY',CLASS=1,MSGCLASS=X
000002 //*-------------------------------------------------------
000003 //STEP1    EXEC PGM=IDCAMS
000004 //ARQUIVO  DD DSN=DSVAABPS.FIN.A999,
000005 //             DISP=SHR
000006 //SYSIN DD *
000007  VERIFY FILE(ARQUIVO)
000008 //SYSPRINT DD SYSOUT=*
...... ============================== B O T T O M ==============================
```

sysouts geradas :

```
> APPLID(ABNROSCD)   USER(OB3,TORI243)                    J,L PENDING
> STA(EPC##ARS,283)  SCRL FULL  COLS 00001 00079
>....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
                        ROSCOE ALTER/STATUS PROCESSOR

 JOB NAME     NO    FILE      LINE    PAGE/   LINE  FIND LIMIT  I/O COUNT
 EPC##ARS    421       1         1       1      1      196608          66


 (1)         (2) (3)              (4)      (5)
  A  FILE STA C  DEST       LINES FORM     CPY NOTES
  _      1 NOP X LOCAL           21 STD      1   JES2.JESMSGLG
  _      2 NOP X LOCAL            7 STD      1   JES2.JESJCL
  _      3 NOP X LOCAL           14 STD      1   JES2.JESYSMSG
  _      4 NOP X LOCAL            8 STD      1   STEP1.SYSPRINT
=========================== END OF OUTPUT FILES ===========================
```

JES2.JESMSGLG

```
                  J E S 2  J O B  L O G  --  S Y S T E M  A B 7 3  --  N O D E  A B N M V S 1
09.00.02 JOB00421 ---- FRIDAY,    16 APR 2004 ----
09.00.02 JOB00421  IRR010I  USERID TORI243  IS ASSIGNED TO THIS JOB.
09.00.04 JOB00421  ICH70001I TORI243  LAST ACCESS AT 08:58:50 ON FRIDAY, APRIL 16, 2004
09.00.04 JOB00421  $HASP373 EPC##ARS STARTED - INIT 1    - CLASS 1 - SYS AB73
09.00.04 JOB00421  IEF403I EPC##ARS - STARTED - TIME=09.00.04
09.00.07 JOB00421  IEC161I 076-002,EPC##ARS,STEP1,ARQUIVO,,,DSVAABPS.FIN.A999
09.00.08 JOB00421  #                                                  --TIMINGS (MINS.)--
----PAGING COUNTS---
09.00.08 JOB00421  # JOBNAME  STEPNAME PROCSTEP PROGRAM     RC   EXCP   CONN    TCB    SRB  CLOCK   SERV
PG  PAGE  SWAP   VIO SWAPS
09.00.08 JOB00421  # EPC##ARS STEP1             IDCAMS      12     16     22    .00    .00     .0    615
0     0     0     0     0
09.00.08 JOB00421  IEF404I EPC##ARS - ENDED - TIME=09.00.08
09.00.08 JOB00421  # EPC##ARS ENDED.  NAME-VERIFY               TOTAL TCB CPU TIME=   .00  TOTAL ELAPSED
TIME=    .0
09.00.08 JOB00421  $HASP395 EPC##ARS ENDED
------ JES2 JOB STATISTICS ------
  16 APR 2004 JOB EXECUTION DATE
           8 CARDS READ
```

JES2.JESJCL

```
1 //EPC##ARS JOB ('VERIFY'),'VERIFY',CLASS=1,MSGCLASS=X
  //*-------------------------------------------------------
2 //STEP1    EXEC PGM=IDCAMS
3 //ARQUIVO  DD DSN=DSVAABPS.FIN.A999,
  //             DISP=SHR
4 //SYSIN DD *
5 //SYSPRINT DD SYSOUT=*
```

JES2.JESYSMSG

```
ICH70001I TORI243  LAST ACCESS AT 08:58:50 ON FRIDAY, APRIL 16, 2004
IEF236I ALLOC. FOR EPC##ARS STEP1
IGD103I SMS ALLOCATED TO DDNAME ARQUIVO
IEF237I JES2 ALLOCATED TO SYSIN
IEF237I JES2 ALLOCATED TO SYSPRINT
IEC161I 076-002,EPC##ARS,STEP1,ARQUIVO,,,DSVAABPS.FIN.A999
IEF142I EPC##ARS STEP1 - STEP WAS EXECUTED - COND CODE 0012
IGD104I DSVAABPS.FIN.A999                              RETAINED,  DDNAME=ARQUIVO
IEF285I   TORI243.EPC##ARS.JOB00421.D0000101.?         SYSIN
IEF285I   TORI243.EPC##ARS.JOB00421.D0000102.?         SYSOUT
IEF373I STEP/STEP1   /START 2004107.0900
IEF374I STEP/STEP1   /STOP  2004107.0900 CPU    0MIN 00.05SEC SRB    0MIN 00.00SEC VIRT   192K SYS   256K
EXT     140K SYS   11696K
IEF375I  JOB/EPC##ARS/START 2004107.0900
IEF376I  JOB/EPC##ARS/STOP  2004107.0900 CPU    0MIN 00.05SEC SRB    0MIN 00.00SEC
```

STEP1.SYSPRINT

```
IDCAMS  SYSTEM SERVICES                                          TIME: 09:00:05
04/16/04     PAGE      1

 VERIFY FILE(ARQUIVO)
IDC3300I  ERROR OPENING DSVAABPS.FIN.A999
IDC3351I ** VSAM OPEN RETURN CODE IS 188
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 12
```

2.2.2 Execução de JOBS

Cada JOB tem uma classe (de seleção para execução). Ex. CLASS=A

A seleção dos jobs para execução é feita pelos initiators (inits), usando como critério a classe. A quantidade de init's, quais as classes que cada um atende, quais ficam "abertos" atendendo à demanda de jobs, é determinada pelo pessoal de suporte e de produção.

Exemplo de initiators e respectivos status, para atendimento das filas de entrada de jobs (para visualizar : comando `DIS INIT`)

```
> APPLID(ABNROSCD)   USER(OB3,TORI243)
>
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
INIT 1    ACTIVE   JOB00716 ADP#6IA  (1) PVAG1      0:17:13 C=12345
INIT 2    ACTIVE   JOB01873 NDVJD000 (1) COPYDIR    0:05:04 C=21345
INIT 3    ACTIVE   JOB00809 SCO#END  (1) NDVRBAT    0:01:15 C=32154
INIT 4    INACTIVE                                          C=0
INIT 5    INACTIVE                                          C=85
INIT 6    ACTIVE   JOB00789 OCC#SCL  (9) NDVRBAT    0:05:05 C=9
INIT 7    INACTIVE                                          C=QP
INIT 8    ACTIVE   JOB00812 FIN#T305 (S) STEP002    0:01:00 C=QS
INIT 9    INACTIVE                                          C=QR
INIT 10   ACTIVE   JOB00605 CPC#BUS  (A) NDVRBAT    0:28:38 C=YA
INIT 11   ACTIVE   JOB00771 SCO#ALB  (Y) NDVRBAT    0:08:08 C=YU
INIT 12   ACTIVE   JOB00819 NDVJPKPR (Z) BC1JPCKG   0:00:05 C=ZB
INIT 13   INACTIVE                                          C=CT
INIT 14   INACTIVE                                          C=ZN
INIT 15   ACTIVE   JOB00782 SCO#A005 (W) STEP1      0:06:13 C=W
```

Uma vez selecionados pelo initiator, o job é executado.

Durante sua execução, o que vai determinar sua seleção para uso de CPU é sua prioridade (PRTY).

À medida que for sendo executado, o job gera a(s) sysout(s) : em princípio, pelo menos as 3 padrões (`JES2.JESMSGLG, JES2.JESJCL e JES2.JESYSMSG`) e, adicionalmente, aquelas que, se for o caso, os programas gerarem.
Tais sysouts vão sendo colocadas na fila de saídas do JOB para que possam ser vistas (browse / view) e / ou impressas.
A visualização pode ser feita enquanto o JOB estiver sendo executado.

2.2.3 Saída de JOBS

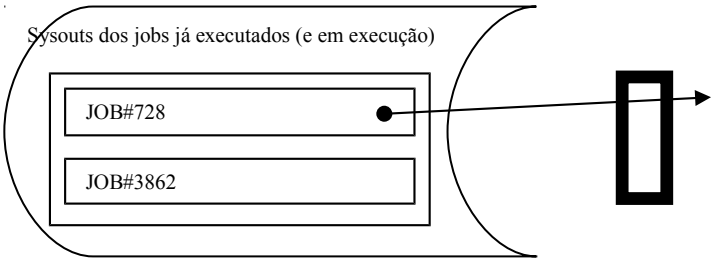
Pode-se efetuar a impressão ou a visualização das saídas de um JOB.

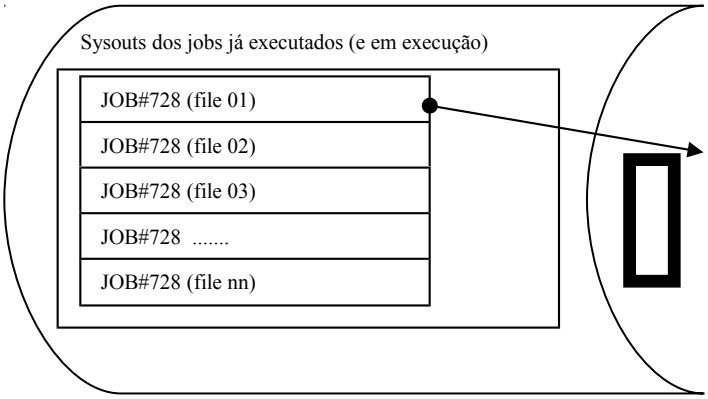
A impressão (se for o caso) é feita pelas WRITERS do sistema.
Para atendimento à fila de impressão, elas baseiam-se em

- identificação da impressora (na realidade associada à classe do relatório : SYSOUT=A; SYSOUT=* indica que a classe da sysout deve ser a mesma classe do MSGCLASS)
- identificação de formulário (FORMS)

A visualização é feita de duas formas :

- Uso do Roscoe ou do TSO
- Uso do Control-P





2.3 JCL

Linguagem de comandos para execução de serviços.

Statements (comandos) básicos : JOB , STEP e DD

Comentários = `//*  nas colunas 1 , 2 e 3`

Identificação = `//  nas colunas 1 e 2`

Conteúdo dos statements

- Identificação = `//  nas colunas 1 e 2`
- Nome = mínimo 1 e máximo 8 posições; início na posição 3
- Operação = JOB ou EXEC ou DD (ou PROC ou PEND ou IF ou END-IF)
- Parâmetros
  - podem ser
    - ✓ posicionais : dependem da posição ou
    - ✓ keyword : palavra chave seguida do sinal de =
  - Se tiver só posicionais, colocá-los na ordem (posição) adequada
  - Se tiver só keyword, colocar em qualquer ordem
  - Se tiver ambos, colocar primeiro os posicionais (na ordem adequada) e depois os keyword (em qualquer ordem)
- Comentários

Entre nome do statement (nome do job, nome do step ou nome do DD) e operação : pelo menos 1 branco; pode ter quantos couber

Entre operação e parâmetros : pelo menos 1 branco; pode ter quantos couber

Após o final do(s) parâmetro(s) , deixando pelo menos um espaço em branco, o conteúdo a seguir é considerado comentário.

Continuação de statements

- statement a ser continuado (se o parâmetro estiver completo) deve terminar com uma vírgula (para indicar que vai haver mais parâmetros)
- no statement seguinte (onde é feita a continuação) o início da continuação pode ser feito em qualquer posição desde a 4 até a 16 (não pode continuar desde a 3 nem depois da 16)
- Exemplos (as duas primeiras linhas são réguas para referência da posição dos caracteres nos statements) :

```
.        1         2         3         4         5         6         7..
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
//ENTRADA   DD               DSN=SYS5.LINKLIB,
// DISP=SHR       BIBLIOTECA DE UTILITARIOS HOME MADE
//ENTRADA   DD               DSN=SYS5.LINKLIB,
//          DISP=SHR      COMENTARIO
//ENTRADA   DD               DSN=SYS5.LINKLIB,
//             DISP=SHR COMENTARIO
//ENTRADA   DD               DSN=SYS5.LINKLIB,
//             DISP=SHR
```

Os casos a seguir mostram casos de PROBLEMAS COMUNS.
Alguns não são acusados como erro pelo sistema, pois são entendidos de forma diferente!

```
.        1         2         3         4         5         6         7..
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

//ENTRADA   DD               DSN=SYS5.LINKLIB
// DISP=SHR       BIBLIOTECA DE UTILITARIOS HOME MADE
```

Neste caso faltou a virgula após o DSN; portanto o DISP=SHR não foi considerado como fazendo parte do statement DD e sim um statement separado ; observar que apesar de não acusar erro de jcl no DD ENTRADA, o sistema vai entender que o arquivo SYS5.LINKLIB deve ser criado , pois faltou o DISP para ele; vai entender também que o statement seguinte está errado pois a operação é inválida (DISP=SHR)

```
.        1         2         3         4         5         6         7..
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

//ENTRADA   DD               DSN=SYS5.LINKLIB,
//              DISP=SHR      COMENTARIO
```

Neste caso a continuação está na col 17 (depois da 16)

```
.        1         2         3         4         5         6         7..
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

//ENTRADA   DD               DSN=SYS5.LINKLIB,
//DISP=SHR COMENTARIO
```

Neste caso a continuação está na 3

```
.        1         2         3         4         5         6         7..
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

//ENTRADA   DD               DSN=SYS5.LINKLIB,
            DISP=SHR
```

Neste caso, o statement de continuação não tem o // nas colunas 1 e 2 : o sistema entende que o statement DD de nome ENTRADA irá continuar na linha seguinte (pois termina com um parâmetro seguido de vírgula) , mas a linha seguinte nem é de jcl (é considerada uma linha de dados de um arquivo de ddname SYSIN)

Conceitos

JOB = unidade máxima de tratamento de serviços batch = execução de um conjunto de programas, serialmente

STEP = unidade mínima de tratamento de serviços batch = execução de um programa; comandada pelo statement EXEC

DD =Data Definition = descrição de um arquivo de um programa

Fim NORMAL x Fim ANORMAL (ABEND) de execução de um programa

O final de um programa pode ocorrer de duas formas distintas : normal e anormal.

Fim NORMAL

É programado, ou seja, o programa determina quando e em qual situação algorítmica o programa deve terminar suas atividades, encerrando sua execução. Ao terminar normalmente , o programa entrega ao sistema operacional um número denominado RETURN CODE (eventualmente é referenciado como COND CODE). Os programas "colocam" o return code disponível para o sistema operacional , colocando o valor desejado numa variável convencionada. Essa variável depende da linguagem utilizada :

```
COBOL       :  MOVE  4  TO RETURN-CODE
Assembler   :  L    15,=F'4'
PL/I        :  CALL PLIRETC (4);
Easytrieve  :  RETURN-CODE = 4
```

Para que o teste do Return Code, utiliza-se o parâmetro COND do statement EXEC.

Normalmente, após a colocação do return code na variável adequada, o programa deve emitir a instrução que finaliza (de forma normal) sua execução:

```
COBOL       :  STOP RUN      ou   GOBACK
Assembler   :  BR     14     ou   RETURN
```

```
PL/I        :  RETURN        ou   END;
Easytrieve  :  STOP  (explícito ou implícito)
```

Em geral, o Return Code emitido por um programa que termina normalmente deve ser zero. Um número diferente indica término normal mas com alguma condição ou evento que deva ser analisado ou que mereça atenção.

Exemplificando : considerar um conjunto de execuções composto por 3 execuções de programas : compilação + linkedição + execução de aplicativo

- O compilador Cobol:
  - se não houver nenhum erro de compilação, termina a execução com return code zero
  - se houver erro simples, que não impeça a execução do programa aplicativo, termina a compilação com return code 4
  - se houver erro de maior severidade, que impeça a execução do programa aplicativo, termina a compilação com return code 8
  - se houver erros mais graves termina a compilação com return code 12 ou 16

  O sistema operacional , ao analisar o return code da compilação (antes de iniciar a execução do linkeditor) , pode determinar se as execuções restantes (linkedição + execução de aplicativo) deve prosseguir ou não;

- O linkeditor:
  - se não houver nenhum erro de linkedição termina a execução com return code zero
  - se houver algum erro, dependendo de sua severidade, termina a linkedição com return code 4 ou 8 ou 12 ou 16

  O sistema operacional , ao analisar o return code da linkedição (antes de iniciar a execução do programa aplicativo) , pode determinar se ela deve ser comandada ou não.

Isso permite estabelecer uma relação de dependência para a execução sequencial de diversos programas, na qual, quando um deles terminar em condições que podem prejudicar os subsequentes, a execução dos seguintes pode ser "pulada".

Fim ANORMAL

O fim ANORMAL é denominado ABEND (ABnormal END). Abend's podem ser:

- Programados : programa emite macro ABEND (que, na verdade, gera uma instrução de máquina SVC – supervisor call) ; em geral usados quando o aplicativo detecta alguma condição extremamente anormal, que irá influenciar, causando danos, a execução de todos os programas subsequentes e, eventualmente, de jobs subsequentes.
  Abends programados são identificados por um código (Unnnn) onde U identifica que ele foi decisão do aplicativo (Usuário do sistema) e nnnn indica um número de 0 a 4095.

Observar que o estabelecimento de tais números é feito por quem concebe os aplicativos que emitem os abends; exemplificando : um programa que deveria ler registros lógicos que tivessem código `'INC'` nas posições 221 a 223 se, eventualmente, lesse um registro que não contivesse tal código, poderia terminar normalmente (inclusive com Return Code = 00) ou anormalmente; se fosse anormalmente, seria com um código que poderia ser desde U0000 até U4095, por escolha de quem estivesse concebendo a aplicação.

- Não programado : componentes externos ao programa identificaram uma situação anormal e emitiram a macro ABEND.
  Abends não programados são identificados por um código (Sxxx) onde S identifica que foi decisão do sistema e xxx é um código que permite identificar o tipo de anormalidade acontecida.
  Exemplos :
  S0C7 = abend provocado pelo sistema, quando ele detectou a tentativa de executar instrução que manipula dados que deveriam estar no formato decimal compactado, mas que não estão no referido formato.

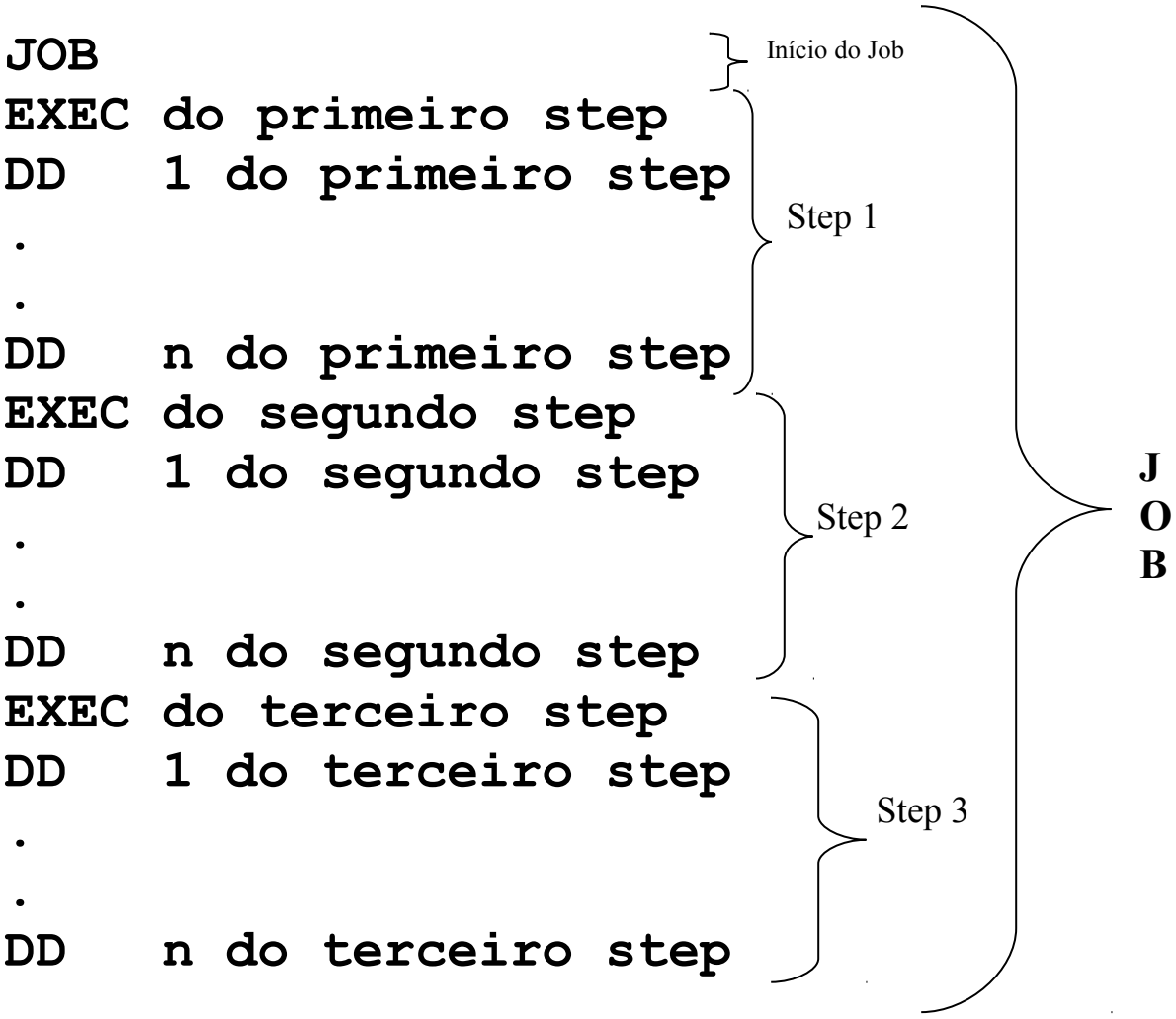
Tabela resumo :

FIM	Return Code (RC)	Abend Code	Próximos Steps
Normal	= 0 ou > 0	**Não aplicável**	**Se tiver COND=(*n,condição*) :** **executados se *n* for *condição* comparado com o RC** **não executados (FLUSHED) se *n* não for *condição* comparado com o RC** **Se não tiver COND :** **executados independentemente dos RC's anteriores**
Anormal	**Não aplicável**	Unnnn ou Sxxx	**Se COND=EVEN : executados mesmo que tenha havido fim anormal** **Se COND=ONLY : executados somente se houve fim anormal** **Se não tiver COND : não são executados (FLUSHED)**

```
J E S 2  J O B  L O G  --  S Y S T E M  A B 7 3  --  N O D E

JOB02648 ---- FRIDAY,    07 MAY 2004 ----
JOB02648  IRR010I  USERID TORI243  IS ASSIGNED TO THIS JOB.
JOB02648  ICH70001I TORI243  LAST ACCESS AT 15:50:25 ON FRIDAY, MAY 7, 20
JOB02648  $HASP373 LSG#@ARS STARTED - INIT 1    - CLASS 1 - SYS AB73
JOB02648  IEF403I LSG#@ARS - STARTED - TIME=15.51.59
JOB02648  #                                                          --T
JOB02648  # JOBNAME  STEPNAME PROCSTEP PROGRAM     RC   EXCP   CONN    TC
JOB02648  # LSG#@ARS AMS237            IDCAMS      12    193    307    .0
JOB02648  # LSG#@ARS CRIAVSAM          EZTPA00     00    100    142    .0
JOB02648  IEC161I 056-084,LSG#@ARS,FINP473V,FINA473,,,DSVAABVS.FIN.A473,
JOB02648  IEC161I DSVAABVS.FIN.A473.DATA,UCATDSV.GERAL
JOB02648  IEC161I 056-084,LSG#@ARS,FINP473V,FINA473,,,DSVAABVS.FIN.A473,
JOB02648  IEC161I DSVAABVS.FIN.A473.INDEX,UCATDSV.GERAL
JOB02648  IEC161I 062-086,LSG#@ARS,FINP473V,FINA473,,,DSVAABVS.FIN.A473,
JOB02648  IEC161I DSVAABVS.FIN.A473.DATA,UCATDSV.GERAL
JOB02648  IEF450I LSG#@ARS FINP473V - ABEND=S000 U0000 REASON=00000000  6
                  TIME=15.52.13
JOB02648  # LSG#@ARS FINP473V          TABTR9C9 U0000    113    187    .0
JOB02648  IEF404I LSG#@ARS - ENDED - TIME=15.52.15
JOB02648  # LSG#@ARS ENDED.  NAME-ROMANO               TOTAL TCB CPU TIME
JOB02648  $HASP395 LSG#@ARS ENDED
S2 JOB STATISTICS ------
2004 JOB EXECUTION DATE
 107 CARDS READ
 340 SYSOUT PRINT RECORDS
   0 SYSOUT PUNCH RECORDS
  18 SYSOUT SPOOL KBYTES
0.26 MINUTES EXECUTION TIME
```

Visão Geral:


JOB
EXEC do primeiro step
DD 1 do primeiro step
.
.
DD n do primeiro step
EXEC do segundo step
DD 1 do segundo step
.
.
DD n do segundo step
EXEC do terceiro step
DD 1 do terceiro step
.
.
DD n do terceiro step
Início do Job
Step 1
Step 2
Step 3
JOB

Exemplo de job batch solicitando SELECT de uma base de dados

```
//PEF#SELE  JOB (B281,DB2),A005864,MSGCLASS=X,CLASS=R,
// NOTIFY=TORI004,GROUP=DB2DPEF
//*PASSWORD=MAR03X
//STEP001  EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=6M
//STEPLIB  DD  DSN=SYS1.DSNDB1D.SDSNLOAD,DISP=SHR
//         DD  DSN=SYS1.DSNDB1D.RUNLIB.LOAD,DISP=SHR
//DBRMLIB  DD  DSN=SYS1.DSNDB1D.DBRMLIB.DATA,
//             DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(DB1D)
 RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP2) -
      LIB('SYS1.DSNDB1D.RUNLIB.LOAD') PARMS('/ALIGN(MID)')
 END
//SYSIN    DD *
 SELECT CD_AGE, NM_AGE
 FROM DB2T.ZZZ100_CADAGE
 WHERE CD_AGE > 20 AND CD_AGE < 100;
```

Saída do job (resultado do Select)

```
PAGE    1
***INPUT STATEMENT:
  SELECT CD_AGE, NM_AGE
  FROM DB2T.ZZZ100_CADAGE
  WHERE CD_AGE > 20 AND CD_AGE < 100;

  +----------------------------------+
  |     CD_AGE     |     NM_AGE      |
  +----------------------------------+
1_|             30 | BARUERI         |
2_|             40 | S.JOSE R.PRETO  |
3_|             50 | SAO CARLOS      |
4_|             60 | PINHEIROS       |
5_|             70 | PERDIZES        |
6_|             80 | MARAMBAIA       |
7_|             90 | JAGUARETE       |
  +----------------------------------+
```

Exemplo de JOB com 1 step (execução do programa IDCAMS)

```
//ABN#ARS1 JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
//*-------------------------------------------------
//*  LISTA INFORMACOES DE CATALOGO
//*
//STEP1 EXEC PGM=IDCAMS
//SYSOUT   DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSABOUT DD  SYSOUT=*
//SYSIN DD *
  LISTCAT  ENTRIES(DSVAABPS.ACD.A701) ALL
//
```

Sua execução gera as seguintes saídas :

JES2 JOB LOG (JESMSGLG)

```
J E S 2  J O B  L O G  --  S Y S T E M  A B N 1  --  N O D E  A B N M V S 1

13.23.26 JOB02321 ---- WEDNESDAY, 31 JUL 2002 ----
13.23.26 JOB02321  IRR010I  USERID TORI141  IS ASSIGNED TO THIS JOB.
13.23.26 JOB02321  ICH70001I TORI141  LAST ACCESS AT 13:21:42 ON WEDNESDA Y, JULY 31, 2002
13.23.26 JOB02321  $HASP373 ABN#ARS1 STARTED - INIT 1    - CLASS 1 - SYS ABN1
13.23.26 JOB02321  IEF403I ABN#ARS1 - STARTED - TIME=13.23.26
13.23.27 JOB02321  #                                                          --TIMINGS
(MINS.)--            ----PAGING COUNTS---
13.23.27 JOB02321  # JOBNAME  STEPNAME PROCSTEP PROGRAM     RC   EXCP   CONN    TCB    SRB
CLOCK   SERV  PG  PAGE  SWAP   VIO SWAPS
13.23.27 JOB02321  # ABN#ARS1 STEP1             IDCAMS      00     24     60    .00    .00
.0  21213 203     0     0     0     0
13.23.27 JOB02321  IEF404I ABN#ARS1 - ENDED - TIME=13.23.27
13.23.27 JOB02321  # ABN#ARS1 ENDED.  NAME-ALBERTO              TOTAL TCB CPU TIME=   .00
TOTAL ELAPSED TIME=    .0
13.23.27 JOB02321  $HASP395 ABN#ARS1 ENDED
------ JES2 JOB STATISTICS ------
  31 JUL 2002 JOB EXECUTION DATE
           13 CARDS READ
           84 SYSOUT PRINT RECORDS
```

JES2 JOB JCL LIST (JESJCL)

```
 1 //ABN#ARS1 JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
   //*-------------------------------------------------------------------
   //*  LISTA INFORMACOES DE CATALOGO
   //*
 2 //STEP1 EXEC PGM=IDCAMS
 3 //SYSOUT   DD  SYSOUT=*
 4 //SYSUDUMP DD  SYSOUT=*
 5 //SYSPRINT DD  SYSOUT=*
 6 //SYSABOUT DD  SYSOUT=*
 7 //SYSIN DD *
 8 //
```

JOB SYSTEM MESSAGES (JESYSMSG)

```
ICH70001I TORI141  LAST ACCESS AT 13:21:42 ON WEDNESDAY, JULY 31, 2002
IEF236I ALLOC. FOR ABN#ARS1 STEP1
IEF237I JES2 ALLOCATED TO SYSOUT
IEF237I JES2 ALLOCATED TO SYSUDUMP
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I JES2 ALLOCATED TO SYSABOUT
IEF237I JES2 ALLOCATED TO SYSIN
IEF142I ABN#ARS1 STEP1 - STEP WAS EXECUTED - COND CODE 0000
IEF285I   TORI141.ABN#ARS1.JOB02321.D0000102.?         SYSOUT
IEF285I   TORI141.ABN#ARS1.JOB02321.D0000103.?         SYSOUT
IEF285I   TORI141.ABN#ARS1.JOB02321.D0000104.?         SYSOUT
IEF285I   TORI141.ABN#ARS1.JOB02321.D0000105.?         SYSOUT
IEF285I   TORI141.ABN#ARS1.JOB02321.D0000101.?         SYSIN
IEF373I STEP/STEP1   /START 2002212.1323
IEF374I STEP/STEP1   /STOP  2002212.1323 CPU    0MIN 00.19SEC SRB    0MIN 00.00SEC VIRT
444K SYS   260K EXT       4K SYS   12076K
IEF375I  JOB/ABN#ARS1/START 2002212.1323
IEF376I  JOB/ABN#ARS1/STOP  2002212.1323 CPU    0MIN 00.19SEC SRB    0MIN 00.00SEC
```

Saída do aplicativo (neste caso SYSPRINT)

```
IDCAMS  SYSTEM SERVICES                                          TIME: 13:23:26
07/31/02     PAGE      1

 LISTCAT  ENTRIES(DSVAABPS.ACD.A701) ALL                              00002100
NONVSAM ------- DSVAABPS.ACD.A701
     IN-CAT --- UCATDSV.GERAL
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------2002.207
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
     SMSDATA
       STORAGECLASS ---STCDSVPS      MANAGEMENTCLASS-MGMDSVPS
       DATACLASS ------------PS      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------D7V013      DEVTYPE------X'3010200F'     FSEQN--- ---------------0
     ASSOCIATIONS--------(NULL)
     ATTRIBUTES
IDCAMS  SYSTEM SERVICES                                  TIME: 13:23:26        07/31/02
PAGE      2
THE NUMBER OF ENTRIES PROCESSED WAS:
                    AIX -------------------0
                    ALIAS -----------------0
                    CLUSTER ---------------0
           etc...          .
                    TOTAL -----------------1
         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

2.3.1 Statement JOB

Sintaxe : **`//nomejob JOB parâmetros_posicionais,parâmetros_keyword`**

nome do job = máximo 8 caracteres

parâmetros = primeiro os posicionais (se houver) e depois os keywords

Parâmetros posicionais:

`programador=` indica o
`account#=` indica o

Parâmetros keyword:

`CLASS=`	indica a classe de execução do job; cada classe é atendida por 1 (ou mais) *initiators*; a quantidade de init's, quais as classes que cada um atende, quais ficam "abertos" atendendo à demanda de jobs, é determinada pelo pessoal de suporte e de produção.
`TIME=`	indica o tempo máximo de CPU do JOB; 1440 = indica que não há tempo máximo
`MSGCLASS=`	indica a classe da sysout das mensagens exibidas (conforme solicitado pelo MSGLEVEL)
`MSGLEVEL=`	indica o que deve ser listado. Especifica-se MSGLEVEL=(a,b) sendo a : mensagens de JCL `0` = somente o statement JOB `1` = exibir tudo: statements JCL submetidos + statements JCL gerados pelo sistema `2` = exibir somente statements JCL submetidos b : mensagens de alocação `0` = não exibir `1` = exibir
`NOTIFY=`	indica o id do usuário de TSO que deve ser notificado quando o job terminar
`REGION=`	indica o tamanho da REGION que deve ser atribuida ao JOB
`RESTART=`	Indica o nome do step a partir do qual o JOB deve (re)iniciar
`TYPRUN=`	indica o tipo de submissão desejada : SCAN = valida se o JCL está OK mas não executa nenhum programa HOLD = deixa o job em espera até ser liberado para execução

Exemplo :

```
//EPC##ARS JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
```

2.3.2 Statement EXEC

Sintaxe : **`//nomestep EXEC parâmetros`**

nome do step = máximo 8 caracteres
parâmetros = primeiro os posicionais e depois os keywords

Parâmetro posicional só existe um : a identificação do nome da PROC; neste caso, se ele for colocado, não pode ser especificado nem PROC= nem PGM= pois o default é (pela especificação do parâmetro posicional que é o nome da proc) nome da PROC ver adiante exemplos).

Principais parâmetros keyword:

COND=	indica a(s) condição(ões) de execução ou não do step **COND=EVEN** executa o step MESMO QUE algum anterior tenha cancelado **COND=ONLY** executa o step SOMENTE SE algum anterior tenha cancelado **COND=(n,*cond*)** NÃO EXECUTA o step se n satisfizer a condição especificada em qualquer step anterior **COND=(n,*cond*,step)** NÃO EXECUTA o step se n satisfizer a condição especificada no step cujo nome foi indicado ***Cond*** pode ser : **LT** ou **GT** ou **EQ** ou **NE** **Ex.:** **COND=(4,LT)** se 4 for menor que o return code de qualquer step anterior, NÃO EXECUTA ou seja : se algum step anterior emitiu return code maior que 4, NÃO EXECUTA **COND=((4,LT,COMP),(4,LT,LKED))** (se 4 for menor que o return code do step COMP E se 4 for menor que o return code do step LKED) NÃO EXECUTA ou seja : (se o step COMP emitiu return code maior que 4 OU se o step LKED emitiu return code maior que 4) NÃO EXECUTA
PGM=	indica o nome do programa a ser executado

PARM=	indica os dados que devem ser passados como parâmetro(s) para o programa chamado para executar
	Ex.:
	`PARM='NOSEQ,QUOTE,OFFSET,LIB'`
	`PARM='30/12/2003'`

PROC=	indica o nome da proc a ser executada. Se não especificado nem PGM= nem PROC= assume PROC= Exemplos : `//STEP1 EXEC TST01` executa a PROC denominada TST01 `//STEP2 EXEC PROC=TST01` executa a PROC denominada TST01 `//STEP1 EXEC PGM=TST01` executa o programa denominado TST01

Exemplos :

```
//COMP  EXEC PGM=IGYCRCTL,COND=(0,LT),PARM='NOSEQ,QUOTE,OFFSET,LIB'
//STP1  EXEC COBCLG
//STP1  EXEC PROG=COBCLG
//STP1  EXEC PGM=IEWL
```

2.3.3 Statement DD

Sintaxe : **`//nomedd DD parâmetros`**

nome do dd = máximo 8 caracteres

parâmetros = primeiro os posicionais (se houver) e depois os keywords

Principais parâmetros posicionais : * e DUMMY

***** indica que os dados estão a partir do próximo statement

Ex.:

```
//SYSIN DD *
JOSE DA SILVA
JUCA
//OUTRO DD SYSOUT=*
```

Dados vão desde o DD * até o anterior ao próximo //

```
//SYSIN DD *
JOSE DA SILVA
JUCA
/*
//OUTRO DD SYSOUT=*
```

Dados vão desde o DD * até o /*

```
//SYSIN DD *,DLM="AA"
   JOSE DA SILVA
JUCA
AA
//OUTRO DD SYSOUT=
```

Dados vão desde o DD * até aquele com o DeLiMitador indicado pelo parâmetro DLM

DUMMY indica que o arquivo não existe, e o programa deve entender como um arquivo com 0 registros (EOF na primeira leitura); válido apenas para arquivos que no programa são declarados com sequenciais.

Ex.:

```
//SYSIN DD DUMMY
```

Principais parâmetros Keyword :

DSN	Data Set Name = indica o nome externo (para o sistema operacional) do arquivo `Ex.:` `//SYSIN DD DSN=DSVAABPS.TER.ALBERTO,DISP=SHR`

DCB	indica as características do arquivo
	Sub-parâmetros :
	DSORG= (Data Set ORGanization) indica a organização do arquivo.
	Principais opções :
	PO (Partitioned Organization = Particionado = PDS)
	PS (Physical Sequential = Sequencial)
	EXPDT= (EXPiration DaTe = data de expiração) = indica a data a partir da qual o arquivo pode ser deletado
	RETPD= (RETention PerioD = período de retenção) indica o período de tempo durante o qual o arquivo não está expirado, não podendo ser deletado
	LRECL= (Logical RECord Length) indica o tamanho, em bytes, do registro lógico
	RECFM= (RECord ForMat) indica o formato dos registros do arquivo.
	Principais opções :
	FB = Fixed Blocked
	F = Fixed (Unblocked)
	FBA = Fixed Blocked Asa
	FA = Fixed (Unblocked) Asa
	V = Variable (Unblocked)
	VB = Variable blocked
	U = Undefined
	BLKSIZE= (BLocKSIZE) indica o tamanho, em bytes, do registro físico
	`Ex.:`
	`//SYSIN DD`
	`DUMMY,DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)`

LRECL	Indica o tamanho do registro lógico
RECFM	Indica o formato dos registros : F, FB, V, VB
BLKSIZE	Indica o tamnaho dos blocos (registros físicos)
DSORG	Indica a organização do arquivo : PS, PO
EXPDT	Indica a data de expiração do arquivo
RETPD	Indica o período durante o qual o arquivo não está expirado
DISP	**Disposition** **DISP=(a,b,c)** indica a ação a ser tomada referente ao arquivo **a = NEW** (arquivo será criado) ou **OLD** (arquivo já existe e será manipulado com exclusividade) ou **SHR** (shared - arquivo já existe e será compartilhada) ou **MOD** (arquivo já existe e será estendido) **b = CATLG** (arquivo será catalogado ao final, se final normal) ou **UNCATLG** (arquivo será descatalogado ao final, se normal) ou **DELETE** (arquivo será deletado ao final, se normal)

	KEEP (arquivo será mantido ao final, se normal) **PASS** (arquivo temporário será passado para steps subsequentes) **c = CATLG** (arquivo será catalogado ao final, se final anormal) ou **UNCATLG** (arquivo será descatalogado ao final, se anormal) ou **DELETE** (arquivo será deletado ao final, se anormal) **KEEP** (arquivo será mantido ao final, se anormal)
UNIT	Indica onde o arquivo deve ser alocado. Em geral : SYSDA, 3390
SYSOUT	SYSOUT=(a,b,c) a = indica a classe da sysout b = c = nome do formulario a utilizar Ex.: //SYSOUT DD SYSOUT=*
SPACE	**SPACE= (a,(b,c),RLSE) ou** **SPACE= (d,(b,c),RLSE)** indica a quantidade de espaço em disco a ser alocada para o arquivo . **a = TRK ou CYL ou ABSTR** unidade de alocação : trilhas, cilindros, ou **a = TRK ou CYL ou ABSTR ou rec_len** unidade de alocação; trilhas, cilindros, ou tamanho do registro lógico **b =** quantidade da área primária **c =** quantidade da área secundária (opcional) **d =** tamanho do registro lógico **RLSE =** (ReLeaSE) (opcional) se sobrar espaço, liberar
KEYOFF=	KEY OFFset = indica o deslocamento do campo chave dentro do registro lógico para arquivos VSAM KSDS
KEYLEN=	KEY LENgth = indica o tamanho do campo chave no registro lógico para arquivos VSAM KSDS
RECORG=	RECord ORGanization = indica qual a organização do arquivo VSAM : KS (KSDS), ES (ESDS), RR (RRDS) ou LS (Linear Space)
DATACLAS=	DATA CLASs = indica VSDEF
AVGREC=	Indica a unidade de especificação da quantidade de registros do parâmetro SPACE ; só pode ser usado se SPACE for especificado com quantidade de registros (não pode ser especificado junto com TRK, CYL ou ABSTR) AVGREC=U indica UNIDADES (quantidade * 1) AVGREC=K indica Ks (quantidade * 1024) AVGREC=M indica Ms (quantidade * 1048576)

Exemplos :

```
//SAIDA    DD  DSN=ATAAABPS.EPC.SE62,
//             DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(100,50)),
//             UNIT=SYSDA,DCB=LRECL=83

//ENTRADA  DD  DSN=ATAAABPS.EPC.SLB005,DISP=SHR

//EPCASLBS DD  DSN=ATAAABPS.EPC.SE64(+1),
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(100,50)),
//             UNIT=SYSDA,
//             DCB=LRECL=83

//EPCASLBS DD  DSN=ATAAABPS.EPC.SE64(+1),
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(100,50)),
//             UNIT=SYSDA,
//             DCB=(LRECL=83,BLKSIZE=830)


//VKSDS1 DD DSN=DSVAABVS.LSG.A889.ALBERTO,DISP=(,CATLG,DELETE),
//             SPACE=(CYL,(10,10)),LRECL=100,KEYOFF=10,
//             KEYLEN=12,RECORG=KS
```

2.4 PROCS

Procs = procedimentos = conjunto de statements não-JOB. Podem ser:

- CATALOGADAS
- IN-STREAM

2.4.1 PROCS Catalogadas

A chamada deve ser feita da seguinte forma : no exemplo, a PROC SSSPRO1 deve estar previamente catalogada numa PROCLIB

```
//SSS##nnn JOB ...
//*------------------------------
//* PRIMEIRO STEP = CHAMA PROGRAMA
//*------------------------------
//STEP0001 EXEC PGM=SSSPGM01
//DDNAME01 DD   ....
//DDNAME02 DD   ....
//DDNAME03 DD   ....
//*------------------------------
//* SEGUNDO STEP CHAMA PROC SSSPRO1
//*------------------------------
//STEP0002 EXEC SSSPRO1
//STEPA.DD1 DD  ....
//STEPA.DD2 DD  ....   DD's do STEPA
//STEPA.DD3 DD  ....
//STEPB.DD1 DD  ....   DD's do STEPB
//STEPB.DD2 DD  ....
//STEPC.DD1 DD  ....
//STEPC.DD2 DD  ....   DD's do STEPC
//STEPC.DD3 DD  ....
//STEPC.DD4 DD  ....
//*------------------------------
//* TERCEIRO STEP CHAMA PROC SSSPRO1
//*------------------------------
//STEP0003 EXEC PROC=SSSPRO1
//STEPA.DD1 DD  ....
//STEPA.DD2 DD  ....   DD's do STEPA
//STEPA.DD3 DD  ....
//STEPB.DD1 DD  ....   DD's do STEPB
//STEPB.DD2 DD  ....
//STEPC.DD1 DD  ....
//STEPC.DD2 DD  ....   DD's do STEPC
//STEPC.DD3 DD  ....
//STEPC.DD4 DD  ....
```

Exemplo :

```
//SSSPRO1  PROC
//STEPA    EXEC PGM=PPPX1
//DD1      DD   DSN=...
//SYSPRINT DD   SYSOUT=*
//STEPB    EXEC PGM=PPPX2
//SYSPRINT DD   SYSOUT=*
//STEPC    EXEC PGM=PPPX3
//SYSPRINT DD   SYSOUT=*
//         PEND
```

2.4.2 PROCS In-Stream

A chamada deve ser feita da seguinte forma : no exemplo, a PROC SSSPRO1 deve estar previamente catalogada numa PROCLIB

```
//SSS##nnn JOB ...
//SSSPRO1  PROC
//STEPA    EXEC PGM=PPPX1
//DD1      DD   DSN=...
//SYSPRINT DD   SYSOUT=*
//STEPB    EXEC PGM=PPPX2
//SYSPRINT DD   SYSOUT=*
//STEPC    EXEC PGM=PPPX3
//SYSPRINT DD   SYSOUT=*
//         PEND
//*------------------------------
//* PRIMEIRO STEP = CHAMA PROGRAMA
//*------------------------------
//STEP0001 EXEC PGM=SSSPGM01
//DDNAME01 DD   ....
//DDNAME02 DD   ....
//DDNAME03 DD   ....
//*------------------------------
//* SEGUNDO STEP = CHAMA PROC
//*------------------------------
//STEP0002 EXEC SSSPRO1
//STEPA.DD1 DD  ....
//STEPA.DD2 DD  ....   DD's do STEPA
//STEPA.DD3 DD  ....
//STEPB.DD1 DD  ....   DD's do STEPB
//STEPB.DD2 DD  ....
//STEPC.DD1 DD  ....
//STEPC.DD2 DD  ....   DD's do STEPC
//STEPC.DD3 DD  ....
//STEPC.DD4 DD  ....
```

2.4.3 Parâmetros simbólicos

Nas PROCS, podemos atribuir valores *default* para determinados valores relacionados a parâmetros.

Na PROC a seguir :

```
//SSSPRO1  PROC
//STEPA    EXEC PGM=PPPX1
//DD1      DD   DSN=...
//SYSPRINT DD   SYSOUT=*
//STEPB    EXEC PGM=PPPX2
//SYSPRINT DD   SYSOUT=*
//STEPC    EXEC PGM=PPPX3
//SYSPRINT DD   SYSOUT=*
//         PEND
```

Se quisermos que o valor padrão da classe da sysout seja A, mas que seja possível alterar quando da chamada da PROC, podemos declará-la da seguinte forma :

```
//SSSPRO1  PROC RELCLA=A
//STEPA    EXEC PGM=PPPX1
//DD1      DD   DSN=...
//SYSPRINT DD   SYSOUT=&RELCLA
//STEPB    EXEC PGM=PPPX2
//SYSPRINT DD   SYSOUT=&RELCLA
//STEPC    EXEC PGM=PPPX3
//SYSPRINT DD   SYSOUT=&RELCLA
//         PEND
```

Se, quando a PROC for chamada, não for feita nenhuma referência ao parâmetro simbólico RELCLA, as sysouts serão usadas com a classe *default* A indicada no statement de declaração da PROC :

```
//STEP1    EXEC PROC=SSSPR01
```

Se quisermos que o SYSPRINT do STEPB tenha classe X, ao chamar a PROC, devemos especificar :

```
//STEP1    EXEC PROC=SSSPR01,RELCLA.STEPB=X
```

Outra forma de estabelecer um padrão para que a substituição seja feita é com o statement SET (ver adiante).

2.4.4 Override e substituição de parâmetros simbólicos

A utilização de PROCS quase sempre causa a necessidade de especificarmos ou mudarmos o conteúdo de determinados parâmetros que fazem parte do corpo da PROC.

Para que tais adequações possam ser feitas usamos os processos de OVERRIDE (a sobreposição de informações àquelas que estejam especificadas na PROC catalogada ou in-stream) ou de substituição de parâmetros simbólicos.

Supor a PROC abaixo, denominada PROCX1 para exemplificar :

```
//STEPA    EXEC PGM=PROCX1
//DD1      DD   DSN=...
//SYSPRINT DD   SYSOUT=*
//STEPB    EXEC PGM=PPPX2
//SYSPRINT DD   SYSOUT=*
//STEPC    EXEC PGM=PPPX3
//SYSPRINT DD   SYSOUT=*
```

Inclusão de DD's :

após a chamada da PROC, especificar os novos DD's na ordem dos steps da PROC , indicando no ddname **nome_do_step_na_proc•ddname**

```
//STEP1    EXEC PROCX1
//STEPA.DD2 DD DSN=...
//STEPA.DD3 DD DSN=...
//STEPA.DD4 DD DSN=...
//*
//STEPB.DD2 DD DSN=...
//STEPB.DD3 DD DSN=...
//STEPB.DD4 DD DSN=...
```

Inclusão de concatenação em DD :

Se na PROC houver DD especificado, e quisermos incluir concatenação nele, especificar um DD para cada DD existente, sem parâmetros, e acrescentar o DD desejado :

Se houver um DD sem concatenação : um DD sem parâmetros para o original e adicionar a concatenação :

```
//nome_do_step_na_proc•ddname DD
//                            DD nova_concatenacao_desejada
```

Exemplo :

```
//STEP1    EXEC PROCX1
//STEPA.DD2 DD
//          DD DSN=MAIS.UM.ARQUIVO...
```

Se houver um DD com uma concatenação : um DD sem parâmetros para o original, um DD sem ddname nem parâmetros para a concatenação já existente, e adicionar a nova concatenação :

```
//nome_do_step_na_proc•ddname DD
//                            DD
//                            DD nova_concatenacao_desejada
```

Exemplo :

```
//STEP1    EXEC PROCX1
//STEPA.DD2 DD
//          DD
//          DD DSN=TST.ARQUIVOX...
```

Se houver um DD com duas concatenações : um DD sem parâmetros para o original, dois DD's sem ddname nem parâmetros (um para cada concatenação já existente), e adicionar a nova concatenação :

```
//nome_do_step_na_proc•ddname DD
//                            DD
//                            DD
//                            DD nova_concatenacao_desejada
```

Exemplo :

```
//STEP1    EXEC PROCX1
//STEPA.DD2 DD
//          DD
//          DD
//          DD DSN=TST.ARQUIVOX...
```

Alteração de parâmetro(s) de EXEC's :

após a chamada da PROC, especificar os novos parms na ordem dos steps da PROC, indicando **parametro•nome_do_step_na_proc =**

Exemplos :

```
//STEP1   EXEC PROCX1,PARM.STEP1='DATA=29/01/1955',PARM.STEP2='SIM'
//STEPA.DD2 DD DSN=...

//STEP1   EXEC PROCX2,
//            COND.STEP1=EVEN,
//            COND.STEP2=ONLY
```

2.5 INCLUDE

O statement INCLUDE permite que se mantenha arquivado / catalogado um conjunto de statements de JCL, e quando necessário incorporá-lo a um stream que será submetido, deve-se invocá-lo por meio do statement INCLUDE.

Exemplo 1 :

```
//LSG#@ARS JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
//         JCLLIB ORDER=(JCL.SIST.SRC)
//         INCLUDE MEMBER=BIBPROD
//*-------------------------------------
//STEP1    EXEC PGM=P001
//STEP2    EXEC PGM=P003
//STEP3    EXEC PGM=P004
```

Exemplo 2 :

```
//LSG#@ARS JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
//         JCLLIB ORDER=(JCL.SIST.SRC)
//         INCLUDE MEMBER=BIBDESEN
//*-------------------------------------
//STEP1    EXEC PGM=P001
//STEP2    EXEC PGM=P003
//STEP3    EXEC PGM=P004
```

Se o arquivo JCL.PROD.SRC tiver o statement

```
//STEPAUX  EXEC PGM=EXIBHORA
```

e se o arquivo JCL.DESENV.SRC tiver o statement

```
//STEPAUX  EXEC PGM=IEFBR14
```

o JCL que será efetivamente submetido será

Exemplo 1 :

```
//LSG#@ARS JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
//*        JCLLIB ORDER=(JCL.PROD.SRC)
//STEPAUX  EXEC PGM=EXIBHORA
//*-------------------------------------
//STEP1    EXEC PGM=P001
//STEP2    EXEC PGM=P003
//STEP3    EXEC PGM=P004
```

Exemplo 2 :

```
//LSG#@ARS JOB ('ALBERTO'),'ALBERTO',CLASS=1,MSGCLASS=X
//*        JCLLIB ORDER=(JCL.DESEN.SRC)
//STEPAUX  EXEC PGM=IEFBR14
//*-------------------------------------
//STEP1    EXEC PGM=P001
//STEP2    EXEC PGM=P003
//STEP3    EXEC PGM=P004
```

2.6 Dicas gerais

2.6.1 Identificação da origem dos statements de JCL

```
Se o statement
listado
começa com     Então

    //         O statement é um dos que foram submetidos
    XX         O statement foi gerado por expansão de PROC ou INCLUDE
```

2.6.2 DD's especiais referentes à localização de programas executáveis

Quando um programa é chamado para ser executado (através do EXEC PGM=), o sistema tem uma relação de arquivos onde ele deve buscar o programa.

Para que essa busca seja feita em outros lugares TAMBÉM, usar os DD's JOBLIB e / ou STEPLIB.

JOBLIB

Vale para todos os steps do job.
Deve ser colocado logo após o JOB.

Indica o(s) arquivos que tem programas executáveis, onde deve ser feita primeiramente a busca de um programa para carga e execução.

Ex.:

```
//TAB##P18 JOB .........
//JOBLIB DD DSN=ATAAABLB.SIS.LOADLIB,DISP=SHR
//STEP1  EXEC PGM=TAB001
//TABAE01 DD  .........
//TABAE02 DD  .........
//STEP2  EXEC PGM=TAB015
//TABAE98 DD  .........
//TABAr012 DD  .........
```

Procura primeiro no dataset especificado no JOBLIB, e depois nos datasets padrão do sistema

STEPLIB

Vale para o step em que o STEPLIB estiver. Deve ser colocado logo após o EXEC.

Indica o(s) arquivos que tem programas executáveis, onde deve ser feita primeiramente a busca de um programa para carga e execução. Ex.:

```
//TAB##P18 JOB .........
//STEP1  EXEC PGM=TAB001
//STEPLIB DD  DSN=ATAAABLB.SIS.LOADLIB,DISP=SHR
//TABAE01 DD  .........
//TABAE02 DD  .........
//STEP2  EXEC PGM=TAB015
//TABAE98 DD  .........
//TABAr012 DD  .........
```

Procura primeiro no dataset especificado no STEPLIB, e depois nos datasets padrão do sistema

Procura só nos datasets padrão do sistema

2.6.3 Direcionando a localização de PROCS e INCLUDES

Quando uma proc é chamada para ser expandida (através do EXEC PROC= ou EXEC nome), o sistema tem uma relação de arquivos onde ele deve buscar a proc.

De forma análoga, se um conjunto de statements estiver catalogado e quisermos incorporá-lo ao JCL que estiver sendo submetido, podemos usar o statement INCLUDE, que busca JCL num arquivo / biblioteca, de forma semelhante ao COPY (Cobol).

Para que se indique ao sistema onde (em qual arquivo / biblioteca) ele deve buscar os statements a incorporar (para PROC ou INCLUDE), usar o statement JCLLIB

JCLLIB

```
Sintaxe :  //[nome]  JCLLIB ORDER=(dsn1[,dsn2...])
```

- O(s) dsn(s) indica(m) o(s) arquivo(s) que tem a(s) PROC(s) ou o(s) INCLUDE(s).
- Vale para todos os steps do job.
- Deve ser colocado logo após o JOB; só pode haver um por job.
- Não pode haver INCLUDE nos statements catalogados

```
//TAB##P18 JOB .........
//       JCLLIB ORDER=(TORI243.T#RPC.SRC)
//STEP1  EXEC TABMENSAL
//STEPLIB DD  DSN=ATAAABLB.SIS.LOADLIB,DISP=SHR
//TABAE01 DD  .........
//TABAE02 DD  .........
//STEP2  EXEC TABSEMAN
//TABAE98 DD  .........
//TABAr012 DD  .........
```

Procura a PROC primeiro no TORI243.T#RPC.SRC e depois nos datasets padrão do sistema

2.7 SET

Serve para estabelecer valor(es) padrão para parâmetros simbólicos, e pode ser alterado no transcorrer do JCL.

```
Sintaxe :  //[nome]  SET variavel_simbolica=[valor]
```

```
//SSSPRO1  PROC
//STEPA    EXEC PGM=PPPX1
//DD1      DD   DSN=...
//SYSPRINT DD   SYSOUT=&RELCLA
//STEPB    EXEC PGM=PPPX2
//SYSPRINT DD   SYSOUT=&RELCLA
//STEPC    EXEC PGM=PPPX3
//SYSPRINT DD   SYSOUT=&RELCLA
//         PEND
//   SET RELCLA=A
//STEP1    EXEC SSSPR01
//STEP2    EXEC SSSPR01
//STEP3    EXEC SSSPR01
//STEP4    EXEC SSSPR01
//STEP5    EXEC SSSPR01
//   SET RELCLA=B
//STEP6    EXEC SSSPR01
//STEP7    EXEC SSSPR01
//STEP8    EXEC SSSPR01
//STEP9    EXEC SSSPR01
//STEP10    EXEC SSSPR01
```

Os steps 01, 02, 03, 04 e 05 terão RELCLA valendo A
Os steps 06, 07, 08, 09 e 10 terão RELCLA valendo B

2.8 IF / THEN / ELSE / ENDIF

Pode ser usado em substituição ao COND (com a vantagem de ser mais fácil de entender...).

```
Sintaxe :
```

```
//     IF  (condicao) THEN
statements a executar se a concição for satisfeita
//     ENDIF
```

ou

```
//     IF  (condicao) THEN
statements a executar se a condição for satisfeita
//     ELSE
statements a executar se a condição não for satisfeita
//     ENDIF
```

Sendo :

Condição = especificação da condição que, se satisfeita, indica que o conjunto de statements que seguem o THEN deve ser executado. Deve ser especificada de uma das seguintes formas :
Obs.: nas sintaxes a seguir, se não for especificado o nome do step, refere-se ao último

- Testando o return code de um step anterior (se não for especificado o nome do step, refere-se ao último) :

```
[stepname.]RC EQ return_code   ou
[stepname.]RC GT return_code   ou
[stepname.]RC LT return_code   ou
[stepname.]RC NE return_code   ou
[stepname.]RC NL return_code   ou
[stepname.]RC NG return_code   ou
[stepname.]RC GE return_code   ou
[stepname.]RC LE return_code
```

- Testando se um (programa de um) step anterior iniciou a execução ou não (flushed) :

```
[stepname.]RUN EQ TRUE   ou
[stepname.]RUN EQ FALSE

[stepname.]¬RUN EQ TRUE  ou
[stepname.]¬RUN EQ FALSE
```

- Testando se um (programa de um) step anterior abendou ou não :

```
[stepname.]ABEND EQ TRUE       ou
[stepname.]ABEND EQ FALSE

[stepname.]¬ABEND EQ TRUE      ou
[stepname.]¬ABEND EQ FALSE
```

- Testando o Completion Code (sistema ou user) de um (programa de um) step anterior :

```
[stepname.]ABENDCC EQ Snnn     ou
[stepname.]ABENDCC EQ Unnnn

[stepname.]¬ABENDCC EQ Snnn    ou
[stepname.]¬ABENDCC EQ Unnnn
```

Condições múltiplas =
- Utilizar parênteses para indicar a prioridade e sequência de análise
- Utilizar & para E (and) e | para OU (or)

ENDIF é obrigatório para indicar o fim da abrangência do que deve ser executado se a condição for satisfeita.

ELSE é opcional.

IMPORTANTE : Usar COND e IF / THEN / ELSE / ENDIF juntos pode ser extremamente confuso. NÃO FAÇA ISSO.

Exemplos :

```
//LSG#@ARS JOB ('ROMANO'),'ROMANO',CLASS=1,MSGCLASS=X
//*-------------------------------------------------------------
//STEP01 EXEC PGM=IEFBR14
//STEP02 EXEC PGM=IEFBR14
//         IF  (STEP01.RC NG 0) THEN
//STEP03 EXEC PGM=IEFBR14
//         ENDIF
//*-------------------------------------------------------------
//STEP04 EXEC PGM=IEFBR14
//         IF  ((STEP01.RC EQ 0)) THEN
//STEP05 EXEC PGM=IEFBR14
//         ELSE
//STEP06 EXEC PGM=IEFBR14
//         ENDIF
//*-------------------------------------------------------------
//STEP07 EXEC PGM=IEFBR14
//         IF  ((STEP01.RUN EQ TRUE)) THEN
//STEP08 EXEC PGM=IEFBR14
//         ENDIF
//         IF  ((STEP01.RUN EQ FALSE)) THEN
//STEP09 EXEC PGM=IEFBR14
//         ENDIF
```

Esse JCL, ao ser executado, gerará :

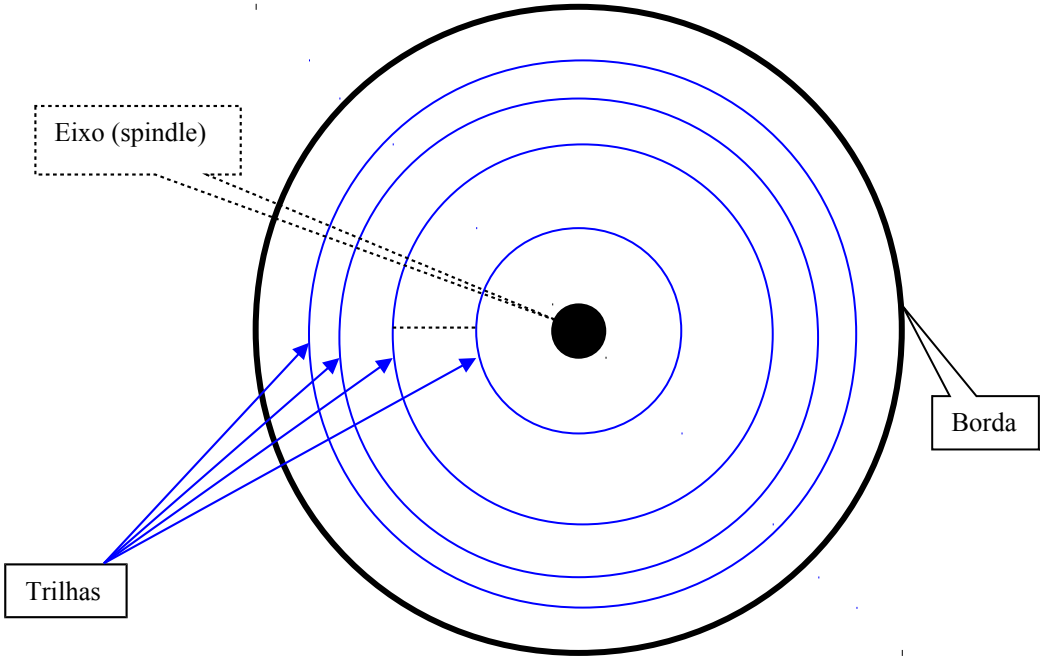
```
$HASP373 LSG#@ARS STARTED - INIT 2
IEF403I LSG#@ARS - STARTED - TIME=14.52.18
#
# JOBNAME  STEPNAME PROCSTEP PROGRAM     RC
# LSG#@ARS STEP01            IEFBR14     00
# LSG#@ARS STEP02            IEFBR14     00
# LSG#@ARS STEP03            IEFBR14     00
# LSG#@ARS STEP04            IEFBR14     00
# LSG#@ARS STEP05            IEFBR14     00
# LSG#@ARS STEP06            IEFBR14  FLUSH
# LSG#@ARS STEP07            IEFBR14     00
# LSG#@ARS STEP08            IEFBR14     00
# LSG#@ARS STEP09            IEFBR14  FLUSH
IEF404I LSG#@ARS - ENDED - TIME=14.52.19
```

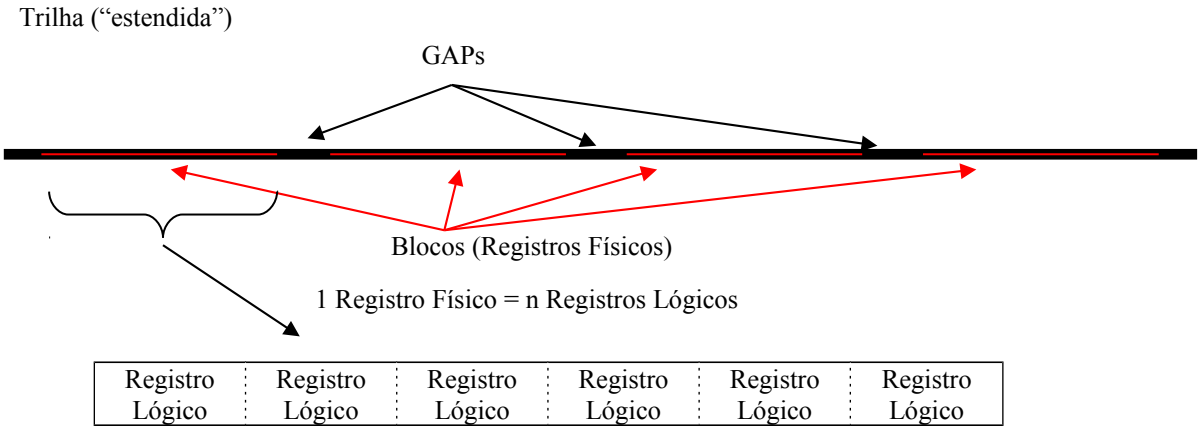
```
//LSG#@ARS JOB ('ROMANO'),'ROMANO',CLASS=1,MSGCLASS=X
//*-------------------------------------------------------------
//STEP01 EXEC PGM=IEFBR14
//STEP02 EXEC PGM=IEFBR14
//         IF  (STEP01.RC EQ 0 & STEP02.RC EQ 0) THEN
//STEP03 EXEC PGM=IEFBR14
//         ENDIF
//*-------------------------------------------------------------
//STEP04 EXEC PGM=IEFBR14
//         IF  (STEP01.RC GT 0 | STEP04.RC GT 0) THEN
```

```
//STEP05 EXEC PGM=IEFBR14
//         ELSE
//STEP06 EXEC PGM=IEFBR14
//         ENDIF
//*-------------------------------------------------------------
//STEP07 EXEC PGM=IEFBR14
//         IF  (ABEND | (STEP04.RC GT 0 & STEP06.RC GT 0)) THEN
//STEP08 EXEC PGM=IEFBR14
//         ELSE
//STEP09 EXEC PGM=IEFBR14
//         ENDIF
//*-------------------------------------------------------------
//STEP10 EXEC PGM=IEFBR14
//         IF  (¬ABEND | (STEP04.RC GT 0 & STEP06.RC GT 0)) THEN
//STEP11 EXEC PGM=IEFBR14
//         ELSE
//STEP12 EXEC PGM=IEFBR14
//         ENDIF
```

3. Arquivos

3.1 Organização de Discos

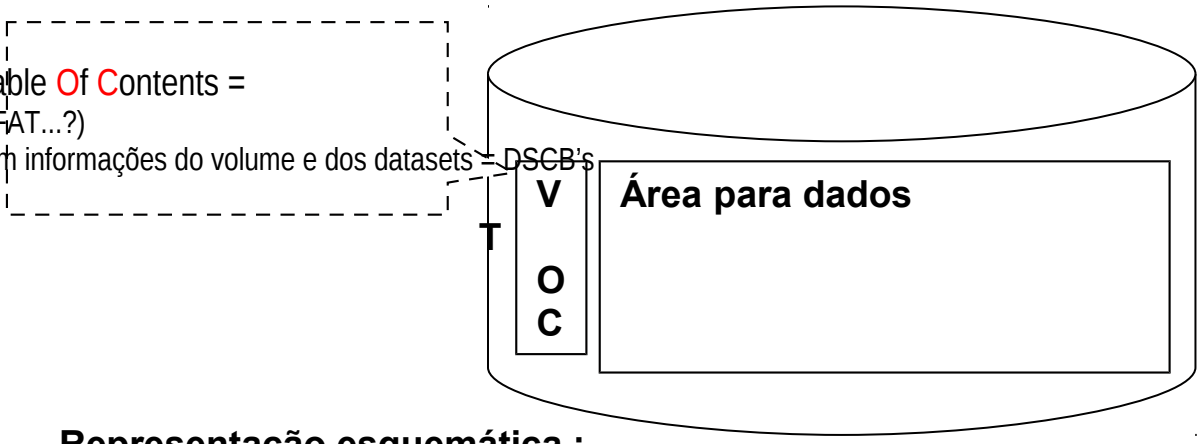




Quantidade de registros lógicos em um registro físico = FATOR de BLOCO

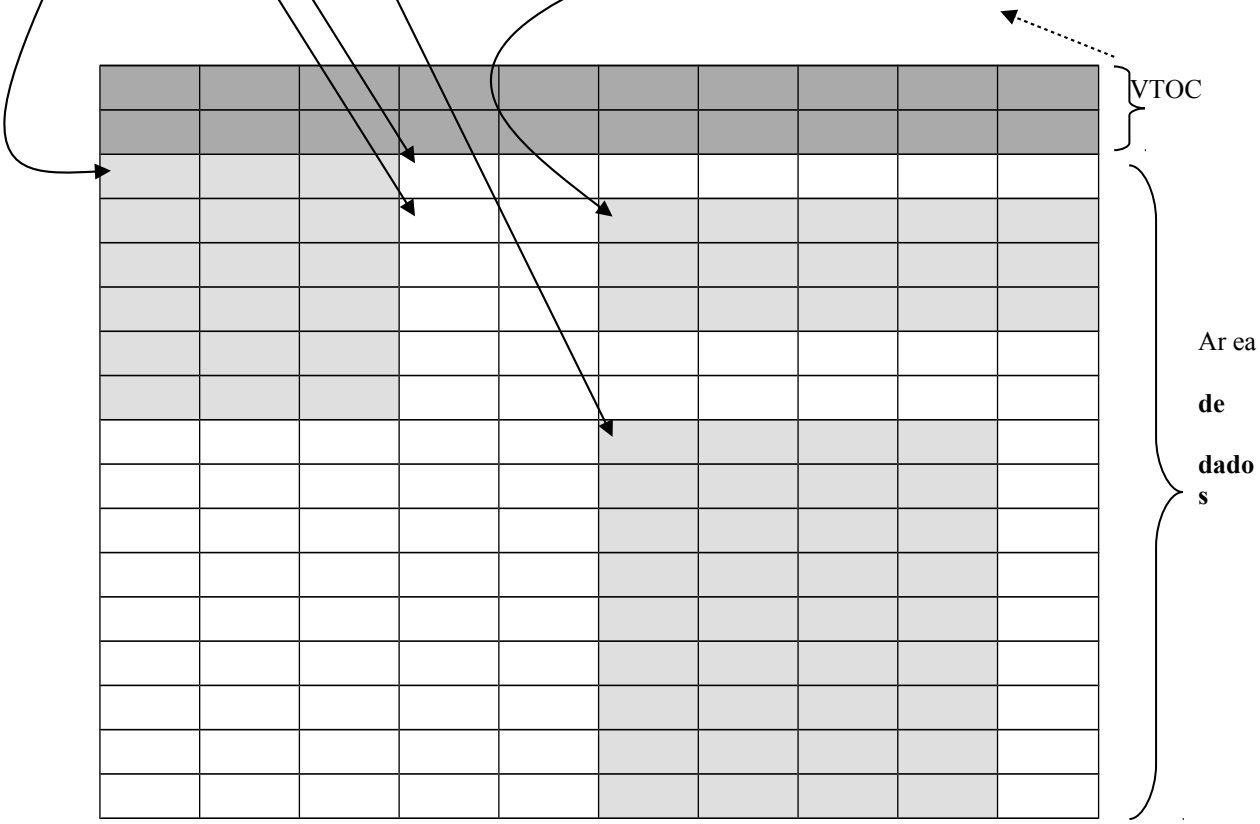
Se Fator de Bloco = 1 : Registros DESBLOCADOS
Se Fator de Bloco > 1 : Registros BLOCADOS

Volume Table Of Contents =
(lembra da FAT...?)
entradas com informações do volume e dos datasets = DSCB's



Representação esquemática :

Arquivo ARQ001 : localizado no cilindro ccc m trilha ttt m
Arquivo ARQ002 alocação primária : localizado no cilindro ccc n trilha ttt n
Arquivo ARQ002 alocação secundária 1 : localizado no cilindro ccc p trilha ttt p
Área livre : localizada no cilindro ccc q trilha q
Área livre : localizada no cilindro ccc r trilha t
Etc...



Área de dados livre

Área de dados ocupada

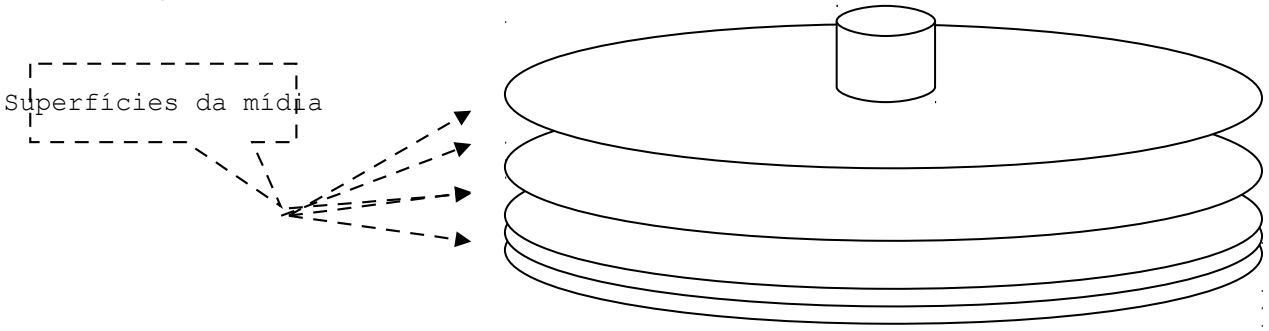
```
.-------------------------- VTOC Summary Information --------------------------.
| Volume . : D7V013                                                            |
| Unit . . : 3390                                                              |
|                                                                              |
|  Volume Data             VTOC Data              Free Space  Tracks   Cyls    |
|  Tracks . :   50,085     Tracks  . :     104    Size  . . :   5,584   358    |
|  %Used  . :       88     %Used . . :       4    Largest . : 1,395      93    |
|  Trks/Cyls:       15     Free DSCBS:   5,016                                 |
|                                                 Free Extents . :      61     |
'------------------------------------------------------------------------------'
```

3390 = MODELO MUITO USADO Capacidade aproximada = 50K trilhas * 27Kb/trilha = 1.4 Gb

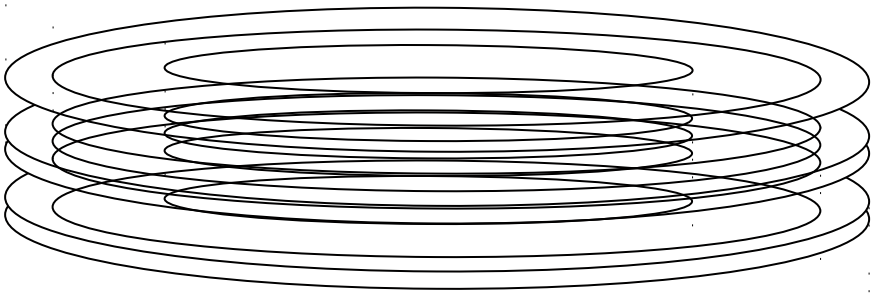
50.085 trilhas total no disco (104 usadas pela VTOC) 15 trilhas por cilindro

3.339 cilindros total no disco (358 livres + 2.981 usados)

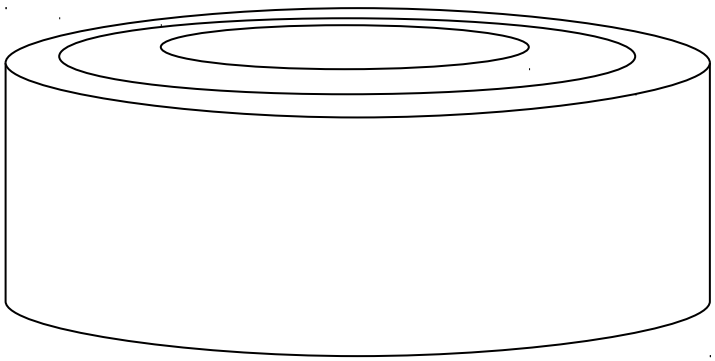
Trilhas, cilindros



Trilhas : Circunferências nas superfícies (faces) da mídia



Cilindro : Conjunto de trilhas equidistantes do eixo



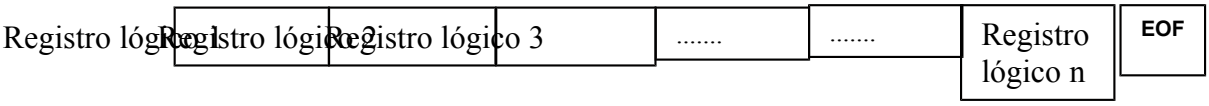
3.2 Organização de Arquivos

Principais : SAM, PAM, VSAM. Outras : **IAM (Innovation Access Method), DAM, ISAM, etc...**

3.2.1 SAM

Método de acesso = Sequential Access Method
Arquivos sequenciais = arquivos PS = physical sequential
Acesso e organização sequenciais
Organização : somente área com os registros de dados, gravados sequencialmente

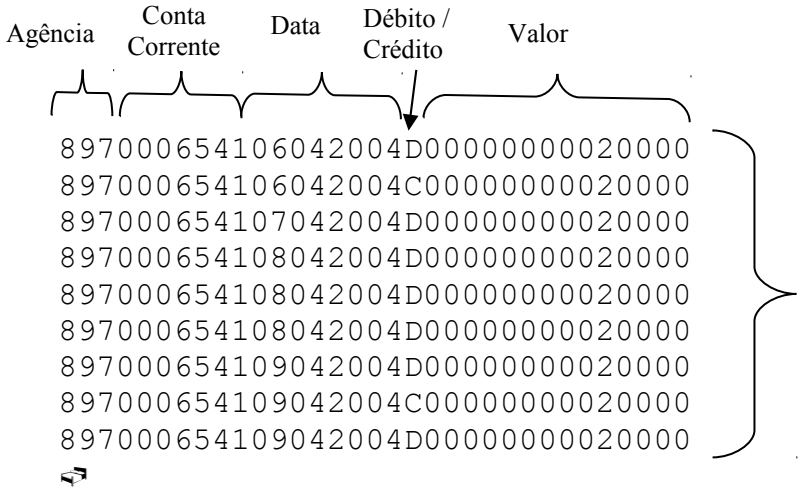
Para acessar (ler) um determinado registro, deve-se, obrigatoriamente, ler todos os anteriores. Portanto, não é possível fazer acesso direto a um determinado registro (acessá-lo diretamente, sem ler outros)



Obs.: EOF = End Of File = marca indicadora de fim de arquivo

Exemplo :

Arquivo com dados referentes a transações de débito ou crédito que um correntista efetuou em sua conta corrente num mes: cada registro lógico corresponde a uma transação.

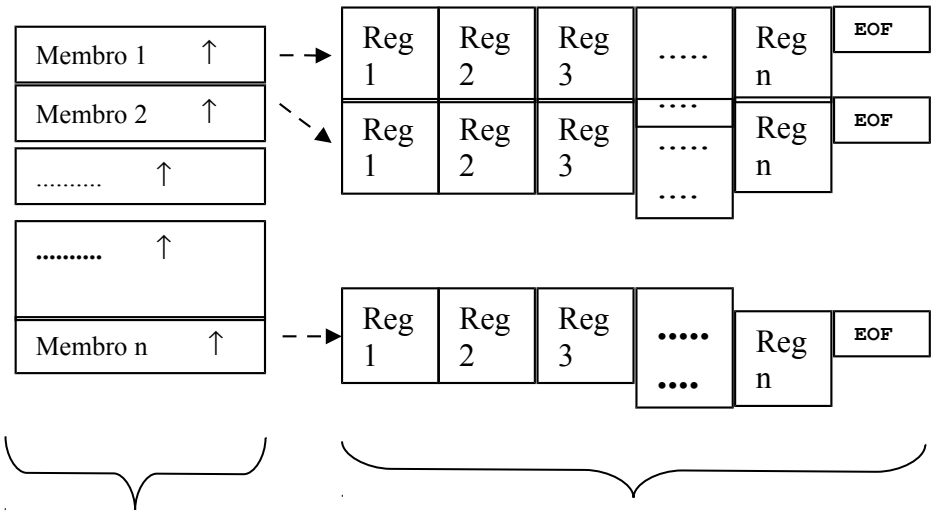


Cada linha corresponde a um registro lógico, e tem os dados referentes a uma transação de débito ou crédito.
Para acessar o 9º lançamento, é necessário ler todos os 8 anteriores.

3.2.2 PAM

Método de acesso = Partitioned Access Method
Arquivo particionado=PDS=partitioned data set
Acesso sequencial
Organização particionada (PO = partitioned organization) :
área para diretório (índice dos membros) +
n membros com dados

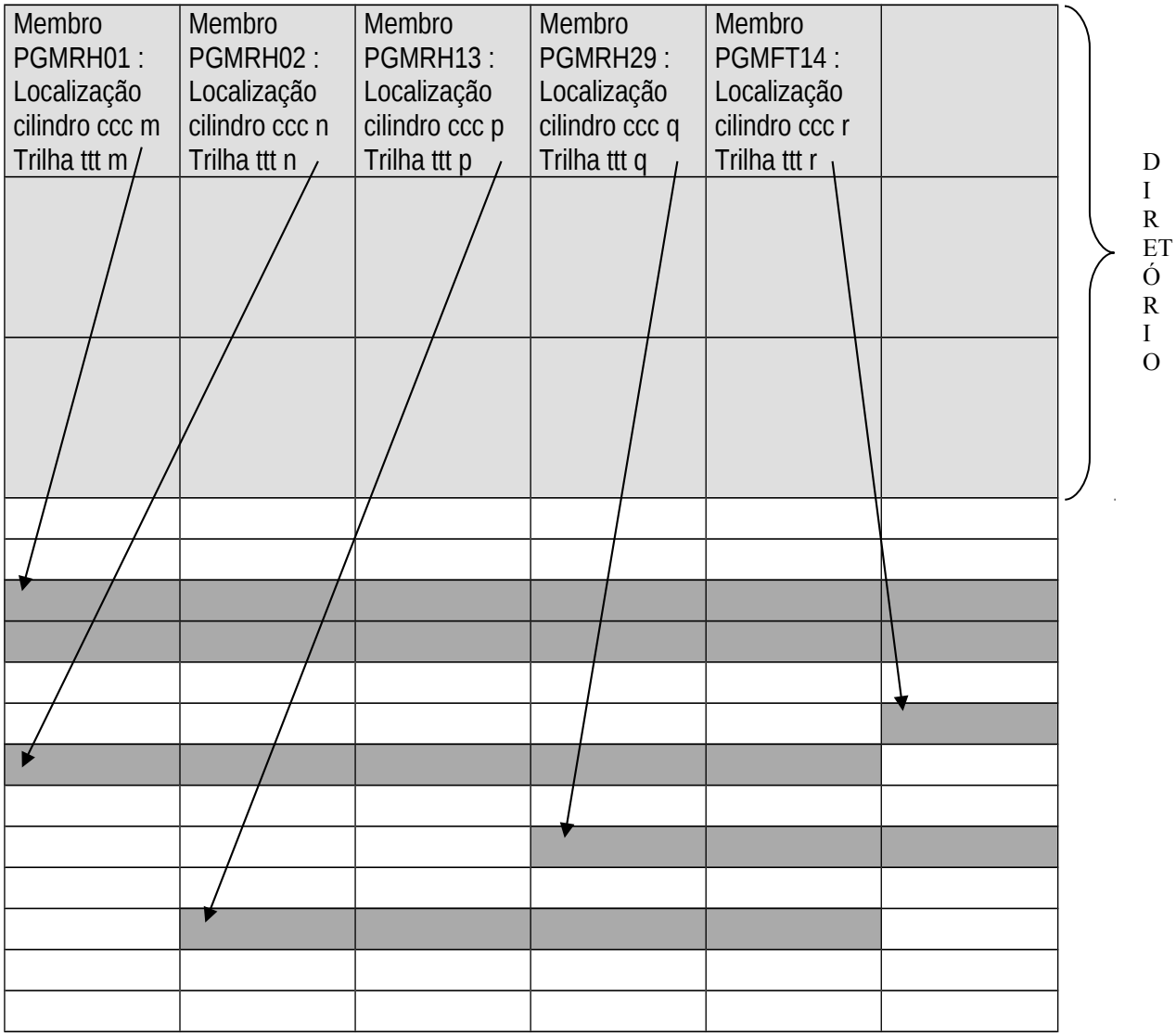
Para determinar a localização de um membro na área de dados, é feita uma pesquisa no diretório.
Uma vez localizado o membro, ele é acessado da mesma forma que um arquivo sequencial, ou seja, é feita a leitura de um registro lógico por vez.



Utilizações típicas de PDS :

- arquivo com programas executáveis (load modules); eles são colocados no PDS pelo linkeditor; são genericamente denominados loadlibs ou linklibs;
- arquivo com fontes de trabalho do TSO
- arquivo com movimentos diários; cada conjunto de registros referentes a um dia é um membro

Arquivo USRTST.LINKLIB



3.2.3 VSAM

VSAM = Virtual Storage Access Method

Tipos de arquivos VSAM :

- Sequenciais (tipo ESDS = entrie sequenced data set) ou
- Indexados (tipo KSDS = key sequenced data set) ou
- Relativos registros tamanho fixo (tipo RRDS = relative record data set)
- Relativos registros tamanho variável (tipo VRRDS = variable relative record data set)
- Lineares (tipo LDS = linear data set)

VSAM KSDS

(a) ORGANIZAÇÃO : área de dados + área(s) de índice(s) ; armazenamento por ordem de chave primária
(b) ACESSO : sequencial ; direto (via chave); direto (via RBA)
(c) ÍNDICES : pelo menos um (índice primário); pode haver outros (alternados = AIX)
(d) RBA : RBA pode ser mudado
(e) FREE SPACE : usado para inserir novos registros ou alterar o tamanho de registros existentes
(f) REGISTROS DELETADOS : o espaço passa a ser free space
(g) REGISTROS SPANNED : sim ; FORMATO EXTENDIDO : sim ; COMPRESSÃO : sim

VSAM ESDS

(a) ORGANIZAÇÃO : área de dados ; armazenamento por ordem de entrada / gravação
(b) ACESSO : sequencial; direto (via RBA)
(c) ÍNDICES : pelo menos um índice (primário); pode haver outros (alternados = AIX)
(d) RBA : RBA não pode ser mudado
(e) FREE SPACE : Usado para inserir novos registros
(f) REGISTROS DELETADOS : não se pode deletar um registro, mas o espaço que ele ocupa pode ser reusado (atualizado) para conter outro registro; em geral deleção l.
(g) REGISTROS SPANNED : sim ; FORMATO EXTENDIDO : sim

VSAM RRDS

(a) ORGANIZAÇÃO : Registros com tamanho fixo; área de dados (não em índice) ; armazenamento por ordem de RRN (relative record number = número do registro relativo)
(b) ACESSO : sequencial ou direto (via RRN)
(c) ÍNDICES : não há
(d) RRN : RRN não pode ser mudado
(e) FREE SPACE : (os slots vazios) são usados para inserir novos registros
(f) REGISTROS DELETADOS : A slot given up by a deleted record can be used

(g) REGISTROS SPANNED : não ; FORMATO EXTENDIDO : sim

VSAM VRRDS

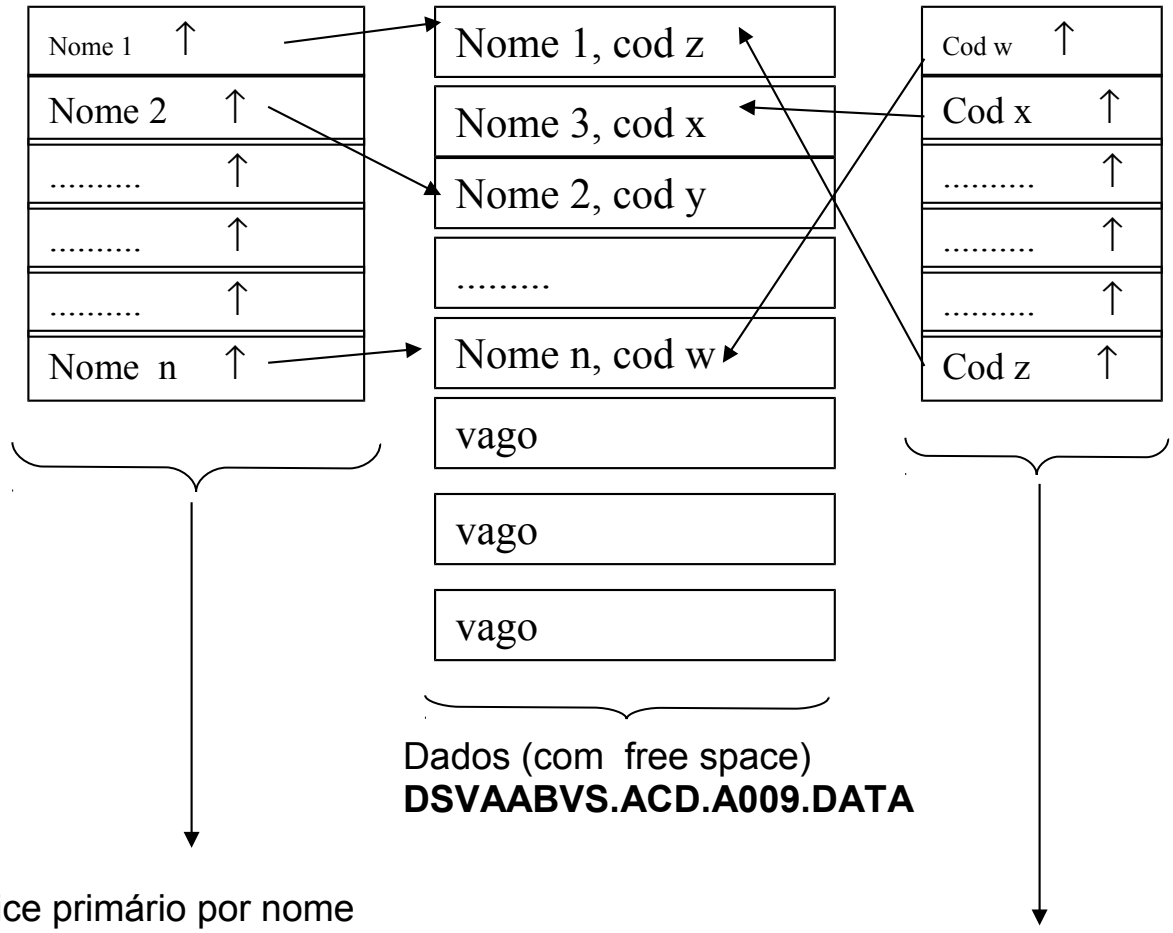
(a) ORGANIZAÇÃO : Registros com tamanho variável ; área de dados (não em índice); armazenamento por ordem de RRN
(b) ACESSO : sequencial ou direto (via RRN)
(c) ÍNDICES : não há
(d) RRN : RRN não pode ser mudado
(e) FREE SPACE : usado para inserir novos registros ou alterar o tamanho de registros existentes
(f) REGISTROS DELETADOS : seu espaço é transformado em free space
(g) REGISTROS SPANNED : não ; FORMATO EXTENDIDO : sim

VSAM LDS

(a) ORGANIZAÇÃO : área de dados ; não trabalha com (o conceito de) registros lógicos; portanto : não tem manipulação / acesso a registros lógicos
(b) ACESSO : sequencial ou direto (via DIV = Data-In-Virtual)
(c) ÍNDICES : não há
(d) RBA : não aplicável (não existe o conceito de registro lógico)
(e) FREE SPACE : não aplicável (não existe o conceito de registro lógico)
(f) REGISTROS DELETADOS : não aplicável (não existe o conceito de registro lógico)
(g) REGISTROS SPANNED : não ; FORMATO EXTENDIDO : sim

Ex.: VSAM KSDS

Dataset DSVAABVS.ACD.A009



Dados (com free space)
DSVAABVS.ACD.A009.DATA

Índice primário por nome

DSVAABVS.ACD.A009.INDEX

Índice alternado por código

DSVAABVS.ACD.A009.AIX1

3.3 Concatenação

2 ou mais arquivos (datasets) lidos como se fossem um único; válido somente para arquivos sequenciais; ao especificar os DD's, o primeiro DEVE ter o DDNAME e os demais NÃO PODEM ter o DDNAME. Exemplos:

```
//ENTRADA  DD  DSN=TST.PROPOST.D040501,DISP=SHR
//         DD  DSN=TST.PROPOST.D040502,DISP=SHR
//         DD  DSN=TST.PROPOST.D040503,DISP=SHR
```

Supondo que o arquivo `TST.PROPOST.D040501` tenha os seguintres registros :

```
00300100010001200404 01
00300100011277520040401
00300104022578200404 01
00300109220012200404 01
00300100200000200404 01
```

E que o arquivo `TST.PROPOST.D040502` tenha os seguintres registros :

```
00300100010001200404 02
00300100010001200404 02
00300100010001200404 02
```

E que o arquivo `TST.PROPOST.D040503` tenha os seguintres registros :

```
00300100155677200404 03
00300100023300200404 03
00300100500000200404 03
```

O programa que ler o arquivo `ENTRADA` declarado com a concatenação acima, lerá "um arquivo" único com o seguinte conteúdo :

```
00300100010001200404 01
00300100011277520040401
00300104022578200404 01
00300109220012200404 01
00300100200000200404 01
00300100010001200404 02
00300100010001200404 02
00300100010001200404 02
00300100155677200404 03
00300100023300200404 03
00300100500000200404 03
```

A condição de fim de arquivo somente será "detectada" quando for efetuada a tentativa de leitura do 4° registro do 3° arquivo da concatenação.

Outros exemplos :

```
//ENTRADA  DD  DSN=ARQSRC.SISRH(BOOKA001),DISP=SHR
//         DD  *
dados


//ARQI01   DD  *
dados
//         DD  DSN=ARQSRC.SISRH(PARMS11),DISP=SHR
//         DD  DSN=ARQPARMS.RH.D040501,DISP=SHR


//SYSLIB   DD  DSN=SYS1.COBLIB,DISP=SHR
//         DD  DSN=SYS2.COBLIB.TESTE,DISP=SHR
//         DD  DSN=SYS2.COBLIB.ACEITE,DISP=SHR
//         DD  DSN=SYS2.COBLIB.PROD,DISP=SHR
```

3.4 Catálogos

É um local para arquivamento de dsnames, onde constam o dsname e o respectivo volser e onde o dataset está fisicamente.

Um arquivo catalogado, ao ser referenciado apenas por seu DSNAME, é localizado pelo sistema operacional. Se não estiver catalogado, para ser referenciado / encontrado é necessário que seja indicada em qual unidade está (UNIT=xxxxx,VOL=SER=yyyyyy).

Catalogando arquivos, portanto, podemos simplesmente referenciá-lo através de seu nome (dsname), o que basta para que ele seja localizado pelo sistema operacional.

Catálogo UCATST

Arquivo	Volser
TST.ARQA01.RH	WRK001
TST.ARQBETA	WRK333
TST.FILETST.CLIENTE.X	WRK111
Etc...	

Catálogo UCATP1

Arquivo	Volser
PROD.$TABELAS.ARQA01	PRD766
PROD.$TABELAS.ARQ2	PRD321
PROD.CLIENTES.Y	PRD422
Etc...	

Catálogo UCATH2

Arquivo	Volser
HOML.$TABELAS.ARQA01	HOML15
HOML.$TABELAS.ARQ2	HOML57
HOML.CLIENTES.Y	HOML44
Etc...	

A criação de catálogos normalmente é feita pelo pessoal de suporte.

3.5 GDG (Generation Data Group)

Generation Data Group = recurso do sistema operacional que permite a existência de diversas versões de um arquivo, com um DSNAMEs (quase) iguais. Exemplos :

```
//EPCASLBS DD  DSN=ATAAABPS.EPC.SE64(+1),
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(100,50)),
//             UNIT=SYSDA,
//             DCB=LRECL=83
```

Se é a primeira "versão" do arquivo, o DSNAME com o qual ele será gravado será ATAAABPS.EPC.SE64.G0001V00

Se é a segunda versão do arquivo, o DSNAME com o qual ele será gravado será ATAAABPS.EPC.SE64.G0002V00

```
//EPCASLBS DD  DSN=ATAAABPS.EPC.SE64(0),DISP=SHR
```

Supondo que existam os arquivos
ATAAABPS.EPC.SE64.G0001V00
ATAAABPS.EPC.SE64.G0002V00
ATAAABPS.EPC.SE64.G0003V00
"pega" o arquivo ATAAABPS.EPC.SE64.G0003V00 (último nível) para usar.

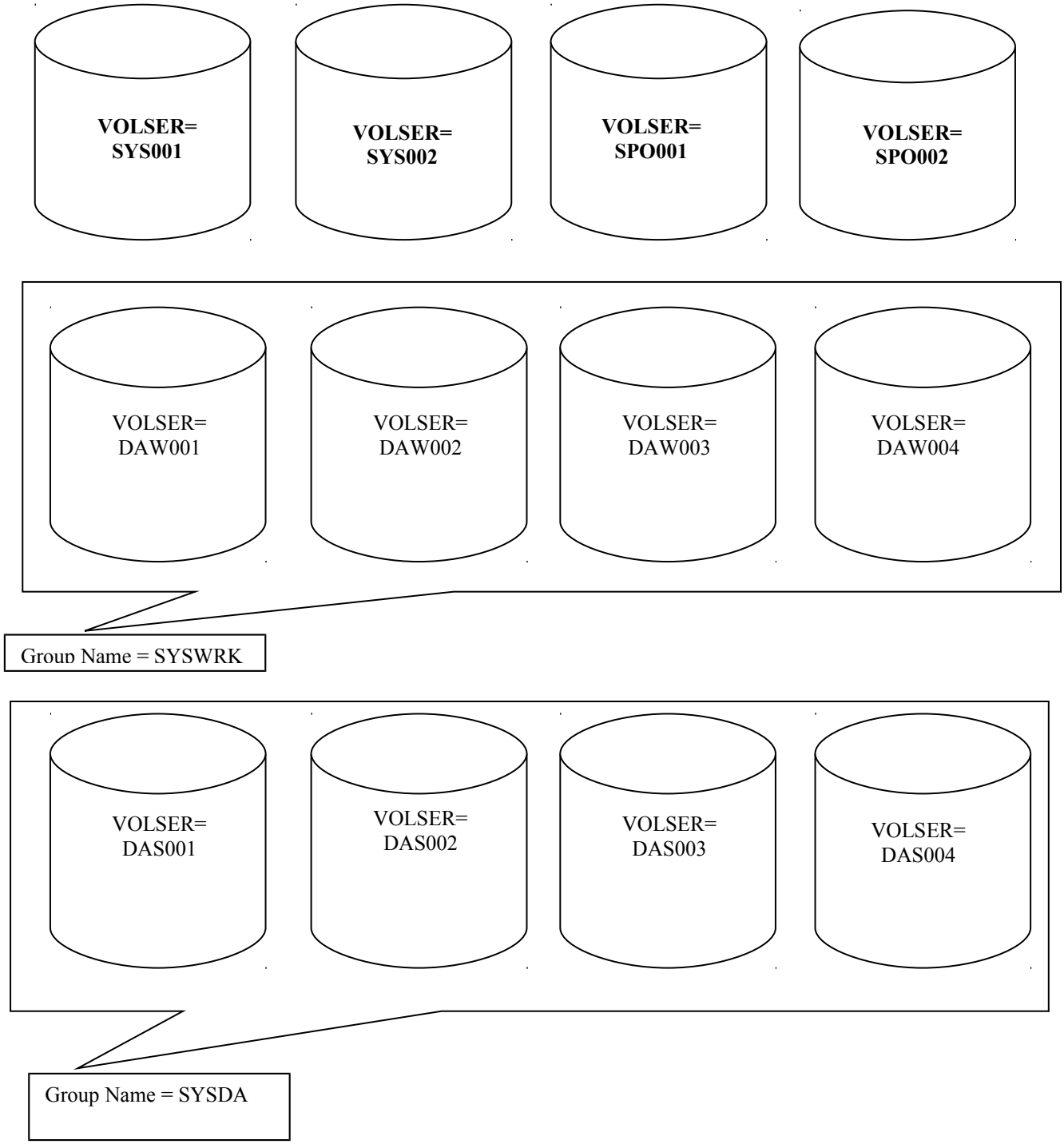
```
//EPCASLBS DD  DSN=ATAAABPS.EPC.SE64(0),DISP=SHR
```

Supondo que existam os arquivos
ATAAABPS.EPC.SE64.G0001V00
ATAAABPS.EPC.SE64.G0002V00
ATAAABPS.EPC.SE64.G0003V00
"pega" o arquivo ATAAABPS.EPC.SE64.G0003V00 (último nível) para usar.

A criação de um GDG pode ser feita pelo Roscoe (DSN, opção DG), pelo TSO (opção 3 / 2 / V / 1 – 4) ; em algumas instalações ela pode ser feita somente pelo pessoal de produção.

3.6 Group Names

Conjunto de discos reunidos sob um mesmo nome (nome do grupo).
Quando um espaço (para alocação) é solicitado em determinado group name, o a busca de espaço disponível para alocação será feita pelo sistema operacional somente nos discos que pertencerem ao group name.



A criação de um group names é normalmente feita pelo pessoal de suporte.

4. Utilitários Batch

4.1 IDCAMS

O Access Method Service (AMS) é um programa utilitário para manipular arquivos VSAM.

DDNAMES - Arquivos :

`SYSIN` = arquivo com os statements de controle para indicar as ações desejadas
`SYSPRINT` = listagem dos statements e resultado das ações

Comandos :

Para especificar a(s) ação(ões) desejadas, pode-se utilizar os commandos funcionais ou modais.

Comandos funcionais : usados para :

`ALTER`	altera informações do catálogo VSAM, do cluster, do(s) índice(s) alternado(s) ou path(s)
`BLDINDEX`	Cria (constrói) índice alternado
`DEFINE AIX`	Define índice alternado (aloca espaço e respectivos dados de controle em disco)
`DEFINE CLUSTER`	Define cluster (aloca espaço e respectivos dados de controle em disco)
`DEFINE PATH`	Define path relacionando cluster e índice alternado (aloca espaço e respectivos dados de controle em disco)
`DELETE`	Deleta entrada de catálogo VSAM, cluster, aix ou path
`LISTCAT`	Lista informações relacionadas a um dataset
`PRINT`	Lista conteúdo (dados) de um dataset
`REPRO`	Copia dados de um arquivo para outro

Comandos modais : usados para execução condicional de comandos funcionais
IF e SET

Exemplos

DELETE – Exemplo

```
//STEP1    EXEC  PGM=IDCAMS
//SYSOUT     DD  SYSOUT=*
//SYSPRINT   DD  SYSOUT=*
//SYSIN DD *
   DELETE  DSVAABVS.EPC.ALBERTO.A001 PURGE
```

LISTCAT – Exemplo

```
//STEP1 EXEC PGM=IDCAMS
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
 LISTCAT  ENTRIES(DSVAABVS.RPC.A011) ALL
```

PRINT – Exemplo

```
//PRINT    EXEC PGM=IDCAMS,COND=(0,NE)
//ENT       DD DSN=DSVAABVS.EPC.A005,DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
  PRINT INFILE (ENT) COUNT (100)

//PRINT2   EXEC PGM=IDCAMS,COND=(0,NE)
//ENT      DD DSN=DSVAABVS.EPC.A005,DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
  PRINT INDATASET (DSVAABVS.EPC.A005) COUNT (100)
```

REPRO – Exemplo

```
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//ENTRADA  DD *
...
//SAIDA    DD DSN=DSVAABPS.EPC.ALBERTO.EXEMPPO(XYZ),DISP=SHR
//SYSIN DD *
 REPRO INFILE(ENTRADA) OUTFILE(SAIDA)
```

DEFINE CLUSTER - Exemplo

```
//*--------------------------------------
//STEP1    EXEC  PGM=IDCAMS
//SYSOUT     DD  SYSOUT=*
//SYSUDUMP   DD  SYSOUT=*
//SYSPRINT   DD  SYSOUT=*
//SYSABOUT   DD  SYSOUT=*
//SYSIN DD *
   DELETE  DSVAABVS.EPC.ALBERTO.A001 PURGE
   IF MAXCC NE 0 THEN SET MAXCC = 0
   DEFINE CLUSTER             -
 ( NAME('DSVAABVS.EPC.ALBERTO.A001')  -
   INDEXED                    -
   FREESPACE(20 10)           -
   KEYS(20 0)                 -
   RECORDSIZE(080 080)        -
   NOREPLICATE                -
   NOREUSE                    -
   SHAREOPTIONS(2 3)          -
            )    /* END OF CLUSTER */  -
          DATA                          -
 ( NAME('DSVAABVS.EPC.ALBERTO.A001.DATA')     -
   TRACKS(5 1)                       -
   CISZ(4096)                           -
   SPEED                                -
            )    /* END OF DATA */     -
          INDEX                         -
 ( NAME('DSVAABVS.EPC.ALBERTO.A001.INDEX')    -
   TRACKS(1 1)                          -
   CISZ(2048)                           -
            )    /* END OF INDEX */
   IF MAXCC NE 0 THEN SET MAXCC EQ 0
//*--------------------------------------
//STEP2 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER -
(NAME (STUD.DATA.KSDS) -
VOLUME (USER01) -
CYLINDERS (1 2) -
CISZ (4096) -
KEYS (8 0) -
INDEXED -
RECORDSIZE (80 80))
/*
```

DEFINE ALTERNATEINDEX ou DEFINE AIX - Exemplo

```
//STEP1    EXEC  PGM=IDCAMS
     //SYSPRINT DD    SYSOUT=A
     //SYSIN    DD    *
          DEFINE ALTERNATEINDEX -
                (NAME(EXAMPLE.AIX) -
                 RELATE(EXAMPLE.KSDS2) -
                 KEYS(3 0) -
                 RECORDSIZE(40 50) -
                 VOLUMES(VSER01) -
                 CYLINDERS(3 1) -
                 NONUNIQUEKEY -
                 UPGRADE) -
                CATALOG(USERCAT)


//STEP1   EXEC  PGM=IDCAMS,COND=(0,NE)
//SYSOUT     DD  SYSOUT=*
//SYSPRINT   DD  SYSOUT=*
//SYSIN      DD  *

     DEFINE AIX (NAME(DSVAABVS.ADP.A004.AIX1)      -
                RELATE(DSVAABVS.ADP.A004)      -
                    RECSZ(26 26)               -
                    KEYS(10 11)          -
                    FSPC(10 10)           -
                    CYL(10 5)           -
                    SPEED               -
                    UPGRADE             -
                    UNIQUEKEY           -
                    SHR(2 3))           -
               DATA (NAME(DSVAABVS.ADP.A004.AIX1.DATA) -
                    CISZ(4096))            -
              INDEX (NAME(DSVAABVS.ADP.A004.AIX1.INDEX) -
                    CISZ(2048))
```

DEFINE PATH - Exemplo

```
     DEFINE PATH(NAME(DSVAABVS.ADP.A004.PATH.AIX1)      -
            PATHENTRY(DSVAABVS.ADP.A004.AIX1))
```

BLDINDEX – Exemplo

```
//S4      EXEC  PGM=IDCAMS,COND=(0,NE)
//SYSOUT     DD  SYSOUT=*
//SYSPRINT   DD  SYSOUT=*
//DD1  DD    DSN=DSVAABVS.ADP.A004,DISP=SHR
//DD2  DD    DSN=DSVAABVS.ADP.A004.AIX1,DISP=SHR
//SYSIN      DD  *
  BLDINDEX INFILE(DD1)  OUTFILE(DD2)
```

`ALTER` – Altera atributos de um dataset ou catálogo VSAM.

`DELETE` – Deleta objetos VSAM.

`DELETE` (*entryname*[*entryname*...]) objeto parâmetros

entryname : nome do(s) dataset(s) a deletar (pode-se usar wildcards)

objeto : identificação do(s) objeto(s) a deletar. Pode ser :

```
ALIAS
ALTERNATEINDEX
CLUSTER
GENERATIONDATAGROUP
LIBRARYENTRY
NONVSAM
NVR
PAGESPACE
PATH
TRUENAME
USERCATALOG
VOLUMEENTRY
VVR
```

Parâmetros :

CATALOG(*catname*) ou CAT : indica o catálogo

ERASE ou ERAS : indica que a area de um cluster ou índice alternado deve ser gravada com zeros binarios se deletada

NOERASE ou NERAS : indica que a area de um cluster ou índice alternado não deve ser gravada com zeros binarios se deletada

FILE(*ddname*) : indica o ddname que declara o objeto a deletar

FORCE|*NOFORCE*
specifies whether entries that are not empty should be deleted.

FORCE ou FRC : indica que a deleção deve ser feita sem que seja assegurado que o dataset esteja vazio

NOFORCE ou NFRC : indica que a deleção não deve ser feita sem que seja assegurado que o dataset esteja vazio

PURGE ou PRG : indica que o objeto deve ser deletado mesmo que a data de expeiração não tenha sido alcançada

NOPURGE ou NPRG : indica que o objeto não deve ser deletado se a data de expeiração não tenha sido alcançada

RECOVERY ou RCVRY : indica que se o objeto for um catálogo do usuário, ele deve ser substituido por uma cópia backup.

NORECOVERY ou NRCVRY : indica que não deve ser substituido

SCRATCH ou SCR : indica que a entrada do dataset deve ser removida da VTOC

NOSCRATCH ou NSCR : indica que a entrada do dataset deve ser removida do catálogo (não há necessidade de montar o volume)

LISTCAT – Lista características e atributos de objetos

LISTCAT (*entryname*[*entryname*...]) *objeto parâmetros*

entryname : nome do objeto

objeto : identificação do(s) objeto(s) a deletar. Pode ser :

ALIAS
ALTERNATEINDEX ou AIX
CLUSTER ou CL
DATA
GENERATIONDATAGROUP ou GDG
INDEX ou IX
LIBRARYENTRIES(*libentries*) ou LIBENTRIES ou LIBENT
NONVSAM ou NVSAM
PAGESPACE ou PGSPC
PATH
USERCATALOG ou UCAT
VOLUMEENTRIES(*volume*) ou VOLENTRIES ou VOLENT

Parâmetros :

CATALOG(*catname*) ou CAT : nome do catálogo

CREATION(*days*) ou CREAT : só listar se a data de criação for igual ou anterior à data atual - *days*

ENTRIES(*entryname*[*entryname*...]) : nome do objeto a listar

LEVEL(*level*) ou LVL : listar as entradas com o nível indicado.

EXPIRATION(*days*) ou EXPIR : só listar se a data de expiração for igual ou anterior à data atual + *days*

FILE(*ddname*) : nome do DD que identifica os columes que contém o VVDS a listar.

LIBRARY*(libname)* ou LIB : nome da tape library entrie cujos volumes devem ser listados

NAME|HISTORY|VOLUME|ALLOCATION|ALL : quais campos devem ser listados

OUTFILE(*ddname*) ou OFILE : nome do dd onde, alternativamente à SYSPRINT, a saída será gravada.

`PRINT` – Lista o conteúdo de datasets

INFILE(*ddname*) ou IFILE : indica o ddname do dataset (ou objeto) alistar. Se for o membro de um PDS, o DD deve especificar *dsname*(membername). Se for uma listagem de um KSDS por chave alternada, o DD deve especificar o dsname do PATH. Mutuamente exclusivo com o INDATASET.

INDATASET(*entryname*) ou IDS : indica o dsname do dataset (ou objeto) a listar. Mutuamente exclusivo com o INFILE.

CHARACTER|*DUMP*|HEX : indica o formato de listagem. Assume DUMP se não especificado.

FROMKEY(*key*) ou FKEY : chave do primeiro registro a listar.

FROMADDRESS(*address*) ou FADDR : RBA do primeiro registro a listar.

FROMNUMBER(*number*) ou FNUM : relative record number do primeiro registro a listar (para RRDS)

SKIP(*number*) : pular *number* registros antes de começar a listar

OUTFILE(*ddname*) ou OFILE : arquivo onde a saída sera listada, alternativamente à SYSPRINT

TOKEY(*key*)|TOADDRESS(*address*)| TONUMBER(*number*)|COUNT(*number*) : chave / endereço / número do registro / qtd regs que indica o fim da listagem

TOKEY(*key*) : chave do ultimo registro a listar

TOADDRESS(*address*) ou TADDR : RBA do ultimo registro a listar

TONUMBER(*number*) ou TNUM : numero do ultimo registro a listar

COUNT(*number*) : quantidade de registros a listar

REPRO – Copia datasets

INFILE(ddname) ou IFILE : indica o ddname do dataset (ou objeto) alistar. Se for o membro de um PDS, o DD deve especificar dsname(membername). Se for uma listagem de um KSDS por chave alternada, o DD deve especificar o dsname do PATH. Mutuamente exclusivo com o INDATASET.

INDATASET(entryname) ou IDS : indica o dsname do dataset (ou objeto) a listar. Mutuamente exclusivo com o INFILE.

OUTFILE(ddname) ou OFILE : indica o ddname do datset de saída.

OUTDATASET(entryname) ou ODS : indica o dsname do dataset de saída.

ENTRIES(entryname[entryname...]) ou ENT : nome do objeto a mergear

LEVEL(level) ou LVL : indica que as entradas que tiverem o nível indicado devem ser mergeadas .

ERRORLIMIT(value) ou ELIMIT : indica o número máximo de erros permitido antes que o IDCAMS cancele a ação.

FILE(ddname) : para VVDS.

FROMKEY(key) ou FKEY : chave do primeiro registro a copiar

FROMADDRESS(address) ou FADDR : RBA do primeiro registro a copiar.

FROMNUMBER(number) ou FNUM : relative record number do primeiro registro a copiar (para RRDS)

SKIP(number) : pular number registros antes de começar a copiar

MERGECAT ou MRGC: para efetuar merge de catálogos

NOMERGECAT ou NOMRGC: para copiar dados de um catálogo para outro

REPLACE ou REP : indica que se um KSDS é copiado, se um registro com chave já existente no arquivo de saída existir no arquivo de entrada, este deve substituir o de saída.

NOREPLACE ou NREP : indica que se um KSDS é copiado, se um registro com chave já existente no arquivo de saída existir no arquivo de entrada, o já existente no arquivo de saída permanece e não é substituído pelo de entrada.

REUSE ou RUS : indica que o dataset de saída deve ser aberto como REUSABLE, independentemente de ter o parâmetro REUSE na sua definição.

NOREUSE ou NRUS : indica que o dataset de saída deve ser aberto como não REUSABLE.

TOKEY(key) : chave do ultimo registro a copiar

TOADDRESS(address) ou TADDR : RBA do ultimo registro a copiar

TONUMBER(number) ou TNUM : numero do ultimo registro a copiar

COUNT(number) : quantidade de registros a copiar

VOLUMEENTRIES(entryname) ou VOLENTRIES ou VOLENT : indica o nome do catálogo de fitas a ser merged ou copiado.

`DEFINE ALTERNATEINDEX ou DEFINE AIX –` define (aloca) um AIX

Parâmetros :

NAME : indica o nome do índice alternado. Ex.: EXAMPLE.AIX.

RELATE : indica o cluster. Ex.: EXAMPLE.KSDS2.

KEYS : indica o tamanho e a posição (relativa a zero) da chave no registro

RECORDSIZE : indica o tamanho do registro lógico

VOLUMES : indica o VOLSER (identificação do disco) onde deve ser alocado o índice alternado

CYLINDERS : indica o espaço que o índice alternado deve ter.

NONUNIQUEKEY : indica que pode haver chaves duplicadas

UPGRADE : indica que o índice alternado deve ser aberto pelo VSAM, e atualizado toda vez que o cluster for aberto para processamento

CATALOG : indica que o índice alternado deve constar do user catalog

ATTEMPTS(n) ou ATT : indica o número de tentativas que o operador pode fazer para entrar com a senha; valores=0 a 7; default=2

AUTHORIZATION(*entrypoint*[*string*]) ou AUTH : indica nome de aplicativo para validação adicional de senha

BUFFERSPACE(*n*) ou BUFSP ou BUFSPC : indica tamanho mínimo dos buffers; pode ser especificado em decimal (n) ou hexa (X'nnnnnn') ou binário (B'nnnnnn...nn'); mínimo = 2 * control_interval_size; máximo=16MB

CATALOG(*catname*) ou CAT : indica o nome do catálogo

CODE(*code*) : indica um nome-código que identifica o índice alternado nas mensagens de senha inválida

CONTROLINTERVALSIZE(*size*) ou CISZ or CNVSZ : indica o tamanho do control interval do índice alternado; se omitido, o VSAM determina um valor, múltiplo de 512 ou 2048.

DATACLASS(*class*) ou DATACLAS : indica o nome da classe usada para referenciar atributos do dataset.

ERASE ou ERAS : indica que a área do índice alternado deve ser gravada com zeros binários quando o índice for deletado.

NOERASE ou NERAS : indica que a área do índice alternado não deve ser gravada com zeros binários quando o índice for deletado.

EXCEPTIONEXIT(*entrypoint*) ou EEXT : indica o nome do aplicativo para tratamento de I/O error.

FILE(*ddname*) : indica o ddname que faz referência ao volser onde deve ser alocado o índice alternado

FREESPACE(*CI-percent*[*CA-percent*]|*0 0*) ou FSPC : indica o tamanho do freespace. Se omitido, assume zero.

KEYRANGES((*lowkey highkey*)[(*lowkey highkey*)...]) ou KRNG : indica as faixas de valores das chaves que devem ser distribuidas por volumes distintos.

KEYS(*length offset*| *64 0*) : indica o tamanho e a posição (relative a zero) da chave nos registros)

MODEL(*entryname*[*catname*]) : indica para usar um índice alternado como modelo deste

ORDERED ou ORD : indica que os volumes (discos) devem ser usados na ordem especificada no parâmetro VOLUMES

UNORDERED ou UNORD : indica que os volumes (discos) não precisam ser usados na ordem especificada no parâmetro VOLUMES

OWNER(*ownerid*) : indica o proprietário do índice alternado

RECATALOG ou RCTLG : indica que as entradas de catálogo devem ser recriadas se entradas VVDS válidas forem achadas no volume primário VVDS

NORECATALOG ou NRCTLG : indica que as entradas de catálogo não devem ser recriadas se entradas VVDS válidas forem achadas no volume primário VVDS

RECORDSIZE(*average maximum*| *4086 32600*) ou RECSZ : indica o tamanho do registro.

REPLICATE ou REPL : indica que um registro do índice deve ser replicado na trilha até seu tamanho máximo, para aumentar desempenho (< rotational delay)

NOREPLICATE ou NREPL : indica que não deve haver a replicação

REUSE ou RUS : indica o reaproveitamento do índice alternado na construção de outro.

NOREUSE ou NRUS : indica para não fazer REUSE

SHAREOPTIONS(*crossregion*[*crosssystem*]|*1 3*) ou SHR : indica o tipo de compartilhamento.

crossregion : indica o compartilhamento do dataset entre aplicações num mesmo sistema
1 : o dataset pode ser acessado por qualquer número de usuários para READ **ou** pode ser acessado por um único usuário para READ e WRITE (UPDATE).
2 : o dataset pode ser acessado por qualquer número de usuários para READ **e** pode ser acessado por um único usuário para WRITE ; é responsabilidade do aplicativo que lê cuidar da integridade dos dados lidos.
3 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; é responsabilidade do aplicativo que lê ou grava cuidar da integridade dos dados.

4 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; para cada operação, é feita atualização dos buffers de I/O

crosssystem : indica o compartilhamento do dataset entre aplicações de diversos sistemas
1 : reservado
2 : reservado
3 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; é responsabilidade do aplicativo que lê ou grava cuidar da integridade dos dados.
4 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; para cada operação, é feita atualização dos buffers de I/O

SPEED : não formata o espaço de dados

RECOVERY ou RCVY : indica que as areas de controle dos dados devem ser formatadas

TO(*date*)|FOR(*days*) : indica o período de retenção (data após a qual pode ser deletado) através de uma data (*date*) ou qtdd dias (*days*)

UNIQUEKEY ou UNQK : indica que não pode haver chave duplicada (mais de um registro com a mesma chave)

NONUNIQUEKEY ou NUNQK : indica que pode haver chave duplicada (mais de um registro com a mesma chave – no máximo 32768 registros)

UPGRADE ou UPG : indica que o índice alternado deve ser atualizado quando os dados da base forem atualizados (incluidos, atualizados ou excluidos)

NOUPGRADE ou NUPG : indica que o índice alternado não deve ser atualizado quando os dados da base forem atualizados

WRITECHECK ou WCK : indica que após um WRITE deve ser feito um READ sem data transfer para teste

NOWRITECHECK ou NWCK : indica que após um WRITE não deve ser feito um READ sem data transfer para teste

`BLDINDEX` – constrói um índice alternado (AIX)

INFILE(*ddname*) ou IFILE : indica o ddname do dataset com os dados que darão origem ao índice alternado

INDATASET(*entryname*) ou IDS : indica o dsname do dataset com os dados que darão origem ao índice alternado

OUTFILE(*ddname*[*ddname*...]) ou OFILE : indica o ddname do DD que especifica o índice alternado.

OUTDATASET(*entryname*[*entryname*...]) ou ODS : indica o dsname do DD que especifica o índice alternado.

CATALOG(*catname*) ou CAT : indica o catálogo que sera usado para os arquivos de trabalho.

EXTERNALSORT ou ESORT : indica que devem ser usados 2 arquivos ESDS como arquivos de trabalho (especificar 2 DD's com os ddname IDCUT1 e IDCUT2)

INTERNALSORT ou ISORT : indica que deve ser usada a memória virtual para efetuar a classificação.

SORTCALL : indica que o AMS deve usar o DFSORT para classificar os dados para gerar o índice.

NOSORTCALL : indica para o AMS deve usar seu algoritmo interno de sort.

SORTDEVICETYPE*(device type)* ou SORTDVT SDVT : indica o tipo de device que deve ser usado pelo DFSORT (no caso de SORTCALL)

SORTFILENUMBER*(number)* ou SORTFN SFN : indica a quantidade de arquivos de trabalho do DFSORT (no caso de SORTCALL)

SORTMESSAGEDD*(ddname)* ou SORTMDD SMDD : indica o ddname do dataset de mensagens do DFSORT (no caso de SORTCALL)

SORTMESSAGELEVEL({ALL|*CRITICAL*|NONE}) ou SORTML SML : indica o nível de mensagens que devem ser exibidas no arquivo de mensagens do DFSORT.

WORKFILES(*ddname ddname*) ou WFILE : indica o(s) ddname(s) do(s) arquivo(s) de trabalho do sort.

`DEFINE CLUSTER` – define (aloca) um dataset VSAM

Visão geral :

```
DEFINE CLUSTER

(NAME(entryname)
  espaço
  parâmetros_cluster
 DATA (
  espaço
  parâmetros_área_dados
 INDEX (
  espaço
  parâmetros_área_índice
```

Parâmetros :

NAME(*entryname*) : dsname do cluster

CYLINDERS(*primary*[*secondary*]) ou CYL ou
KILOBYTES(*primary*[*secondary*]) ou KB ou
MEGABYTES(*primary*[*secondary*]) ou MB ou
RECORDS(*primary*[*secondary*]) ou REC ou
TRACKS(*primary*[*secondary*]) ou TRK
indica o espaço primário e secundário a ser alocado para o cluster ou para a área de dados ou para a area de índice

VOLUMES(*volser*[*volser*...]) ou VOL : indica o(s) id(s) do(s) volume(s) onde deve ser feita a alocação do cluster, da área de dados ou da area de índice

ATTEMPTS(*number*|*2*) ou ATT : indica o número de tentativas que o operador pode fazer para entrar com a senha; valores=0 a 7; default=2

AUTHORIZATION(*entrypoint*[*string*]) ou AUTH : indica nome de aplicativo para validação adicional de senha

BUFFERSPACE(*size*) ou BUFSP ou BUFSPC : indica tamanho mínimo dos buffers; pode ser especificado em decimal (n) ou hexa (X'nnnnnn') ou binário (B'nnnnnn...nn'); mínimo = 2 * control_interval_size; máximo=16MB

CATALOG(*catname*) ou CAT : indica o nome do catálogo onde deve ser colocado o cluster

CODE(*code*) : indica um nome-código que identifica o índice alternado nas mensagens de senha inválida

CONTROLINTERVALSIZE(*size*) ou CISZ or CNVSZ : indica o tamanho do control interval do índice alternado; se omitido, o VSAM determina um valor, múltiplo de 512 ou 2048.

is the size of the control interval for the cluster or component.

DATACLASS(*class*) ou DATACLAS : indica o nome da classe usada para referenciar atributos do dataset.

ERASE ou ERAS : indica que a área do índice alternado deve ser gravada com zeros binários quando o índice for deletado.

NOERASE ou NERAS : indica que a área do índice alternado não deve ser gravada com zeros binários quando o índice for deletado.

EXCEPTIONEXIT(*entrypoint*) ou EEXT : indica o nome do aplicativo para tratamento de I/O error.

FILE(*ddname*) : indica o ddname que faz referência ao volser onde deve ser alocado

FREESPACE(*CI-percent*[*CA-percent*]|*0 0*) ou FSPC : indica o tamanho do freespace. Se omitido, assume zero.

INDEXED|LINEAR|NONINDEXED|NUMBERED : indica o tipo de dataset. Para KSDS é INDEXED; para ESDS é NONINDEXED; para RRDS é NUMBERED;

KEYRANGES((*lowkey highkey*)[(*lowkey highkey*)...]) ou KRNG : indica as faixas de valores das chaves que devem ser distribuidas por volumes distintos.

KEYS(*length offset*| *64 0*) : indica o tamanho e a posição (relative a zero) da chave nos registros)

MODEL(*entryname*[*catname*]) : indica para usar um índice alternado como modelo deste

OWNER(*ownerid*) : indica o proprietário do índice alternado

RECATALOG ou RCTLG : indica que as entradas de catálogo devem ser recriadas se entradas VVDS válidas forem achadas no volume primário VVDS

NORECATALOG ou NRCTLG : indica que as entradas de catálogo não devem ser recriadas se entradas VVDS válidas forem achadas no volume primário VVDS

RECORDSIZE(*average maximum*| *4086 32600*) ou RECSZ : indica o tamanho do registro.

REUSE ou RUS : indica o reaproveitamento do índice alternado na construção de outro.

NOREUSE ou NRUS : indica para não fazer REUSE

SHAREOPTIONS(*crossregion*[*crosssystem*]|*1 3*) ou SHR : indica o tipo de compartilhamento.

crossregion : indica o compartilhamento do dataset entre aplicações num mesmo sistema
1 : o dataset pode ser acessado por qualquer número de usuários para READ **ou** pode ser acessado por um único usuário para READ e WRITE (UPDATE).
2 : o dataset pode ser acessado por qualquer número de usuários para READ **e** pode ser acessado por um único usuário para WRITE ; é responsabilidade do aplicativo que lê cuidar da integridade dos dados lidos.

3 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; é responsabilidade do aplicativo que lê ou grava cuidar da integridade dos dados.
4 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; para cada operação, é feita atualização dos buffers de I/O

crosssystem : indica o compartilhamento do dataset entre aplicações de diversos sistemas
1 : reservado
2 : reservado
3 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; é responsabilidade do aplicativo que lê ou grava cuidar da integridade dos dados.
4 : o dataset pode ser acessado por qualquer número de usuários para qualquer operação; para cada operação, é feita atualização dos buffers de I/O

SPEED : não formata o espaço de dados

RECOVERY ou RCVY : indica que as areas de controle dos dados devem ser formatadas

TO(*date*)|FOR(*days*) : indica o período de retenção (data após a qual pode ser deletado) através de uma data (*date*) ou qtdd dias (*days*)

WRITECHECK ou WCK : indica que após um WRITE deve ser feito um READ sem data transfer para teste

NOWRITECHECK ou NWCK : indica que após um WRITE não deve ser feito um READ sem data transfer para teste

DEFINE PATH

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/DGT1I205/CONTENTS?SHELF=&DT=19981130115827#14.1

4.2 SORT / MERGE

Utilitário para efetuar classificação (SORT) ou intercalação (MERGE) ou cópia (COPY) de arquivos.

DDNAMES - Arquivos :

```
SYSIN      = arquivo com os statements de controle para indicar como o SORT / MERGE será
executado
SYSOUT     = listagem dos statements conforme entendidos pelo SORT e das ocorrências
SORTDIAG   = listagem dos statements conforme entendidos pelo SORT e das ocorrências
SYSPRINT   = listagem dos statements conforme entendidos pelo SORT e das ocorrências
SORTIN     = arquivo de entrada para o SORT (máximo 31 arquivos concatenados)
SORTIN1    = primeiro arquivo de entrada para o MERGE
SORTIN2    = segundo arquivo de entrada para o MERGE
...
SORTINnn   = n-ésimo arquivo de entrada para o MERGE
SORTOUT    = arquivo de saída para o SORT ou MERGE
SORTWK01   = arquivo de trabalho 01 (especificar UNIT e SPACE)
...
SORTWKnn   = arquivo de trabalho nn (especificar UNIT e SPACE)
```

Comentários nos statements de controle :

* na posição 1
brancos da 1 até a 72

Continuação nos statements de controle: o statement que terminar numa linha com vírgula indica que a próxima linha é a continuação

Statements (iniciar a partir da posição 2)

```
MERGE FIELDS ...
SORT FIELDS ...
SUM FIELDS ...
INCLUDE COND ...
OMIT COND ...
RECORD TYPE ...
END
```

Obs.:
1 – `SORT` e `MERGE` são mutuamente exclusivos
2 - `INCLUDE` e `OMIT` são mutuamente exclusivos

Cálculo do espaço dos arquivos de trabalho: (em quantidade de cilindros)
(LRECL * qtdd registros lógicos) /1.560.000
Exemplo : LRECL = 80, qtdd = 75.000 registros :

(80 * 75000) / 1560000 = 3,85 = 4 cilindros

```
INCLUDE COND
```

Indica quais registros do arquivo de entrada devem ser selecionados para classificação e conseqüentemente gravados no arquivo de saída.

```
INCLUDE COND=(condição)  ou
INCLUDE COND=(condição1,operador_relacional,condição2[,etc...])
```

Condição : *operando1,operador_lógico,operando2*

operando1 = posição_inicial,tamanho,formato

operador_lógico = `EQ` ou `NE` ou `GT` ou `GE` ou `LT` ou `LE`

operando2 = constante ou posição_inicial,tamanho, formato
posição_inicial = relativa a 1
tamanho = em bytes
formato =

```
CH (character)
BI (binary)
ZD (zoned decimal)
PD (packed decimal)
FI (fixed point com sinal)
```

Operador_ relacional : `AND` ou `OR`

Ou

```
INCLUDE COND=(condição),FORMAT=f  ou
INCLUDE COND=(condição1,operador_relacional,condição2[,etc...])
```

Condição : operando1,operador_lógico,operando2

operando1 = posição_inicial,tamanho

operador lógico = EQ , NE , GT , GE , LT , LE

operando2 = constante ou posição_inicial,tamanho
posição_inicial = relativa a 1
tamanho = em bytes
formato =

```
CH (character)
BI (binary)
ZD (zoned decimal)
PD (packed decimal)
FI (fixed point com sinal)
```

`OMIT COND`

Indica quais registros do arquivo de entrada não devem ser selecionados para classificação e conseqüentemente gravados no arquivo de saída.

Sintaxe análoga ao `INCLUDE COND` .

`SORT FIELDS`

Indica quais campos formam a chave de classificação, e a respectiva ordem de classificação (crescente ou decrescente). Não é necessário que sejam contíguos.

`SORT FIELDS=COPY` Efetua a cópia do(s) arquivo(s) de entrada para o arquivo de saída

Ou

`SORT FIELDS=(`*`Campo1`*`[,`*`campo2,campo3...`*`])[,FORMAT=f] [,EQUALS] [,NOEQUALS]`

`FORMAT=f` caso o formato individual de cada campo não seja informado, informa-se este formato, que vale para todos os campos. Pode ser :

```
CH (character)
BI (binary)
ZD (zoned decimal)
PD (packed decimal)
FI (fixed point com sinal)
```

`EQUALS` caso haja registros com chaves iguais, eles devem ser gravados na mesma ordem que estão no(s) arquivo(s) de entrada

`NOEQUALS` caso haja registros com chaves iguais, a ordem de gravação pode não ser a mesma que está no(s) arquivo(s) de entrada

Especificação dos campos de classificação :

Se for especificado o formato individual, não especificar o parâmetro FORMAT

`SORT FIELDS=(`*`Campo1`*`[,`*`campo2,campo3...`*`])[,EQUALS][,NOEQUALS]`

Campo : *posição_inicial,tamanho,ordem,formato*

Posição_inicial = posição_inicial do campo no registro, relativa a 1
Tamanho = em bytes
ordem = `A` (ascendente) ou `D` (descendente)
formato = valem as mesmas alternativas do FORMAT :
`CH, BI, ZD, PD ou FI`

Se não for especificado o formato individual, especificar o parâmetro FORMAT

```
SORT FIELDS=(Campo1[,campo2,campo3...]),FORMAT=f[,EQUALS] [,NOEQUALS]
```

Campo : *posição_inicial,tamanho,ordem*

Posição_inicial = posição_inicial do campo no registro, relativa a 1
Tamanho = em bytes
ordem = `A` (ascendente) ou `D` (descendente)

```
MERGE FIELDS
```

Indica quais campos formam a chave de intercalação. Não é necessário que sejam contíguos.

```
MERGE FIELDS=(Campo1[,campo2,campo3...]) [,FORMAT=f]
```

`FORMAT=f` caso o formato individual de cada campo não seja informado, informa-se este formato, que vale para todos os campos. Pode ser :

```
CH (character)
BI (binary)
ZD (zoned decimal)
PD (packed decimal)
FI (fixed point com sinal)
```

Especificação dos campos de classificação :

Se for especificado o formato individual, não especificar o parâmetro FORMAT

```
MERGE FIELDS=(Campo1[,campo2,campo3...])
```

Campo : *posição_inicial,tamanho,formato*
Posição_inicial = posição_inicial do campo no registro, relativa a 1
Tamanho = em bytes
formato = valem as mesmas alternativas do FORMAT :
`CH, BI, ZD, PD ou FI`

Ou

Se não for especificado o formato individual, especificar o parâmetro FORMAT

```
MERGE FIELDS=(Campo1[,campo2,campo3...]),FORMAT=f
```

Campo : *posição_inicial,tamanho*
Posição_inicial = posição_inicial do campo no registro, relativa a 1
Tamanho = em bytes
f = valem as mesmas alternativas do FORMAT :
`CH, BI, ZD, PD ou FI`

SUM FIELDS

Indica quais campos devem ser sumarizados, isto é, para chaves que se repetem, o conteúdo deles deve ser somado e gravado um único registro no arquivo de saída.

`SUM FIELDS=NONE` Não sumariza mas elimina duplicidades

Ou

`SUM FIELDS=(`*`Campo1`*`[,`*`campo2,campo3...`*`]),FORMAT=`*`f`*

Campo : *posição_inicial,tamanho*

Posição_inicial = posição_inicial do campo no registro, relativa a 1

Tamanho = em bytes

f =

```
BI (binary)
ZD (zoned decimal)
PD (packed decimal)
FI (fixed point com sinal)
```

Exemplos :

```
SUM FIELDS=(1,5),FORMAT=ZD
SORT FIELDS=(6,7,D),FORMAT=BI
RECORD TYPE=F

SORT FIELDS=(1,3,A),FORMAT=BI
SUM FIELDS=(4,2),FORMAT=ZD
RECORD TYPE=F
```

RECORD TYPE

RECORD TYPE=F

Ou

RECORD TYPE=V

END

Exemplos :

```
//SORTHSMP EXEC  PGM=ICEMAN
//*=======================
//*
//SYSOUT   DD SYSOUT=*
//*
//SORTIN   DD DISP=SHR,DSN=EXPL69.LST3.HSM
//         DD DISP=SHR,DSN=EXPL69.LST5.HSM
//SORTOUT  DD DSN=EXPL69.HSM.PRIK7,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA,RECFM=FBA,LRECL=80
/*
//SORTDIAG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 SORT     FIELDS=(2,6,CH,A)
 INCLUDE  COND=(2,1,CH,EQ,C'4',OR,2,1,CH,EQ,C'L')
 END
```

```
//MERGEHSM EXEC  PGM=ICEMAN
//*=======================
//*
//SYSOUT   DD SYSOUT=*
//*
//SORTIN1  DD DSN=EXPL69.HSM.PRIK7,DISP=OLD
//SORTIN2  DD DSN=EXPL69.HSM.ALTK7,DISP=OLD
//SORTIN3  DD DSN=EXPL69.HSM.ABRK7,DISP=OLD
//SORTOUT  DD DSN=EXPL69.HSM.ALLK7,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA,RECFM=FBA,LRECL=121
/*
//SORTDIAG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 MERGE    FIELDS=(2,6,CH,A)
 END
```

Para os demais exemplos, assumir arquivo de entrada com o seguinte conjunto de registros :

```
MOHANK  23423423434534344 KIRAN
MOHANK  13342345345345345 RAJEEV
ARAMES  34535345325354324 SURESH
SURESH  98347385385933987 PULI
RAMESH  67575789769876785 MADHU
KRISHN  50830948530859340 OIIED
KRISHN  30495849572938495 MADHU
SURESH  98347385385933987 PULI
```

```
//SYSIN    DD *
   SORT FIELDS=(1,3,CH,A,9,3,CH,A)
```

O arquivo de saída será :

```
ARAMES  34535345325354324 SURESH
KRISHN  30495849572938495 MADHU
KRISHN  50830948530859340 OIIED
MOHANK  13342345345345345 RAJEEV
MOHANK  23423423434534344 KIRAN
RAMESH  67575789769876785 MADHU
SURESH  98347385385933987 PULI
SURESH  98347385385933987 PULI
```

```
//SYSIN    DD *
      SORT FIELDS=(1,3,CH,A)
      SUM FIELDS=NONE
```

O arquivo de saída será :

```
ARAMES  34535345325354324 SURESH
KRISHN  50830948530859340 OIIED
MOHANK  23423423434534344 KIRAN
RAMESH  67575789769876785 MADHU
SURESH  98347385385933987 PULI
```

```
//SYSIN    DD *
    SORT FIELDS=COPY
     INCLUDE COND=(1,6,CH,EQ,C'SURESH')
```

O arquivo de saída será :

```
SURESH  98347385385933987 PULI
SURESH  98347385385933987 PULI
```

Sites super interessantes

http://www.geocities.com/srcsinc/drona/programming/languages/jcl/jcl.sort
.html

MVS JCL USERS GUIDE

http://ranch.state.nd.us/pdf/pdf/iea1b540.pdf.

Ou

http://216.239.37.104/search?
q=cache:UIXvVSN28lsJ:ranch.state.nd.us/pdf/pdf/iea1b540.pdf+%2Bsortin+
%2Bsortout&hl=en&ie=UTF-8#38

4.3 IEFBR14

"Utilitário" que na realidade não efetua nenhuma função; é um programa que tão logo inicia, emite o comando para terminar.

Na realidade o objetivo de sua utilização é o aproveitamento das funções de alocação (e desalocação) que o sistema operacional executa antes e depois de executar um programa.

Portanto, a grande aplicação deste "utilitário" é para poder alocar, deletar, catalogar e descatalogar arquivos.

Essas funções são determinadas por meio da especificação indicada no parâmetro DISP do statement DD.

Como o IEFBR14 não irá abendar quando for executado, vale a ação indicada para término normal no DISP :

```
   DISP=(NEW,KEEP,KEEP)        para criar (alocar) um arquivo

   DISP=(OLD,DELETE,DELETE)    para deletar um arquivo

   DISP=(OLD,CATLG,KEEP)       para catalogar um arquivo

   DISP=(OLD,UNCATALG,KEEP)    para descatalogar um arquivo
```

```
//JOB1  JOB ,'Example'
//BR14STEP  EXEC  PGM=IEFBR14
//DELETE1   DD    DSN=Z123456.MYLIB,UNIT=DISK,VOL=SER=ACA302,
//     DISP=(OLD,DELETE,DELETE)
//CREATE1   DD DSN=Z123456.NEWFILE,UNIT=DISK,VOL=SER=ACA302,
//     DISP=(NEW,KEEP,KEEP)
//
```

4.4 IEBCOPY

- Copia membros de um PDS para outro PDS. Formato:

```
//stepname  EXEC  PGM=IEBCOPY
//ddname1   DD    DSN=dsn_do_arquivo_de_entrada,outros_parametros
//ddname2   DD    DSN= dsn_do_arquivo_de_entrada,outros_parametros
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
  control statements
```

COPY	COPY INDD=ddname1,OUTDD=ddname2 Os membros do PDS ddname1 são copiados para o PDS ddname2 COPY INDD=ddname1,OUTDD=ddname2[,LIST=YES][,LIST=NO] Os membros do PDS ddname1 são copiados para o PDS ddname2 YES : os nomes dos membros copiados são listados; default: YES COPY INDD=((ddname1,R)),OUTDD=ddname2 Os membros do PDS ddname1 cujo nome tenham equivalente no PDS ddname2 substituiem os do PDS ddname2
SELECT	SELECT MEMBER=name Copia o membro indicado do PDS de entrada para o PDS de saida SELECT MEMBER=(name1,name2,...) Copia os membros indicados do PDS de entrada para o PDS de saida SELECT MEMBER=((name,newname,R)) Copia o membro indicado do PDS de entrada para o PDS de saida alterando o seu nome para newname SELECT MEMBER=((name,,R)) Copia (com replace) o membro do PDS de entrada para o PDS de saida
EXCLUDE	EXCLUDE MEMBER=name Copia todos os membros do PDS de entrada para o PDS de saida, EXCETO o membro indicado EXCLUDE MEMBER=(name1,name2,...) Copia todos os membros do PDS de entrada para o PDS de saida, EXCETO os membros indicados

Exemplos :

```
//COPYSTEP  EXEC  PGM=IEBCOPY
//INFILE    DD    DSN=Z123456.INPUT,DISP=SHR
//OUTFILE   DD    DSN=Z123456.OUTPUT,DISP=SHR
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
  COPY  INDD=INFILE,OUTDD=OUTFILE
  SELECT  MEMBER=(MEM1,MEM3,MEM5)
/*
```

```
//HERC01C JOB HERC01,'HERC01',CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//STEP010 EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3 DD SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSUT4 DD SPACE=(TRK,(1,1)),UNIT=SYSDA
//DDI1 DD DSN=SYS2.DEV.LOADLIB,DISP=SHR
//DDO1 DD DSN=SYS3.LINKLIB,DISP=SHR
//SYSIN DD *
 COPY OUTDD=DDO1,INDD=DDI1

//HERC01C JOB (HERC01),'COMPRESS',CLASS=A,MSGCLASS=X
//STEP010 EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3 DD SPACE=(TRK,(1,1)),UNIT=SYSDA
//SYSUT4 DD SPACE=(TRK,(1,1)),UNIT=SYSDA
//DDI1 DD DSN=HERC01.DEV.LOADLIB,
//     DISP=OLD
//DDO1 DD DSN=HERC01.DEV.LOADLIB,
//     DISP=OLD
//SYSIN DD *
 COPY OUTDD=DDO1,INDD=DDI1
```

4.5 IEBPTPCH

- Imprime registros do arquivo de entrada no arquivo de saida. Formato :

```
//stepname  EXEC  PGM=IEBPTPCH
//ddname1   DD    DSN=dsn_do_arquivo_de_entrada,outros_parametros
//ddname2   DD    DSN=dsn_do_arquivo_de_entrada,outros_parametros
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
  control statements
```

Statement	Options
PRINT Ou PUNCH	indica saida em impressão ou (existe também o statement PUNCH para indicar a saída em cartão perfurado). (PRINT ou PUNCH : é obrigatório um deles) PREFORM=A ou PREFORM=M Tipo de caracter de controle : ASA ou Maquina. Se especificado, todas as outras opções (exceto TYPORG) são ignoradas. TYPORG=PO ou TYPORG=PS Tipo de arquivo (SYSUT1). PO = particionado; PS = sequencial TOTCONV=XE ou TOTCONV=PZ Indica o tipo de conversão de daos que deve ser feita (não tem default). XE : imprime 2 caracteres por byte PZ : converte decimal compactado para decimal zonado CNTRL=1 ou CNTRL=2 ou CNTRL=3 Indica o tipo de espacejamento. Default: 1 STRTAFT=m Indica o número de registros que devem ser pulados (não impressos) antes do início da impressão. Default: 0. m <= 32767 STOPAFT=n Indica o número de registros a imprimir. Default: : imprime tudo. n <= 32767 SKIP=p Indica que todo p-ésimo registro deve ser pulado. Default: imprime sem pular MAXNAME=q Indica a quantidade máxima de membros que vão constar nos subsequentes statements MEMBER; necessário se houver statement(s) MEMBER. q <= 32767 MAXFLDS=r Indica a quantidade máxima de parâmetros FIELD que vão constar nos statements FIELD subsequentes. r <= 32767 Não é necessário incluir os campos em branco MAXGPS=s Indica a quantidade de parâmetros IDENT que vão constar nos statements RECORD subsequentes. Necessário se houver IDENT. s <= 32767 MAXLITS=t Indica o número de caracteres que vao constar nas literais IDENT nos statements

	RECORD subsequentes; necessário se houver literais. t <= 32767
	
	`INITPG=u MAXLITS=t`
	Indica o número da página inicial na saída. 1 < u < 9999 . Default: 1
	
	`MAXLINE=x`
	Indica o número máximo de linhas por página. Default: 60

`TITLE`	Indica o cabeçalho e/ou coluna de cabeçalho para as 2 primeiras linhas de cada página; um TITLE pode Ter diversos statements ITEM; pode haver no máximo 2 statements TITLE.
	
	`ITEM=('title',m)`
	`title = literal a imprimir(máximo 40 caracteres)`
	`m = indica a coluna inicial do conteúdo do cabeçalho; default: 1`
	
	`Exemplo :`
	
	`TITLE ITEM=('Banco California',20)`
	`TITLE ITEM=('Nome Agencia',10),`
	`ITEM=('Nome Gerente',28)`
	
	`12345678901234567890123456789012345678901`
	`Banco California`
	`Nome Agencia Nome Gerente`

`MEMBER`	Indica qual(quais) membro(s) do PDS devem ser impresso(s); pode haver diversos statements MEMBER; se não for especificado MEMBER e se o dataset em SYSUT1 for PDS, todos os membros serão listados. `NAME=member-name`

`RECORD`	Indica um grupo de registros
	
	`IDENT=(m,'string',n)`
	Indica o último registro num grupo de registros (argumento de pesquisa)
	m tamanho do campo para casar (argumento) com o registro. m < 8
	string conteúdo para casar (argumento) com o conteúdo do registro
	n posição inicial do campo no registro
	
	`FIELD=(p,q,conversion,r)`
	Para editar o registro na saída
	p tamanho do campo
	q posição inicial; default: 1
	conversion tipo de conversão a efetuar; se não especificado, não é feita nenhuma conversão
	PZ decimal compactado para decimal zonado
	XE hexadecimal (2 bytes para cada byte)
	r posição inicial na saída; default: 1

Exemplos :

```
  //* listar um arquivo SAM duplo espaco pulando
  //* reg 4 reg 8 reg 12 e assim por diante
  //PRNTSTEP  EXEC  PGM=IEBPTPCH
  //SYSUT1    DD    DSN=T90RAZ1.CS465S02.NAMEFILE,DISP=SHR
  //SYSUT2    DD    SYSOUT=*
  //SYSPRINT  DD    SYSOUT=*
  //SYSIN     DD    *
   PRINT  TYPORG=PS,CNTRL=2,SKIP=4

  //* listar um arquivo SAM
  //PRNTSTEP  EXEC  PGM=IEBPTPCH
  //SYSUT1    DD    DSN=T90RAZ1.CS465S02.NAMEFILE,DISP=SHR
  //SYSUT2    DD    SYSOUT=*
  //SYSPRINT  DD    SYSOUT=*
  //SYSIN     DD    *
    PRINT  TYPORG=PS,CNTRL=2,SKIP=4,MAXFLDS=5
    TITLE  ITEM=('Big 9 Conference',35)
    TITLE  ITEM=('School Name',10),ITEM=('Team Name',30),
           ITEM=('Score 1',55),ITEM=('Score 2',65)
    RECORD FIELD=(20,1,,10),FIELD=(20,21,,30),
           FIELD=(3,41,,56),FIELD=(3,44,,66)

  //* listar um membro de um arquivo PDS
  //* DSN=nome_pds no DD ; TYPORG=PO e MEMBER=nome_membro no PRINT
  //PRNTSTEP  EXEC  PGM=IEBPTPCH
  //SYSUT1    DD    DSN=T90RAZ1.NAMEPDS,DISP=SHR
  //SYSUT2    DD    SYSOUT=*
  //SYSPRINT  DD    SYSOUT=*
  //SYSIN     DD    *
    PRINT  TYPORG=PO,MEMBER=FONT3001,CNTRL=2,MAXFLDS=5
    TITLE  ITEM=('LINHA CABEC',35)
    TITLE  ITEM=('SUBTITULO1',10),ITEM=('SUBTIT2',30)
    RECORD FIELD=(20,1,,10),FIELD=(20,21,,30)

  //* listar um membro de um arquivo PDS
  //* DSN=nome_pds(nome_membro) no DD ; TYPORG=PS no PRINT
  //PRNTSTEP  EXEC  PGM=IEBPTPCH
  //SYSUT1    DD    DSN=T90RAZ1.NAMEPDS(FONT3001),DISP=SHR
  //SYSUT2    DD    SYSOUT=*
  //SYSPRINT  DD    SYSOUT=*
  //SYSIN     DD    *
    PRINT  TYPORG=PS,CNTRL=2,MAXFLDS=5
    TITLE  ITEM=('LINHA CABEC',35)
    TITLE  ITEM=('SUBTITULO1',10),ITEM=('SUBTIT2',30)
    RECORD FIELD=(20,1,,10),FIELD=(20,21,,30)

------------------------------------------------------------
//SYSIN  DD  *
PRINT   MAXFLDS=3
RECORD  FIELD=
(5,1,,10),                                              *
FIELD=
(20,6,,20),                                             *
FIELD=(6,38,,45)
```

4.6 IEBGENER

- Copia registro(s) do arquivo de entrada para o de saída
- Os registros na saída poder ser ou não editados
- O arquivo de saída pode ser PS ou PDS
- O mais comum é copiar dados in-stream para um mebro de um PDS
- Os parâmetros da DCB devem ser especificados no SYSUT2

```
Format:

//stepname  EXEC  PGM=IEBGENER
//SYSUT1   DD    DSN=dsn_do_arquivo_de_entrada,outros_parametros
//SYSUT2   DD    DSN=dsn_do_arquivo_de_entrada,outros_parametros
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
  control statements
```

Statement	Options
GENERATE	❑ Indica que os dados copiados / editados estão sendo criados ❑ Duas atividades podem ser indicadas com esta opção : 1. A group of records from a sequential data set can be copied in new members of a PDS. If this is the case, it is necessary to provide a name for each member and to specify the end of the group. 2. Edit a group of records as they are being copied. MAXNAME=q Specifies the number of member names appearing on subsequent MEMBER statements; required if there are any MEMBER statements MAXFLDS=r Specifies the number of FIELD parameters appearing on subsequent RECORD statements; required if there are any FIELD statements MAXGPS=s Specifies the number of IDENT parameters appearing on subsequent RECORD statements; required if there are any IDENT statements MAXLITS=t Specifies the number of characters appearing in the IDENT literals of Any subsequent RECORD statements; required if there are any literals At all
MEMBER	NAME=member-name Used when a member of a PDS is being created; must be one for each member being created; if not specified, the SYSUT2 data set is organized sequentially.
RECORD	Used to define a group of records in a member of a PDS or in a sequential data set

	IDENT=(m,'string',n) Identifies the last record in a group of records m length of the field to match in the record; m < 8 string field in the record to match n starting position (column) of the field to match FIELD=(p,q,conversion,r) Used to edit a record p length of the field to be edited; default: 80 q starting position (column) of the field to edit; optional; default: 1 conversion type of conversion to perform on the field if not specified, no conversion takes place PZ packed decimal to zoned decimal ZP zoned decimal to packed decimal r starting position (column) to place the field in the output; optional; default: 1 FIELD=(p,'literal',conversion,r) Used to put a literal into a record p length of the specified literal; p < 40 literal string to put into the record conversion type of conversion to perform on the literal if not specified, no conversion takes place PZ packed decimal to zoned decimal ZP zoned decimal to packed decimal r starting position (column) to place the literal in the output; optional; default: 1

Exemplos :

```
//GENSTEP1  EXEC  PGM=IEBGENER
//SYSUT1    DD    DSN=T90AJB1.FILE1,DISP=SHR
//SYSUT2    DD    DSN=T90AJB1.NEWFILE,DISP=SHR
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
  GENERATE  MAXNAME=3,MAXGPS=2
  MEMBER  NAME=MEMBER1
  RECORD  IDENT=(8,'FIRSTMEM',1)
  MEMBER  NAME=MEMBER2
  RECORD  IDENT=(8,'SECNDMEM',40)
  MEMBER  NAME=MEMBER3
/*
```

The above step will create three PDS members in T90AJB1.NEWFILE.
Their names will be MEMBER1, MEMBER2, and MEMBER3.

MEMBER1 will contain everything from the beginning of the
file through a record with 'FIRSTMEM' in the first 8 columns

MEMBER2 will contain everything from the record with 'FIRSTMEM'

in the first 8 columns through a record with 'SECNDMEM'
starting in column 40

MEMBER3 will contain everything from the record
with 'SECNDMEM' starting in column 40 through the end of the file

```
//GENSTEP2  EXEC  PGM=IEBGENER
//SYSUT1    DD    DSN=T90RAZ1.CS465S02.NAMEFILE,DISP=SHR
//SYSUT2    DD    DSN=T90AJB1.NEWFILE,DISP=SHR,
//     DCB=(LRECL=55,BLKSIZE=550,RECFM=FB)
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
  GENERATE  MAXFLDS=3,MAXLITS=14
  RECORD  FIELD=(20,1,,35),
            FIELD=(14,'ARE A TEAM AT ',21),
            FIELD=(20,21,,1)
/*
```

The above step will create a sequential file from the records In
T90RAZ1.CS465S02.NAMEFILE. The records in the new file will
have a format similar to:

```
BLUE HENS           ARE A TEAM AT DELAWARE ST.
HEDGE HOGS          ARE A TEAM AT DELAWARE ST.
```

4.7 IEHLIST

Para listar PDS.

```
Formato:

//* listar todos os membros do PDS
//stepname  EXEC PGM=IEHLIST
//SYSPRINT  DD   SYSOUT=*
//ANY1      DD   UNIT=DISK,VOL=SER=volume#,SPACE=(TRK,0),DISP=OLD
//SYSIN     DD   *
  LISTPDS  DSNAME=PDS name goes here,VOL=DISK=volume#
/*
```

Pode ter mais de um LISTPDS, mas cada um deve ter um DD

```
//* this will list all of the members in the specified PDS in an
//* unedited format such that only the member names will be readable
//stepname  EXEC PGM=IEHLIST
//SYSPRINT  DD   SYSOUT=*
//ANY1      DD   UNIT=DISK,VOL=SER=volume#,SPACE=(TRK,0),DISP=OLD
//SYSIN     DD   *
  LISTPDS  DSNAME=PDS name goes here,VOL=DISK=volume#,DUMP
/*


//* this will list all of the members in the specified PDS in
//* an edited format intended ONLY for listing load modules
//stepname  EXEC PGM=IEHLIST
//SYSPRINT  DD   SYSOUT=*
//ANY1      DD   UNIT=DISK,VOL=SER=volume#,SPACE=(TRK,0),DISP=OLD
//SYSIN     DD   *
  LISTPDS  DSNAME=PDS name goes here,VOL=DISK=volume#,FORMAT
/*
```

5. ENDEVOR

5.1 Conceitos gerais

Estrutura de armazenamento

```
ENVIRONMENT ===>  DESENV
SYSTEM      ===>  RPC
SUBSYSTEM   ===>  RPC
ELEMENT     ===>  RPCP450
TYPE        ===>  COBPB
STAGE       ===>  D


ENVIRONMENT = DESENV ou PRODUCAO


TYPE
      COBPB = cobol, programas, batch
      BOOK = books
      COBRB = cobol, rotinas, batch
      APPL = aplicativos on-line CSP
      COBPO = cobol, programas, on-line
      COBRO = cobol, rotinas, on-line
      COBPBD = cobol, programa, batch com DB2
      MAPAC = mapa BMS


STAGE  = D  ou  A  ou  P
```

Como opera o Endevor?

Ver algo = on-line
Fazer algo = preparar on-line o job a ser executado batch

Fazer algo : O QUE ?

RETRIEVE = trazer um elemento do ENDEVOR para um arquivo do MVS (arquivo sequencial ou membro de um PDS) (necessário para efetuar UPDATE também)
ADD = adicionar um elemento novo na estrutura (base de dados) do ENDEVOR
UPDATE = atualizar um elemento já existente na estrutura do ENDEVOR
Criar PACKAGE = criar um pacote com 1 ou mais programas para transferi-lo(s) de um estágio para o seguinte

Para entrar no Endevor :

TSO

E – ENDEVOR

1 - DESENVOLVIMENTO

Para exibir um elemento :

1 (display)
1 (elemento)
Preencher estrutura de referência
(se não preencher type, traz de todos os tipos)
enter

Para efetuar retrieve de um elemento :

3 (batch)
Indicar o nome do arquivo/membro onde será gravado o request (SCL)

```
REQUEST DATA SET:
   PROJECT  ===> TORI243
   GROUP    ===> T#RPC
   TYPE     ===> SRC
   MEMBER   ===> pegar
```

OPTION = 1 (build scl)

tela de BATCH OPTION MENU
OPTION = 3 (retrieve)

Preencher a estrutura de referência FROM (para indicar qual o elemento desejado) e
CUIDADO com as ACTION OPTIONS (SIGNOUT ELEMENT = N) !!!!

```
FROM ENDEVOR:                          ACTION OPTIONS:
  ENVIRONMENT ===> DESENV                 CCID             ===>
  SYSTEM      ===> ADP                    EXPAND INCLUDES  ===> N (Y/N)
  SUBSYSTEM   ===> ADP                    SIGNOUT ELEMENT  ===> n (Y/N)
  ELEMENT     ===> adpb010                OVERRIDE SIGNOUT ===> N (Y/N)
  TYPE        ===> BOOK                   REPLACE MEMBER   ===> N (Y/N)
  STAGE       ===> D        D - DSV            A - ATA
  COMMENT     ===>
```

Preencher onde o elemento deverá ser gravado pelo Endevor

```
TO ISPF LIBRARY:                       LIST OPTIONS:
    PROJECT ===> TORI243                   DISPLAY LIST      ===> Y (Y/N)
    LIBRARY ===> T#RPC                     WHERE CCID EQ     ===>
    TYPE    ===> SRC                       WHERE PROC GRP EQ ===>
    MEMBER  ===> adpb010                   BUILD USING MAP   ===> Y (Y/N)
                                           FIRST FOUND       ===> Y (Y/N)
 TO OTHER PARTITIONED OR SEQUENTIAL DATA SET:
```

OPTION = R
Voltar (PF3) até a tela de BATCH OPTION MENU
Para verificar o SCL gerado / confirmar / alterar, entrar em 2 (edit), e sair por PF3
Quando o SCL estiver OK,
verificar / acertar o statement JOB nesta tela

OPTION = 3 (SUBMIT)

O job será submetido e deve ser acompanhado (pelos métodos normais) via TSO ou ROSCOE.

Para incluir um elemento :

3 (batch)
Indicar o nome do arquivo/membro onde será gravado o request (SCL)

```
REQUEST DATA SET:
   PROJECT  ===> TORI243
   GROUP    ===> T#RPC
   TYPE     ===> SRC
   MEMBER   ===> pegar
```

OPTION = 1 (build scl)

tela de BATCH OPTION MENU
OPTION = 2 (add/update)

Preencher onde o elemento deve ser adicionado na estrutura do ENDEVOR (só pode ser em DESENV).

```
TO ENDEVOR:                          ACTION OPTIONS:
   ENVIRONMENT ===> DESENV              CCID                 ===>
   SYSTEM      ===> ADP                 GENERATE ELEMENT     ===> Y (Y/N)
   SUBSYSTEM   ===> ADP                 DELETE INPUT SOURCE  ===> N (Y/N)
   ELEMENT     ===> novo                NEW VERSION          ===>
   TYPE        ===> BOOK                OVERRIDE SIGNOUT     ===> N (Y/N)
   STAGE:           D                   PROCESSOR GROUP      ===>
                                        UPDATE IF PRESENT    ===> N (Y/N)
   COMMENT     ===>
```

Preencher de onde o elemento deve ser obtido pelo ENDEVOR.

```
FROM ISPF LIBRARY:                   LIST OPTIONS:
   PROJECT ===> TORI243                 DISPLAY LIST      ===> Y (Y/N)
   LIBRARY ===> T#RPC
   TYPE    ===> SRC
   MEMBER  ===> novo        THRU MEMBER ===>

FROM OTHER PARTITIONED OR SEQUENTIAL DATA SET:
```

OPTION = A (add)

Voltar (PF3) até a tela de BATCH OPTION MENU
Para verificar o SCL gerado / confirmar / alterar, entrar em 2 (edit), e sair por PF3
Quando o SCL estiver OK,
verificar / acertar o statement JOB nesta tela
OPTION = 3 (SUBMIT)

O job será submetido e deve ser acompanhado (pelos métodos normais) via TSO ou ROSCOE.

Para efetuar update de um elemento :

3 (batch)
Indicar o nome do arquivo/membro onde será gravado o request (SCL)

```
REQUEST DATA SET:
   PROJECT  ===> TORI243
   GROUP    ===> T#RPC
   TYPE     ===> SRC
   MEMBER   ===> pegar
```

OPTION = 1 (build scl)
tela de BATCH OPTION MENU
OPTION = 2 (add/update)

Preencher onde o elemento deve ser adicionado na estrutura do ENDEVOR (só pode ser em DESENV).

```
TO ENDEVOR:                          ACTION OPTIONS:
   ENVIRONMENT ===> DESENV              CCID                 ===>
   SYSTEM      ===> ADP                 GENERATE ELEMENT     ===> Y (Y/N)
   SUBSYSTEM   ===> ADP                 DELETE INPUT SOURCE  ===> N (Y/N)
   ELEMENT     ===> novo                NEW VERSION          ===>
   TYPE        ===> BOOK                OVERRIDE SIGNOUT     ===> N (Y/N)
   STAGE:           D                   PROCESSOR GROUP      ===>
                                        UPDATE IF PRESENT    ===> N (Y/N)
   COMMENT     ===>
```

Preencher de onde o elemento deve ser obtido pelo ENDEVOR.

```
FROM ISPF LIBRARY:                   LIST OPTIONS:
   PROJECT ===> TORI243                 DISPLAY LIST      ===> Y (Y/N)
   LIBRARY ===> T#RPC
   TYPE    ===> SRC
   MEMBER  ===> novo        THRU MEMBER ===>

FROM OTHER PARTITIONED OR SEQUENTIAL DATA SET:
```

OPTION = U (update)

Voltar (PF3) até a tela de BATCH OPTION MENU
Para verificar o SCL gerado / confirmar / alterar, entrar em 2 (edit), e sair por PF3
Quando o SCL estiver OK,
verificar / acertar o statement JOB nesta tela
OPTION = 3 (SUBMIT)

O job será submetido e deve ser acompanhado (pelos métodos normais) via TSO ou ROSCOE.

6. FILE-AID

6.1 Conceitos gerais

Para entrar :

TSO

PP – DIVERSOS

FA – FILE-AID /MVS

Para exibir conteúdo de um arquivo :
1 (browse)
em geral :
Browse mode = C
Datasetname = nome do arquivo a ver
Member name = nome do membro (se for o membro de um PDS)
Selection criteria = N (exibir todos os registros, sem filtrar)
se quiser especificar filtro : colocar q (quick)
CMD = para editar as linhas de especificacao de filtros
RO = relational operator =
= (equal) EQ (equal)
¬= (not equal) NE (not equal)
> (greater than) GT (greater than)
>= (greater or equal) GE (greater or equal)
< (less than) LT (less than)
<= (less or equal) LE (less or equal)
CO (contained) NC (not contained)
BT (between) NB (not between)
VA () NV ()
MX () NO ()

Para editar um arquivo :
2 (edit)
em geral :
Edit mode = C
Datasetname = nome do arquivo a ver
Member name = nome do membro (se for o membro de um PDS)
Disposition = OLD ???
Selection criteria = N (exibir todos os registros, sem filtrar)
se quiser especificar filtro : colocar q (quick)
CMD = para editar as linhas de especificacao de filtros
RO = relational operator =
=, EQ, ¬=, NE, >, GT, >=, GE, <, LT, <=, LE, CO, NC, BT, NB, VA,

NV, MX, NO

```
Banco Real               ISPF Master Application Menu            ABN AMRO Bank

 P  PDF       ISPF/Program Development Facility             Userid   : TORI243
 SD SDSF      System Display and Search Facility            Time     : 13:33
 DB DB2I      DB2 Interactive                               Date     : 03/10/08
 RT RT        Opcoes do Racf                                Julian   : 03.281
 E  ENDEVOR   CA-Endevor for OS/390                         Sysid    : AB73
 C  CRIAMSL   Remontagem APLIC CSP para ADD/UPDT Endevor    Release  : ISPF 5.0
 PP DIVERSOS  Produtos Diversos                             Procedure: IKJ@RPL
 X  EXIT      Terminate ISPF using list/log defaults









 Enter END command to terminate application

 5647-A01 (C) COPYRIGHT IBM CORP 1982, 1997
 Option ===>   pp
  F1=Help      F2=Split     F3=Exit      F9=Swap     F10=Actions  F12=Cancel
```

```
Banco Real               Produtos Diversos                       ABN AMRO Bank

 FA FILE-AID  File-AID/MVS                                  Userid   : TORI243
 FD FILE-AID  File-AID/DB2                                  Time     : 13:35
 RD FILE-AID  File-AID/RDX                                  Date     : 03/10/08
 DS FILE-AID  File-AID/Data Solutions                       Julian   : 03.281
                                                            Sysid    : AB73
 ST STROBE    Strobe                                        Release  : ISPF 5.0
                                                            Procedure: IKJ@RPL
 X  EXIT      Exit









 Enter END command to terminate application



 Option ===>  fa
  F1=Help      F2=Split     F3=Exit      F9=Swap     F10=Actions  F12=Cancel
```

```
 IKJ56247I FILE SYSUT1 NOT FREED, IS NOT ALLOCATED
 IKJ56247I FILE SYSUT2 NOT FREED, IS NOT ALLOCATED
 IKJ56893I DATA SET DSVAABPS.ZZZ.TORI243.DATEHOUR.FAIDMVS NOT ALLOCATED+
 IGD17400I REFERENCED DATA SET DSVAABPS.ZZZ.GUSHA.LRECL26 NOT CATALOGED
 IGD17409I FAILURE OCCURRED IN DATA SET PROPERTIES MERGE WHILE ATTEMPTING TO
 DEFINE DATA SET DSVAABPS.ZZZ.TORI243.DATEHOUR.FAIDMVS
 IRX0555E The input or output file SYSUT1 is not allocated. It cannot be opene
for I/O.
 IRX0670E EXECIO error while trying to GET or PUT a record.
 IDCAMS  SYSTEM SERVICES                                          TIME: 13:31
3        10/08/03     PAGE      1

  REPRO IFILE(SYSUT1) OFILE(SYSUT2)
 IDC2908I SYSUT1 NOT FOUND IN SYSTEM
 IDC3300I  ERROR OPENING SYSUT1
 IDC3304I ** JCL STATEMENT MISSING
 IDC0005I NUMBER OF RECORDS PROCESSED WAS 0
 IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12

 IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 12
 IKJ56247I FILE SYSUT1 NOT FREED, IS NOT ALLOCATED
 ***
```

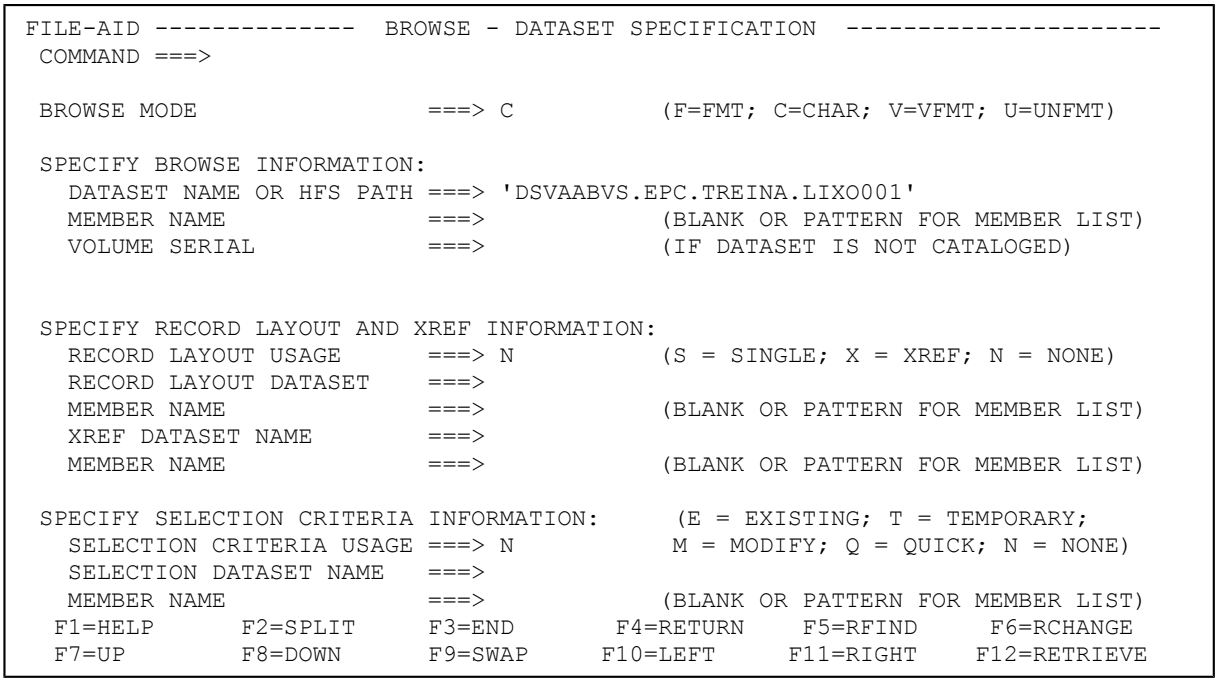
```
FILE-AID 8.8.0 -------------  PRIMARY OPTION MENU  -----------------------------
 OPTION ===>

   0  PARAMETERS  - SPECIFY ISPF AND FILE-AID PARAMETERS     USERID   - TORI243
   1  BROWSE      - DISPLAY FILE CONTENTS                    PF KEYS  - 24
   2  EDIT        - CREATE OR CHANGE FILE CONTENTS           TERMINAL - 3278
   3  UTILITIES   - FILE-AID/SPF EXTENDED UTILITIES          TIME     - 13:36
   5  PRINT       - PRINT FILE CONTENTS                      JULIAN   - 03.281
   6  SELECTION   - CREATE OR CHANGE SELECTION CRITERIA      DATE     - 03/10/08
   7  XREF        - CREATE OR CHANGE RECORD LAYOUT CROSS REFERENCE
   8  VIEW        - VIEW INTERPRETED RECORD LAYOUT
   9  REFORMAT    - CONVERT FILE FROM ONE FORMAT TO ANOTHER
  10  COMPARE     - COMPARE FILE CONTENTS
   C  CHANGES     - DISPLAY SUMMARY OF FILE-AID CHANGES
   T  TUTORIAL    - DISPLAY INFORMATION ABOUT FILE-AID
   X  EXIT        - TERMINATE FILE-AID AND RETURN TO ISPF

      USE END TO TERMINATE FILE-AID

      ONLINE TECHNICAL SUPPORT AVAILABLE AT:  FRONTLINE.COMPUWARE.COM

      COPYRIGHT (C) 1982 - 2001.  ALL RIGHTS RESERVED.  UNPUBLISHED
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```


FILE-AID -------------- BROWSE - DATASET SPECIFICATION ----------------------
COMMAND ===>
BROWSE MODE ===> C (F=FMT; C=CHAR; V=VFMT; U=UNFMT)
SPECIFY BROWSE INFORMATION:
DATASET NAME OR HFS PATH ===> 'DSVAABVS.EPC.TREINA.LIXO001'
MEMBER NAME ===> (BLANK OR PATTERN FOR MEMBER LIST)
VOLUME SERIAL ===> (IF DATASET IS NOT CATALOGED)
SPECIFY RECORD LAYOUT AND XREF INFORMATION:
RECORD LAYOUT USAGE ===> N (S = SINGLE; X = XREF; N = NONE)
RECORD LAYOUT DATASET ===>
MEMBER NAME ===> (BLANK OR PATTERN FOR MEMBER LIST)
XREF DATASET NAME ===>
MEMBER NAME ===> (BLANK OR PATTERN FOR MEMBER LIST)
SPECIFY SELECTION CRITERIA INFORMATION: (E = EXISTING; T = TEMPORARY;
SELECTION CRITERIA USAGE ===> N M = MODIFY; Q = QUICK; N = NONE)
SELECTION DATASET NAME ===>
MEMBER NAME ===> (BLANK OR PATTERN FOR MEMBER LIST)
F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFIND F6=RCHANGE
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEVE

Hardware

Baixa Plataforma	Ítem	Mainframe
(Dias)	MTBF (Mean Time Between Failures	(Anos)
(Minutos)	MTTR (Mean Time To Repair)	(Horas)
Milhares de US$	Custo	Dezenas ou Centenas de milhares de US$
Em geral de arquitetura aberta	Conexões	Em geral arquitetura proprietária
Em geral admite faixa mais ampla de condições	Ambiente	Condições (mais) Controladas
Curto prazo, sujeito sujeito a flutuações de mercado e evolução tecnológica; "atrelada" a outras empresas (software – Microsoft, etc)	Desenvolvimento de produto	Longo prazo, menos sujeito a flutuações de mercado e evolução tecnológica; em geral desenvolvimento próprio

Software

Baixa Plataforma	Ítem	Mainframe
Transacional	Sistema Operacional	Batch-oriented A parte transacional é feita pelo CICS e VTAM
(Dias)	MTBF Sist. Operacional (Mean Time Between Failures	(Anos (?))
Orientação a Objeto, Visual, etc...	Desenvolvimento de Aplicativos	Estruturada, "spaghetti code" (CSP diferente)

Apêndice 1 – Tabela Ascii / Ebcdic

HW IBM - tabela Ascii/Ebcdic - de 00h a 1Fh

ASCII		Dec	Hex	EBCDIC	
NUL	**Null**	**0**	**00**	**NUL**	**Null**
SOH	**Start of Heading (CC)**	**1**	**01**	**SOH**	**Start of Heading**
STX	**Start of Text (CC)**	**2**	**02**	**STX**	**Start of Text**
ETX	**End of Text (CC)**	**3**	**03**	**ETX**	**End of Text**
EOT	**End of Transmission (CC)**	**4**	**04**	**PF**	**Punch Off**
ENQ	**Enquiry (CC)**	**5**	**05**	**HT**	**Horizontal Tab**
ACK	**Acknowledge (CC)**	**6**	**06**	**LC**	**Lower Case**
BEL	**Bell**	**7**	**07**	**DEL**	**Delete**
BS	**Backspace (FE)**	**8**	**08**	**nada**	**nada**
HT	**Horizontal Tabulation (FE)**	**9**	**09**	**nada**	**nada**
LF	**Line Feed (FE)**	**10**	**0A**	**SMM**	**Start of Manual Message**
VT	**Vertical Tabulation (FE)**	**11**	**0B**	**VT**	**Vertical Tab**
FF	**Form Feed (FE)**	**12**	**0C**	**FF**	**Form Feed**
CR	**Carriage Return (FE)**	**13**	**0D**	**CR**	**Carriage Return**
SO	**Shift Out**	**14**	**0E**	**SO**	**Shift Out**
SI	**Shift In**	**15**	**0F**	**SI**	**Shift In**
DLE	**Data Link Escape (CC)**	**16**	**10**	**DLE**	**Data Link Escape**
DC1	**Device Control 1**	**17**	**11**	**DC1**	**Device Control 1**
DC2	**Device Control 2**	**18**	**12**	**DC2**	**Device Control 2**
DC3	**Device Control 3**	**19**	**13**	**TM**	**Tape Mark**
DC4	**Device Control 4**	**20**	**14**	**RES**	**Restore**
NAK	**Negative Acknowledge (CC)**	**21**	**15**	**NL**	**New Line**
SYN	**Synchronous Idle (CC)**	**22**	**16**	**BS**	**Backspace**
ETB	**End of Transmission Block (CC)**	**23**	**17**	**IL**	**Idle**
CAN	**Cancel**	**24**	**18**	**CAN**	**Cancel**
EM	**End of Medium**	**25**	**19**	**EM**	**End of Medium**
SUB	**Substitute**	**26**	**1A**	**CC**	**Cursor Control**
ESC	**Escape**	**27**	**1B**	**CU1**	**Customer Use 1**
FS	**File Separator (IS)**	**28**	**1C**	**IFS**	**Interchange File Separator**
GS	**Group Separator (IS)**	**29**	**1D**	**IGS**	**Interchange Group Separator**
RS	**Record Separator (IS)**	**30**	**1E**	**IRS**	**Interchange Record Separator**
US	**Unit Separator (IS)**	**31**	**1F**	**IUS**	**Interchange Unit Separator**

HW IBM - tabela Ascii/Ebcdic - de 20h a 3Fh

ASCII		Dec	Hex	EBCDIC	
SP	**Space**	**32**	**20**	**DS**	**Digit Select**
!	**Exclamation Point**	**33**	**21**	**SOS**	**Start of Significance**
"	**Quotation Mark**	**34**	**22**	**FS**	**Field Separator**
# "pound"	**Number Sign, Octothorp,**	**35**	**23**	**nada**	**nada**
$	**Dollar Sign**	**36**	**24**	**BYP**	**Bypass**
%	**Percent**	**37**	**25**	**LF**	**Line Feed**
&	**Ampersand**	**38**	**26**	**ETB**	**End of Transmission Block**
'	**Apostrophe, Prime**	**39**	**27**	**ESC**	**Escape**
(	**Left Parenthesis**	**40**	**28**	**nada**	**nada**
)	**Right Parenthesis**	**41**	**29**	**nada**	**nada**
*****	**Asterisk, "star"**	**42**	**2A**	**SM**	**Set Mode**
+	**Plus Sign**	**43**	**2B**	**CU2**	**Customer Use 2**
,	**Comma**	**44**	**2C**	**nada**	**nada**
-	**Hyphen, Minus Sign**	**45**	**2D**	**ENQ**	**Enquiry**
.	**Period, Decimal Point, "dot"**	**46**	**2E**	**ACK**	**Acknowledge**
/	**Slash, Virgule**	**47**	**2F**	**BEL**	**Bell**
0	**0**	**48**	**30**	**nada**	**nada**
1	**1**	**49**	**31**	**nada**	**nada**
2	**2**	**50**	**32**	**SYN**	**Synchronous Idle**
3	**3**	**51**	**33**	**nada**	**nada**
4	**4**	**52**	**34**	**PN**	**Punch On**
5	**5**	**53**	**35**	**RS**	**Reader Stop**
6	**6**	**54**	**36**	**UC**	**Upper Case**
7	**7**	**55**	**37**	**EOT**	**End of Transmission**
8	**8**	**56**	**38**	**nada**	**nada**
9	**9**	**57**	**39**	**nada**	**nada**
:	**Colon**	**58**	**3A**	**nada**	**nada**
;	**Semicolon**	**59**	**3B**	**CU3**	**Customer Use 3**
<	**Less-than Sign**	**60**	**3C**	**DC4**	**Device Control 4**
=	**Equal Sign**	**61**	**3D**	**NAK**	**Negative Acknowledge**
>	**Greater-than Sign**	**62**	**3E**	**nada**	**nada**
?	**Question Mark**	**63**	**3F**	**SUB**	**Substitute**

HW IBM - tabela Ascii/Ebcdic - de 40h a 5Fh

ASCII		Dec	Hex	EBCDIC	
@	**At Sign**	**64**	**40**	**SP**	**Space**
A	**A**	**65**	**41**	**nada**	**nada**
B	**B**	**66**	**42**	**nada**	**nada**
C	**C**	**67**	**43**	**nada**	**nada**
D	**D**	**68**	**44**	**nada**	**nada**
E	**E**	**69**	**45**	**nada**	**nada**
F	**F**	**70**	**46**	**nada**	**nada**
G	**G**	**71**	**47**	**nada**	**nada**
H	**H**	**72**	**48**	**nada**	**nada**
I	**I**	**73**	**49**	**nada**	**nada**
J	**J**	**74**	**4A**	**¢**	**Cent Sign**
K	**K**	**75**	**4B**	**.**	**Period, Decimal Point, "dot"**
L	**L**	**76**	**4C**	**<**	**Less-than Sign**
M	**M**	**77**	**4D**	**(**	**Left Parenthesis**
N	**N**	**78**	**4E**	**+**	**Plus Sign**
O	**O**	**79**	**4F**	**|**	**Logical OR**
P	**P**	**80**	**50**	**&**	**Ampersand**
Q	**Q**	**81**	**51**	**nada**	**nada**
R	**R**	**82**	**52**	**nada**	**nada**
S	**S**	**83**	**53**	**nada**	**nada**
T	**T**	**84**	**54**	**nada**	**nada**
U	**U**	**85**	**55**	**nada**	**nada**
V	**V**	**86**	**56**	**nada**	**nada**
W	**W**	**87**	**57**	**nada**	**nada**
X	**X**	**88**	**58**	**nada**	**nada**
Y	**Y**	**89**	**59**	**nada**	**nada**
Z	**Z**	**90**	**5A**	**!**	**Exclamation Point**
[	**Opening Bracket**	**91**	**5B**	**$**	**Dollar Sign**
****	**Reverse Slant**	**92**	**5C**	*****	**Asterisk, "star"**
]	**Closing Bracket**	**93**	**5D**	**)**	**Right Parenthesis**
^	**Circumflex, Caret**	**94**	**5E**	**;**	**Semicolon**
_	**Underline, Underscore**	**95**	**5F**	**¬**	**Logical NOT**

HW IBM - tabela Ascii/Ebcdic - de 60h a 7Fh

ASCII		Dec	Hex	EBCDIC	
`	Grave Accent	96	60	-	Hyphen, Minus Sign
a	a	97	61	/	Slash, Virgule
b	b	98	62	nada	nada
c	c	99	63	nada	nada
d	d	100	64	nada	nada
e	e	101	65	nada	nada
f	f	102	66	nada	nada
g	g	103	67	nada	nada
h	h	104	68	nada	nada
i	i	105	69	nada	nada
j	j	106	6A	nada	nada
k	k	107	6B	,	Comma
l	l	108	6C	%	Percent
m	m	109	6D	_	Underline, Underscore
n	n	110	6E	>	Greater-than Sign
o	o	111	6F	?	Question Mark
p	p	112	70	nada	nada
q	q	113	71	nada	nada
r	r	114	72	nada	nada
s	s	115	73	nada	nada
t	t	116	74	nada	nada
u	u	117	75	nada	nada
v	v	118	76	nada	nada
w	w	119	77	nada	nada
x	x	120	78	nada	nada
y	y	121	79	nada	nada
z	z	122	7A	:	Colon
{	Opening Brace	123	7B	#	Number Sign, Octothorp, "pound”
|	Vertical Line	124	7C	@	At Sign
}	Closing Brace	125	7D	'	Apostrophe, Prime
~	Tilde	126	7E	=	Equal Sign
DEL	Delete	127	7F	"	Quotation Mark

HW IBM - tabela Ascii/Ebcdic - de 80h a 9Fh

ASCII		Dec	Hex	EBCDIC	
nada	**Reserved**	**128**	**80**	**nada**	**nada**
nada	**Reserved**	**129**	**81**	**a**	**a**
nada	**Reserved**	**130**	**82**	**b**	**b**
nada	**Reserved**	**131**	**83**	**c**	**c**
IND	**Index (FE)**	**132**	**84**	**d**	**d**
NEL	**Next Line (FE)**	**133**	**85**	**e**	**e**
SSA	**Start of Selected Area**	**134**	**86**	**f**	**f**
ESA	**End of Selected Area**	**135**	**87**	**g**	**g**
HTS	**Horizontal Tabulation Set (FE)**	**136**	**88**	**h**	**h**
HTJ	**Horizontal Tab with Justification (FE)**	**137**	**89**	**i**	**i**
VTS	**Vertical Tabulation Set (FE)**	**138**	**8A**	**nada**	**nada**
PLD	**Partial Line Down (FE)**	**139**	**8B**	**nada**	**nada**
PLU	**Partial Line Up (FE)**	**140**	**8C**	**nada**	**nada**
RI	**Reverse Index (FE)**	**141**	**8D**	**nada**	**nada**
SS2	**Single Shift Two (1)**	**142**	**8E**	**nada**	**nada**
SS3	**Single Shift Three (1)**	**143**	**8F**	**nada**	**nada**
DCS	**Device Control String (2)**	**144**	**90**	**nada**	**nada**
PU1	**Private Use One**	**145**	**91**	**j**	**j**
PU2	**Private Use Two**	**146**	**92**	**k**	**k**
STS	**Set Transmit State**	**147**	**93**	**l**	**l**
CCH	**Cancel Character**	**148**	**94**	**m**	**m**
MW	**Message Waiting**	**149**	**95**	**n**	**n**
SPA	**Start of Protected Area**	**150**	**96**	**o**	**o**
EPA	**End of Protected Area**	**151**	**97**	**p**	**p**
Nada	**Reserved**	**152**	**98**	**q**	**q**
Nada	**Reserved**	**153**	**99**	**r**	**r**
Nada	**Reserved**	**154**	**9A**	**nada**	**nada**
CSI	**Control Sequence Introducer (1)**	**155**	**9B**	**nada**	**nada**
ST	**String Terminator (2)**	**156**	**9C**	**nada**	**nada**
OSC	**Operating System Command (2)**	**157**	**9D**	**nada**	**nada**
PM	**Privacy Message (2)**	**158**	**9E**	**nada**	**nada**
APC	**Application Program Command (2)**	**159**	**9F**	**nada**	**nada**

HW IBM - tabela Ascii/Ebcdic - de A0h a BFh

ASCII	Dec	Hex	EBCDIC
nada nada	**160**	**A0**	**nada nada**
nada nada	**161**	**A1**	**nada nada**
nada nada	**162**	**A2**	**s s**
nada nada	**163**	**A3**	**t t**
nada nada	**164**	**A4**	**u u**
nada nada	**165**	**A5**	**v v**
nada nada	**166**	**A6**	**w w**
nada nada	**167**	**A7**	**x x**
nada nada	**168**	**A8**	**y y**
nada nada	**169**	**A9**	**z z**
nada nada	**170**	**AA**	**nada nada**
nada nada	**171**	**AB**	**nada nada**
nada nada	**172**	**AC**	**nada nada**
nada nada	**173**	**AD**	**nada nada**
nada nada	**174**	**AE**	**nada nada**
nada nada	**175**	**AF**	**nada nada**
nada nada	**176**	**B0**	**nada nada**
nada nada	**177**	**B1**	**nada nada**
nada nada	**178**	**B2**	**nada nada**
nada nada	**179**	**B3**	**nada nada**
nada nada	**180**	**B4**	**nada nada**
nada nada	**181**	**B5**	**nada nada**
nada nada	**182**	**B6**	**nada nada**
nada nada	**183**	**B7**	**nada nada**
nada nada	**184**	**B8**	**nada nada**
nada nada	**185**	**B9**	**` Grave Accent**
nada nada	**186**	**BA**	**nada nada**
nada nada	**187**	**BB**	**nada nada**
nada nada	**188**	**BC**	**nada nada**
nada nada	**189**	**BD**	**nada nada**
nada nada	**190**	**BE**	**nada nada**
nada nada	**191**	**BF**	**nada nada**

HW IBM - tabela Ascii/Ebcdic - de C0h a DFh

ASCII		Dec	Hex	EBCDIC	
nada	**nada**	**192**	**C0**	**nada**	**nada**
nada	**nada**	**193**	**C1**	**A**	**A**
nada	**nada**	**194**	**C2**	**B**	**B**
nada	**nada**	**195**	**C3**	**C**	**C**
nada	**nada**	**196**	**C4**	**D**	**D**
nada	**nada**	**197**	**C5**	**E**	**E**
nada	**nada**	**198**	**C6**	**F**	**F**
nada	**nada**	**199**	**C7**	**G**	**G**
nada	**nada**	**200**	**C8**	**H**	**H**
nada	**nada**	**201**	**C9**	**I**	**I**
nada	**nada**	**202**	**CA**	**nada**	**nada**
nada	**nada**	**203**	**CB**	**nada**	**nada**
nada	**nada**	**204**	**CC**	**nada**	**nada**
nada	**nada**	**205**	**CD**	**nada**	**nada**
nada	**nada**	**206**	**CE**	**nada**	**nada**
nada	**nada**	**207**	**CF**	**nada**	**nada**
nada	**nada**	**208**	**D0**	**nada**	**nada**
nada	**nada**	**209**	**D1**	**J**	**J**
nada	**nada**	**210**	**D2**	**K**	**K**
nada	**nada**	**211**	**D3**	**L**	**L**
nada	**nada**	**212**	**D4**	**M**	**M**
nada	**nada**	**213**	**D5**	**N**	**N**
nada	**nada**	**214**	**D6**	**O**	**O**
nada	**nada**	**215**	**D7**	**P**	**P**
nada	**nada**	**216**	**D8**	**Q**	**Q**
nada	**nada**	**217**	**D9**	**R**	**R**
nada	**nada**	**218**	**DA**	**nada**	**nada**
nada	**nada**	**219**	**DB**	**nada**	**nada**
nada	**nada**	**220**	**DC**	**nada**	**nada**
nada	**nada**	**221**	**DD**	**nada**	**nada**
nada	**nada**	**222**	**DE**	**nada**	**nada**
nada	**nada**	**223**	**DF**	**nada**	**nada**

HW IBM - tabela Ascii/Ebcdic - de E0h a FFh

ASCII	Dec	Hex	EBCDIC
nada nada	**224**	**E0**	**nada nada**
nada nada	**225**	**E1**	**nada nada**
nada nada	**226**	**E2**	**S S**
nada nada	**227**	**E3**	**T T**
nada nada	**228**	**E4**	**U U**
nada nada	**229**	**E5**	**V V**
nada nada	**230**	**E6**	**W W**
nada nada	**231**	**E7**	**X X**
nada nada	**232**	**E8**	**Y Y**
nada nada	**233**	**E9**	**Z Z**
nada nada	**234**	**EA**	**nada nada**
nada nada	**235**	**EB**	**nada nada**
nada nada	**236**	**EC**	**nada nada**
nada nada	**237**	**ED**	**nada nada**
nada nada	**238**	**EE**	**nada nada**
nada nada	**239**	**EF**	**nada nada**
nada nada	**240**	**F0**	**0 0**
nada nada	**241**	**F1**	**1 1**
nada nada	**242**	**F2**	**2 2**
nada nada	**243**	**F3**	**3 3**
nada nada	**244**	**F4**	**4 4**
nada nada	**245**	**F5**	**5 5**
nada nada	**246**	**F6**	**6 6**
nada nada	**247**	**F7**	**7 7**
nada nada	**248**	**F8**	**8 8**
nada nada	**249**	**F9**	**9 9**
nada nada	**250**	**FA**	**nada nada**
nada nada	**251**	**FB**	**nada nada**
nada nada	**252**	**FC**	**nada nada**
nada nada	**253**	**FD**	**nada nada**
nada nada	**254**	**FE**	**nada nada**
nada nada	**255**	**FF**	**nada nada**