CONTROLE DO SISTEMA – J. C. L. – (PARA O SISTEMA IBM/360 – 370)

Roberto Bersotti

INFORMAÇÃO IEA 56 CPD 7	ABRIL/1977

INF. IEA 56
CPD 7

ABRIL/1977

CONTROLE DO SISTEMA – J. C. L. – (PARA O SISTEMA IBM/360 – 370)

Roberto Barsotti

CENTRO DE PROCESSAMENTO DE DADOS
(CPD)

INSTITUTO DE ENERGIA ATÔMICA
SÃO PAULO – BRASIL

APROVADO PARA PUBLICAÇÃO EM NOVEMBRO/1976

INSTITUTO DE ENERGIA ATÔMICA
Caixa Postal 11.049 (Pinheiros)
Cidade Universitária "Armando de Salles Oliveira"
SÃO PAULO – BRASIL

NOTA: Este trabalho foi conferido pelo autor depois de composto e sua redação está conforme o original, sem quaisquer correção ou mudança.

ÍNDICE

CONTROLE DO SISTEMA – J. C. L. (PARA O SISTEMA IBM/370)

Roberto Barsotti*

RESUMO

O J.C.L. – JOB CONTROL LANGUAGE – é uma linguagem que permite a comunicação entre o Sistema e o usuário fornecendo a este último as ferramentas para resolver seus problemas ao nível do computador.

Como toda linguagem possue sua estrutura própria e uma série de normas que a regem. O perfeito domínio do J.C.L. permite ao usuário uma utilização eficiente e otimizada da máquina

PREFÁCIO

Esta publicação, tratando do controle do Sistema – J.C.L. –, é, como a anterior sobre O.S., o resultado de um curso interno que tivemos oportunidade de desenvolver no período de setembro-outubro de 75, para alguns programadores do nosso C.P.D.

O que se pretende neste pequeno trabalho não é formar mas sim informar os eventuais leitores sobre o que seja o JOB CONTROL LANGUAGE, tentando apresentar esta linguagem de controle como um todo para que possa ser percebida sua utilização, seus recursos e sua estrutura.

Acreditamos que após a leitura cuidadosa deste trabalho o leitor possa prosseguir seus estudos de J.C.L. com menos dificuldades e com o espírito um pouco mais crítico. Se isto acontecer a presente publicação terá atingido seu objetivo e nós nos daremos por satisfeitos.

Por outro lado, para um melhor entendimento, principalmente por parte dos iniciantes, sugerimos que antes da leitura do presente trabalho seja lida uma outra publicação, também de nossa autoria, sobre O.S. na qual são apresentados os conceitos básicos do Sistema Operacional O.S. a muitos dos quais este trabalho se refere.

Finalmente, queremos deixar claro que a aplicação eficiente do JOB CONTROL LANGUAGE exige a utilização de manuais apropriados, fornecidos pelo fabricante, sendo este trabalho insuficiente para este fim, por mais que tenhamos tido todo o cuidado para apresentar as informações nele contidas da forma mais precisa possível. A insuficiência desta publicação acreditamos não estar na precisão mas sim no grau de detalhamento o qual não podia ser maior considerando o objetivo do trabalho.

1 – INTRODUÇÃO

O computador é uma máquina projetada para executar um número praticamente ilimitado de tarefas, sem com isso ter de modificar sua estrutura a cada nova tarefa exigida. Isto é possível graças ao Sistema Operacional o qual, por sua concepção versátil, possibilita esta diferenciação de trabalho executado pela máquina propriamente dita. Portanto cabe ao Sistema Operacional controlar a máquina e servir de ligação entre esta e o programa do usuário já que, se é o Sistema Operacional que controla a

* Analista de Sistemas – Centro de Processamento de Dados; Instituto de Energia Atômica; São Paulo, SP.

máquina, cabe ao programa de aplicação definir as instruções que dizem ao computador o que fazer. Como é que o usuário diz ao Sistema Operacional o que fazer e que recursos são necessários? Estas informações são passadas ao sistema através do J.C.L. – JOB CONTROL LANGUAGE – que é uma linguagem usada para permitir a comunicação entre o usuário e o sistema operacional. Para sermos mais exatos convém dizer que o J.C.L. permite a comunicação entre o usuário e o JOB MANAGEMENT e, em especial, com o JOB SCHEDULER. O JOB MANAGEMENT constitui-se de três partes:

- READER/INTERPRETER – cuja função é ler e analisar os cartões de controle do job, verificando se não há erros de codificação; colocar as informações contidas nos J.C.C. – JOB CONTROL CARDS – numa série de tabelas de uso do sistema.

- INITIATOR/TERMINATOR – cuja função é alocar os recursos requeridos para a execução de um step do job bem como carregar e transferir o controle ao programa a ser executado. Além disso terminar o step quando a execução do programa se completa e selecionar um job da input work queue.

- OUTPUT/WRITER – controla a gravação dos dados de saída do job.

2 – GENERALIDADES DO J. C. L.

2.1 – Comandos do Job Control Language

Existem nove comandos do J.C.L. para descrever o job ao sistema. São eles:

- Comando JOB – usado para identificar o job, marcando seu início e o fim do anterior.

- Comando EXEC – usado para marcar o início de um step e o fim do step anterior. Identifica o programa a ser executado ou a procedure a ser chamada.

- Comando DD – identifica um data set e descreve seus atributos.

- Comando Delimitador – usado no input stream para indicar o fim dos dados.

- Comando Nulo – usado para indicar o fim de um job stream.

- Comando PROC – marca o início de uma procedure in-stream e opcionalmente de uma procedure catalogada. No primeiro caso pode ser usado para atribuir valores a parâmetros simbólicos. No segundo caso é usado para atribuir valores a parâmetros simbólicos.

- Comando PEND – marca o fim de uma procedure in stream.

- Comando de Comentários – usado para enviar informações auxiliares que não são consideradas pelo sistema mas apenas impressas para utilização do usuário.

- Comando de Comandos – usado para entrar com comandos através do input-stream.

Nota – É muito comum utilizar a palavra cartão em lugar de comando para referir-se aos "STATEMENTS" do J.C.L. Daí ser mais frequente alguém falar em cartão job do que falar em comando job, cartão DD em vez de comando DD, e assim por diante. Do ponto de vista pragmático isto pouco importa, importando mesmo a utilização correta e eficiente do comando, ou cartão como se queira. Porém a rigor o certo é comando, não somente por ser a tradução de "statement" mas por se tratar realmente de comandos dos quais os cartões são apenas suportes físicos.

2.2 – Campos nos Cartões de J. C. L.

2.2.1 – Campo Inicial – este campo ocupa as colunas 1 e 2 e, com exceção do cartão delimitador que contém /*, contém sempre //.

2.2.2 – Campo de Nome – identifica o cartão permitindo a outros cartões e blocos de controle do sistema fazerem referências a ele. Este campo deve iniciar na coluna 3 do cartão e pode comportar dois tipos de nomes: nomes simples e nomes qualificados, obedecendo às características de formação de cada um deles.

2.2.3 – Campo de Operação – especifica o tipo de cartão de controle ou, no caso de um comando, o comando. Este campo deve ser precedido e seguido de, pelo menos, um branco.

2.2.4 – Campo de Operando – contém parâmetros separados por vírgulas. Deve ser precedido e seguido de, pelo menos, um branco.

2.2.5 – Campo de Comentários – pode conter qualquer informação que seja útil. Deve ser precedido de, pelo menos, um branco.

2.2.6 – Campo de Continuação – este campo, ocupando a coluna 72, pode conter qualquer caracter inclusive branco, quando o cartão que o contém tiver continuação. Se não tiver continuação este campo deve conter somente branco. Não precisa ser precedido ou seguido por um branco.

2.2.7 – Campo de Identificação – ocupando as colunas 73 a 80, pode ser usado para conter códigos que indentifiquem o cartão. Não precisa ser precedido nem seguido de branco.

Observação – Embora os campos existentes sejam estes, nem todos os comandos de controle contêm todos os campos.

A Figura 1 mostra os campos obrigatórios e os opcionais.

(MANDO)	CAMPOS			
	COLS. 1 e 2	COLUNAS 3 a 71	COL. 72	COLS. 73 a 80
JOB	//	NOME OPERAÇÃO (JOB) OPERANDO[1] COMENTARIOS[2]	CONT[1]	IDENTIF[1]
EXECUTE	//	NOME[1] OPERAÇÃO (EXEC) OPERANDO COMENTARIOS[1]	CONT[1]	IDENTIF[1]
DATA DEFINITION	//	NOME[1] OPERAÇÃO (DD) OPERANDO COMENTARIOS[1]	CONT[1]	IDENTIF[1]
PROC (Cabeçalho)	//	NOME[1] OPERAÇÃO (PROC) OPERANDO COMENTARIOS[1]	CONT[1]	IDENTIF[1]
PROC (Intermed.)	//	NOME OPERAÇÃO (PROC) OPERANDO[1] COMENTARIOS[2]	CONT[1]	IDENTIF[1]
PROC ESQUELEND	//	NOME[1] OPERAÇÃO (PEND) COMENTARIOS[1]	CONT[1]	IDENTIF[1]
COMANDO	/	OPERAÇÃO (COMANDO) OPERANDO COMENTARIOS[1]	CONT[1]	IDENTIF[1]
DELIMITER	/*	COMENTARIOS[1]	CONT[1]	IDENTIF[1]
NULL	//	BRANCOS		IDENTIF[1]
COMMENT	/*	COMENTARIOS		IDENTIF[1]

1 – OPCIONAL

2 – Não são [illegible] comentários e menos [illegible] pelo menos um operando [illegible]

Figura 1 – [illegible] Obrigatórios e Opcionais nos Cartões de J. C. L.

2.3 – Regras para Continuar um Cartão de J. C. L.

– A interrupção somente pode se dar após a codificação de um parâmetro ou subparâmetro completo, isto é, após uma vírgula.

– A vírgula não deve ultrapassar a coluna 71.

– O cartão continuação deve ter // codificado nas colunas 1 e 2.

– A codificação da continuação deve iniciar entre as colunas 4 e 16.

OBSERVAÇÕES

I – Os comandos abaixo não podem continuar em outro cartão:

– COMANDO (//)

– DELIMITADOR (/*)

– COMENTARIO (//*)

– NULO (//)

No caso dos comandos: COMENTÁRIO e COMANDO, embora não possam ter continuação em outro cartão, podem ser codificados tantos cartões quantos forem necessários.

II – Não é necessário perfurar um caracter diferente de branco na coluna 72 para indicar que há continuação no cartão seguinte.

2.4 – Tipos de Parâmetros no J. C. L.

Existem 4 tipos de parâmetros em J.C.L.

2.4.1 – Opcionais – usados somente quando necessário

2.4.2 – Obrigatórios – sempre usados

2.4.3 – Posicionais – cujo significado depende da posição relativa dentro do conjunto de parâmetros. A ausência de um destes parâmetros deve ser indicada por uma vírgula, porém, se o parâmetro ausente for o último ou se todos os parâmetros posicionais forem omitidos, não haverá necessidade de serem codificadas as vírgulas correspondentes.

2.4.4 – Palavras-chave – cujo significado é dado pela própria palavra independentemente da posição relativa ocupada. São codificadas sempre após os parâmetros posicionais (se houver).

2.5 – Uso dos Parenteses e Apóstrofes no J. C. L.

– Os parenteses são comumente usados nos casos de subparâmetros dentro de parâmetros.

– O apóstrofe é usado principalmente para casos em que seja necessário o uso de caracteres especiais ou nomes com mais de oito caracteres.

Observação – há casos em que o apóstrofe pode ser substituído por parênteses.

Na figura 2 podemos ver o resumo esquemático de tudo o que foi abordado neste ítem 2 – Generalidades do J.C.L.

COMANDO	SÍMBOLO	CAMPOS				FUNÇÃO	PARÂMETROS	OBSERVAÇÕES
		NOME	OPERANDO	PARÂMETROS	COMENTÁRIOS			
JOB	//	NOME obrigatório	JOB	Parâmetros posicionais e/ou palavras chave	Pode ter	Não tem	Pode ter	Identifica o início de um job e fornece informações para o sistema
EXECUTE	//	NOME opcional em certas circunstâncias mas obrigatório em outras	EXEC	Nome a ser executado ou proc. catalogada a ser chamada. Após isto, os parâmetros desejados	Pode ter	Pode ter	Pode ter	Marca o início de um step e o fim do step prior. Identifica o programa a ser executado ou a procedure catalogada a ser chamada
DATA DEFINITION	//	NOME opcional em certas circunstâncias mas obrigatório em outras	DD	Parametros posicionais e/ou palavras chave	Pode ter	Pode ter	Pode ter	Identifica o data set e descreve seus atributos
PROCEDURE	//	PROCNAME obrigatório em procs. in-stream opcional em procs catalogadas	PROC	Parametros simbolicos e seus valores separados por virgulas	Comete um erro se campo de comentário for modificado	Pode ter	Pode ter	Marca o início de uma procedure in-stream e opcionalmente de uma procedure catalogada
PROCEDURE END	//	PENDNAME opcional	PEND	C O M E N T A R I O S		Não tem	Pode ter	Marca o fim de uma procedure in-stream
COMMAND	//	C O M A N D O S			Pode ter	Não tem	Pode ter	Usado para entrar com comandos at[illegible] no input stream
DELIMITER	/*	C O M E N T A R I O S				Não tem	Pode ter	Usado no input stream para indicar o fim dos dados
COMMENT	//	*	C O M E N T A R I O S			Não tem	Pode ter	Usado para entrar com informações auxiliares
NULL	//	B R A N C O S				Não tem	Pode ter	Pode ser usado para indicar o fim de um job stream

Figura 2 — Comandos do J.C.L. — seus Campos e suas Funções.

3 – COMANDO JOB

O comando JOB marca o início de um job e o término do job anterior. Todos os parâmetros em seu campo de operando são opcionais e se nenhum deles for codificado não poderão ser codificados os comentários.

3.1 – Formato geral

```
//JOBNAME ⎵ JOB ⎵ operandos ⎵ comentários
```

O JOBNAME é obrigatório e deve ter todas as características de um nome simples.

Trabalhando em multiprogramação não devem existir dois ou mais jobs com o mesmo JOBNAME.

3.2 – Regras para Codificação

– Codificar os caracteres // nas colunas 1 e 2.

– Codificar o jobname começando na coluna 3.

– Após o jobname deixar pelo menos um branco (⎵).

– Codificar a palavra JOB.

– Após a palavra JOB deixar pelo menos um branco.

– Codificar os parâmetros posicionais necessários separando uns dos outros com vírgulas.

– Codificar os parâmetros "palavras-chave" necessários, separando uns dos outros com vírgulas.

– Após o último parâmetro deixar pelo menos um branco.

– Codificar os comentários necessários.

Observação – As regras a que nos referimos no ítem 3.2 são regras gerais de codificação dentro das quais cada instalação, de acordo com suas características e com certas normas adotadas pelo CPD à qual pertence, pode codificar ou omitir aquelas informações que lhe pareça necessário desde que sejam observadas as regras gerais. No nosso caso CPD do IEA - o comando JOB é codificado em dois cartões cujas estruturas estão descritas no Boletim Informativo e devem ser observadas rigorosamente. A não observância do formato do cartão JOB descrito no Boletim Informativo implicará no cancelamento do JOB. No decorrer desta publicação trataremos o J.C.L. de forma genérica não nos preocupando com situações especiais desta ou daquela instalação.

3.3 – Parâmetros do Comando JOB

Existem dois tipos de parâmetros que podem ser codificados no comando JOB: os posicionais e as palavras-chave. Embora os parâmetros possam ser obrigatórios ou opcionais, no caso do comando JOB todos os parâmetros são opcionais.

3.3.1 – Parâmetros Posicionais

Devem preceder os parâmetros "palavra-chave" e se usados devem ser codificados na seguinte ordem:

– Informações de contabilização

– Identificação do programador

Faltando o primeiro deve ser codificada uma vírgula em seu lugar. Faltando o último ou ambos não há necessidade de codificar vírgula.

3.3.1.1. – Informações de Contabilização

Este parâmetro pode estar constituído por subparâmetros. Neste caso os subparâmetros devem ser codificados entre parênteses ou apóstrofes.

Ex: (745,GRUPO6,ADIN) ou '745,GRUPO6,ADIN'

As informações de contabilização **não podem exceder** 142 caracteres incluindo as vírgulas que separam os subparâmetros. Se as informações de contabilização tiverem que ser continuadas em outro cartão os subparâmetros devem ser colocados entre parênteses e não entre apóstrofes.

Havendo caracteres especiais nos subparâmetros, com exceção do hífem (–), fechar as informações de contabilização entre apóstrofes '5432,10/02/75' ou então entre parênteses mas tendo o cuidado de fechar os subparâmetros que contêm os caracteres especiais entre apóstrofes; ex: (5432,'10/02/75'), neste caso os apóstrofes **não serão** considerados parte da informação.

3.3.1.2 – Identificação do Programador

Não pode exceder 20 caracteres, incluindo caracteres especiais. Se este parâmetro contiver caracteres especiais, com exceção do ponto, ele deve ser colocado entre apóstrofes. Ex:

```
* //EXPLO1    JOB    37C96A
* //EXPLO2    JOB    (45B3,SECAO21,999),JOAO
* //EXPLO3    JOB    (45B3,'SEC1/2',43),MARIA
* //EXPLO4    JOB    '45B3,SEC7/8,43',MARIA
* //EXPLO5    JOB    ,J.S.SILVA
* //EXPLO6    JOB    37C 'P. L. SILVA'
* //EXPLO7    JOB
```

3.3.2 – Parâmetros Palavras-Chave

Podem ser codificados em qualquer ordem desde que sejam precedidos pelos parâmetros posicionais, caso haja

Os parâmetros palavras chave do comando JOB são os seguintes:

- CLASS
- COND
- MSGCLASS
- MSGLEVEL
- NOTIFY (MVT com TSO)
- PRTY
- RD
- REGION (MVT)
- RESTART
- ROLL (MVT)
- TIME
- TYPRUN

Destes parâmetros veremos apenas os mais usados normalmente:

3.3.2.1 – CLASS

```
CLASS=jobclass
```

jobclass – indica a qual das 15 classes (A,O) o job vai pertencer.

ex:

```
CLASS=J
```

o **default** é

```
CLASS=A
```

Nota – default é aquele valor que, quando omitido o parâmetro, é assumido pelo computador. No nosso caso acima, se o parâmetro CLASS não for codificado o computador assume, por default, A.

3.3.2.2 – COND

```
COND=((CODIGO,OPERADOR),. . . . )
```

código – número decimal compreendido entre **0** e **4095**. Este número é comparado com o código de retorno emitido por cada job step.

operador – determina que tipo de comparação deve ser efetuada com o código de retorno. Os operadores são:

GT (GREATER THAN) – Maior que

GE (GREATER THAN OR EQUAL TO) – Maior ou igual a

EQ (EQUAL TO) – Igual a

LT (LESS THAN) – Menor que

LE (LESS THAN OR EQUAL TO) – Menor ou igual a

NE (NOT EQUAL TO) – Diferente de

Cada job step emite um código de retorno no final do procedimento. O parâmetro COND condiciona a continuação do processamento dos job steps seguintes ao resultado da comparação entre o código de retorno dos job-steps anteriores e o valor especificado no parâmetro COND. O código de retorno pode ser testado até 8 vezes e se um dos testes for satisfeito os steps restantes do job não serão executados.

Ex:

COND=((50,GE),(60,LT))

significa que:

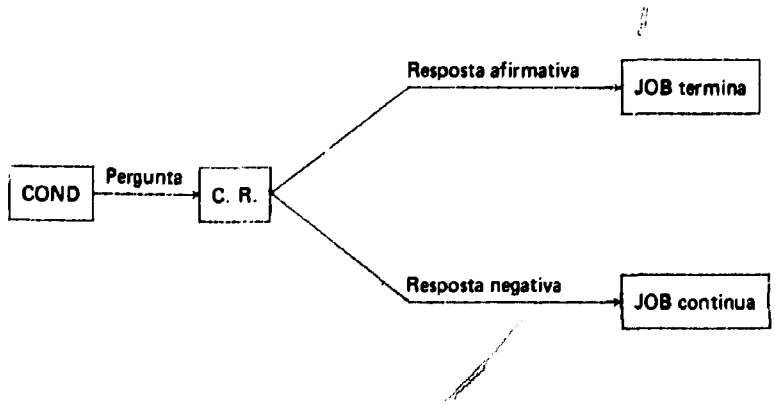
Se **50** for maior ou igual ao código de retorno, ou se **60** for menor que o código de retorno os jobs steps seguintes não serão executados. Em outras palavras, os steps deste job serão executados enquanto os códigos de retorno ficarem entre 51 e **60**.

COND=(9,LT)

significa que:

se 9 for menor que o código de retorno o job termina.

O mecanismo de funcionamento do parâmetro COND está esquematizado abaixo.



3.3.2.3 – MSGCLASS

MSGCLASS=output class

output class – indica para que classe de saída as mensagens do sistema (relativas ao job) devem ser dirigidas. As classes de saída são representadas por letras (A,Z) ou números (0,9).

ex:

MSGCLASS=F

MSGCLASS=4

o **default** é

MSGCLASS=A

3.3.2.4 – MSGLEVEL

MSGLEVEL=(comandos,Mensagens)

comandos – especifica quais comandos de controle devem ser listados na saída do job.

Pode ter valor de 0, 1 ou 2

0 – lista somente o cartão JOB

1 – lista todos os cartões de controle do job, ou seja: os do Input Stream e os da SYS1 PROCLIB.

2 – lista somente os cartões de controle do job que entraram via Input Stream.

Observação – Na listagem, os cartões de controle do Input Stream são reconhecidos por // nas posições 1 e 2 ao passo que os cartões de controle da SYS1.PROCLIB são caracterizados por XX nas mesmas posições.

Mensagens : diz quais mensagens de alocação e término devem ser listadas na saída do job.

Pode ter valores 0 e 1.

0 – nenhuma mensagem de alocação e término é listada, a menos que o Job termine anormalmente.

1 – todas as mensagens de alocação e término são listadas.

ex: MSGLEVEL=(0,0)

MSGLEVEL=(,0)

MSGLEVEL=(2,1)

o default é:

MSGLEVEL=(1,1)

3.3.2.5 – PRTY

PRTY=prioridade

prioridade – designa a prioridade do job e varia de 0 a 13, onde 13, por ser a mais alta, é aconselhado evitar de usar.

Nota – quando o job é iniciado, o sistema converte a prioridade do job numa prioridade de execução (Dispatching Priority) permitindo à task ou tasks deste job competirem com outras tasks para o uso da U.C.P. e da memória principal.

ex:

PRTY=4

o default é

PRTY=1

3.3.2.6 – REGION

REGION=ValorK

ValorK – *'valor' especifica o número de áreas contíguas de* 1024 bytes da memória principal a ser alocada para cada step do job.

ex:

REGION=248K

3.3.2.7 – ROLL

ROLL=(X,Y)

X – declara se os steps do job podem ser rolados fora da memória ou não.

Se X for substituído por **YES** os jobs steps **podem** ser rolados.

Se X for substituído por **NO** os jobs steps **não podem** ser rolados.

Y – declara se os steps do job podem rolar fora da memória um step de outro job.

Se Y for substituído por **YES** os jobs step **podem** rolar o step de outro job.

Se Y for substituído por **NO** os jobs steps **não podem** rolar steps de outro job.

Resumindo temos:

ROLL = ({ YES,YES / NO,NO })

- 1º → declara a possibilidade dos steps poderem rolar
- 2º → declara a possibilidade dos steps poderem ser rolados

ex:

ROLL=(NO,YES)

ROLL=(YES, YES)

ROLL=(YES,NO)

ROLL=(NO,NO)

o **default** é

ROLL=(YES,NO)

3.3.2.8 – TIME

TIME={(minutos,segundos) / 1440}

minutos – especifica o número máximo de minutos que o job pode usar a U.C.P. Este número deve ser menor que 1440 (24 horas).

segundos – especifica o número máximo de segundos que o job pode usar a U.C.P., além dos minutos que eventualmente tenham sido especificados. Se os minutos não forem especificados o job poderá usar a U.C.P. somente pelo número de segundos codificados.

1440 – especifica que o tempo não será contado.

ex:

TIME=(14,8)

TIME=3

TIME=(,3)

TIME=1440

3.3.2.9 – TYPRUN

```
TYPRUN=HOLD
```

HOLD – especifica que o job ficará preso na fila de entrada (SYS1.SYSJOBQE) até ser liberado por um comando RELEASE.

4 – COMANDO EXEC

É o primeiro comando de cada step, sendo seguido por comandos DD e dados pertencentes ao step.

A função principal do comando EXEC é identificar o programa a ser executado ou a procedure catalogada a ser chamada. Todos os parâmetros no campo de operando são opcionais. Um job não pode conter mais do que 255 steps.

4.1 – Formato Geral

```
//STEPNAME ŏ EXEC ŏ operandos ŏ comentários
```

O stepname identifica o step dentro do job. É opcional mas quando usado deve preencher os seguintes requisitos:

- Nome simples com as características de nome simples.
- Começar na coluna 3 do cartão que contém o comando EXEC.
- Deve ser único dentro do job.

4.2 – Regras para Codificação

- Codificar // nas colunas 1 e 2
- Opcionalmente pode ser codificado o nome do step, mas se o for deve iniciar na coluna 3.
- Codificar, após stepname (se for codificado), ou após // um ou mais brancos.
- Codificar EXEC.
- Após o EXEC codificar um ou mais brancos.
- Identificar o programa a ser executado (PGM) ou a procedure catalogada a ser chamada (PROC). Quando se tratar de procedure catalogada pode ser omitida a palavra PROC.
- Codificar qualquer parâmetro palavra chave separando um do outro com vírgulas.

– Codificar pelo menos um branco

– Codificar os comentários desejados.

4.3 – Parâmetros do Comando EXEC

Existem dois tipos de parâmetros no comando EXEC, os posicionais e as palavras chave, ambos opcionais no caso do comando EXEC.

4.3.1 – Parâmetros Posicionais

Os parâmetros posicionais do comando EXEC são dois e são *mutuamente exclusivos*:

***PGM**

***PROC**

4.3.1.1 – PGM

```
PGM={nome do programa
    {*.stepname.ddname
    {*.stepname.procstepname.ddname
```

nome do programa – é o nome do membro, ou aliás, do programa a ser executado. O programa deve ser membro de um data set particionado residente numa biblioteca do sistema ou particular

***.stepname.ddname** – é uma referência (BACKWARD ⟶ para trás) a um cartão DD que *define, como membro de um P.D.S., o programa a ser executado*; Stepname é o nome do step no qual aparece o cartão DD. Normalmente esta forma é usada quando um PDS temporário é criado previamente num step, para armazenar um programa até o momento de requisitá-lo.

*** .stepname.procstepname.ddname** – é uma referência (BACKWARD) para um cartão DD que pertence a um certo step dentro de uma procedure catalogada que foi selecionada *por um step anterior deste mesmo job. Stepname é o nome* do step que chama a procedure. Procstepname é o nome da procedure que contém o cartão DD. Normalmente esta forma é usada quando o step de uma procedure catalogada, chamada por um step do mesmo job, cria um PDS temporário para armazenar o programa até o momento de requisitá-lo.

Exs:

1 – //STEP1 EXEC PGM=CONTROLE

especifica que o programa chamado CONTROLE é membro do PDS SYS1.LINKLIB, ou a ele concatenado.

2 – //JOB8 JOB MSGLEVEL=(2,0)

```
//STEP2  EXEC  PGM=UPDT
//DDA    DD    DSNAME=SYS1.LINKLIB(P40),DISP=OLD
//STEP3  EXEC  PGM=*.STEP2.DDA
```

```
3 – //STEP1  EXEG  PROC=RRA
//STEP2  EXEC  PGM=*.STEP1.RRA.DD2
```

Procedure RRA		
//A	EXEC	PGM=AC
//DD1	DD	.....
//DD2	DD	DSN=.....
//B	EXEC	PGM=BD
//DD3	DD	.....
//DD4	DD	.....

4.3.1.2 – PROC

```
PROC=nome da procedure
nome da procedure
```

nome da procedure – nome do membro, ou aliás, de um procedimento catalogado ou o nome de um procedimento in-stream a ser chamado.

Exemplo:

```
//STEPX  EXEC  PROC=ROTIN
//STEPX  EXEC  ROTIN
```

4.3.2 – Parâmetros Palavras-Chave

Podem ser codificados em qualquer ordem desde que sejam precedidos pelos parâmetros posicionais. Os parâmetros palavra chave do comando EXEC são os seguintes:

- ACCT
- COND
- DPRTY (MVT)
- PARM
- RD

- REGION (MVT)
- ROLL (MVT)
- TIME

Aqui tambem veremos os parâmetros mais utilizados usualmente

4.3.2.1 – ACCT

```
ACCT=(informações de contabilização. . .)
```

Informações de contabilização – pode estar constituido de subparâmetros separados por vírgulas.

Ex:

```
//STPA  EXEC  PGM=PAG,ACCT=(IEA,CPD,'30/12')
```

4.3.2.2 – COND

```
       / [(codigo, operador)                        ]    [EVEN] \
COND= (  [(codigo,operador,stepname)                ] ,  [    ]  )
       \ [(codigo, operador,stepname.procstepname)  ]    [ONLY] /
```

Código – número decimal entre 0 e 4095. Este número é comparado com o código de retorno emitido por todos os steps anteriores ou por determinado step.

Operador – estabelece o tipo de comparação a efetuar com o código de retorno.

Os operadores são:

GT – Greater than

GE – Greater than or Equal to

EQ – Equal to

LT – less than

LE – less than or Equal to

NE – Not Equal to

Stepname – nome de um step anterior com cujo código de retorno desejamos efetuar a comparação.

Stepname.procstepname – nome do step de uma procedure (procstepname) com cujo código de retorno desejamos efetuar a comparação. A procedure em questão foi chamada por um step anterior (stepname).

Even – especifica que o step deve ser executado sempre, tenha ou não havido términos anormais em steps anteriores.

Only – especifica que o step deve ser executado somente se um ou mais steps anteriores tiverem terminado anormalmente.

Nota – O código de retorno pode ser testado até 8 vezes se não forem codificados nem EVEN nem ONLY Caso contrário apenas 7 testes podem ser efetuados.

Explicação 1

Ex:

COND=((10,LT,STEPA),(20,EQ),ONLY)

leitura – Este step será executado somente se um ou mais steps anteriores tiverem terminado anormalmente

Contudo será ignorado se 10 for menor que o código de retorno emitido por STEPA ou se algum dos steps que terminaram normalmente tiver emitido código de retorno igual a 20.

resumo – Este step será executado se:

1 – alguns dos steps anteriores tiver abendado

2 – e o C R. do STEPA $\leqslant$ 10

3 – e todos os steps anteriores tiverem emitido C. R. $\neq$ 20.

Explicação 2

EX:

COND=((10,LT,STEPA),(20,EQ),EVEN)

leitura – Este step será ignorado se 10 for menor que o código de retorno emitido por STEPA ou se qualquer um dos steps anteriores tiver emitido um código de retorno igual a 20. Se não, será executado mesmo que algum dos steps anteriores tiver terminado anormalmente.

resumo – Este step será executado se:

1 – O C. R. de STEPA $\leqslant$ 10

2 – Nenhum dos steps anteriores tenha emitido C. R. = 20.

* Não importa se houve ou não términos anormais.

4.3.2.3 – PARM

PARM=Valor

Valor – Consiste de até 100 caracteres de informações que serão enviados ao programa de processamento.

Se trabalharmos com uma procedure é preciso determinar para qual etapa da procedure irão as informações.

Ex: //STEP1 EXEC COBUCLG,PARM.GO='16/02/76'

No exemplo acima o PARM GO diz que a data deve ir para o programa durante a fase de execução, da etapa GO da procedure COBUCLG.

4.3.2.4 – REGION

```
REGION=ValorK
```

ValorK – especifica o número de áreas contíguas de 1024 bytes – (1K) da memória principal a ser alocada para o step.

Ex:

```
//STEPUM EXEC PGM=A,REGION=400K
```

4.3.2.5 – ROLL

```
Roll=(x,y)
```

Funciona da mesma forma que o ROLL do comando JOB só que apenas para o step que o tem codificado.

Nota – Se o parâmetro ROLL for codificado no comando JOB, os parâmetros ROLL dos comandos EXEC daquele JOB serão ignorados.

4.3.2.6 – TIME

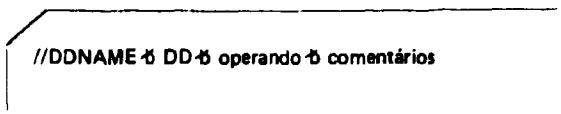
```
TIME={(minutos,segundos)
     {1440
```

Assim como o parâmetro ROLL o TIME funciona da mesma forma que o TIME do comando JOB só que apenas para o step que o tem codificado.

5 – COMANDO DD

O comando DD – DATA DEFINITION – descreve um data set a ser usado num step e especifica as facilidades de entrada e saída requeridas para uso do data set. Cada data set a ser usado num step requer um comando DD que o descreva.

5.1 – Formato Geral



O DDNAME é obrigatório exceto quando se tratar de data sets concatenados ou quando se tratar de segundos e terceiros DDs de um data set seqüencial indexado (áreas : PRIME,INDEX,OVERFLOW). O DDNAME deve ser um nome simples e não pode haver dois DDNAMEs iguais dentro de um mesmo step.

5.2 – Regras para codificação:

– Codificar os caracteres // nas colunas 1 e 2.

– Codificar o DDNAME (se for o caso) começando na coluna 3.

– Após o DDNAME deixar pelo menos um branco

– Codificar DD

– Após DD deixar pelo menos um branco

– Codificar os parâmetros posicionais necessários separando um do outro por vírgulas

– Codificar os parâmetros palavras-chave necessários separando um do outro por vírgulas

– Após o último parâmetro deixar pelo menos um branco

– Codificar os comentários necessários

5.3 – Parâmetro do comando DD

Todos os parâmetros do comando DD são opcionais mas pelo menos um deles deve estar presente.

No comando DD também temos dois tipos de parâmetros: os posicionais e as palavras-chave.

5.3.1 – Parâmetros posicionais

Devem preceder os parâmetros palavras-chave e são os seguintes:

– *

– DATA

– DUMMY

– DYNAM

Destes veremos apenas os mais utilizados.

5.3.1.1 – *

```
//DDNAME DD *
```

Especifica que o data set que segue este cartão deve entrar no input stream para uso do programa de processamento. Nenhum registro deve possuir // nas colunas 1 e 2. O fim dos dados é determinado por /*. (se for omitido, um cartão com // nas colunas 1 e 2 também indica o fim dos dados).

5.3.1.2 – DATA

```
//DDNAME DD DATA
```

Tem o mesmo uso do * com a diferença que o input stream contém cartões de JCL que devem ser tratados como dados. O fim é determinado apenas por /*. Um exemplo de aplicação é a criação de uma Procedure Catalogada.

5.3.1.3 – DUMMY

```
//DDNAME DD DUMMY
```

Permite executar o programa de processamento sem realizar operações de entrada e saída, alocação de espaço e disposição relativa àquele data set.

Quando o programa pedir para gravar um data set dummy, a requisição de gravação é reconhecida, mas não há transmissão de dados. Quando o programa pede para ler um data set dummy, a rotina de fim de data set é tomada imediatamente.

5.3.2 – Parâmetros palavras-chave

Podem ser codificados em qualquer ordem desde que sejam precedidos pelos parâmetros posicionais, se houver. Os parâmetros palavras-chave do comando DD são os seguintes:

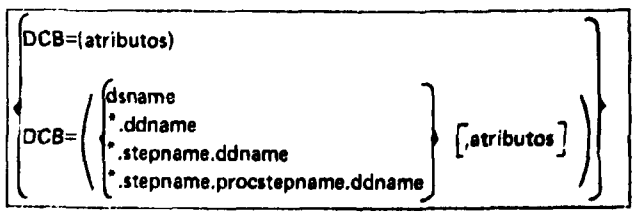
- AFF
- DCB
- DDNAME
- DISP

- DLM
- DSNAME ou DSN
- FCB
- LABEL
- OUTLIM
- QNAME (MFT ou MVT com TCAM)
- SEP
- SPACE
- SPLIT
- SUBALLOC
- SYSOUT
- TERM (MVT com TSO)
- UCS
- UNIT
- VOLUME ou VOL

Destes veremos os mais utilizados normalmente.

5.3.2.1 – DCB

Permite completar os campos do bloco de controle de dados originalmente construído pelo programa de processamento pela macro DCB.



atributos

BLKSIZE = tamanho máximo de um bloco, em bytes

BUFNO = número de buffers a ser designado

RECFM = formato dos registros do data set

LRECL = tamanho de um registro lógico, em bytes

DSORG = organização do data set

DEN = densidade de gravação em fita magnética

KEYLEN = tamanho das chaves usadas na data set, em bytes (máximo 255)

CODE = código da fita de papel

PRTSP = espacejamento na impressora

.

.

.

etc.

Dsname – permite copiar as informações do label de um data set similar (Dsname) desde que este seja catalogado e montado, além de estar gravado num dispositivo de acesso direto.

***.ddname** – permite copiar as informações da DCB de um comando DD anterior dentro do mesmo step (ddname).

***.stepname.ddname** – permite copiar as informações da DCB de um comando DD (ddname) que pertence a um step anterior (stepname) do mesmo JOB.

***.stepname.procstepname.ddname** – permite copiar as informações da DCB de um comando DD (ddname) que pertence a um step de uma procedure (procstepname) chamada por um step anterior (stepname) deste mesmo JOB.

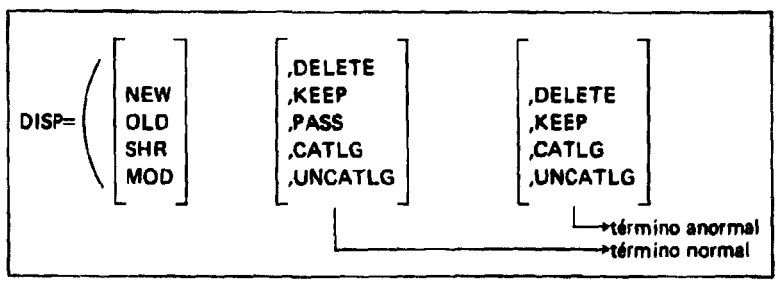
Exemplos:

```
//STEP01  DD  DSN=ARQUIVO.DE.PAGTO,DCB=(BLKSIZE=1024,LRECL=256)
//STEP02  DD  DSN=ARQUIVO.DE.VENCTO,DCB=*.STEP01
```

5.3.2.2 – DISP

O parâmetro descreve para o sistema o estado do data set e indica o que deve ser feito com o mesmo após o término do step que o processa ou no fim do JOB.

Pode-se indicar o que fazer com o data set caso o step termine normalmente (segundo subparâmetro posicional) e o que fazer caso o step termine de forma anormal (tercairo subparâmetro posicional).



NEW – O data set vai ser criado neste step

OLD – O data set já existe e é de uso exclusivo do JOB

SHR – O data set já existe e pode ser compartilhado até por todos os JOBs que estiverem correndo na mesma hora. O data set reside num volume de acesso direto.

MOD – O data set vai sofrer acréscimo de registros. O data set será aberto para a saída e o mecanismo de leitura/gravação ficará posicionado após o último registro do data set.

DELETE – O data set será deletado no final do step

KEEP – Mantém o data set para o step ou JOB subseqüente até que seja deletado.

PASS – O data set é passado para que steps seguintes tenham acesso a ele.

CATLG – Para que o data set seja catalogado.

UNCATLG – O data set é mantido mas a entrada no catálogo é removida.

O default é : DISP=(NEW,DELETE,DELETE) e é válido considerando o parâmetro como um todo. Parcialmente a regra é a seguinte: Quando o segundo subparâmetro for omitido o sistema conserva o data set que já existia antes do job e deleta aquele que não existia antes do job.

Quando o terceiro subparâmetro for omitido, o sistema usa a disposição especificada no segundo.

Ex:

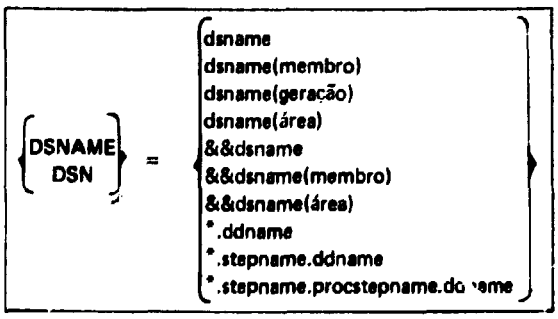
DISP=OLD

DISP=(,KEEP,DELETE)

DISP=(OLD,DELETE,KEEP)

5.3.2.3 – DSNAME

É utilizado para dar nome a um data set que posteriormente servirá para identificá-lo.



dsname – para definir o nome de um data set

ex: DSN=ABC

dsname(membro) – para data set particionado

ex: DSN=ARQ.AB(CD)

dsname(geração) – para data sets em GDG (Generation Data Group)

ex: DSN=ARQ.AB(+1) - - - geração relativa

DSN=ARQ.AB.G0001TO

dsname(area) – para data set sequencial indexado

ex: DSN=ABC(PRIME)

&&dsname – para definir o nome de um data set temporário

ex: DSN=&&ABC

&&dsname(membro) – para data set particionado temporário

ex: DSN=&&ABC(DE)

&&dsname(area) – para data set sequencial indexado temporário

ex:DSN=&&ABC(PRIME)

***.ddname** – para copiar o nome do data set de um comando DD anterior dentro do mesmo step.

ex: DSN= *.DD9

***.stepname.ddname** – para copiar o nome do data set de um comando DD (ddname) pertencente a um step (stepname) do mesmo job.

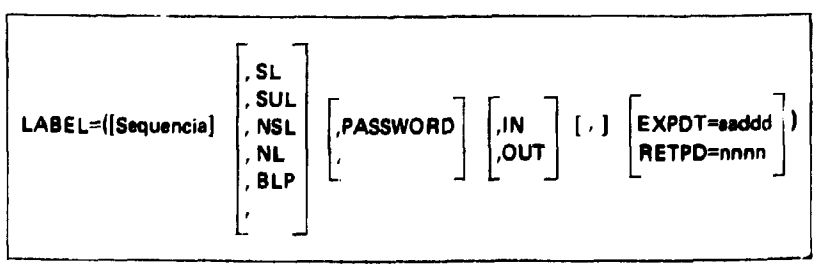
ex: DSN= *.STP1.DD3

***.stepname.procstepname.ddname** – para copiar o nome do data set de um comando DD (ddname) que pertence a uma procedure (procstepname) chamada por um step anterior (stepname) do mesmo job.

ex: DSN=*.STP5.GO.DD2

5.3.2.4 – LABEL

Os labels são usados pelo sistema operacional para identificar volumes, data sets e seus atributos.



Sequência – primeiro posicional – indica a ordem do data set no carretel de fita (1º, 8º, etc).

Tipo – segundo posicional – indica que tipo de rótulo (label) deve ser associado ao data set.

SL – Standard Label - label padrão

SUL – Standard User Label - label padrão e do usuário

NSL – No Standard Label - label não padrão

NL – No Label – sem label

BLP – Bypass Label Processing - ignora processamento de labels

PASSWORD – terceiro posicional – o data set só poderá ser aberto mediante a utilização de uma senha.

IN/OUT – quarto posicional – indica se o data set será usado como entrada (IN) ou saída (OUT).

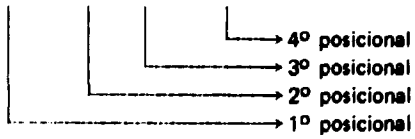
Retenção – usado para reter o data set por determinado período.

EXPDT=aaddd (ano e dia) estabelece a data de expiração

RETPD=nnnn(de 1 a 9999) expressa o número de dias que o data set deve ser retido.

Resumindo:

LABEL=(Sequência,Tipo,Password,IN/OUT,Retenção)



Ex:

LABEL=(5,SUL,PASSWORD,IN,RETPD=34)

LABEL=(,NSL)

LABEL=(,SUL,EXPDT=78097)

LABEL=(3,,PASSWORD)

LABEL=3

5.3.2.5 – SPACE

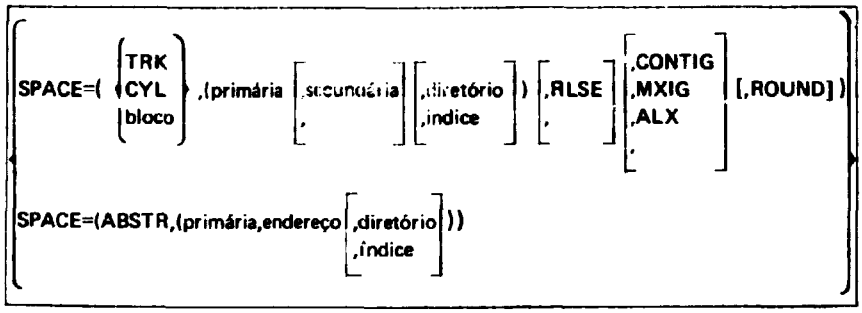
O parâmetro SPACE é utilizado para reservar espaço em dispositivo de acesso direto – DASD –

Unidade de Medida

TRK – especifica que o espaço é alocado em trilhas

CYL – especifica que o espaço é alocado em cilindros

bloco – especifica que o espaço é alocado pelo tamanho médio do bloco. (Neste caso o sistema calcula o número de trilhas).



Primária – especifica a quantidade primária de unidade de medida que deve ser alocada.

Secundária – especifica a quantidade de unidade de medida a ser alocada como incremento, toda vez que o sistema solicitar mais espaço (o incremento é alocado até 15 vezes no máximo).

Diretório – especifica o número de registros de 256 bytes a serem reservados para o diretório de um data set particionado.

Índice – especifica o número de cilindros a serem reservados para o índice de um data set organizado de forma sequencial indexada (DSORG = IS na DCB).

RLSE – especifica que o espaço alocado e não utilizado deve ser liberado.

CONTIG – especifica que o espaço alocado deve ser contíguo. Se não houver espaço contíguo o job termina.

MXIG – especifica que deve ser alocada a maior área contígua existente no volume, no mínimo igual à área solicitada.

ALX – especifica que as 5 maiores áreas contíguas diferentes devem ser alocadas sendo que todas devem ser no mínimo iguais à área solicitada.

ROUND – especifica que o espaço requisitado em blocos seja arredondado em número inteiro de cilindros.

ABSTR – especifica que o espaço é requisitado em trilhas absolutas e o endereço físico inicial é determinado.

Primária – especifica o número de trilhas a serem alocadas.

Endereço – especifica o número de trilha da primeira trilha a ser alocada.

Diretório – especifica o número de registros de 256 bytes a serem reservados para o diretório de um data set particionado.

Índice – especifica o número de cilindros a serem reservados para o índice de um data set sequencial indexado.

Exs :

SPACE=(400,(2000,50))

SPACE=(TRK,(286,8))

SPACE=(CYL,(29,1))

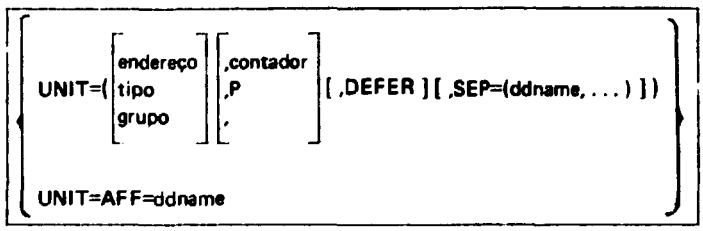
SPACE=(CYL,(21,1,3),RLSE,CONTG)

SPACE=(1024,(100,25),,,ROUND)

SPACE=(ABSTR,(50,12)

5.3.2.6 – UNIT

O parâmetro UNIT é usado para identificar a unidade desejada.



Endereço – identifica uma determinada unidade através de três números : número de canal, número da unidade de controle e número da unidade.

Ex: UNIT=182

significa: canal 1, unidade de controle 8 e unidade 2.

Tipo – identifica um determinado tipo de dispositivo permitindo ao sistema designar qualquer unidade disponível daquele tipo.

Ex: UNIT=2311

Grupo – identifica um determinado grupo de dispositivos composto de um ou mais tipos.

Exemplo:

SYSDA – discos

TAPE – fitas de 800 e 1600 bpi

TAPES – fitas de 1600 bpi

TAPE800 – fitas de 800 bpi

NOTA – bpi=bits per inch – bits por polegada

Contador – número de dispositivos designados (máximo 59 e o default é 1)

P – especifica montagem em paralelo (número de dispositivos igual ao número de volumes).

DEFER – usado para indicar que a montagem do volume deve ser feita somente no instante do OPEN

SEP – usado para indicar que os data set devem estar em dispositivos diferentes.

(ddname,...) – nome do(s) comando(s) DD no step que define(m) o data set para os quais são desejados dispositivos diferentes (até 8).

AFF – especifica que o data set utilize a mesma unidade física usada por um data set anterior definido no mesmo step.

ddname – nome de um comando DD anterior, no mesmo step, que define o data set com o qual desejamos usar AFF.

Exs:

UNIT=(DISK,,DEFER)

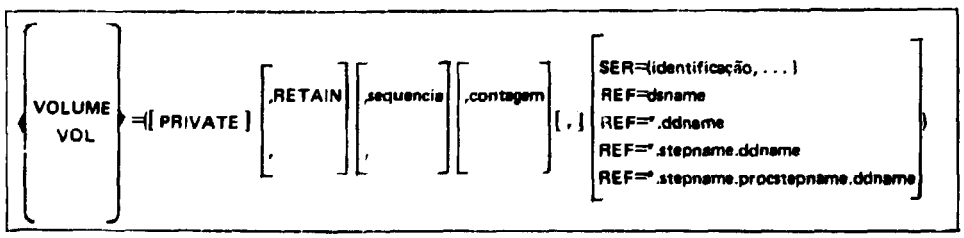
UNIT=(2311,SEP=(DD1,DD2)

UNIT=AFF=DDA

5.3.2.7 – VOLUME

O parâmetro volume é usado para descrever o volume em que o data set reside.

Só é usado para fita magnética e dispositivo de acesso direto.



PRIVATE – indica que o data set é lido ou gravado de um volume particular que será desmontado ao final do step.

RETAIN – indica que o volume deve ficar montado até outro step usá-lo ou até o fim do job (o que ocorrer antes). Só tem sentido usado com PRIVATE.

sequência – um número de 1 a 255 que especifica em qual volume deseja-se começar o processamento de um data set multivolume.

contagem – um número de 1 a 255 que especifica qual o número máximo de volumes que pode ser requisitado para um data set.

SER – especifica os números seriais dos volumes a serem utilizados.

identificação – número serial do volume.

REF – o sistema obtém informações de volume na fonte indicada neste subparâmetro.

5.3.2.8 – SYSOUT

Este parâmetro serve para dirigir o data set para o output stream.

SYSOUT=(classe[,programa] [,formulário])

classe – a classe associada ao dispositivo de saída para onde deve ser dirigido o data set (A – Z, 0 – 9)

programa – nome do programa que transfere os dados do DASD para dispositivos de saída.

formulário – número do formulário especial onde deve ser impressa ou perfurada a saída (de 1 a 4 caracteres)

Exs:

SYSOUT=4

SYSOUT=(F,,1234)

5.4 – Classificação dos Parâmetros Quanto ao Uso

Neste ítem serão apresentados os parâmetros do comando DD separados, esquematicamente, em classes de acordo com a finalidade a que se destinam.

5.4.1 – Parâmetros para Informações do Data Set

- DSNAME (DSN) – marca o nome do data set. Se omitido o data set é considerado temporário

- DISP – especifica o estado (status) corrente do data set bem como se o mesmo deve ser retido ou não após o step.

5.4.2 – Parâmetros para a Localização do Data Set

- UNIT – especifica o tipo de dispositivo a ser usado para o data set.

- VOLUME (VOL) – descreve o volume em que o data set reside. Só é usado para fita magnética e dispositivos de acesso direto.

- LABEL – descreve o rótulo (label) do volume. Usado só para fita magnética e dispositivos de acesso direto.

- SYSOUT – dirige o data set para o output stream

– * – indica que o data set está no input stream

– DATA – indica que o data set está no input stream e contém cartões de J.C.L. que devem ser tratados como dados.

5.4.3 – Parâmetros para o Tamanho do Data Set

– SPACE – requisita espaço em dispositivos de acesso direto

– SPLIT – divide o cilindro entre data sets

– SUBALLOC – especifica que o data set compartilha o espaço de acesso direto com outros data sets.

5.4.4 – Parâmetros para os Atributos de Dados

– DCB – permite completar os campos da tabela DCB na memória.

5.4.5 – Parâmetros para Processamentos Especiais

– AFF – permite a separação de canais. Usado sempre após o parâmetro SEP (desde que no mesmo step).

– DUMMY – permite executar o programa de processamento sem realizar operações de entrada e saída, alocação de espaço e disposição relativos àquele data set.

– DYNAM – usado em T.S.O. para permitir alocação dinâmica do data set a ser usado. Quando usado em esquema batch equivale ao parâmetro DUMMY.

– DDNAME – permite fazer referência a um comando DD posterior, no mesmo step, que define um data set com as mesmas características daquele ao qual o parâmetro DDNAME se refere.

– DLM – usado para indicar o fim de um grupo de dados no input stream.

– FCB – usado somente para impressoras 3211 com a finalidade de especificar informações de controle de formulário.

– OUTLIM – usado para especificar o número máximo de registros lógicos a serem incluídos num data set de saída.

– SEP – permite a separação de canais.

– TERM – indica ao sistema se os dados estão entrando ou saindo por um terminal. Só é válido para T.S.O.

– UCS – usado para descrever um conjunto de caracteres a ser usado para imprimir um data set de saída.

5.5 – DDNAMES Especiais

Existem cinco ddnames especiais que permitem utilizar vantagens particulares do sistema. Esses ddnames, JOBLIB, STEPLIB, SYSUDUMP, SYSABEND e SYSCHK, são reunidos em três categorias de acordo com a função exercida.

5.5.1 – DDNAMES para Uso de Bibliotecas Particulares

Quando o programa requisitado no cartão EXEC reside numa biblioteca particular ou temporária, é preciso fornecer ao sistema o nome desta biblioteca para que o mesmo não vá procurar o programa na biblioteca do sistema SYS1.LINKLIB e concatenadas. O nome da biblioteca é fornecido via comandos JOBLIB, STEPLIB

Incluindo o cartão JOBLIB – logo após o cartão JOB – e/ou o cartão STEPLIB logo após o cartão EXEC – cada vez que um programa for requisitado o sistema irá procurá-lo primeiro na biblioteca particular e, somente se não o encontrar lá, irá procurá-lo posteriormente na SYS1.LINKLIB.

Quando dentro de um mesmo job, for codificado um JOBLIB e um ou mais STEPLIBs, a definição do STEPLIB tem precedência sobre o JOBLIB.

5.5.2 – DDNAMES que Definem Data Sets quanto a DUMPs

Estes comandos DD definem um data set em que o dump, no caso de um término anormal, possa ser gravado.

O DD SYSABEND faz o sistema emitir um dump que inclui o núcleo do sistema e a área de memória do programa de processamento.

O DD SYSUDUMP fornece o dump da área de memória do programa de processamento.

Caso ambos os DD – SYSABEND e SYSUDUMP – forem codificados prevalece o SYSUDUMP.

Ex:

```
//SYSUDUMP DD SYSOUT=A

//SYSABEND DD SYSOUT=A
```

5.5.3 – *DDNAME que Define o Data Set quanto a Checkpoint*

Se, durante a execução de um programa, as macros CHKPT forem executadas, as entradas de checkpoint serão gravadas num data set de checkpoint.

Se, posteriormente, o job for submetido a um restart e a execução tiver que ser reiniciada num determinado checkpoint, é preciso incluir o DD SYSCHK (na nova execução). Este DD define o data set em que a entrada do checkpoint foi gravada e antecede o comando EXEC.

Ex: `//SYSCHK DD DSNAME=BIBLCHK`

5.6 – Esquemas de Utilização do Comando DD

Geralmente a maior dificuldade com que se defrontam os usuários de JCL, é a de saber quais os parâmetros que devem ser codificados e quando codificá-los, dentro de uma determinada aplicação.

Vejamos três tabelas que relacionam esquematicamente o tipo de dispositivo e organização com os parâmetros utilizados.

Tabela 1

Criação de um DATA SET Temporário

DISPOSITIVO ORGANIZAÇÃO	PARÂMETROS SEMPRE EXIGIDOS	PARÂMETROS QUE PODEM SER EXIGIDOS
UNIT RECORD	UNIT	DCB UCS
FITA	UNIT	DCB VOLUME LABEL
OUTPUT STREAM	SYSOUT	DCB UNIT SPACE
ACESSO DIRETO	UNIT SPACE	DCB VOLUME LABEL

Tabela 2

Criação de um DATA SET não Temporário

DISPOSITIVO ORGANIZAÇÃO	PARÂMETROS SEMPRE EXIGIDOS	PARÂMETROS QUE PODEM SER EXIGIDOS
FITA	UNIT DSNAME DISP	LABEL DCB VOLUME
ACESSO DIRETO (SEQUENCIAL)	UNIT DSNAME DISP SPACE	LABEL DCB VOLUME
ORGANIZAÇÃO DIRETA	UNIT DSNAME DISP SPACE DCB	LABEL VOLUME
PARTICIONADO	UNIT DSNAME DISP SPACE	LABEL VOLUME DCB
NOVO MEMBRO DO PARTICIONADO	DISP DSNAME	UNIT VOLUME
SEQUENCIAL INDEXADO	UNIT DSNAME DISP DCB SPACE	VOLUME LABEL

Tabela 3

Leitura de um DATA SET

DISPOSITIVO ORGANIZAÇÃO	PARÂMETROS SEMPRE EXIGIDOS	PARÂMETROS QUE PODEM SER EXIGIDOS
DATA SET CATALOGADO	DSNAME DISP	DCB LABEL UNIT
FITA (NÃO CATALOGADA)	DSNAME UNIT VOLUME DISP	LABEL DCB
ACESSO DIRETO (SEQUENCIAL NÃO CATALOGADO	UNIT VOLUME DSNAME DISP	LABEL
ACESSO DIRETO (ORGANIZ.DIRETA NÃO CATALOGADA)	UNIT VOLUME DSNAME DISP	LABEL
MEMBRO DE PARTICIONADO	DISP DSNAME	UNIT VOLUME
SEQUENCIAL INDEXADO	DSNAME UNIT VOLUME DCB DISP	
DATA SET PASSADO	DSNAME DISP	LABEL DCB VOLUME UNIT

6 – 'PROCEDURE' CATALOGADA

Chamamos 'procedure catalogada a todo conjunto de comandos de controle ao qual foi designado um nome e que fica alojado num data set particionado também chamado biblioteca de 'procedures'

A biblioteca de procedures do sistema é chamada de SYS1.PROCLIB e além desta, cada instalação pode ter outras bibliotecas de 'procedures'.

Quando uma 'procedure' catalogada é chamada via comando EXEC – os comandos controle da mesma são anexados ao input stream pelo Reader/Interpreter.

Neste conjunto de comandos de controle que constituem uma 'procedure' não podem estar presentes os seguintes comandos de J C L.:

JOB

DELIMITER

NULL

DD*

DD DATA

EXEC com o parâmetro PROC

A vantagem das 'procedures' catalogadas consiste principalmente em fazer com que comandos de controle usados frequentemente não tenham que ser anexados ao input stream a cada utilização.

Por exemplo; um programa de aplicação possue as etapas: Compilação, Link-edição e Execução. Estas etapas são iguais para todos os programas Para evitar de ter que anexar no input stream as três etapas, sempre que for necessário testar um programa, as mesmas foram convertidas em 'procedures' catalogadas

Então temos por exemplo as procedures:

COBUC – para compilação em COBOL

COBUCL – para compilação e link-edição em COBOL

COBCLG – para compilação, link-edição e execução em COBOL

FORTHC – para compilação em FORTRAN

FORTHCLG – para compilação, link-edição e execução em FORTRAN.

etc.

Pode acontecer também que uma certa procedure possa ser utilizada por algum usuário desde que um ou outro parâmetro ou subparâmetro sejam modificados. Para tanto basta 'chamar' esta procedure, proceder à alteração ou alterações desejadas e utilizá-la. É claro que a alteração é efetuada após a procedure estar na memória, isto significa que a procedure permanece na biblioteca na forma original.

Suponhamos por exemplo que desejemos modificar a REGION do step COB da procedure COBUC, que está com 100K, para 80K. Para isto fazemos:

```
//STPA EXEC COBUC,REGION.COB=80K.
```

Desta forma alteramos a REGION para 80K mas na biblioteca a mesma permanece com os 100K originais.

Ao processo de alteração de parâmetros ou subparâmetros de uma 'procedure' chamamos de **"override"**.

Além de fazer "override" é possível também anular ou adicionar parâmetros e subparâmetros a uma procedure.

Se desejamos obter uma listagem dos comandos de controle, quer seja daqueles que entraram no input stream, quer seja daqueles da 'procedure' é necessário codificar o parâmetro MSGLEVEL fazendo a=1.

O reconhecimento dos comandos é feito da seguinte maneira:

// identifica os comandos vindos do input stream

XX identifica os comandos da 'procedure' sem override

X/ identifica os comandos da 'procedure' com override

Existem basicamente 5 regras que regem as alterações, inclusões ou exclusões numa procedure.

1º – Os comandos DD de override no input stream devem estar na mesma ordem que os comandos correspondentes da 'procedure'.

2º – Os comandos DD de adição devem seguir os de override.

3º – O parâmetro a sofrer um 'override' deve ser inteiramente recodificado (exceto a DCB) ou deve ser codificado um parâmetro mutuamente exclusivo.

4º – Para anular um parâmetro basta codificar o parâmetro seguido do sinal igual (=) e nada mais.

5º – Para alterar um comando DD de um data set concatenado é necessário copiar os comandos DD da 'procedure' até aquele que deve ser alterado.

Vejamos um exemplo de alteração (override), inclusão e exclusão de parâmetros e subparâmetros de uma 'procedure'.

```
                          PROCEDURE : EXEMPLO

//LOOKUP        EXEC      PGM=SEARCH
//IN1           DD        DSN=A.B.C,DISP=OLD
//OUT1          DD        UNIT=2311,SPACE=(TRK,(10,2)),DISP=(,PASS)
//REDUCE        EXEC      PGM=TRUNCATE
//IN2           DD        DSN=*.LOOKUP.OUT1,DISP=(OLD,DELETE)
//WORK          DD        UNIT=TAPE
```

```
//OUT2          DD      UNIT=2311,SPACE=(TRK,(5,1)),DISP=(,PASS)
//DISPLAY       EXEC    PGM=PRINT
//IN3           DD      DSN=*.REDUCE.OUT2,DISP=(OLD,DELETE)
//OUT3          DD      SYSOUT=A
```

INPUT STREAM

```
//STEP1         EXEC    EXEMPLO
//LOOKUP.OUT1   DD      UNIT=2400,LABEL=(,NL)
//REDUCE.WORK   DD      UNIT=180
//REDUCE.XTRA   DD      UNIT=181
//DISPLAY.OUT3  DD      DSN=TEXT,UNIT=2400,DISP=(,KEEP)
```

RESULTADO:

```
//LOOKUP        EXEC    PGM=SEARCH
//IN1           DD      DSN=A.B.C,DISP=OLD
//OUT1          DD      UNIT=2400,SPACE=(TRK,(10,2)),DISP=(,PASS),LABEL=(,NL)
//REDUCE        EXEC    PGM=TRUNCATE
//IN2           DD      DSN=*.LOOKUP.OUT1,DISP=(OLD,DELETE)
//WORK          DD      UNIT=180
//OUT2          DD      UNIT=2311,SPACE=(TRK,(5,1)),DISP=(,PASS)
//XTRA          DD      UNIT=181
//DISPLAY       EXEC    PGM=PRINT
//IN3           DD      DSN=*.REDUCE.OUT2,DISP=(OLD,DELETE)
//OUT3          DD      DSN=TEXT,UNIT=2400,DISP=(,KEEP)
```

OBS.: SPACE,no cartão OUT1, será ignorado,face ao parâmetro UNIT.

7 -- ANEXOS

7.1 – DCB Subparameters (ANEXO 1)

Values by Access Method

Keyword	BISAM	QISAM	BPAM	BDAM	BSAM	QSAM
BFALN=	F or D	F or D	F or D	F or D	F or D	F or D
BFTEK=				R	R	S,E,orA
BLKSIZE=		number of bytes	number of bytes	number of bytes	number of bytes	number of bytes
BUFL=			number of bytes	number of bytes	number of bytes	number of bytes
BUFNO=	number of buffers	number of buffers	number of buffers	number of buffers	number of buffers	number of buffers
CODE=					see glossary	see glossary
CYLOFL=		number of tracks				
DEN=					0,1,2,or3 (Tape)	0,1,2,or3 (Tape)
DSORG=	IS or ISU	IS or ISU		DA orDAU	PS or PSU (Disk)	PS or PSU (Disk)
EROPT=						ACC, SKP, or ABE
HIARCHY=	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1
KEYLEN=		number of bytes	number of bytes	number of bytes	number of bytes	
LIMCT=				number of trks or blks		
LRECL=		see glossary	see glossary		see glossary	see glossary
MODE=					C or E (Rdr/Punch)	C or E (Rdr/Punch)
NCP=	number of chan'l pgms.		number of chan'l pgms.		number of chan'l pgms.	
NTM=		number of tracks				
OPTCD=		see glossary	see glossary	see glossary	see glossary	see glossary
PRTSP=					0,1,2,or3 (Printer)	0,1,2,or3 (Printer)
RECFM=		see glossary	see glossary	see glossary	see glossary	see glossary
RKP=		byte number				
STACK=					1 or 2 (Rdr/Punch)	1 or 2 (Rdr/Punch)
TRTCH=					C,E,T,orET (Tape)	C,E,T,orET (Tape)

Glossary

BFALN	F = fullword or doubleword boundary alignment of each buffer (default = D)
BFTEK	Type of buffering to be supplied by the control program (A, B, S, or E)
BLKSIZE	Maximum block size in bytes (a number).
BUFL	Length, in bytes, of each buffer to be obtained for a buffer pool (a number).
BUFNO	Number of buffers to be assigned to the data control block.
BUFRQ	Number of buffers to be read in advance from the direct access device queue. (For use with teleprocessing access methods.)
CODE	Paper tape code in which the data is punched. 1 IBM BCD perforated tape and transmission code (8 tracks) F Friden (7 tracks) H Burroughs (7 tracks) O National Cash Register (8 tracks) A ASCII (8 tracks) T Teletype (5 tracks) N No conversion
CPRI	Relative priority to be given to sending and receiving operations. (For use with teleprocessing access methods.)
CYLOFL	Number of tracks to be reserved on each cylinder to store records that overflow from tracks on that cylinder.
DEN	Tape recording density. 0 200 bits/inch (7 track only) 1 556 bits/inch (7 track only) 2 800 bits/inch 3 1600 bits/inch (9 track only)
DSORG	Organization of the data set. PS Physical sequential PSU Physical sequential unmovable PO Partitioned organization POU Partitioned organization unmovable IS Indexed sequential ISU Indexed sequential unmovable DA Direct access DAU Direct access unmovable
EROPT	Option to be executed if an error occurs. ACC Accept SKP Skip ABE Abnormal end of task
GRPDSORG	Organization of a graphic data set. (For use with the graphics access method.)
IOAREA	Maximum number of input/output macro instructions that will be issued before a WAIT macro instruction. (For use with graphics access method.)
MEASURE	The number of the storage hierarchy in which the buffer pool is to be formed (0 or 1)
MNTVL	Number of seconds of intentional delay between passes through a polling list. (For use with teleprocessing access methods.)
KEYLEN	Length, in bytes, of a key (a number).
LIMCT	Search limits, in tracks or blocks, if the extended search option (OPTCD=E) is chosen (a number).
LRECL	Actual or maximum length, in bytes, of a logical record (a number). For variable length spanned records where the logical record length exceeds 32,756, specify LRECL=X.
MODE	Mode of operation (column binary or EBCDIC) for a card reader or card punch (C or E).
NCP	Maximum number of READ or WRITE macro instructions issued before a CHECK macro instruction (number of channel programs).
NTM	Number of tracks to be contained in the master indexes of an indexed sequential data set, required when a master index (OPTCD=M) is requested. Through this master index facility, extensive serial search through a large cylinder index can be avoided.
OPTCD	Optional services to be provided by the control program. W Perform a WRITE validity check. F Feedback will be requested in READ or WRITE macro instructions in the program. E Use extended search. R For BDAM, relative block addresses are used. For OISAM, place normalization criteria information in the RORG1, RORG2, and RORG3 fields of the DCB. A Actual addresses are used. C Use chained scheduling method. M Create master indexes as required. Y Use cylinder overflow areas. I Use independent overflow area. L Delete marked records when new records are added to the data set. U Unblock data check (universal character set). Z Reduced error recovery procedure occurs for an input data set on magnetic tape when a data check is encountered (QSAM, BSAM, and EXCP only). Request only when it is know that the tape contains errors and not all records need to be processed. T Request user totaling facility. H Requests hopper empty exit for Optical Readers (BSAM). O Requests online correction for Optical Readers (QSAM). B Requests that end-of-file recognition be disregarded for tapes.
PRTSP	Line spacing on a printer (0, 1, 2, or 3).
RECFM	Characteristics of the records in a data set. These can be used in combination, as required. U Undefined length, records. V Variable length records. F Fixed-length records. B Blocked records. T Track overflow used. S With fixed-length records, standard blocks, no truncated blocks or unfilled tracks within the data set. With variable-length records, logical records spanned over more than one physical block. A ASA control character. M Machine code control character.
REPOS	Repositioning for tape devices. (For use only with EXCP.) Y Repositioning. N No repositioning.
RKP	Relative position of the first byte of the record key within each logical record (a number).
SOWA	Size, in bytes, of the user-provided work area. (For use with teleprocessing access methods.)
STACK	Stacker bin on a card reader or card punch (1 or 2).
TRTCH	Tape recording technique for seven-track tape. C Data conversion feature. E Even parity. ET Even parity, and BCD to EBCDIC translation is required when reading; EBCDIC to BCD when writing. T Odd parity, and BCD to EBCDIC translation is required when reading; EBCDIC to BCD when writing.

Note The DCB subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive.

ANEXO 2

7.2 – Regras de Codificação do Cartão JOB Adotadas no CPD do IEA

Para utilização do Sistema IBM/370 do I.E.A., o cartão JOB deverá ter o formato abaixo descrito, e a sua não observância implicará em cancelamento do JOB.

O comando JOB deverá ter no mínimo 2 cartões, e o primeiro terá o seguinte formato:

COL 1 – 2 //

COL 3 – 4 Alfabéticos, especificam o código de identificação do usuário. Este código é determinado pelo C.P.D.
Todo usuário deverá colocar o seu em qualquer Job por ele submetido.

COL 5 – 10 Qualquer caráter, alfanumérico, diferente de branco utilizado pelo usuário para identificar o seu Job.

COL 11 – Branco

COL 12 – 14 JOB

COL 15 – Branco

COL 16 – (

COL 17 – 19 pano-Programmer's Accounting Number – Este campo deve ser codificado obrigatoriamente e corresponde ao número da pesquisa para os usuários pertencentes ao I.E.A. Os usuários não funcionários do I.E.A. deverão preencher com um valor numérico, que lhes será atribuído pelo C.P.D.

COL 20 – ,

COL 21 – 23 room – Programmer's room number – Este campo deve ser numérico especificando o código do usuário na instituição atribuído pelo C.P.D. – Também este campo deve ser codificado.

COL 24 – ,

1º Cartão de Continuação

O primeiro cartão de continuação consta de uma parte obrigatória e os restantes sub-campos são opcionais e separados por vírgulas no campo total de "Accounting Information"; a sua codificação será feita conforme as regras seguintes:

COL 1 – 2 //

COL 3 – Branco

COL 4 – 7 time – tempo estimado de execução – Este tempo não é o tempo estimado de utilização de CPU, porém o tempo total de execução estimado, expresso em minutos – Deve ser codificado obrigatoriamente, e com 4 dígitos.

COL 8 e seguintes), se não for codificado nenhum dos sub-campos opcionais seguintes. Caso contrário, codificar obedecendo às seguintes normas:

lines : número aproximado de linhas impressas para todo o JOB, expresso em milhares de linhas. É recomendável a sua codificação e deverá ser obrigatòriamente com 4 dígitos. Se não codificado será assumido o 'default' '0002'. Se o número real de linhas impressas exceder demasiadamente o estimado, o Job poderá ser cancelado pelo operador.

Ex.: Para indicar um número aproximado de 5.000 linhas impressas, codificar "0005".

Cards – Número aproximado de cartões perfurados; consiste de até 4 dígitos e é expresso em unidade de cartões perfurados.

Ex : Para indicar 500 cartões perfurados codificar "500". OBS. O "default" é "100"

Forms – Número do formulário especial para o Job inteiro. Se codificado, deverá ter obrigatoriamente 4 dígitos numéricos especificado no "Slip" de submissão.

Copies – Quantidade de cópias desejadas para o Job. Pode ser codificado com até 2 dígitos

Ex : Para obter 2 copias do Job codificar "2".

Log – HASP System Log Option – Contém informações gerais sobre o Job, tais como tempo de início e fim do Job, mensagens de erros, etc..
Codificar "N" se o usuário não quiser esta opção, que é porém recomendável. Se se codificar qualquer outro caráter ou se omitir este sub-campo, o Sistema emitirá essas informações.

Linecnt – Número de linhas a serem impressas por página. Consta de até dois dígitos, se codificado

Ex : "40" se for pedida a impressão de 40 linhas por página – "0" para não haver salto automático de página.

A partir daqui, a ordem dos parâmetros citados abaixo deverá ser seguida rigorosamente e sua codificação é obrigatória.

Programmer's name – Nome do programador – codificado entre aspas simples ocupando 10 colunas; portanto o campo total deverá ocupar 12 posições.

TIME – Parâmetro de JCL, especificando o tempo de CPU em minutos, com 4 posições.
Observar que time > TIME. Não codificar TIME = 1440

CLASS – Classe correspondente ao Job segundo a Tabela I, em função da maior Região exigida pelo JOB.
Este é o último parâmetro que deverá ser codificado neste primeiro cartão de continuação. Qualquer parâmetro de JCL, válido para o cartão JOB deverá ser codificado num segundo cartão de continuação.

Tabela I

MEMÓRIA	CLASSE
0 – 60 K	A
61 – 120 K	B
121 – 180 K	C
181 – 240 K	D
241 – 320 K	E
321 – 700 K	N
> 700 K	K

OBS 1: Para Jobs de classe K é obrigatório colocar no Slip de Submissão "CLASSE = K"

NOTA – Este ANEXO 2 foi extraído do Boletim Informativo Nº 1 do CPD do IEA

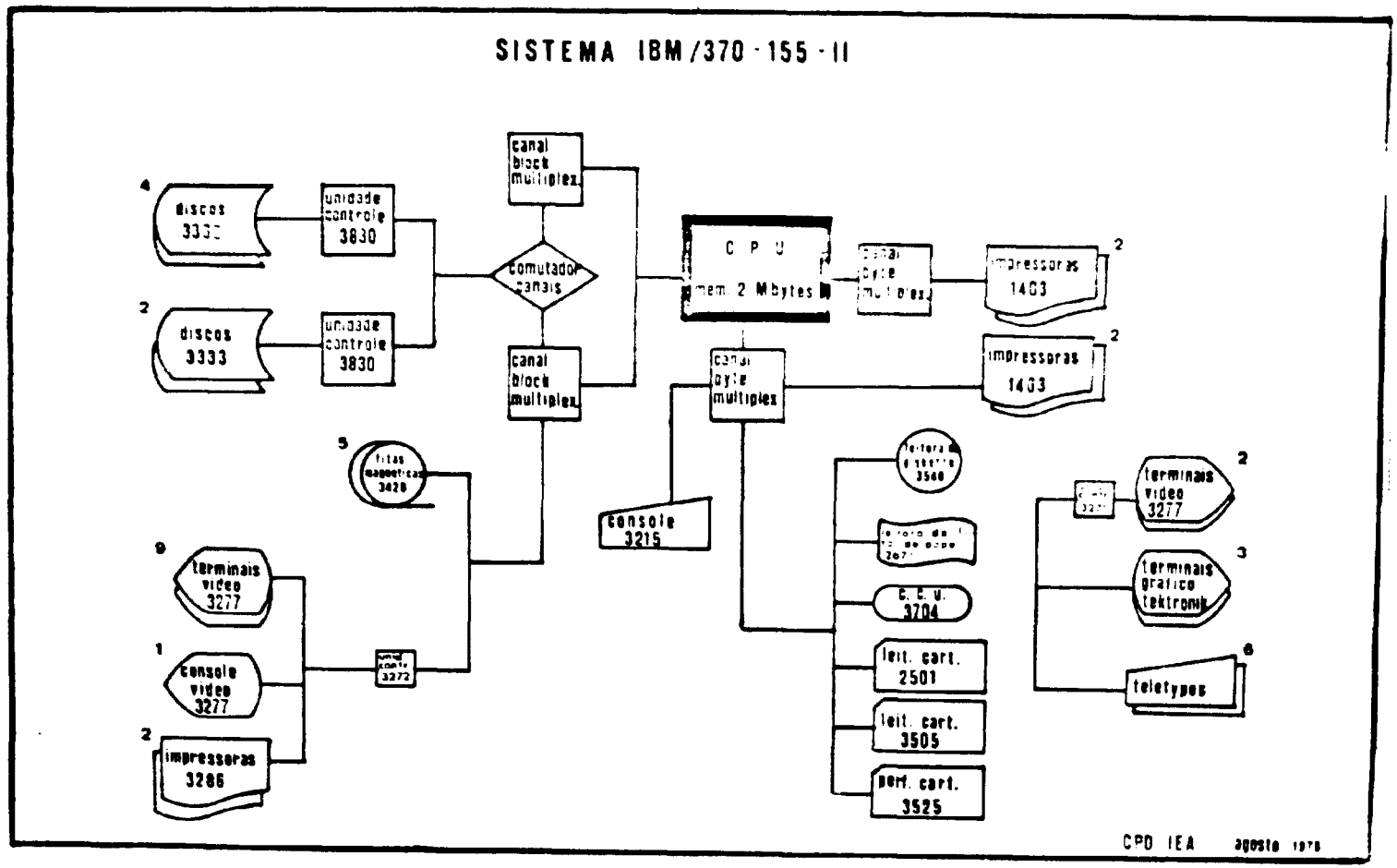
ANEXO 3

7 3 – Configuração Atual do CPD do IEA

ESPECIE	NOME	MODELO	NUMERO DA MAQUINA	CAPACIDADE OU VELOCIDADE
C P U	C P U	—	3155	115 Nano Segundos
MEMORIA	MEMORIA 1	003	3360	512 K Bytes
	MEMORIA 2	003	3360	512 K Bytes
	MEMORIA 3	003	3360	512 K Bytes
	MEMORIA 4	003	3360	512 K Bytes
CONSOLE	CONSOLE	001	3215	85 Caract/seg. 132 Caract/linha
UNIDADES DE FITA	UN FITA 280	005	3420	800 ou 1600 BPI
	UN FITA 281	005	3420	1600 BPI
	UN FITA 282	005	3420	1600 BPI
	UN FITA 283	005	3420	1600 BPI
	UN FITA 284	005	3420	1600 BPI
	UN CONTROLE	001	3803	———
UNIDADES DE DISCOS	UN DISCO A	002	3330	———
	UN DISCO B	002	3330	———
	UN DISCO C	001	3330	———
	UN DISCO D	001	3330	———
	UN DISCO E	002	3330	———
	UN DISCO F	002	3330	———
	UN DISCO G	002	3330	———
	UN DISCO H	002	3330	———
	UN CONTROLE	001	3830	———
	UN DISCO A	002	3333	———
	UN DISCO B	002	3333	———
	UN DISCO E	002	3333	———
	UN DISCO F	002	3333	———
	UN CONTROLE	002	3830	———
LEITORAS	LEIT CARTÕES	B02	2501	1000 Caract/minuto
	LEIT CARTÕES	B00	3505	1100 Caract/minuto
	DISQUETES	—	3540	———
	FITA PAPEL	—	2671	———
	UN CONTROLE	—	2822	———
PERFURADORA	PERF CARTÕES	—	3525	600 Cart/minuto
IMPRESSORAS	IMPRESSORA 00F	N01	1403	1100 Linhas/minuto
	UN CONTROLE	002	2821	———
	IMPRESSORA 00E	N01	1403	1100 Linhas/minuto
	UN CONTROLE	002	2821	———

	IMPRESSORA 40E	N01	1403	1100 Linhas/minuto
	UN. CONTROLE	002	2821	————
	IMPRESSORA 40F	N01	1403	1100 Linhas/minuto
	UN. CONTROLE	002	2821	————
	IMPRESSORA	002	3286	85 Caract/seg.
	IMPRESSORA	002	3286	85 Carct/seg.
UNIDADE	UN. CONTR.(L)	002	3272	16 Terminais
CONTROLE	UN. CONTR.(R)	002	3271	16 Terminais
TERMINAIS	UN. CONTROLE	A03	3704	8 Linhas
VIDEOS	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
	VIDEO	002	3277	1920 Caracteres
MODEM	MODEM	1	3872	————
	MODEM	1	3872	————
TETRONIKS	VIDEO	957-A	—	2800 Caract. 35X80
	VIDEO	957-A	—	2800 Caract. 35X80
	VIDEO	957-A	—	2800 Caract. 35X80

37 DISK PACK, 3336, COM 100 MILHÕES DE BYTES CADA
4 CADEIAS DE TIPOS HN, 1416
1 CADEIA DE TIPOS TN, 1416
1 CADEIA DE TIPOS SN, 1416


SISTEMA IBM/370 - 155 - II
4
discos
3330
unidade controle
3830
2
discos
3333
unidade controle
3830
comutador canais
canal block multiplex
canal block multiplex
C P U
mem. 2 Mbytes
canal byte multiplex
2
impressoras
1403
canal byte multiplex
2
impressoras
1403
5
fitas magneticas
3420
console
3215
leitora de cartões
3540
C. E. V.
3704
leit. cart.
2501
leit. cart.
3505
perf. cart.
3525
2
terminais video
3277
3
terminais grafico tektronik
6
teletypes
9
terminais video
3277
1
console video
3277
2
impressoras
3286
CPD IEA agosto 1978

ABSTRACT

Job Control Language is a language that permits a communication between the system and the user such that the problems may be solved in a computer level.

Like all the languages, J.C.L. has its structure and rules. The knowledge of the language permits an efficient and optimized use of the machine.

AGRADECIMENTOS

- Ao Engº CIBAR CÁCERES AGUILERA, Coordenador Geral do C.P.D. do I.E.A. pelo apoio e pelo incentivo que sempre prestou no decorrer da preparação do presente trabalho.

- À Sra. LÚCIA FARIA SILVA, por ter sugerido e apoiado a publicação do trabalho.

- À Srta. ELENICE MAZZILLI, pela revisão final do trabalho.

- À Srta. ODETTE GUEDES, por ter oferecido condições para a realização do trabalho bem como pelas sugestões a respeito do conteúdo do mesmo.

- **Agradeço de maneira especial** à Srta. VALENTINA MONTIEL FERNANDEZ, com a qual foram discutidos muitos dos ítens do trabalho e, principalmente, pela valiosa colaboração que prestou ao ler e sugerir alterações quanto à forma e conteúdo do mesmo.

- A todos aqueles que, de uma forma ou de outra, contribuiram para a realização deste trabalho, principalmente ao grupo de alunos que assistiram ao curso do qual esta publicação é outro resultado.

REFERÊNCIAS BIBLIOGRÁFICAS

1. FRANCO, R. S. *Sistema operacional OS–IBM/360: conceitos e controle de sistema.* São Paulo, PRODESP, Set. 1974.

2. INTERNATIONAL BUSINESS MACHINES CORPORATION, Poughkeepsie, N. Y. *IBM system/360 operating system: job control language reference – OS release 21.7.* Poughkeepsie, N. Y. Apr. 1973. (File n. S360-36, Order n. GC28-6704-3).

3. _____, *IBM system/360 operating system: job control language user's guide.* Poughkeepsie, N. Y., June 1971. (File n. S360-36, Order n. GC28-6703-2).