

Introdução ao JCL

I n d i c e

| | |
|---|----|
| 1 JCL | 1 |
| JOB | 2 |
| Jobname | 4 |
| 2.2 Account | 4 |
| 2.3 Region | 4 |
| 2.4 Class | 4 |
| 2.5 Msgclass | 4 |
| 2.6 Msglevel | 5 |
| 2.7 Time | 5 |
| 2.8 Typrun | 5 |
| 2.9 Joblib | 6 |
| EXEC | 6 |
| Stepname | 6 |
| 3.2 Pgm | 6 |
| 3.3 Proc | 6 |
| 3.4 Cond | 6 |
| 3.5 Region | 7 |
| 3.6 Time | 7 |
| 3.7 Steplib | 7 |
| DD | 8 |
| Ddname | 8 |
| 4.2 Dsname (DSN) | 8 |
| 4.3 Disp | 8 |
| 4.4 Unit | 9 |
| 4.5 Space | 9 |
| 4.6 Dcb | 10 |
| 4.7 * Input Stream | 10 |
| 4.8 Dummy | 10 |
| 4.9 Sysout | 10 |
| 4.10 Sysudump / Sysabend / Sysabout | 11 |
| 5 Essencial | 12 |
| 6 JCL comentado | 12 |

1 JCL

São as iniciais de Job Control Language (linguagem de controle de serviço).

Gerado através de instruções para dirigir a execução de programas.

É no JCL que informamos:

- Nome do programa / procedure que será executado;
- Nome dos arquivos que serão tratados pelo programa;
- Tamanho dos registros;
- Condição para a continuação do processamento; e; Prioridade de sua execução.

O JCL se caracteriza pela codificação de duas barras (//) nas colunas 1 e 2, sendo basicamente composto por três comandos (instruções), também chamados de cartões de controle:

- **JOB**;
- **EXEC**; e; **DD**

O JCL é interpretado pelo JES (Job Entry Subsystem).

Todas as informações codificadas em um JCL, constituem um serviço, ou seja, um **JOB**.

2 JOB

É através dos JOBS que se executam os programas e utilitários BATCH. Eles podem ser executados manualmente (através do comando SUB) ou automaticamente (através de um sistema apropriado - OPC).

Sintaxe:

```
//Jobname JOB parâmetros
  ↓      ↓      ↓
Nome do Fixo Account, Region, Class,
JOB Msgclass, Msglevel, Time,
Typrun
```

2.1 Jobname

Deve começar na coluna 3 e possuir de 1 a 8 caracteres.

Exemplo:

```
//JFUTURE1 JOB
```

2.2 Account

Parâmetro que pode ser utilizado para indicar de qual departamento será cobrado o tempo de processamento gasto. Deve ser informado entre apóstrofes.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001' Explicação:
```

O job JCTB0001 será cobrado do departamento 91100 (contabilidade geral), pertencente ao sistema CTB (contabilidade), ao ser executado, chamando como primeiro programa CTB0001.

2.3 Region

Usado para alocar espaço em memória (equivale para todo os passos (STEPS) do JOB).

Seu valor default é 512 Kbytes.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K
```

2.4 Class

Usado para especificar a classe em que o JOB será executado.

Essas classes são disponibilizadas de acordo com a necessidade da execução do JOB.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K,CLASS=Z
```

2.5 Msgclass

Usado para especificar a classe de saída das mensagens do sistema. Essas classes são disponibilizadas de acordo com a necessidade da execução do JOB.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K,CLASS=Z ,  
//          MSGCLASS=Y
```

2.6 Msglevel

Usado para estabelecer o nível de detalhe das informações sobre o JOB que o sistema listará.

Sintaxe:

MSGLEVEL=(A,B)

Onde A = 0, 1 ou 2 (informações do JCL)
 B = 0 ou 1 (informações de alocações)

O default é MSGLEVEL(1,1), pode ser utilizado também da seguinte maneira: MSGLEVEL=0.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K,CLASS=Z ,  
//          MSGCLASS=Y,MSGLEVEL=(1,1)
```

2.7 Time

Usado para estabelecer um tempo limite de CPU, com minutos podendo ser variado entre 1 e 1440 (24 horas) e segundos de 1 a 59.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K,CLASS=Z ,  
//          MSGCLASS=Y,MSGLEVEL=(1,1),TIME=(0,45)
```

2.8 Typrun

Usado para solicitar um tratamento especial ao JOB.

Pode ser HOLD (reter o job na fila de entrada aguardando um comando do operados), ou SCAN (detecta erros de sintaxe no job e inibe a execução do mesmo).

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K,CLASS=Z ,  
//          MSGCLASS=Y,MSGLEVEL=(1,1),TIME=(0,45),TYPRUN=HOLD
```

2.9 Joblib

É o parâmetro utilizado para definir a bibliotecas (library) a serem usadas na execução do JCL.

Deve ser descrita, imediatamente após a descrição do parâmetro jobname.

Exemplo:

```
//JCTB0001 JOB '91100,CG,CTB,CTB00001',REGION=1024K,CLASS=Z,
//          MSGCLASS=Y,MSGLEVEL=(1,1),TIME=(0,45),TYPRUN=HOLD
//JOB LIB DD DSN=FUTURE.LIB.LMODUL
```

3 EXEC

É usado para executar programas / utilitários. São conhecidos como os passos dentro de um Job.

Pode ter até 255 cartões EXECs por JOB.

Sintaxe:

```
//Stepname EXEC parâmetros
   ↓      ↓      ↓
Nome do Fixo PGM, PROC, PARM,
STEP COND, REGION, TIME
```

3.1 Stepname

Utilizado para definir o nome do step, devendo começar na coluna 3 e possuir de 1 a 8 caracteres.

Exemplo:

```
//STEP1 EXEC
```

3.2 Pgm

Utilizado para definir o nome do programa a ser executado.

Exemplo:

```
//STEP1 EXEC PGM=FUTU5431
```

3.3 Proc

Abreviatura de PROCNAME, é utilizada para denominar o procedimento a ser utilizado

Exemplo:

```
//STEP1 EXEC PROC=FUTU0000
```

3.4 Cond

Utilizado para condicionar a execução de um determinado Step.

Sintaxe:

COND= ((CODIGO, OPERADOR, STEPNAME / PROCNAME) , EVEN / ONLY) Onde:

Código Código de retorno (RC) do Step a ser testado Operador

GT (maior que), GE (maior ou igual),

LT (menor que), LE (menor ou igual),
 EQ (igual), NE (diferente)
 Stepname Nome do Step a ser testado;
 Procname Nome da Proc a ser testada;
 Even Executa o Step, mesmo que os anteriores tenham
 terminado de forma anormal (abend)
 Only Executa o Step, somente se um dos passos anteriores
 Terminou de forma anormal (abend) **Exemplo:**

```
//STEP3 EXEC PGM=FUTU0010
// COND=(4,GT,STEP2)
```

Explicação:

A lógica da interpretação do COND é para que o STEP **não seja executado**.

Com isso, podemos interpretar a execução do STEP3 das seguintes maneiras:

- O STEP3 não será executado se o código de retorno do STEP2 for maior que 4; ou;
- O STEP3 somente será executado se o código de retorno do STEP4 **não for maior** que 4.

3.5 Region

Idêntica à parametrização do JOB, porém, se parametrizada no Job, anula a utilização dentro do Step.

3.6 Time

Idêntica à parametrização do JOB, porém, a utilização na parametrização do Job, não anula o time para o Step.

3.7 Steplib

É o parâmetro utilizado para definir as bibliotecas (library) a serem usadas na execução do STEP.

Deve ser descrita, imediatamente após a descrição do parâmetro Stepname.

Exemplo:

```
//STEP3 EXEC PGM=FUTU0010
// COND=(4,GT,STEP2)
//STEPLIB DD DSN=FUTURE.LIB.LMODUL
```

4 DD

É usado para definir as características dos arquivos a serem tratados no STEP.

Pode ter até 1635 cartões DDs por EXEC.

Sintaxe:

```
//ddname DD parâmetros
```



```
Nome do Fixo DDNAME, DSNAME,DISP, arquivo
UNIT, SPACE, DCB,*,
```

DUMMY, SYSOUT, SYSUDUMP

4.1 Ddname

Utilizado para identificar o arquivo descrito no comando select do COBOL, também chamado de IFN (Internal File Name).

Exemplo:

```
//ARQUIVO1 DD parâmetros
```

4.2 Dsname (DSN)

É o parâmetro que define o nome do arquivo catalogado, também chamado de EFN (External File Name), podendo ser temporários ou não temporários (serão deletados ao término do JOB) Pode ter até 44 caracteres.

Se omitido o sistema atribui um nome aleatório, tornando-o temporário (será deletado ao término do job).

Exemplos:

```
//ARQUIVO1 DD DSN=FUTURE.ARQUIVOS.ARQUIVO1  
//ARQUIVO2 DD DSN=* .STEP1.ARQENTRA  
//ARQUIVO3 DD DSN=* .PROCSTEP1.ARQSAIDA
```

4.3 Disp

Indica o status (disposição do arquivo).

Seu formato é DISP=(ESTADO ATUAL, FIM NORMAL, FIM ANORMAL). Onde:

ESTADO ATUAL pode ser:

- NEW = Novo (será gravado);
- OLD = Já existe (modo exclusivo);
- SHR = Já existe (modo compartilhado); e;
- MOD = Tratamento para acrescentar registros (modo exclusivo)

FIM NORMAL pode ser:

- CATLG = Catalogar um arquivo permanente;
- PASS = Passar o arquivo para os passos subseqüentes;
- KEEP = Manter os arquivos não temporários; □ UNCATLG = Retira um arquivo do catálogo; e; □ DELETE = Exclui o arquivo do disco.

- **FIM ANORMAL** é idêntico ao FIM NORMAL, com exceção do PASS.

Caso FIM ANORMAL não seja codificado, valerá a disposição de FIM NORMAL, ou seja, DISP=(OLD,KEEP) ⇔ DISP=(OLD,KEEP,KEEP).

FIM ANORMAL para arquivos temporários é sempre DELETE, independente do que foi codificado..

Exemplos:

```
//ARQUIVO1 DD DSN=FUTURE.ARQUIVOS.ARQUIVO1,  
//          DISP=(,PASS)  
//ARQUIVO3 DD DSN=FUTURE.ARQUIVO.ARQUIVO3,  
//          DISP=(OLD,PASS)  
//ARQUIVO4 DD DSN=FUTURE.ARQUIVO.ARQUIVO3,  
//          DISP=(,CATLG,DELETEP)
```

4.4 Unit

Especifica o tipo do periférico onde o arquivo está gravado ou será gravado.

Exemplo:

```
//ARQUIVO4 DD DSN=FUTURE.ARQUIVO.ARQUIVO3,  
//          DISP=(,CATLG,DELETEP),  
//          UNIT=PR39D7
```

4.5 Space

Especifica o espaço que o sistema deve alocar para o arquivo que está sendo gravado em disco.

Seu formato é:

```
SPACE=(Tamanho bloco, (PRIM, SEC, DIR), RLSE, CONTIG, ROUND)  
          TRK                      MXIG  
          CYL                      ALX Onde:
```

Tamanho bloco = Tamanho do bloco (em bytes);

TRK = Quantidade em bytes de uma trilha;

CYL = Quantidade em bytes de um cilindro;

PRIM = Quantidade primária de alocação;

SEC = Quantidade secundária de alocação;

DIR = Quantidade em unidades de blocos de 256 bytes;

RLSE = Liberar o espaço alocado e não utilizado;

CONTIG = Faz o espaço primário ser alocado apenas em trilhas ou cilindros contíguos;

MXIG = O espaço alocado deve ser a maior área contígua;

ALX = Aloca a área primária criando uma lista das 5 maiores áreas contíguas livres no disco

ROUND = Arredonda para número inteiro de cilindros, e aloca em cilindros, quando o espaço solicitado é em blocos

A alocação do espaço funciona da seguinte maneira:

Cálculo para obter tamanho em trilhas

- **(Quantidade re registros x tamanho do registro) / 32700)**

Exemplo:

```
//ARQUIVO4 DD DSN=FUTURE.ARQUIVO.ARQUIVO3,  
//          DISP=(,CATLG,DELETEP),  
//          UNIT=PR39D7,  
//          SPACE=(TRK, (2, 1), RLSE)
```

4.6 Dcb

Especifica as demais características físicas do arquivo.

Seus subparâmetros são:

- **BLKSIZE** = Define o tamanho da blocagem (quantidade * tamanho do registro);
- **LRECL** = Define o tamanho em bytes do registro. Para arquivos de forma variável, o LRECL deve ser o tamanho do registro mais 4 (quatro) bytes;
- **RECFM** = Define o formato do arquivo, podendo ser: - **F** = para registros de tamanho fixo;

- **V** = para registros de tamanhos variáveis;
 - **B** = para registros bloqueados;
 - **A** = para registros cuja 1ª posição contém o caractere asa para impressão
 - **M** = para registros cuja 1ª posição contém o caractere asa para impressão; e;
 - **U** = para registros de tamanhos indefinidos.
- **DSORG** = Define a organização do arquivo, podendo ser:
 - **DA** = Acesso direto (disco);
 - **PO** = Particionado (disco); e;
 - **PS** = Físico seqüencial (fita, cartucho etc)

Exemplo:

```
//ARQUIVO4 DD DSN=FUTURE.ARQUIVO.ARQUIVO3,
//          DISP=(,CATLG,DELETEP),
// UNIT=PR39D7,
//          SPACE=(TRK,(2,1),RLSE)
//          DCB=(LRECL=80,RECFM=FB,DSORG=PO)
```

4.7 * Input Stream

Utilizado para passar dados ao programa, através de um arquivo, via JCL; **Exemplo:**

```
20061231 //ARQUIVO4 DD *
//SYSIN DD *
(Fonte COBOL)
```

4.8 Dummy

Utilizado para receber o arquivo em estado “nulo”; **Exemplo:**

```
//ARQUIVO4 DD DUMMY;
//ARQUIVO5 DD DUMMY, SYSOUT=*, DCB=(LRECL=133)
```

4.9 Sysout

Utilizado para oferecer uma forma conveniente de direcionar as saídas para a impressora.

Exemplo:

```
//RELATO DD SYSOUT=Z
//IMPRESS DD SYSOUT=*
```

4.10 Sysudump / Sysabend / Sysabout

Estes três DDNAMEs são utilizados pelo produto ABEND-AID para imprimir um DUMP formatado quando de um término anormal do STEP; **Exemplo:**

```
//SYSUDUMP DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSABOUT DD SYSOUT=*
```

5 Essencial

| Cartão | Parâmetros |
|---------------|-------------------|
| JOB | JOBNAME |
| | ACCOUNT |
| | MSGCLASS |
| | CLASS |
| EXEC | STEPNAME |
| | PGM ou PROC |
| | PARM |
| | COND |
| DD | DDNAME |
| | DSNAME |
| | DISP |
| | UNIT |
| | SPACE |
| | DCB |
| | SYSOUT |
| | SYSUDUMP |

6 Veja um JOB de exemplo, depois iremos detalhar, passo à passo.

```
//JOB00001 JOB 'FUTU,4250,FU,PROGRA01',CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1)
//STEP1 EXEC PGM=PROGRA01,
// PARM='20020831'
//ARQENT DD DSN=AD.DC40.BRPG.UPDT.ARQENT,DISP=SHR
//ARQSAI DD DSN=AD.DC40.BRPG.UPDT.ARQSAI,DISP(,CATLG),UNIT=DISCO,
// DCB=(BLKSIZE=10000,LRECL=100,RECFM=FB),
// SPACE=(TRK,(10,2),RLSE)
//ARQRELAT DD SYSOUT=*
//*
//JOB00002 JOB 'FUTU,4250,FU,PROGRA02',CLASS=B,MSGCLASS=Y,MSGLEVEL=(0,0)
//STEP1 EXEC PGM=PROGRA02
//DDENT DD DSN=AD.DC40.BRPG.UPDT.DDENT,
// DISP=OLD
//DDSAI DD DSN=AD.DC40.BRPG.UPDT.DDSAID,
// DISP=(NEW,CATDLG,DELETE),
// UNIT=DISCO,
// SPACE=(CYL,(000001,000002)),
// DCB=(LRECL=80,RECFM=FB)
//RELATO DD SYSOUT=Z
//*
//JOB00003 JOB 'FUTU,4250,FU,PROGRA03',CLASS=B,MSGCLASS=Z
//STEP1 EXEC PGM=PROGRA03
//CADASTRO DD DSN=AD.DC40.BRPG.UPDAT.CADASTRO,
// DISP=OLD
//SAIDA DD DSN=AD.DC40.TEMPOR,
// DISP=(,PASS),
// UNIT=DISCO,
// SPACE=(480,(1,2),RLSE),
// DCB=(LRECL=160,RECFM=FB)
//*
//STEP2 EXEC PGM=PROGRA04,COND=(4,LT,STEP1)
//ARQENT DD DSN=AD.DC40.STEP1.SAIDA,
// DISP=SHR
//CADAIS DD DSN=AD.DC40.BRPG.UPDT.CADAIS,
// DISP=(,CATLG,DELETE),
// UNIT=DISCO,
// SPACE=(TRK,(1,2),RLSE),
// DCB=(LRECL=160,RECFM=FB)
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

```
//JOB00001 JOB 'FUTU,4250,FU,PROGRA01',CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1)
JOB00001 = nome do JCL (JOB)
JOB = palavra chave; deve ser o primeiro comando
(cartão) de um JCL
'FUTU,4250,FU,PROGRA01' = account (p/ contabilização); codificado entre
apóstrofes, sendo:
```

- 4 posições para centro de custo = 4 primeiras posições do nome do programa;
- 4 posições para o código do depto proprietário do centro de custo;
- fixo FU;
- 8 posições para o nome do programa; separar os dados acima por vírgulas; CLASS=A = nome da fila de execução; no ambiente Future, os valores permitidos são "A" e "B"; MSGCLASS=Z = fila de mensagens do sistema; no ambiente Future, o valor permitido para Terceiros é "Z".

MSGLEVEL=(1,1) = Nível das mensagens do sistema que se deseja na saída (visualizar ou imprimir). Os valores válidos são:

- (0,0) = exibe só o comando JOB
- (0,1) = exibe o comando JOB + alocações - (1,0) = exibe todos comandos de JCL (codificados e expandidos)
- (1,1) = exibe todos comandos de JCL (codificados e expandidos)+ alocações
- (2,0) = exibe somente os comandos JCL codificados
- (2,1) = exibe somente os comandos JCL codificados e as alocações

```
//STEP1 EXEC PGM=PROGRA01,
// PARM='20020831'
```

STEP1 = nome do step (passo); cada EXEC é um step (passo) dentro do JCL.
 EXEC = palavra chave; indica qual programa ou procedure será executado.
 PGM = indica que um programa será executado (p/ procedure é PROC).
 PROGRA01 = nome do programa a ser executado.
 PARM = parâmetro a ser enviado ao programa quando necessário. '20020831' = parâmetro enviado ao programa. Deve estar entre apóstrofes e no formato esperado pelo programa.

```
//ARQENT DD DSN=AD.DC40.BRPG.UPDT.ARQENT,DISP=SHR
```

ARQENT = nome do comando DD (Data Definition). Deve ser igual ao definido dentro do programa;
 DD = palavra chave; indica a definição dos dados.
 DSN = nome do arquivo catalogado. Neste caso é AD.DC40.BRPG.UPDT.ARQENT.
 DISP = indica a disposição do arquivo (status).
 - SHR = arquivo já existe, acesso compartilhado

```
//ARQSAI DD DSN=AD.DC40.BRPG.UPDT.ARQSAI,DISP(,CATLG),UNIT=DISCO,
// DCB=(BLKSIZE=10000,LRECL=100,RECFM=FB),
// SPACE=(TRK,(10,2),RLSE)
```

ARQSAI = nome do comando DD (Data Definition). Deve ser igual ao definido dentro do programa;
 DD = palavra chave; indica a definição dos dados.
 DSN = nome do arquivo catalogado. Neste caso é AD.DC40.BRPG.UPDT.ARQSAI.
 DISP = indica a disposição do arquivo (status). Neste caso, indique que o arquivo é novo e será catalogado se o step terminar ok.
 UNIT = indica que o arquivo será gravado em DISCO.
 SPACE = indica o espaço que deverá ser alocado para o arquivo. Neste caso, o sistema irá alocar até ((2*15)+10) trilhas;
 TRK = trilhas;
 RLSE = libera o espaço não utilizado.

DCB = contém informações a respeito dos registros do arquivo:
 - BLKSIZE = tamanho do bloco em bytes; deve ser múltiplo do LRECL;
 - LRECL = tamanho do registro em bytes;
 - RECFM = formato do registro. Neste caso indica que o registro é fixo bloqueado.

```
//ARQRELAT DD SYSOUT=*
```

```

//*
ARQRELAT = nome do comando DD (Data Definition). Deve ser igual ao definido
dentro do programa; DD = palavra chave; indica a definição dos dados.
SYSOUT = indica que o arquivo será direcionado para impressora.
* - indica que o relatório será direcionado para a fila do parâ-
metro MSGCLASS.

*****
*****
//JOB00002 JOB 'FUTU,4250,FU,PROGRA02',CLASS=B,MSGCLASS=Y,MSGLEVEL=(0,0)
JOB00002 = nome do JCL (JOB)
JOB = palavra chave; deve ser o primeiro comando
(cartão) de um JCL
'FUTU,4250,FU,PROGRA02' = account (p/ contabilização); codificado entre
apóstrofes, sendo:
- 4 posições para centro de custo = 4 primeiras
posições do nome do programa;
- 4 posições para o código do depto proprietário do
centro de custo;
- fixo FU;
- 8 posições para o nome do programa; separar os
dados acima por vírgulas; CLASS=B = nome da fila de
execução; no ambiente Future, os valores
permitidos são "A" e "B"; MSGCLASS=Y = fila de mensagens do
sistema; no ambiente Future, o valor permitido
para Terceiros é "Z".
MSGLEVEL=(0,0) = Nível das mensagens do sistema que se deseja na
saída (visualizar ou imprimir). Os valores válidos
são:
- (0,0) = exibe só o comando JOB
- (0,1) = exibe o comando JOB + alocações - (1,0) =
exibe todos comandos de JCL (codificados
e expandidos)
- (1,1) = exibe todos comandos de JCL (codificados
e expandidos)+ alocações
- (2,0) = exibe somente os comandos JCL codificados
- (2,1) = exibe somente os comandos JCL codificados
e as alocações
//STEP1 EXEC PGM=PROGRA02
STEP1 = nome do step (passo); cada EXEC é um step (passo) dentro do JCL.
EXEC = palavra chave; indica qual programa ou procedure será executado.
PGM = indica que um programa será executado (p/ procedure é PROC).
PROGRA02 = nome do programa a ser executado.

//DDENT DD DSN=AD.DC40.BRPG.UPDT.DDENT,
// DISP=OLD
DDENT = nome do comando DD (Data Definition). Deve ser igual ao definido
dentro do programa;
DD = palavra chave; indica a definição dos dados.
DSN = nome do arquivo catalogado. Neste caso é AD.DC40.BRPG.UPDT.DDENT.
DISP = indica a disposição do arquivo (status).
- OLD = arquivo já existe, acesso exclusivo.
//DDSAI DD DSN=AD.DC40.BRPG.UPDT.DDSAI,
// DISP=(NEW,CATDLG,DELETE),
// UNIT=DISCO,
// SPACE=(CYL,(000001,000002)),
// DCB=(LRECL=80,RECFM=FB)
DDSAI = nome do comando DD (Data Definition). Deve ser igual ao definido
dentro do programa;
DD = palavra chave; indica a definição dos dados.

```

DSN = nome do arquivo catalogado. Neste caso é AD.DC40.BRPG.UPDT.DDSAI.
 DISP = indica a disposição do arquivo (status). Neste caso, indica que o arquivo é novo, será catalogado se o step terminar ok e será excluído se terminar com abend.
 UNIT = indica que o arquivo será gravado em DISCO.
 SPACE = indica o espaço que deverá ser alocado para o arquivo. Neste caso, o sistema irá alocar até ((000002*15)+000001) cilindros.
 CYL = cilindros
 DCB = contém informações a respeito dos registros do arquivo:
 - LRECL = tamanho do registro em bytes;
 - RECFM = formato do registro. Neste caso indica que o registro é fixo blocado.

```
//RELATO DD SYSOUT=Z
//*
```

RELATO = nome do comando DD (Data Definition). Deve ser igual ao definido dentro do programa;
 DD = palavra chave; indica a definição dos dados.
 SYSOUT = indica que o arquivo será direcionado para impressora.
 Z - indica que o relatório será direcionado para a fila "Z".


```
//JOB00003 JOB 'FUTU,4250,FU,PROGRA03',CLASS=B,MSGCLASS=Z
```

JOB00003 = nome do JCL (JOB)
 JOB = palavra chave; deve ser o primeiro comando (cartão) de um JCL
 'FUTU,4250,FU,PROGRA03' = account (p/ contabilização); codificado entre apóstrofes, sendo:
 - 4 posições para centro de custo = 4 primeiras posições do nome do programa;
 - 4 posições para o código do depto proprietário do centro de custo;
 - fixo FU;
 - 8 posições para o nome do programa; separar os dados acima por vírgulas; CLASS=B = nome da fila de execução; no ambiente Future, os valores permitidos são "A" e "B"; MSGCLASS=Z = fila de mensagens do sistema; no ambiente Future, o valor permitido para Terceiros é "Z".

```
//STEP1 EXEC PGM=PROGRA03
```

STEP1 = nome do step (passo); cada EXEC é um step (passo) dentro do JCL.
 EXEC = palavra chave; indica qual programa ou procedure será executado.
 PGM = indica que um programa será executado (p/ procedure é PROC).
 PROGRA03 = nome do programa a ser executado.

```
//CADASTRO DD DSN=AD.DC40.BRPG.UPDAT.CADASTRO,
// DISP=OLD
```

CADASTRO = nome do comando DD (Data Definition). Deve ser igual ao definido dentro do programa;
 DD = palavra chave; indica a definição dos dados. DSN = nome do arquivo catalogado. Neste caso é AD.DC40.BRPG.UPDT.CADASTRO.
 DISP = indica a disposição do arquivo (status).
 - OLD = arquivo já existe, acesso exclusivo.

```
//SAIDA DD DSN=&&TEMPOR,
// DISP=(,PASS),
// UNIT=DISCO,
// SPACE=(480,(1,2),RLSE),
```

```
//      DCB=(LRECL=160,RECFM=FB)
SAIDA = nome do comando DD (Data Definition). Deve ser igual ao definido
dentro do programa;
DD     = palavra chave; indica a definição dos dados.
DSN    = nome do arquivo catalogado. Neste caso é &&TEMPOR. Os dois && indicam
que o arquivo é temporário, ou seja, será excluído assim que terminar
o JOB.
DISP   = indica a disposição do arquivo (status). Neste caso, indica que o
arquivo é novo e será passado para o step seguinte.
UNIT   = indica que o arquivo será gravado em DISCO.
SPACE  = indica o espaço que deverá ser alocado para o arquivo. Neste caso,
o sistema irá alocar até ((000002*15)+000001)*480 bytes.
      480 = tamanho do bloco em bytes.
DCB    = contém informações a respeito dos registros do arquivo:
- LRECL = tamanho do registro em bytes;
- RECFM = formato do registro. Neste caso indica que o registro é
      fixo bloqueado.

//*
//* = comentário

//STEP2   EXEC PGM=PROGRA04,COND=(4,LT,STEP1) STEP2      = nome do step
(passo); cada EXEC é um step (passo) dentro do JCL.
EXEC     = palavra chave; indica qual programa ou procedure será executado.
PGM      = indica que um programa será executado (p/ procedure é PROC).
PROGRA04 = nome do programa a ser executado;
COND     = impõe uma condição para a execução do step. Serve para testar
os códigos de retorno (return code) dos step's anteriores.           Neste
caso, a condição é: 4 é menor que o return code do step1?
      Se sim, o step2 não é executado. Se não, o Step2 é executado;
//ARQENT  DD DSN=*.STEP1.SAIDA,
//      DISP=SHR
ARQENT = nome do comando DD (Data Definition). Deve ser igual ao definido
dentro do programa;
DD     = palavra chave; indica a definição dos dados.
DSN    = nome do arquivo catalogado. Neste caso, o arquivo referenciado é o
arquivo de DDNAME SAIDA do step STEP1. "*" significa "referback"      ou
referência anterior, ou seja, fazer referência a um arquivo
utilizado anteriormente.
DISP   = indica a disposição do arquivo (status).
      - SHR = arquivo já existe, acesso compartilhado
//CADSAI  DD DSN=AD.DC40.BRPG.UPDT.CADSAI,
//      DISP=(,CATLG,DELETE),
//      UNIT=DISCO,
//      SPACE=(TRK,(1,2),RLSE),
//      DCB=(LRECL=160,RECFM=FB)
CADSAI = nome do comando DD (Data Definition). Deve ser igual ao definido
dentro do programa;
DD     = palavra chave; indica a definição dos dados.
DSN    = nome do arquivo catalogado. Neste caso é AD.DC40.BRPG.UPDT.CADSAI.
DISP   = indica a disposição do arquivo (status). Neste caso, indica que o
arquivo é novo, será catalogado se o step terminar ok e será      excluído
se terminar com abend.
UNIT   = indica que o arquivo será gravado em DISCO.
SPACE  = indica o espaço que deverá ser alocado para o arquivo. Neste caso,
o sistema irá alocar até ((2*15)+1) trilhas;
      TRK = trilhas;
      RLSE = libera o espaço não utilizado.
DCB    = contém informações a respeito dos registros do arquivo:
```

- LRECL = tamanho do registro em bytes;
- RECFM = formato do registro. Neste caso indica que o registro é fixo bloqueado.

Informações necessárias para se codificar um JCL:

Informação Necessária Comando/Parâmetro

| | |
|------------------------------------|-------------------|
| Nome do JCL (JOB) | JOB/JOBNAME |
| Contabilização | JOB/ACCOUNT |
| Classe de execução (fila) | JOB/CLASS |
| Classe de mensagens do sistema | JOB/MSGCLASS |
| Onde está o programa | JOBLIB ou STEPLIB |
| Nome do step (passo) | EXEC/STEPNAME |
| Nome do Programa | EXEC/PGM |
| Condição para execução do step | EXEC/COND |
| Parâmetros para o Programa | EXEC/PARM |
| Nome da definição de Dados | DD/DDNAME |
| Nome do arquivo | DD/DSNAME |
| Disposição (Status) do arquivo | DD/DISP |
| Periférico do arquivo | DD/UNIT |
| Espaço necessário para o arquivo | DD/SPACE |
| Informações a respeito do registro | DD/DCB |
| Classe do relatório | DD/SYSOUT |
| Mapa da memória em caso de abends | DD/SYSUDUMP |

Informações mínimas para um cartão JOB:

JOBNAME
ACCOUNT
CLASS
MSGCLASS

Informações mínimas para um cartão EXEC:

STEPNAME
PGM (para executar programas)
ou
Nome-da-procedure (para executar PROC's)

Informações mínimas para um arquivo de entrada (já existe):

DSNAME
DISP (SHR/OLD)

Informações mínimas para um arquivo de saída:

DSNAME
DISP (NEW,CATLG)
UNIT (DISCO)
SPACE
DCB

Informações necessárias para um arquivo relatório:

SYSOUT
DCB

É sempre bom Ter em todos os JCLs

```
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Editaremos na biblioteca ALUNONA.FONTES.COBOL o membro EXER0001,
Para melhor aprendizado, provocaremos alguns erros, então altere na linha 002900 "PIC para POC".

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      ALUNONA.FONTES.COBOL(EXER0001) - 01.10      Columns 00001 00072  002000
DATA      DIVISION.
002100    FILE              SECTION.
002200    *
002300    FD CADPECA
002400    RECORD CONTAINS 80 CHARACTERS
002500    RECORDING MODE IS F
002600    LABEL RECORD IS STANDARD                      002700
DATA      RECORD IS REG-CADPECA.
002800    01 REG-CADPECA.
002900    05 COD-PECA POC 9(05).
003000    05 NOME-PECA PIC X(30).
003100    05 QTD-PECA PIC 9(05).
003200    05 QTD-MINIMA PIC 9(05).
003300    05 QTD-MAXIMA PIC 9(05).
003400    05 FORNECEDOR PIC X(05).
003500    05 PR-UNITARIO PIC 9(07)V99.
003600    05 PERDA PIC 9(07)V99.
003700    05 TIPO PIC X(01).
003800    05 FILLER PIC X(06).
003900    *
Command ===>                      Scroll ===> CSR

```

Altere na linha 012500 "MOVE para MODE".

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      ALUNONA.FONTES.COBOL(EXER0001) - 01.11      Columns 00001 00072
012000
012100    *=====
012200    001-02-FS-CADPECA SECTION.
012300    *=====
012400
012500    MODE 'CADPECA' TO FS-ARQUIVO.
012600    MOVE FS-CADPECA TO FS-COD-STATUS.                      012700
012800
012900    IF FS-CADPECA NOT EQUAL '00' AND '10'
013000    PERFORM 900-00-ERRO                      013100
END-IF.
013200
013300    001-02-FIM. EXIT.
013400
013500    *=====
013600    001-03-FS-CADATU SECTION.
013700    *=====
013800
013900    MOVE 'CADATU' TO FS-ARQUIVO. Command ===>
Scroll ===> CSR

```

TECLE PF3

Editaremos com a letra **E** <ENTER>ao lado esquerdo do compilador que na escola damos o nome de JCOMPJOB.

```

Menu  Functions  Confirm  Utilities  Help
-----
EDIT          ALUNONA.FONTES.COBOL          Row 00001 of 00005
Name          Prompt          VV MM          Changed          Size  Init  Mod  ID
-----
EXER0001     *Edited          01.11 10/09/04 11:37          281  198  281  IBMUSER
E JCOMPJOB          01.42 10/08/23 22:35           35   35   0  ALUNOJ2
TESTE222          01.00 10/08/17 16:37           5    5   0  ALUNOJ2
TESTE555          01.00 10/08/17 16:37           5    5   0  ALUNOJ2
TESTE777          01.00 10/08/17 16:37           5    5   0  ALUNOJ2
**End**

```

Command ==>>

Scroll ==>> CSR

Mude NOTIFY=IBMUSER para seu NOTIFY=ALUNOXX.

```

File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-----
EDIT          ALUNONA.FONTES.COBOL(JCOMPJOB) - 01.42          Columns 00001 00072
***** ***** Top of Data *****
000001 //JCOMPJOB JOB IBMUSERB,MSGCLASS=A,CLASS=C,NOTIFY=ALUNONA

```

```

000002 //IGYWCLG PROC SYSLBLK=3200,
000003 //          LIBPRFX='CEE'
000004 //*
000005 //COBOL EXEC PGM=IGYCRCTL,REGION=2048K,
000006 //          PARM=(LIST,MAP,APOST,DYNAM)
000007 //SYSPRINT DD SYSOUT=*
000008 //STEPLIB DD DISP=SHR,DSN=IGY.V2R1M0.SIGYCOMP
000009 //SYSLIN DD DSNNAME=&&LOADSET,UNIT=SYSDA,
000010 //          DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
000011 //          DCB=(BLKSIZE=&SYSLBLK)
000012 //SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000013 //SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000014 //SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000015 //SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000016 //SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000017 //SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000018 //SYSUT7 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000019 //LKED EXEC PGM=HEWL,COND=(4,LT,COBOL),REGION=1024K
Command ==>

```

Scroll ==> CSR

Podemos observar que na linha 000026 entre parênteses digite o membro a ser compilado EXER0001, na linha 000035 mude DSN=FUTURE para seu aluno DSN=ALUNONX e entre parênteses digite o membro a ser compilado EXER0001.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          ALUNONA.FONTES.COBO(JCOMPJOB) - 01.42          Columns 00001 00072
000020 //SYSLIB DD DSNNAME=&LIBPRFX..SCEELKED,DISP=SHR
000021 //          DD DSN=CSF.SCSFMOD0,DISP=SHR
000022 //          DD DSN=IBMUSER.PGMS.LOADLIB,DISP=SHR
000023 //SYSPRINT DD SYSOUT=*
000024 //SYSLIN DD DSNNAME=&&LOADSET,DISP=(OLD,DELETE)
000025 //          DD DDNAME=SYSIN
000026 //SYSIMOD DD DSN=FUTURE.LMODUL.COBO(EXER0001),DISP=SHR
000027 //SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(10,10))
000028 //STEPLIB DD DSNNAME=&LIBPRFX..SCEERUN,DISP=SHR
000029 //SYSPRINT DD SYSOUT=A
000030 //CEEDUMP DD SYSOUT=*
000031 //SYSUDUMP DD SYSOUT=*
000032 //SYSOUT DD SYSOUT=A
000033 //          PEND
000034 //COBGO EXEC IGYWCLG
000035 //COBO.SYSIN DD DSN=ALUNONA.FONTES.COBO(EXER0001),DISP=SHR
***** Bottom of Data *****

```

Command ==>

Scroll ==> CSR

Digite na linha de comando **SUBMIT** ou **SUB**<ENTER>.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          ALUNONA.FONTES.COBOL(JCOMPJOB) - 01.42          Columns 00001 00072
000020 //SYSLIB DD DSN=&LIBPRFX..SCEELKED,DISP=SHR
000021 //      DD DSN=CSF.SCSFMOD0,DISP=SHR
000022 //      DD DSN=IBMUSER.PGMS.LOADLIB,DISP=SHR
000023 //SYSPRINT DD SYSOUT=*
000024 //SYSLIN DD DSN=&&LOADSET,DISP=(OLD,DELETE)
000025 //      DD DDNAME=SYSIN
000026 //SYSLMOD DD DSN=FUTURE.LMODUL.COBOL(EXER0001),DISP=SHR
000027 //SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(10,10))
000028 //STEPLIB DD DSN=&LIBPRFX..SCEERUN,DISP=SHR
000029 //SYSPRINT DD SYSOUT=A
000030 //CEEDUMP DD SYSOUT=*
000031 //SYSUDUMP DD SYSOUT=*
000032 //SYSOUT DD SYSOUT=A
000033 //      PEND
000034 //COBGO EXEC IGYWCLG
000035 //COBOL.SYSIN DD DSN=ALUNONA.FONTES.COBOL(EXER0001),DISP=SHR
***** ***** Bottom of Data *****
```

IKJ56250I JOB JCOMPJOB(JOB06192) SUBMITTED

Command ==> SUB

Scroll ==> CSR

Mensagem na tela de job submetido.
Podemos observar que o Return code foi Maxcc=12

21.05.51 JOB06192 \$HASP165 JCOMPJOB ENDED AT N1 MAXCC=12 CN(INTERNAL)

O compilador indica ao programador se deu erro ou não através do Return Code 'retornando um código', veremos abaixo;

MAXCC=0 → indica que não houve erro.

MAXCC=4 → indica que existem alguns comentários do compilador, mas não houve erro.

MAXCC=8 → indica que houve erro de compilação é necessário a correção.

MAXCC=12 → indica que houve erro grave de compilação é necessário a correção.

Como visualizamos acima o compilador indicou Return Code MAXCC=12

```
21.05.51 JOB06192 $HASP165 JCOMPJOB ENDED AT N1 MAXCC=12 CN (INTERNAL)
***
```

TECLE <ENTER>.

Vamos visualizar os erros do membro EXER0001 no spool, para isso vamos abrir outra janela, vamos observar os erros e na outra corrigiremos os erros no programa, digite na linha de comando **START** <ENTER>;

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      ALUNONA.FONTES.COBOL(JCOMPJOB) - 01.42          Columns 00001 00072
*****  ***** Top of Data *****
000001 //JCOMPJOB JOB  IBMUSERB,MSGCLASS=A,CLASS=C,NOTIFY=ALUNONA
000002 //IGYWCLG PROC  SYSBLK=3200,
000003 //                      LIBPREFIX='CEE'
000004 //*
000005 //COBOL  EXEC  PGM=IGYCRCTL,REGION=2048K,
000006 //          PARM=(LIST,MAP,APOST,DYNAM)
000007 //SYSPRINT DD  SYSOUT=*
000008 //STEPLIB  DD  DISP=SHR,DSN=IGY.V2R1M0.SIGYCOMP
000009 //SYSLIN   DD  DSNNAME=&&LOADSET,UNIT=SYSDA,
000010 //          DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
000011 //          DCB=(BLKSIZE=&SYSLBLK)
000012 //SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
000013 //SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
000014 //SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
000015 //SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
000016 //SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
000017 //SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
000018 //SYSUT7   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
Command ==> START                               Scroll ==> CSR
F1=Help      F3=Exit      F5=Rfind      F6=Rchange    F12=Cancel

```

Opção SD <ENTER>;

```
OPTION ==> SD CUSTOMPAC MASTER APPLICATION MENU          SCROLL ==> PAGE
USERID - ALUNONA
TIME - 21:58

IS ISMF - Interactive Storage Management Facility
P PDF - ISPF/Program Development Facility
IP IPCS - Interactive Problem Control Facility
OS SUPPORT - OS/390 ISPF System Support Options
OU USER - OS/390 ISPF User Options
SM SMP/E - SMP/E Dialogs
IC ICSF - Integrated Cryptographic Service Facility
SD SDSF - System Display and Search Facility
R RACF - Resource Access Control Facility
HC HCD - Hardware Configuration Definition
BMB BMR BLD - BookManager Build (Create Online Documentation)
BMR BMR READ - BookManager Read (Read Online Documentation)
BMI BMR INDX - BookManager Read (Create Bookshelf Index)
X EXIT - Terminate ISPF using list/log defaults

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE
```

Opção ST <ENTER>;

```
Display Filter View Print Options Help
-----
HGX1900----- SDSF PRIMARY OPTION MENU -----

LOG - Display the system log
DA - Display active users in the sysplex
I - Display jobs in the JES2 input queue
O - Display jobs in the JES2 output queue
H - Display jobs in the JES2 held output queue
ST - Display status of jobs in the JES2 queues
PR - Display JES2 printers on this system
INIT - Display JES2 initiators on this system
MAS - Display JES2 members in the MAS
LINE - Display JES2 lines on this system
NODE - Display JES2 nodes on this system
SO - Display JES2 spool offload for this system

Licensed Materials - Property of IBM

5647-A01 (C) Copyright IBM Corp. 1981, 1997. All rights reserved.
COMMAND INPUT ==> ST SCROLL ==> PAGE
F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=IFIND      F6=BOOK
F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE
```

Digite **s** ao lado esquerdo do compilador requerido <ENTER>;

```

Display Filter View Print Options Help
-----
SDSF STATUS DISPLAY ALL CLASSES          DATA SET DISPLAYED
NP  JOBNAME  JOBID   OWNER    PRTY QUEUE      C  POS  SAF  ASYS STATUS
   ALUNONA  TSU06191 ALUNONA  15 EXECUTION   C   12  SYS1 SYS1
   JCOMPCOB JOB06081 ALUNONA   1 PRINT        C   13
   JCOMPCOB JOB06082 ALUNONA   1 PRINT        C   14
   JCOMPCOB JOB06083 ALUNONA   1 PRINT        C   44
   ALUNONA  TSU06128 ALUNONA   1 PRINT        C   79
   JCOMPCOB JOB06174 ALUNONA   1 PRINT        C   89
   ALUNONA  TSU06182 ALUNONA   1 PRINT        C   92
   ALUNONA  TSU06188 ALUNONA   1 PRINT        C   93
   JCOMPCOB JOB06192 ALUNONA   1 PRINT        C   93

```

```

COMMAND INPUT ===>
  F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=IFIND    F6=BOOK
F7=UP         F8=DOWN     F9=SWAP     F10=LEFT    F11=RIGHT   F12=RETRIEVE

```

Digite na linha de comando **M** e tecla PF8.
Podemos observar o Return Code 12.

```

Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY JCOMPCOB JOB06192 DSID      2 LINE 0      COLUMNS 02- 81
COMMAND INPUT ==> M                                     SCROLL ==> PAGE
***** TOP OF DATA *****
                J E S 2  J O B  L O G  --  S Y S T E M  S Y S 1  --  N

21.05.48 JOB06192 ---- MONDAY,      06 SEP 2010 ----                21.05.48
JOB06192 IRR010I  USERID ALUNONA  IS ASSIGNED TO THIS JOB.          21.05.49
JOB06192 IEF677I  WARNING MESSAGE(S) FOR JOB JCOMPCOB ISSUED        21.05.49
JOB06192 ICH70001I ALUNONA  LAST ACCESS AT 21:05:09 ON MONDAY, SEPTEMB 21.05.49
JOB06192 $HASP373 JCOMPCOB  STARTED - INIT 3      - CLASS C - SYS SYS1 21.05.49
JOB06192 IEF403I  JCOMPCOB  - STARTED - TIME=21.05.49                21.05.51
JOB06192 -                                           --TIMINGS (M 21.05.51
JOB06192 -JOBNAME STEPNAME PROCSTEP   RC   EXCP  CONN   TCB   SRB 21.05.51
JOB06192 -JCOMPCOB      COBGO       12   418    0    .01   .00 21.05.51
JOB06192 -JCOMPCOB      COBGO      FLUSH    0    0    .00   .00 21.05.51
JOB06192 IEF404I  JCOMPCOB  - ENDED - TIME=21.05.51                21.05.51
JOB06192 -JCOMPCOB  ENDED.  NAME-                               TOTAL TCB CPU TIM 21.05.51
JOB06192 $HASP395 JCOMPCOB  ENDED
JES2 JOB STATISTICS -----
      F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=IFIND      F6=BOOK
F7=UP             F8=DOWN      F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE
  
```

Visualizamos abaixo o final do compilador com algumas linhas de erros;

```

Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY JCOMPCOB JOB06192 DSID      101 LINE 408    COLUMNS 02- 81
COMMAND INPUT ==>                                     SCROLL ==> PAGE

104 IGYPS2052-S   An error was found in the definition of file "CADPECA". T
                Same message on line:      214

125 IGYPS2072-S   "MODE" was invalid.  Skipped to the next verb, period or p

175 IGYPS2053-S   An error was found in the definition of file "CADPECA". T
Messages      Total      Informational      Warning      Error      Severe      Terminating
Printed:      8
Statistics for COBOL program EXER0001:
*   Source records = 281
*   Data Division statements = 33
*   Procedure Division statements = 77
    compilation 1, program EXER0001, highest severity 12.
Return code 12
***** BOTTOM OF DATA *****
      F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=IFIND      F6=BOOK
F7=UP             F8=DOWN      F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE
  
```

Tecla PF7 procuro a linha LineID Message code Message text, após essa linha visualizo o início das linhas de contém erros do programa '.

```

Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY JCOMPJOB JOB06192 DSID 101 LINE 391 COLUMNS 02- 81
COMMAND INPUT ===> SCROLL ===> PAGE
 75 77 FS-OPERACAO . . . . . BLW=0000 060
 76 77 FS-ABERTURA . . . . . BLW=0000 070
 77 77 FS-LEITURA . . . . . BLW=0000 080
 78 77 FS-GRAVACAO . . . . . BLW=0000 090
 79 77 FS-FECHAMENTO . . . . . BLW=0000 0A0
PP 5648-A25 IBM COBOL for OS/390 & VM 2.1.2 EXER0001 Date 09
LineID Message code Message text

 17 IGYDS1028-E The "SELECT" entry was found outside of the "INPUT-OUTPUT
 the "INPUT-OUTPUT SECTION".
 23 IGYGR1232-S No "SELECT" statement was specified for file "CADPECA". T
 29 IGYDS1089-S "POC" was invalid. Scanning was resumed at the next area
 clause.
 29 IGYDS1159-E A "PICTURE" clause was not found for elementary item "COD-
F1=HELP F2=SPLIT F3=END F4=RETURN F5=IFIND F6=BOOK
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEVE

```

PF9 Navegar entre duas sessões

Tecla PF9 e navegue, variando entre as telas

Na 1º tela estamos com o compilador editado (JCOMPJOB) na outra com as linhas que estão os erros.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ALUNONA.FONTES.COBOL(JCOMPJOB) - 01.42 Columns 00001 00072
***** ***** Top of Data *****
000001 //JCOMPJOB JOB IBMUSERB,MSGCLASS=A,CLASS=C,NOTIFY=ALUNONA
000002 //IGYWCLG PROC SYSLBLK=3200,
000003 // LIBPRFX='CEE'
000004 //*
000005 //COBOL EXEC PGM=IGYCRCTL,REGION=2048K,
000006 // PARM=(LIST,MAP,APOST,DYNAM)
000007 //SYSPRINT DD SYSOUT=*
000008 //STEPLIB DD DISP=SHR,DSN=IGY.V2R1M0.SIGYCOMP
000009 //SYSLIN DD DSNNAME=&&LOADSET,UNIT=SYSDA,
000010 // DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
000011 // DCB=(BLKSIZE=&SYSLBLK)
000012 //SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000013 //SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000014 //SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000015 //SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000016 //SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
000017 //SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(1,1))

```

Command ===> Scroll ===> CSR
 F1=Help F3=Exit F5=Rfind F6=Rchange F12=Cancel

Tecla PF3

Edite o membro

```

Menu Functions Confirm Utilities Help
-----
EDIT          ALUNONA.FONTES.COBOL          Row 00001 of 00005
  E           Name      Prompt              VV MM   Changed      Size  Init  Mod  ID
  _____  _____  _____  _____  _____  _____  _____  _____  _____
           EXER0001      JCOMPJOB          01.12 10/09/06 21:03   281   198   281  IBMUSER
           JCOMPJOB          01.42 10/08/23 22:35    35    35    0  ALUNOJ2
           TESTE222          01.00 10/08/17 16:37    5     5     0  ALUNOJ2
           TESTE555          01.00 10/08/17 16:37    5     5     0  ALUNOJ2
           TESTE777          01.00 10/08/17 16:37    5     5     0  ALUNOJ2
           **End**
  
```

Command ===> Scroll ===> PAGE
 F1=Help F3=Exit F10=Actions F12=Cancel

Tecla PF9 e navegue, variando entre as telas.

Tire os erros que aprenderemos na apostila de COBOL e vamos submeter novamente o job de compilação.

Editaremos novamente o compilador JCOMPJOB.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          ALUNONA.FONTES.COBOL(EXER0001) - 01.13          Columns 00001 00072
***** ***** Top of Data ***** 000001
IDENTIFICATION DIVISION.
000002          PROGRAM-ID.          EXER0001.
000003          AUTHOR.              FUTURE SCHOOL CURSOS DE COMPUTACAO.
  
```

```
000004      *
000005      * GERAR ARQUIVO DE PECAS SELECIONADAS
000006      *
000007
ENVIRONMENT  DIVISION.
000008      CONFIGURATION SECTION.
000009      SPECIAL-NAMES.
000010      DECIMAL-POINT IS COMMA.
000011      *
000012
*INPUT-OUTPUT SECTION.
000013      *FILE-CONTROL.
000014      * SELECT CADPECA ASSIGN TO UT-S-CADPECA
000015      * FILE STATUS IS FS-CADPECA.
000016      *
000017      SELECT CADATU ASSIGN TO UT-S-CADATU
Command ====> Scroll ====> CSR
F1=Help      F3=Exit      F5=Rfind      F6=Rchange  F12=Cancel
```

Após tirar os erros você deve compilar novamente;

```
IKJ56250I JOB JCOMPJOB(JOB06197) SUBMITTED
***
```

Objetivo é MAXCC=0 ;

```
22.52.45 JOB06197 $HASP165 JCOMPJOB ENDED AT N1 MAXCC=0 CN(INTERNAL)
***
```