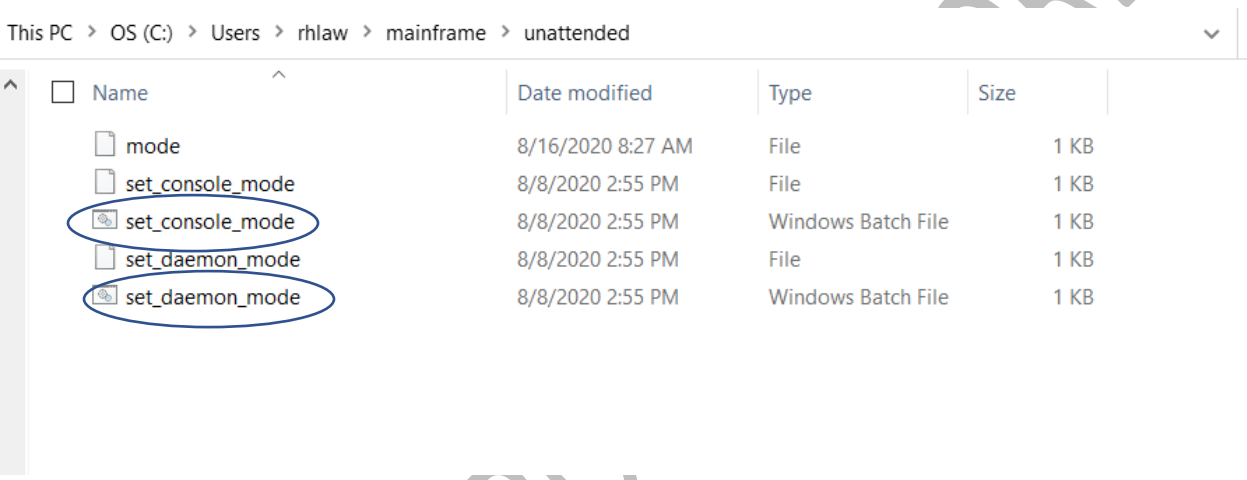
Iniciando MVS 3.8j Turnkey TK4-

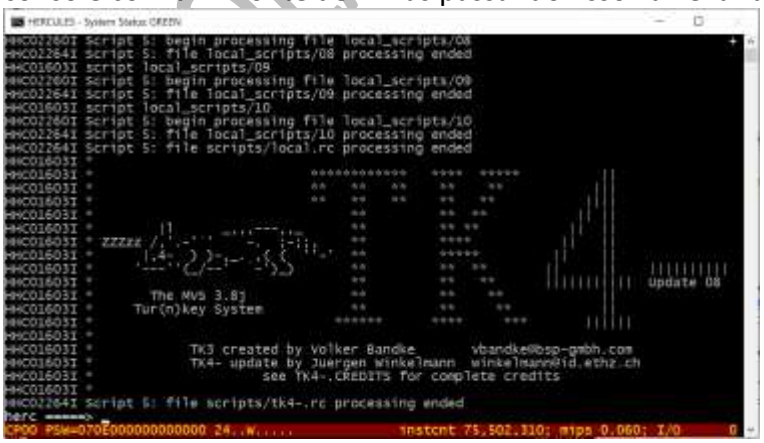
Uma vez que a instalação Turnkey TK4 tenha sido descompactada, mude para esse diretório (UNATTENDED). É possivel executar o servidor de duas maneiras diferentes. O primeiro é um modo daemon (**set_daemon_mode.batch**) onde você não pode montar fitas, variar dispositivos ligados e desligados, e outras tarefas de limpeza relacionadas a Hércules e a MVS.

Na outra opção (**set_console_mode.batch**) você terá um console que permite essas coisas. É fortemente recomendável que você executá-lo dessa forma. No diretório UNATTENDED há cinco arquivos dois dos quais são arquivos em lote.



Execute os arquivos em lote que quiser. O arquivo de lote 'set_console_mode' configurará o sistema em execução com um console. O arquivo de lote 'set_daemon_mode' configurará o sistema para ser executado em um modo autônomo.

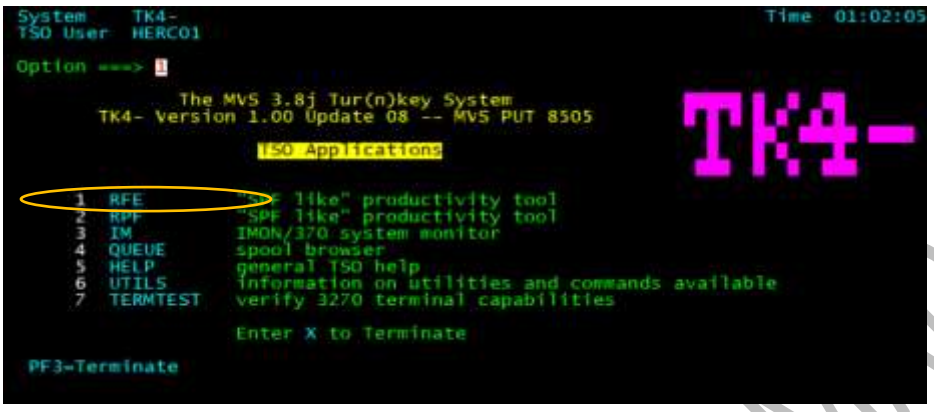
Depois de definir o modo de execução, você pode iniciar o MVS. No Windows volte uma pasta (cd ..) e execute mvs.bat. No Linux ou Mac execute ./mvs. Você verá a seguinte janela ou console com um monte de linhas passando. Isso vai levar alguns minutos para começar.

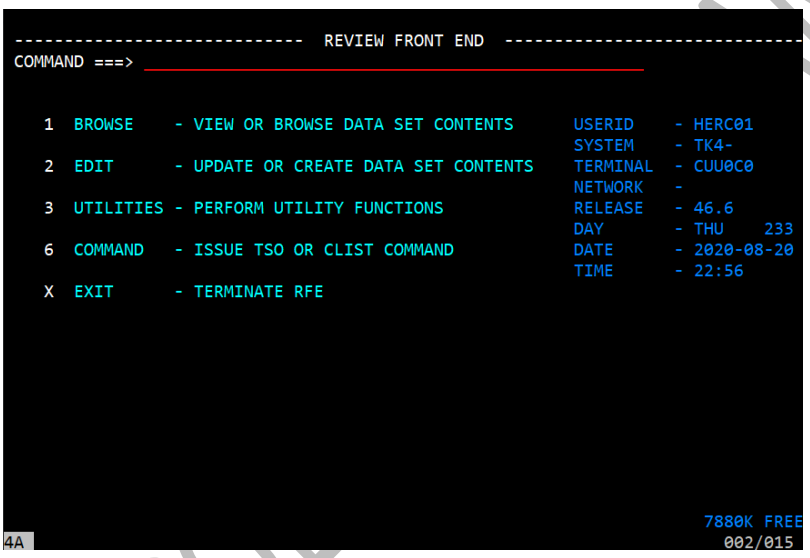


Quando o console ou janela exibe o grande TK4-, MVS 3.8J está pronto para ser conectado pelo TN3270.

Visão geral do uso do aplicativo TSO RFE

Na maioria das vezes, a administração ou programação de MVS 3.8j você estará usando o aplicativo RFE. Tem uma série de recursos que o ISPF da IBM tem. Esta seção descreverá algumas das tarefas de administração e programação que serão pré-formadas no dia a dia. Não cobre tudo o que a RFE pode fazer, apenas as principais tarefas.



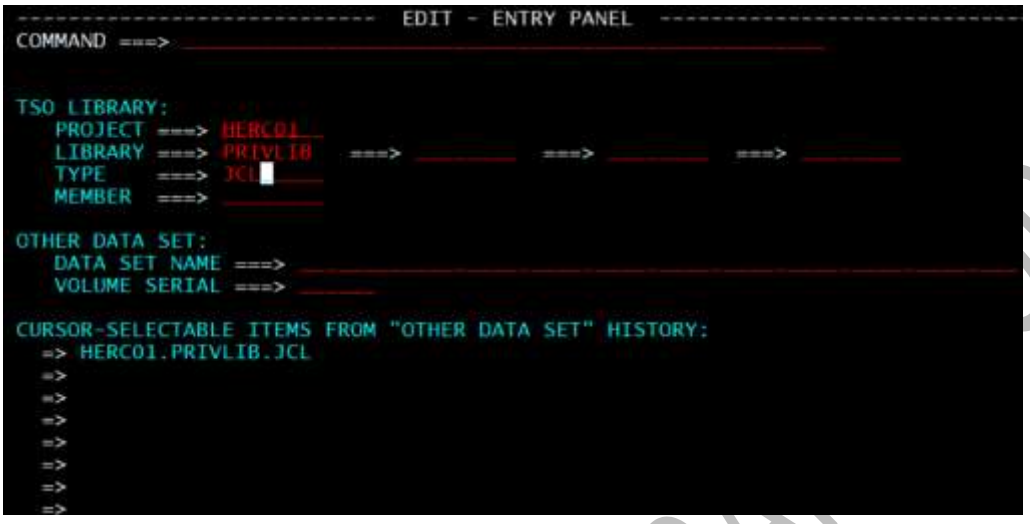


O menu principal da RFE tem as seguintes opções:

1. Browse
2. Edit
3. Utilities
4. Command
5. Exit

Edit Panel

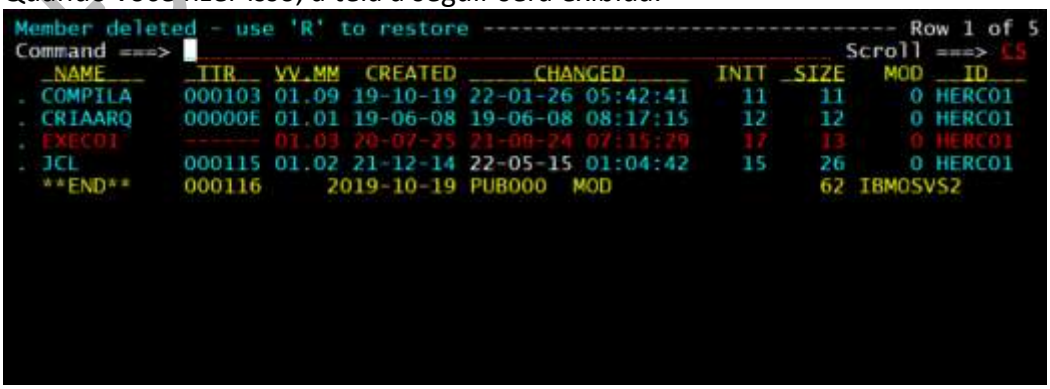
A opção Editar é usada para olhar e editar membros de conjuntos de dados. Se você sabe que o conjunto de dados com o qual está trabalhando é uma boa opção.



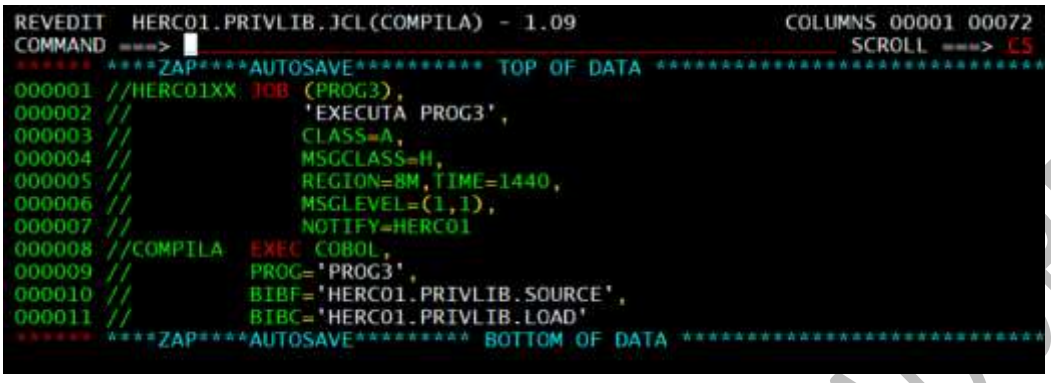
Se você souber o nome do conjunto de dados e deixar o nome do membro em branco, você poderá obter uma exibição de todos os membros no conjunto de dados. Em seguida, você pode selecionar um dos membros e a tela de revedit será exibida e você pode trabalhar com o membro. Se você conhece o conjunto de dados e o nome do membro, o programa levará o diretório para a tela de revedit para que você trabalhe com ele.

Há várias maneiras de chegar a esta tela. O que usamos foi através do Painel de Edição RFE. Esta tela mostra uma lista dos membros. Os Conjuntos de Dados que têm membros são chamados de PDS (Partitioned Data Set, conjunto de dados particionados). Quase toda a entrada de código é feita em membros do PDS. Posicione o cursor no campo de seleção do membro com o qual deseja trabalhar e digite um 's' ou 'e' para exibir o conteúdo do membro. Também é possível excluir o membro digitar 'd' na seleção field.

Quando você fizer isso, a tela a seguir será exibida.



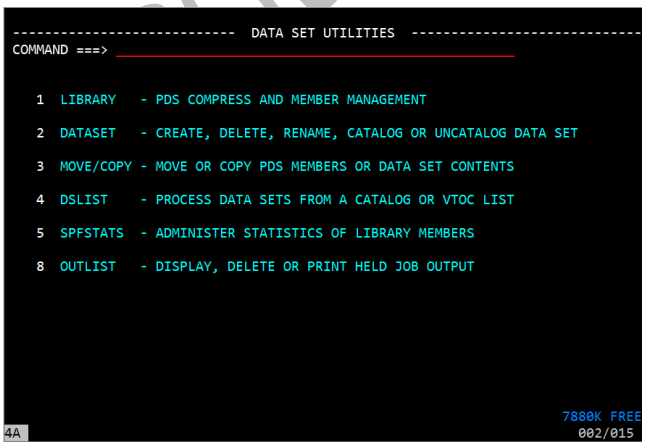
Observe que o membro do EXEC01 está em uma espécie de cor vermelha/marrom e há uma mensagem na parte superior da tela dizendo 'Membro excluído - use 'R' para restaurar ----------'. Se você fosse ver outro membro ou sair da tela, o membro teria ido embora. Se você tiver acertado a tecla 'd' por engano, posicione o cursor no campo de entrada no membro que você acabou de excluir e digite 'R' ou 'r' para recupera o membro.



A tela de edição é exibida quando você digita um 's' ou 'e' na frente do nome do membro na tela REVEDIT. Observe que a linha superior indica que o membro COMPILA do HERC01.PRIVLIB conjunto de dados particionados JCL.

Na linha de comando você pode inserir opções como 'salvar', para salvar o membro ou 'sub' para enviar o membro como um trabalho. Claro que se você fizer isso, ele precisa conter JCL. Na frente de cada linha há um número de linha. Este é um número sequencial das linhas no membro. Nesta linha você pode inserir valores como 'i' para inserir uma linha em branco, 'd' para excluir uma linha, 'c' para copiar, 'm' para mover e uma série de outros valores. Isso permite uma boa capacidade de edição do arquivo. Para ver as outras opções, digite 'ajuda' na linha de comando. Você pode mudar a linha posicionando seu cursor onde deseja alterar e digitar a mudança.

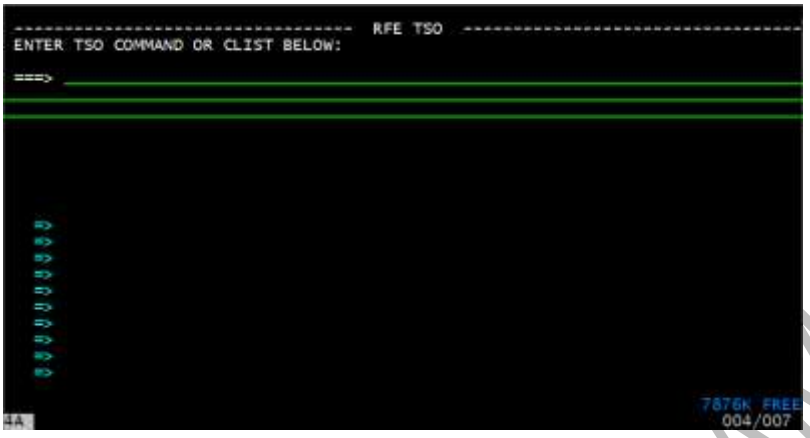
Utilities



A opção Escolha da opção Utilitários exibe o menu Data Set Utilities. Os itens do menu são:

1. Library
2. DataSet
3. Move/Copy
4. DSList
5. SPFStats
6. Outlist

Command

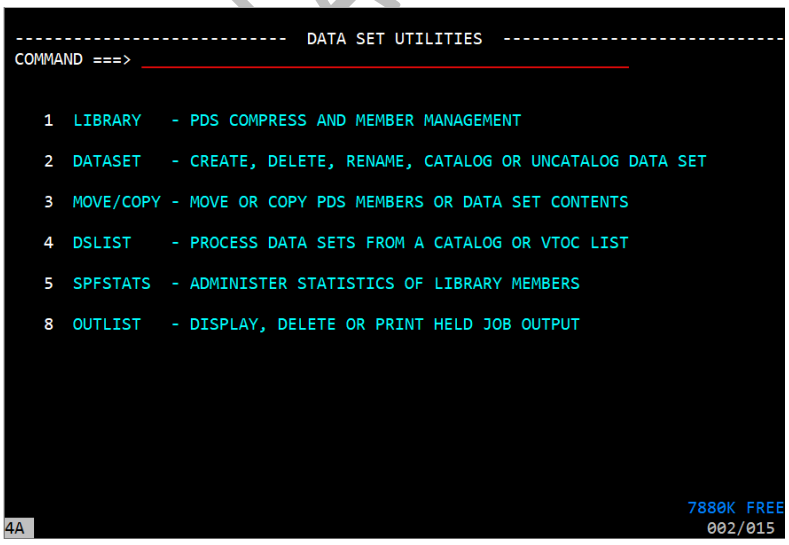


A tela de comando permite executar um comando TSO ou CLIST sem sair do RFE.

Exit

Digite a opção de saída encerrará os Utilitários de conjunto de dados e o devolverá ao menu de aplicativos TSO.

Data Set Utilities Panel

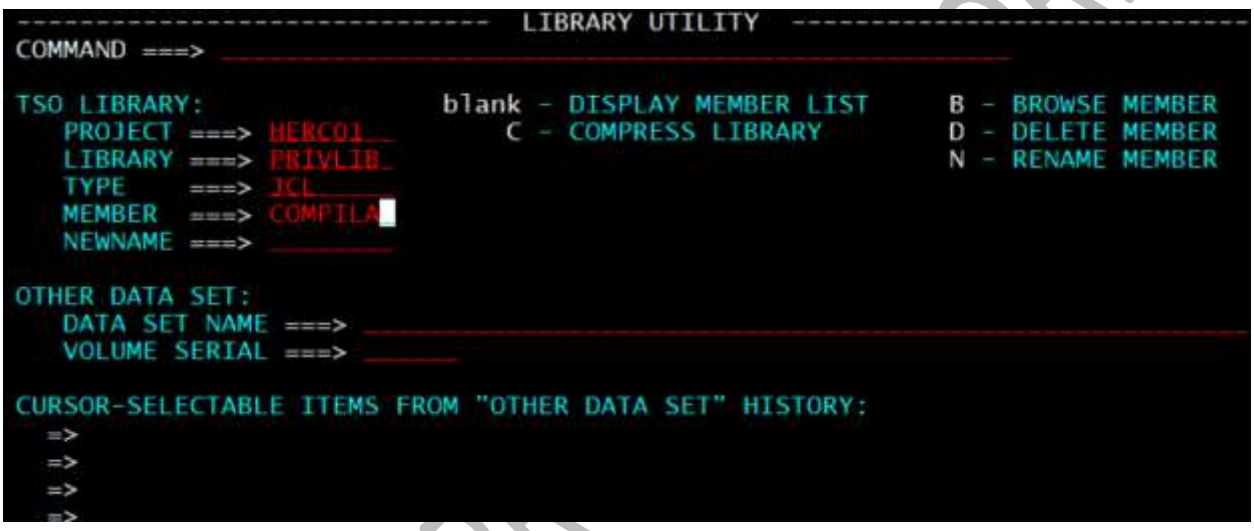


ESCOLADEPROGRAMADORES

O menu Data Set Utilities permite que você execute o número de tarefas em conjuntos de dados e seus membros.

Opção Library

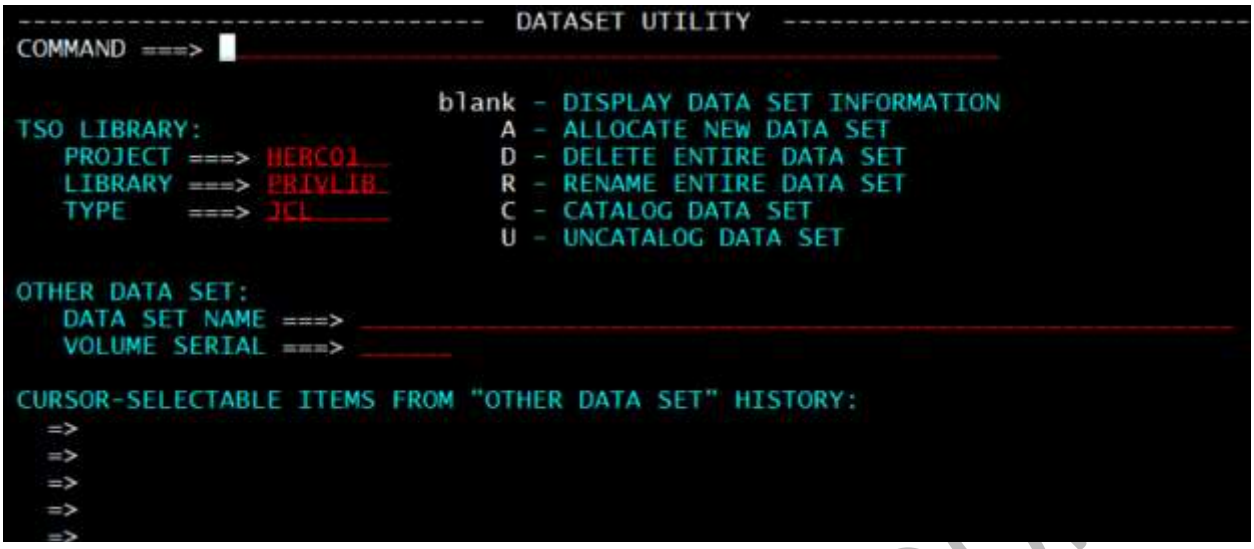
Branco	Exibir uma lista de membros para um PDS
B	Exibir o código-fonte de um membro PDS
N	Renomeie um membro
D	Exclua um membro
C	Comprimir um membro



DATASET UTILITY

Branco	Exibir informações sobre o conjunto de dados
A	Aloque um novo conjunto de dados
D	Exclua um conjunto de dados e se for um PDS, exclua seus membros
R	Renomeie um conjunto de dados
C	Catalogar um conjunto de dados
U	Descataloga um conjunto de dados

ESCOLADEPROGRAMADORES



ALLOCATE

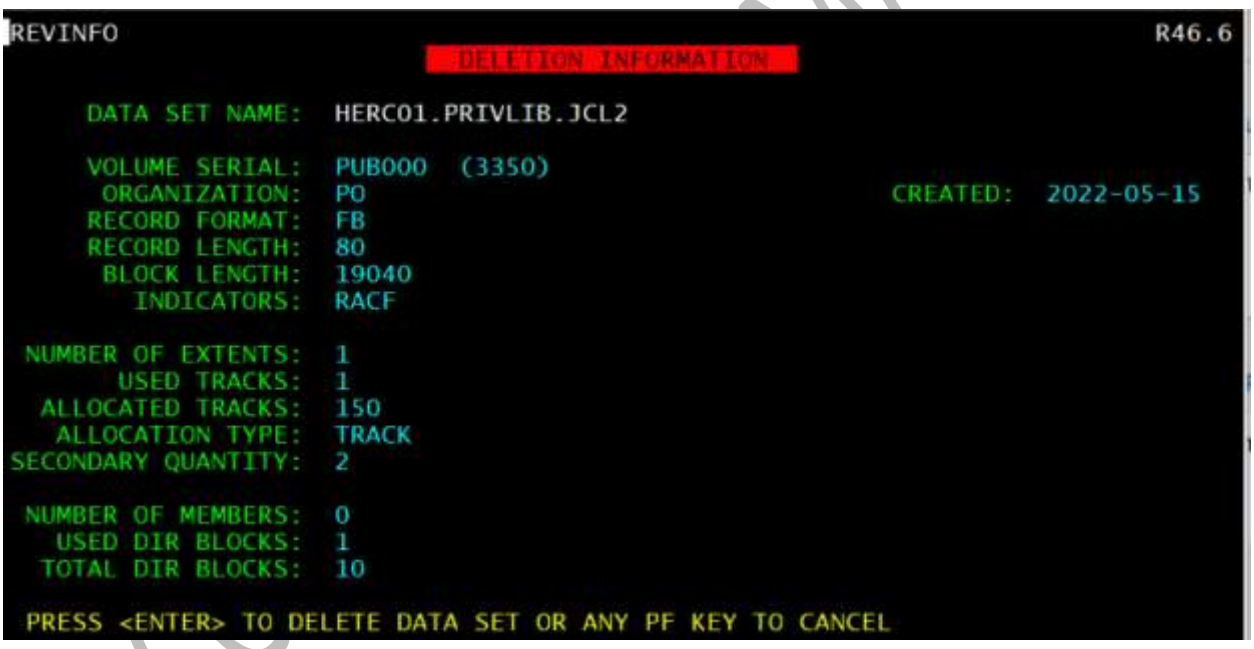
Quando você optar por alocar um novo conjunto de dados, a tela acima é exibida. Ele exibe o nome do conjunto de dados inserido, mas você tem a opção de alterar o nome.

RECORD FORMAT ==> O formato de registro pode ser 'F' - Comprimento fixo, 'FB' - Comprimento fixo e bloqueado, 'V' - Comprimento variável, 'VB' - Comprimento variável, bloqueado.
LOGICAL RECORD LENGTH ==>Tamanho do registro (numero de caracteres). Para bibliotecas de Cobol, JCL usar 80. Para arquivos sequenciais, definir através do layout dos dados.
PHYSICAL BLOCK SIZE ==> O tamanho do bloco físico é do tamanho de um bloco. Para registros Blocks (FB ou V) , deve ser um múltiplo do Comprimento do Registro Lógico.
VOLUME ==> o volume para colocá-lo. Unidade do tipo de unidade (ou seja, 3350) deve ser armazenada.
UNIT ==> Unidade de disco – deixar em branco. Tk4 assume o melhor disco
ALLOCATION SPACE UNIT ==> Unidade de espaço de alocação é 'T' para TRILHA e 'C' para cilindro.
PRIMARY SPACE QUANTITY ==> A Quantidade de Espaço Primário é o número de faixas ou ciclodes para alocar intitalmente.

SECONDARY SPACE QUANTITY ==> A Quantidade de Espaço Secundário é a quantidade de espaço para alocar se o espaço principal estiver cheio. Sob certas condições, pode ser alocado em até 15 vezes.
NUMBER OF DIRECTORY BLOCKS ==> O número de blocos de diretório deve ser deixado vazio a menos que um conjunto de dados particionado (biblioteca) esteja sendo criado. Então isso precisa ser introduzido. É o número de blocos que você deseja alocar para as entradas do diretório. Uma entrada de diretório é de comprimento variável e um bloco pode conter mais de uma entrada de diretório. Quando terminar, pressione a tecla enter para executar a alocação.

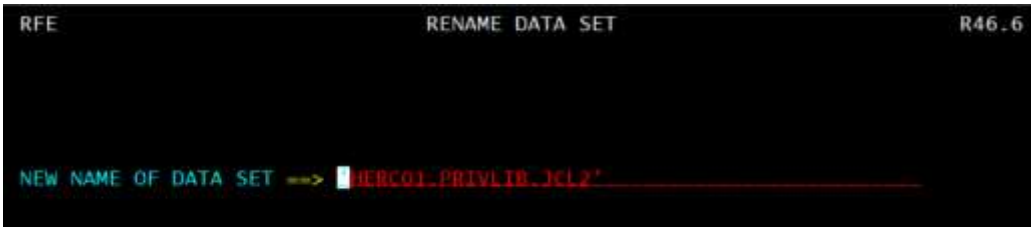
ESCOLADEPROGRAMADORES

Esta tela é exibida quando você escolhe excluir um conjunto de dados. Pressione enter para concluir a exclusão ou qualquer PFkey para cancelar a exclusão.



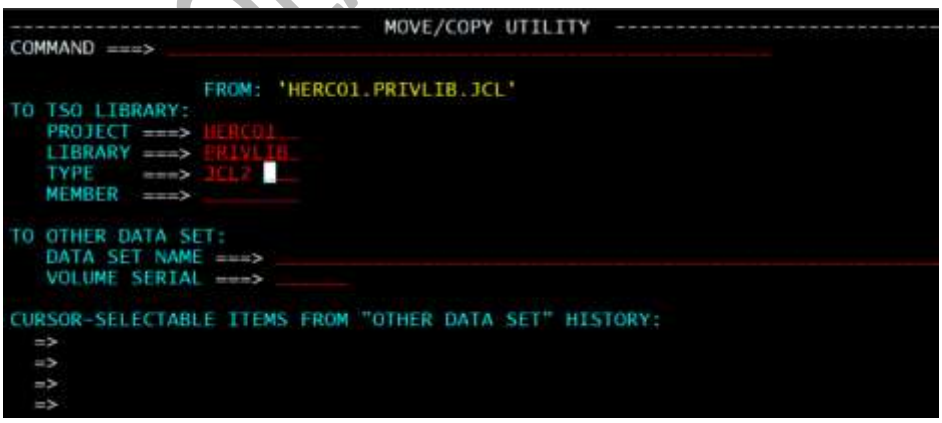
ESCOLADEPROGRAMADORES

Quando você optar por renomear um conjunto de dados, a tela abaixo será exibida. Ele permite que você insira um novo nome para o conjunto de dados.



Move/Copy Panel

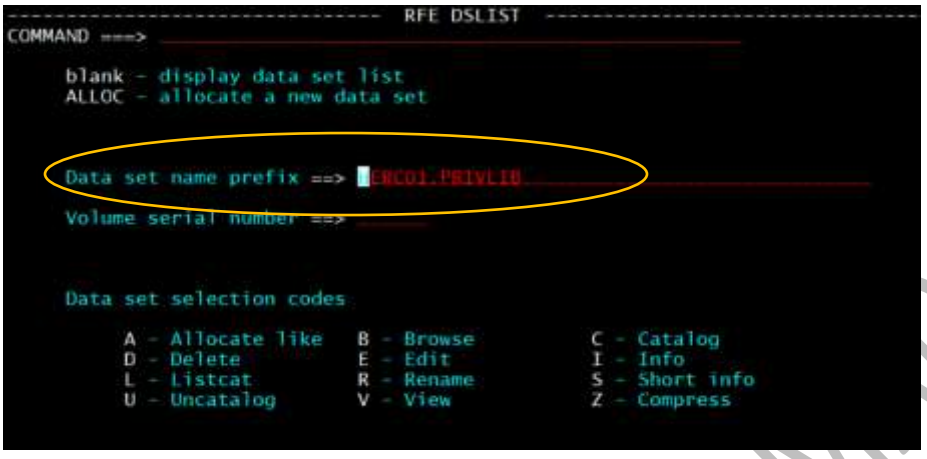
O painel Move/Copy do painel permite especificar um conjunto de dados e, opcionalmente, um membro ou todos os membros para outro conjunto de dados. Na FROM TSO LIBRARY coloque as informações do conjunto de dados que você vai copiar ou mover. Se ele vai copiar ou mover depende de entrar em um 'c' ou 'm' na linha de comando. Depois de pressionar a tecla Enter, o painel Mover/Copiar para o painel será exibido.

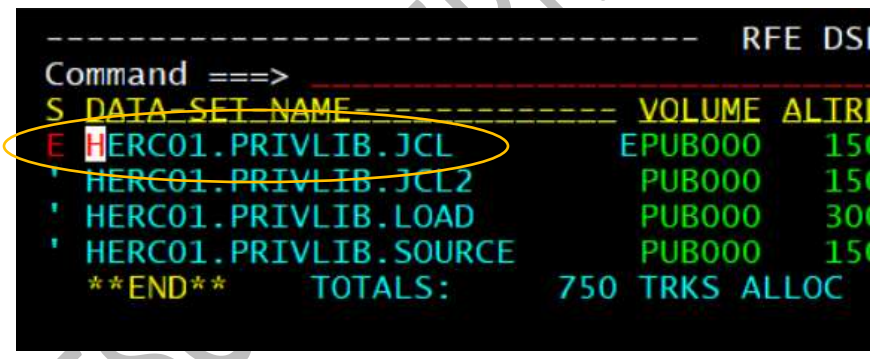


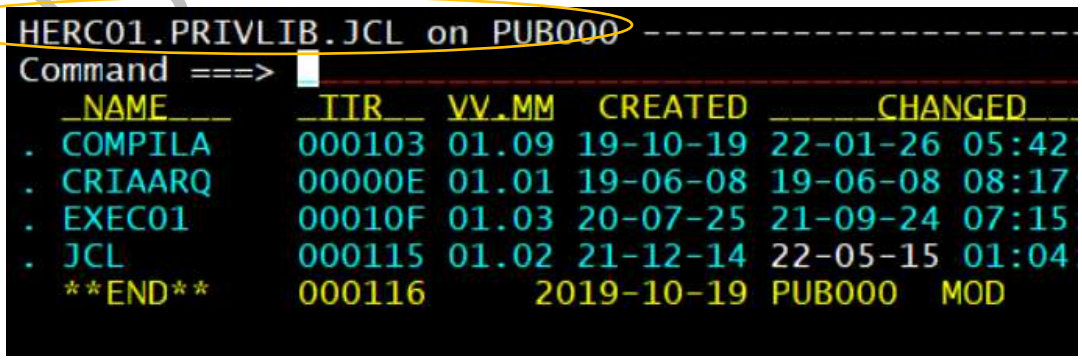
Digite as informações da TO TSO LIBRARY e pressione digitar copiar para executar o movimento ou cópia.

DSLIST

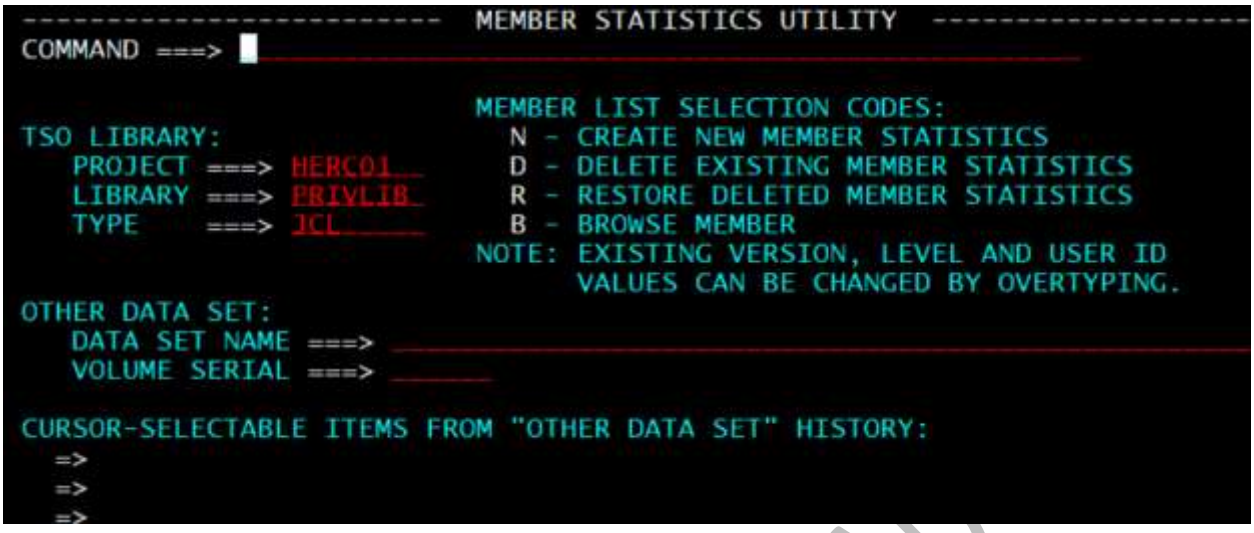
Quando você escolhe a opção 4 DSLIST no menu, a tela abaixo é exibida. O uso mais comum disso é encontrar um conjunto de dados. Você pode entrar em cada parte desde a primeira; primeiro e segundo; ou primeira, segunda e terceira partes da Biblioteca TSO e o sistema encontrarão todos os conjuntos de dados correspondentes e os mostrarão na tela a seguir.







SPFStats



Estatísticas de membros de dados.

OUTLIST

Saida das execuções dos JCLS. Ele mostra a saída para trabalhos que começam com o usuárioid (HERC01) na cartão JOB do JCL.

Na imagem acima o id do usuário é 'HERC01'.

ESCOLADEPROGRAMADORES

É possível ver todos os arquivos digitando em COMMAND ➔ ST *. (para voltar basta digitar ST)

```
REVOUT
COMMAND ===> ST *
S Q JOBNAME  JOBIDENT QUEUE  EXEC STATUS
' ' HERC01   TSU00085 XEQ @  TK4-
' ' HERC01XX JOB00266 PRTPUN
' ' HERC01XX JOB00267 PRTPUN
```

```
REVOUT   *
COMMAND ===>
S Q JOBNAME  JOBIDENT QUEUE  EXEC STATUS          STEPNAME
' ' SYSLOG   STC00707 XEQ $  TK4-
' ' BSPPILOT STC00708 XEQ $  TK4-                 BSPPILOT
' ' INIT     STC00709 XEQ $  TK4-                 INIT
' ' INIT     STC00710 XEQ $  TK4-                 INIT
' ' INIT     STC00711 XEQ $  TK4-                 INIT
' ' INIT     STC00712 XEQ $  TK4-                 INIT
' ' INIT     STC00713 XEQ $  TK4-                 INIT
' ' INIT     STC00714 XEQ $  TK4-                 INIT
' ' NET      STC00717 XEQ $  TK4-                 NET
' ' TP       STC00718 XEQ $  TK4-                 TP
' ' MF1      STC00719 XEQ $  TK4-                 MF1
' ' TSO      STC00720 XEQ $  TK4-                 TSO
' ' SNASOL   STC00721 XEQ $  TK4-                 SNASOL
' ' IRP      STC00722 XEQ $  TK4-                 IRP
```

Isso é útil quando o nome do trabalho não começa com a identificação do usuário.

Para ver o conteudo do JOB basta inserir um 'S' e pressionar enter.
Isso fará com que o painel de exibição de saída apareça.

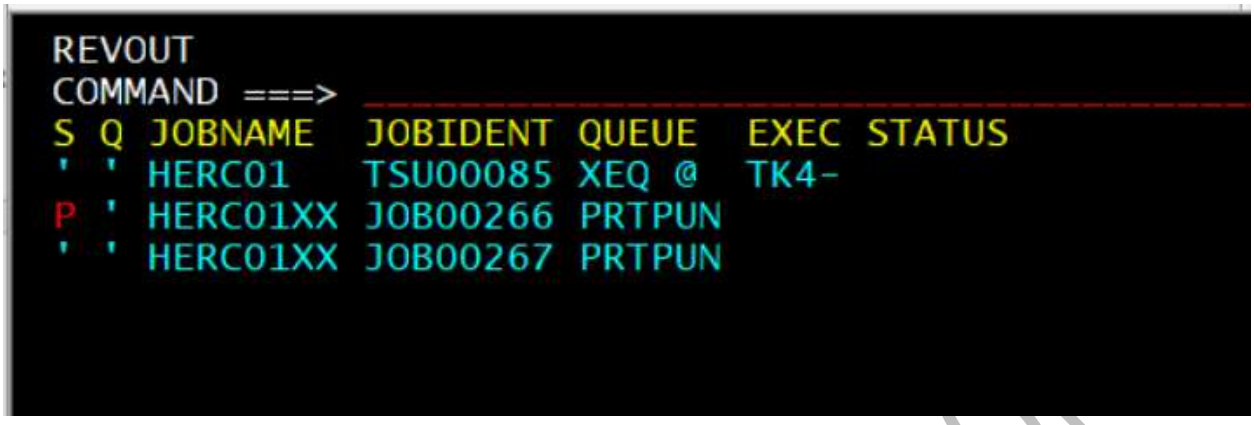
```
REVOUT
COMMAND ===>
S Q JOBNAME  JOBIDENT QUEUE  EXEC STATUS          STE
' ' HERC01   TSU00085 XEQ @  TK4-                 TSO
S ' HERC01XX JOB00266 PRTPUN
' ' HERC01XX JOB00267 PRTPUN
```

```
SYS22135.T014358.RA000.HERC01.JOB00266 ----------------------- Line
Command ===>                                                   Scro
      10        20        30        40        50        60        70
---+----+----+----+----+----+----+----+----+----+----+----+----+----+

                                                 J E S 2   J O B   L O
21.59.24 JOB  266  IEF452I JOBFAIL  JOB NOT RUN - JCL ERROR
    1      //HERC01XX JOB (FORTRAN),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),
    2      //FIRSTPGM  EXEC FORTHCLD,REGION.FORT=384K,PARM.FORT='LIST'
    3      XXFORTHCLD PROC SOUT='*'
    4      XXFORT     EXEC PGM=IEKAA00
    5      XXSYSPRINT DD SYSOUT=&SOUT
    6      XXSYSPUNCH DD SYSOUT=B
    7      XXSYSLIN   DD DSNAME=&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),
           XX           SPACE=(400,(200,50),RLSE)
    8      //FORT.SYSIN DD *
    9      XXGO       EXEC PGM=LOADER,PARM=(MAP),COND=(4,LT,FORT)
   10      XXSYSLIB   DD DSNAME=SYS1.FORTLIB,DISP=SHR
   11      XXSYSLOUT  DD SYSOUT=&SOUT
   12      XXSYSLIN   DD DSNAME=*.FORT.SYSLIN,DISP=(OLD,PASS)
   13      XXFT05F001 DD DDNAME=SYSIN
```

ESCOLA DE PROGRAMADORES

Quando feito com a visualização da saída de JOB é possível limpar o trabalho colocando um 'P' na frente do trabalho na coluna de seleção.



ESCOLADEPROGRAMADORES

JCL Visão geral

Introdução ao JCL

Esta é uma breve introdução ao JCL e como usá-lo. Ele mostra uma série de parâmetros para as várias instruções JCL, mas não descreve todos os parâmetros possíveis.

O JCL foi introduzido com o primeiro Mainframe IBM e ainda é usado no mais novo sistema z/OS. À medida que novos sistemas foram adicionados, os recursos foram adicionados pelo JCL no computador OS/360, está muito vivo em máquinas mais novas. Os computadores originais do mainframe tinham leitores de cartões, fitas e impressoras. Essa foi a situação com a MFT e a MVT, os antecessores imediatos da MVS. Até hoje a maioria dos mainframes ainda tem leitores de cartões.

Para executar um trabalho, um conjunto de cartões seria criado e lido no sistema pelo leitor de cartas. O sistema veria o jcl e o executaria. No MVS 3.8J é possível enviar um arquivo de texto contendo linhas jcl para o leitor de cartão. Também é possível e bastante comum colocar o JCL em um conjunto de dados particionado e submeter o trabalho a partir daí. Aqui está uma foto de um JCL.

```
REVEDIT  HERC01.PRIVLIB.JCL(CRIAARQ) - 1.01              COLUMNS 00001 00072
COMMAND ===>                                              SCROLL ===> CS
****** ****ZAP****AUTOSAVE********** TOP OF DATA ********************************
000001 //HERC010A JOB (COBOL),
000002 //            'COMPILACION COBOL',
000003 //            CLASS=A,
000004 //            MSGCLASS=H,
000005 //            REGION=8M,TIME=1440
000006 //IEFBR14   EXEC PGM=IEFBR14
000007 //SYSPRINT  DD   SYSOUT=*
000008 //ENTRADA   DD   DUMMY
000009 //ARQUIVO   DD   DSN=HERC01.ARQ.CLIENTES,DISP=(NEW,CATLG,DELETE),
000010 //            SPACE=(TRK,(1,1),RLSE),
000011 //            UNIT=SYSDA,
000012 //            DCB=(RECFM=FB,LRECL=50,BLKSIZE=19050,BUFNO=2)
****** ****ZAP****AUTOSAVE********* BOTTOM OF DATA ******************************
```

Um trabalho consiste em três tipos de cartões ; JOB, EXEC e DD. A instrução JOB fornece parâmetros e palavras-chave sobre todo o trabalho. A declaração do EXEC descreve programas a serem executados. Pode haver mais de uma declaração da EXEC em um fluxo de trabalho. Cada declaração exec é chamada de passo. A instrução DD descreve os conjuntos de dados, impressoras e outros dispositivos que o programa EXEC requer para executar.

Ao falar sobre jcl é comum usar o termo "cartão" em vez de "gravar" ou "linha". Neste tutorial será usado "cartão".

ESCOLA DE PROGRAMADORES

Definição do cartão JCL

Uma cartão JCL será dividida em quatro seções.

1. Campo identificador
2. Campo de nomes
3. Campo de Operação
4. Campo de ParaMeters

```
000001 //HERC010A JOB (COBOL),
000002 //            'COMPILACAO COBOL',
000003 //            CLASS=A,
000004 //            MSGCLASS=H,
000005 //            REGION=8M,TIME=1440
```

A cartão JOB é que define o trabalho. Inicia com duas barras (//).
Na sequencia até oito caracteres é o nome do trabalho. (HERC010A) .
O nome pode ter de 1 a 8 caracteres de comprimento. É seguido por pelo menos um espaço após o qual é a palavra "JOB". Seguir a palavra é pelo menos um espaço.

Após o trabalho, o JOB são valores posicionais e de palavras-chave que fornecem alguns parâmetros para o trabalho. Cada parâmetro é selado por uma vírgula (,). Se houver espaços em um parametro, ele deve ser fechado por parenteses.

O primeiro parâmetro é posicional e é informação contábil de trabalho (COBOL).
Muitos sistemas de produção usam parâmetros contábeis no JCL para difundir custos de computação para vários departamentos. O segundo parâmetro também é posicional e é o nome do programador. Tanto o primeiro quanto o segundo parâmetros são opcionais.

Os parâmetros da palavra-chave começam com a palavra-chave e são imediatamente seguidos por um sinal igual (=). As palavras-chave para o cartão JOB são:

PALAVRA	PARA QUE SERVE
USER	O usuário abaixo do qual o trabalho será executado
PASSWORD	A senha do usuário
NOTIFY	O usuário do usuário TSO deve ser notificado quando o trabalho estiver concluído
CLASS	Classe para processamento do JOB
MSGLEVEL	CLasse de saida do processamento e dados do JOB
REGION	Area de memoria onde será executado o JOB

ESCOLADEPROGRAMADORES

CARTAO EXEC

```
000006 //IEFBR14   EXEC PGM=IEFBR14
```

O CARTÃO EXEC começa com duas barras (//), seguido por um nome, seguido por pelo menos um em branco, seguido pela palavra EXEC, seguido por pelo menos um em branco seguido por PGM=programa onde o Program é um programa que pode ser carregado do sistema (Seu programa COBOL por exemplo).
Um cartão EXEC é considerado um passo. É seguido pelos cartões DD. Após os cartões DD, outro passo pode ser iniciado por ter outra declaração exec. Alguns trabalhos podem ter 20 ou mais etapas.

CARTÃO DD

```
//SYSPRINT  DD   SYSOUT=*
//ENTRADA   DD   DUMMY
//ARQUIVO   DD   DSN=HERC01.ARQ.CLIENTES,DISP=(NEW,CATLG,DELETE),
//          SPACE=(TRK,(1,1),RLSE),
//          UNIT=SYSDA,
//          DCB=(RECFM=FB,LRECL=50,BLKSIZE=19050,BUFNO=2)
```

Uma instrução DD permite que um programa se comunique com dispositivos fora do próprio programa. Uma declaração DD consiste de um a qualquer número de cartões. A declaração começa com duas barras (//) seguidos do nome que o programa usa para se referir à declaração. Esse nome é seguido por pelo menos um espaço e, em seguida, a palavra DSN e o arquivo que será lido
O DD pode ser seguido por um dos três tipos de parâmetros. Estes parâmetros são:

Parâmetro	Descrição
*	DD consiste em dados inline. As linhas imediatamente após a instrução DD são dados que o programa usa. Os dados são encerrados quando uma linha composta por '/*' é encontrada.
DUMMY	O arquivo precisa que o DD seja definido, mas não há dados para um DD de entrada e nenhum destino para qualquer saída usando o DD
PALAVRA-CHAVE=	Uma das várias palavras-chave seguidas pelo sinal de equals seguido pelo valor(s) para a palavra-chave. Alguns deles são DSNAME, DSN, DISP, SYSOUT, SYSDA, UNIT, VOL=SER, SPACE e DCB.

ESCOLADEPROGRAMADORES

Terminando um JCL

Quando todas as etapas do EXEC e suas definições de DD foram inseridas, um fluxo de trabalho é fechado por um cartão com duas barras(//). Tecnicamente, o cartão de duas barras não é necessário. No entanto, se houver uma submissão de mais de um trabalho em um fluxo de cartões, os dois cortes são como o sistema sabe que o trabalho está no fim. Em seguida, quando ler a próxima carteira JOB, não registrará um erro.

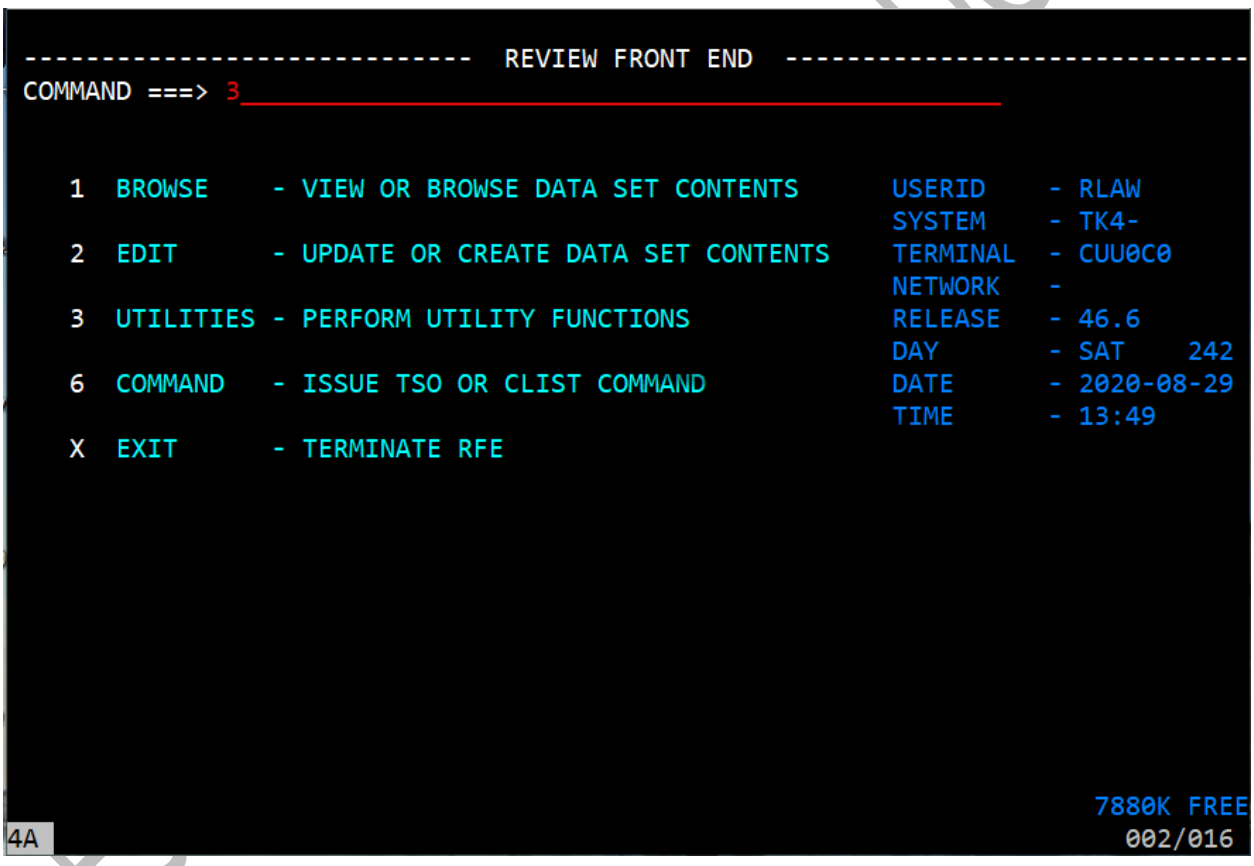
ESCOLADEPROGRAMADORES

Seu primeiro programa COBOL

O nome COBOL é um acrônimo para "linguagem comum orientada para negócios". COBOL foi uma das primeiras linguagens de programação. Na verdade, as únicas línguas mais antigas que o COBOL são FORTRAN, LISP e ALGOL. COBOL foi basicamente projetado por um comitê e como seu nome implica é uma linguagem orientada para negócios, você não gostaria de projetar sistemas de missle com COBOL. A versão do COBOL que está disponível com MVS 3.8J é a COBOL 65.

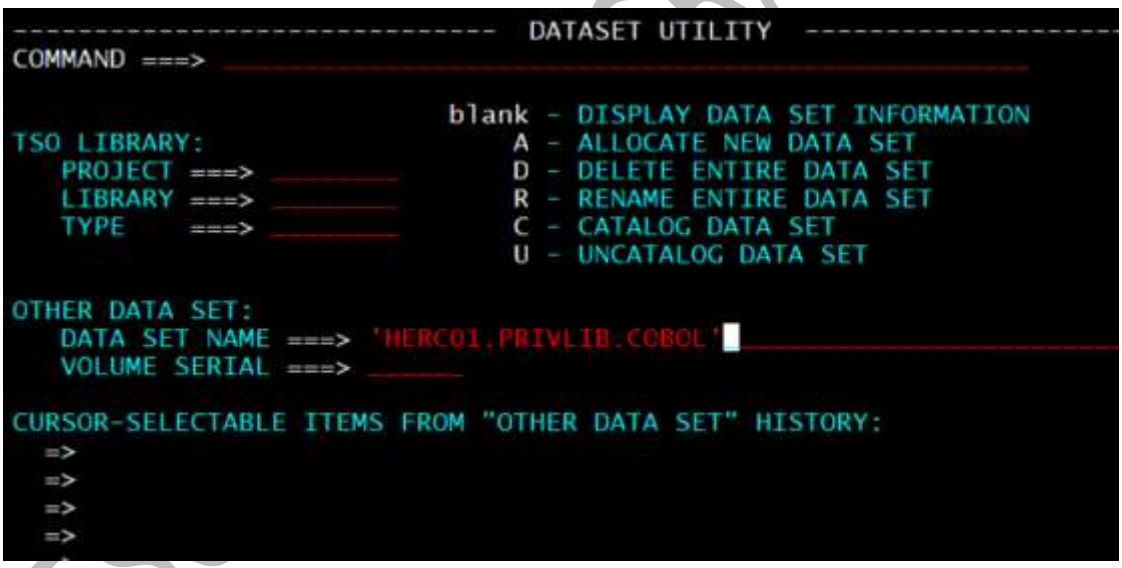
O programa que está sendo desenvolvido aqui é um programa simples de COBOL e dará uma rápida introdução às partes essenciais do idioma.

Criando um dataset



Como esta é a primeira vez nessas lições onde o desenvolvimento real ocorrerá, recuamos um passo atrás e criamos um conjunto de dados particionado para salvar nossos programas COBOL. Entre no RFE, pegue a opção 3 para trazer o menu Utilities.

Uma vez que o menu UTILITIES seja exibido, tome a opção 2 para trabalhar com conjuntos de dados.

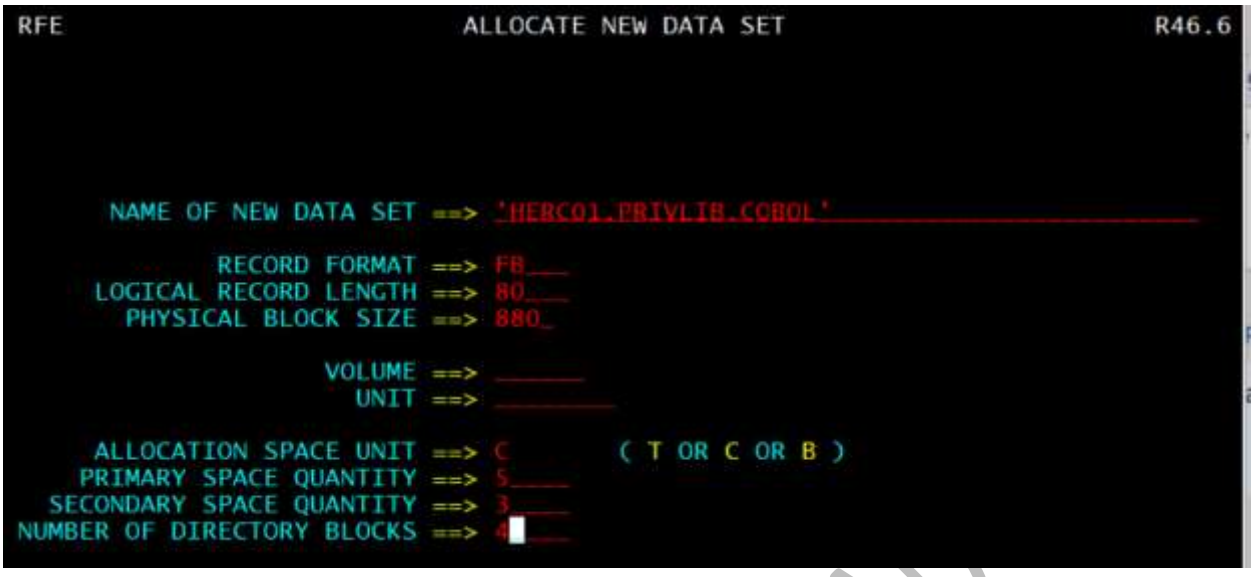


Na tela do conjunto de dados, na área de entrada COMMAND =====> digite A, uma vez que estaremos alocando um novo conjunto de dados.

Na área DATA SET NAME, insira ‘HERC01.PRIVLIB.COBOL’ incluindo as aspas.

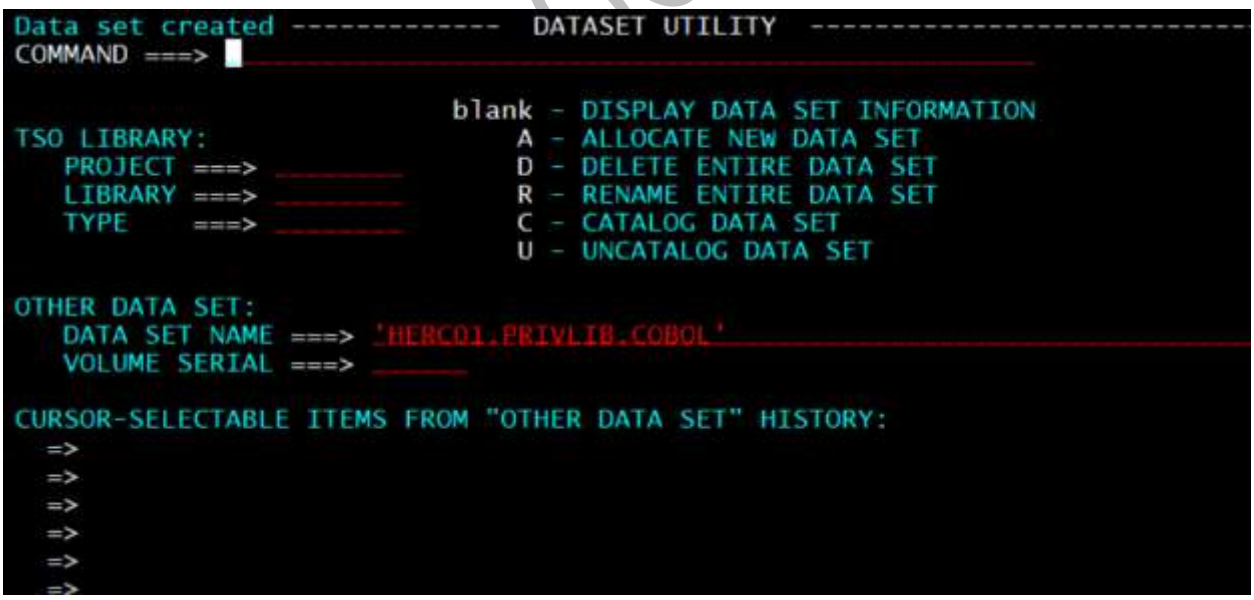
Na área VOLUME SERIAL, deixe em branco. Pressione ENTER para prosseguir para o menu de criação do conjunto de dados.

ESCOLADEPROGRAMADORES

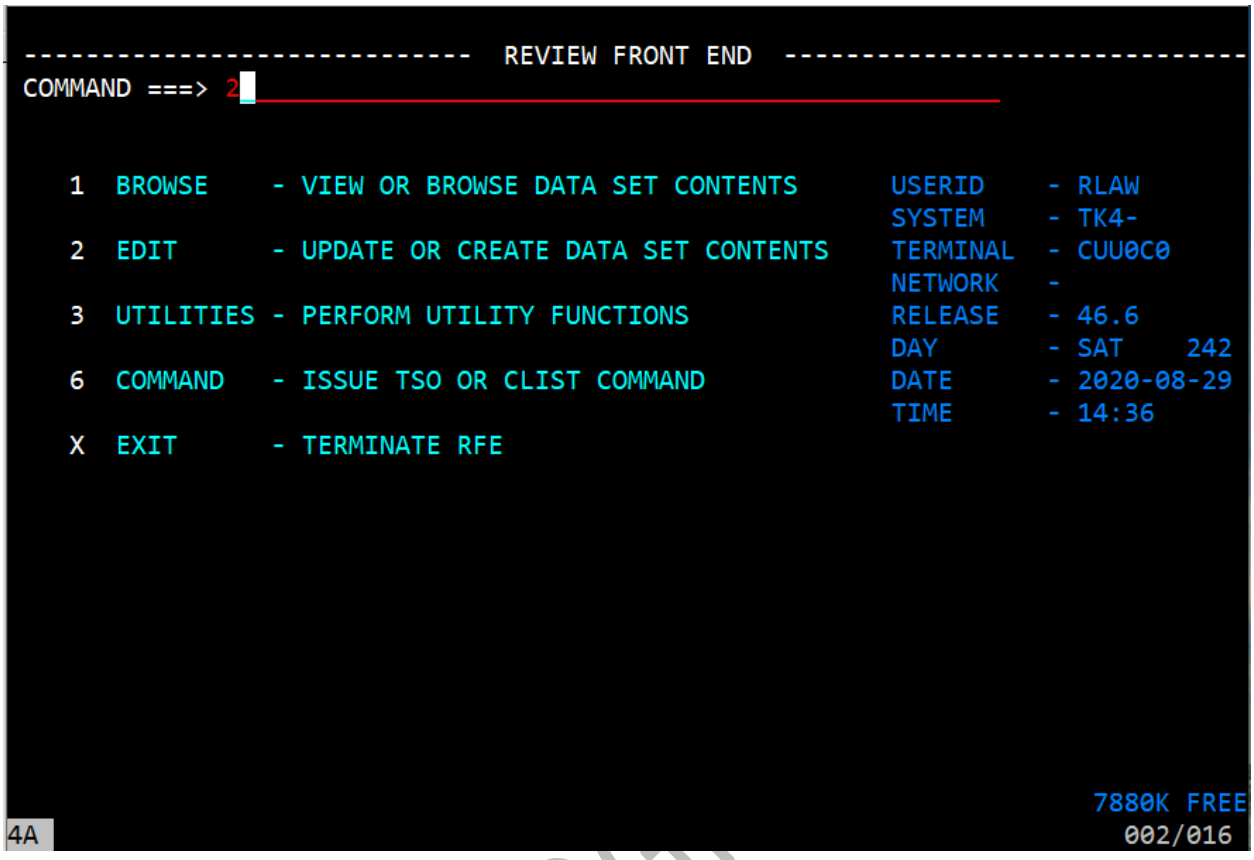


Nesta tela entre FB no formato de gravação. Dê-lhe um tamanho de registro com 80 e um múltiplo de 80 no tamanho do bloco físico. A alocação pode ser em trilhos ou cilindros. O tamanho inicial é de 5 cilinders com um espaço secundário de 3 e número de blocos de diretório como 4.

Pressione enter.

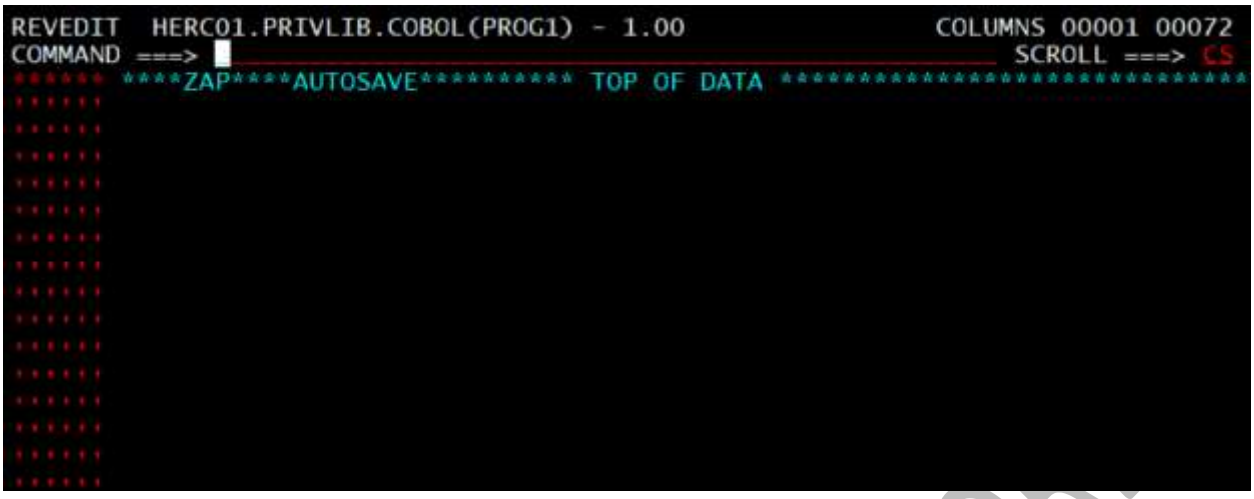


Após o enter é pressionado a tela do utilitário do conjunto de dados é exibida e no canto superior esquerdo ele dirá 'Data set created'.



Volte para o Review Front End (aperte F3 duas vezes) e selecione a opção 2 para EDIT.

No Painel editar você digita os valores que permitem ao programa identificar o conjunto de dados particionado que você vai editar. Digite os valores mostrados na imagem. Observe que há uma correlação entre 'HERC01.PRIVLIB.COBOL' que é o conjunto de dados particionado que foi criado. Você verá mais tarde que vamos referenciar HERC01.PRIVLIB.COBOL (PROG1). Depois de digitar os valores, pressione enter para trazer a tela de edição do membro.

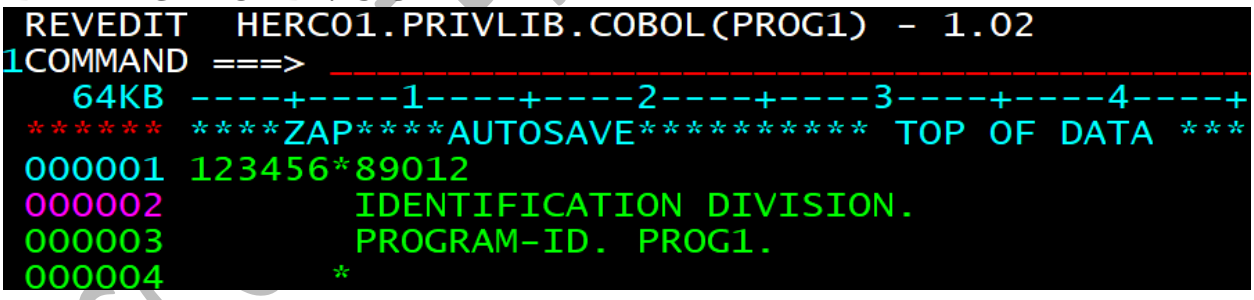


Quando o membro de origem é exibido pela primeira vez, ele está vazio e toda a tela está disponível para entrada. Quando a tecla enter é pressionada, as linhas que não têm nada nelas desaparecerão. A tela é dividida em três áreas. 1. A linha de comando, 2. Os números da seção/linhas de comando em cada linha individual (os seis pontos), 3. A área de entrada onde podemos escrever código.

Seções de código

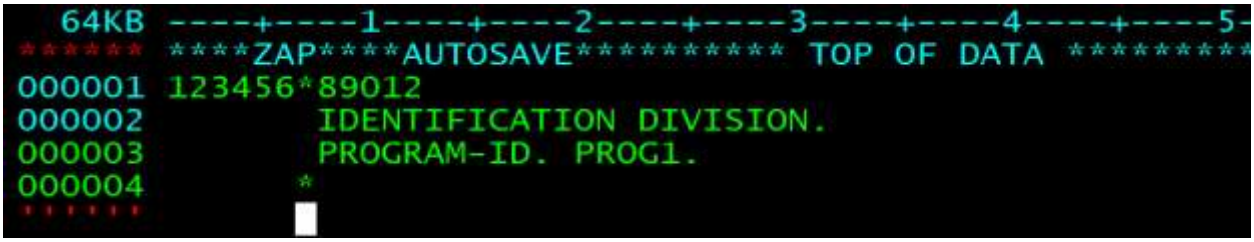
Cada programa COBOL é composto por quatro divisões, Divisão de Identificação, Divisão de Meio Ambiente, Divisão de Dados e Divisão de Procedimentos. Cada uma dessas divisões também pode ter seções, que têm parágrafos, que têm sentenças. Isso fará mais sentido à medida que avançamos.

IDENTIFICATION DIVISION



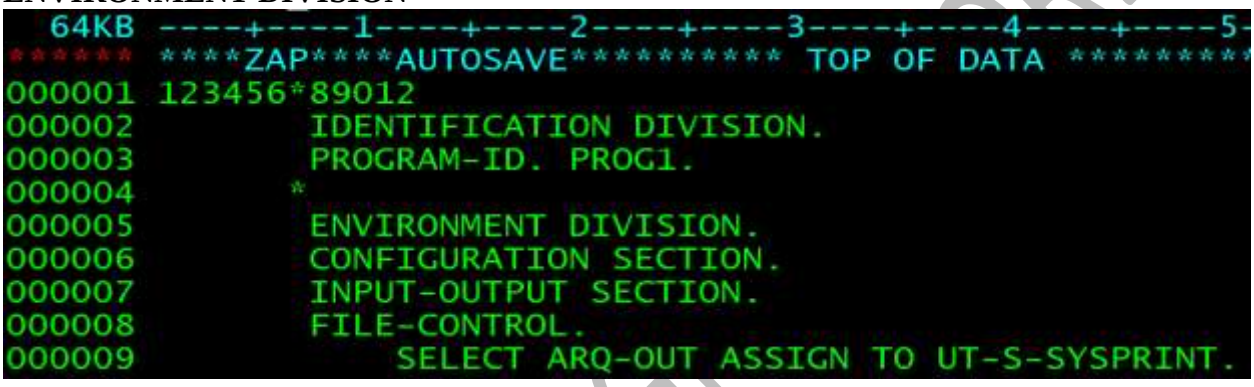
Na imagem acima a primeira linha é um comentário mostrando os números da coluna significativos para COBOL. Não é necessário, mas entrei lá para facilitar a escoar onde começam a coluna 8 e a coluna 12. O asterisco na coluna 7 significa que esta é uma linha de comentários. A linha a seguir diz "IDENTIFICATION DIVISION.". O seguinte diz "PROGRAM-ID. PROG1.'. Observe os PONTOS FINAIS, eles são necessários. Tanto a divisão de identificação quanto as linhas de identificação do programa são necessárias nesta versão do COBOL.

ESCOLA DE PROGRAMADORES



Na última linha digite 'i' e pressione enter. Isso abrirá uma única linha para entrada imediatamente após a linha em que você entrou no 'i' on. Se você inserir 'i10' ele vai inserir 10 linhas em branco.

ENVIRONMENT DIVISION



A **ENVIRONMENT DIVISION** descreve como o programa se comunica com as partes físicas do computador. Comece entrando em uma linha dizendo **ENVIRONMENT DIVISION**. Deve começar na coluna 8. A próxima linha é CONFIGURATION SECTION. Dentro da Seção há três parágrafos.

- SOURCE-COMPUTER.
- OBJECT-COMPUTER.
- SPECIAL-NAMES.

Todas as três linhas são opcionais. O SOURCE-COMPUTER e o OBJECT-COMPUTER são opcionais, mas se usados devem identificar o computador com uma frase como o IBM-370. Nomes especiais é um parágrafo que é usado quando você vai acessar certas coisas por outra palavra, por exemplo se queremos trocar o separador decimal de PONTO para VIRGULA.

```
SPECIAL-NAMES.
    DECIMAL-POINT IS COMMA.
```

ESCOLADEPROGRAMADORES

Após a CONFIGURATION SECTION, a **ENVIRONMENT DIVISION** tem outra seção que negocia arquivos chamado INPUT-OUTPUT SECTION. Esta seção tem que controlar parágrafos

INPUT-OUTPUT SECTION.
FILE-CONTROL.

Só cobriremos o FILE-CONTROL. O FILE-CONTROL é onde as definições dos arquivos usados no programa são inseridas. O programa que estamos escrevendo é referenciado no JCL como SYSPRINT. No programa, fazemos referência ao arquivo usando o RPT-OUT de trabalho, mas pode ser qualquer nome desejado. O formato é SELECT ASSIGN TO . A definição do arquivo é dividida em três seções. A primeira seção tem as seguintes definções:

- 'UR' - card reader, card punch, or printer
- 'UT' - tape drive, disk drive, (impressora também funciona
- 'DA' - Disk Drive - acesso direto, como ISAM ou VSAM

A segunda seção tem esses valores:

- 'I' - Indexado
- 'S' - Sequencial

Dadas as definiçoes acima, podemos dizer que o UT-S-SYSPRINT é uma fita, unidade de disco ou impressora que cujos registros são acessados sequencialmente, e são definidos ainda mais em uma instrução JCL DD que se chama SYSPRINT.

Para certos tipos de arquivo, como o ISAM, existem frases adicionais que são usadas na instrução selecionada. Eles serão cobertos quando escrevermos um programa usando o ISAM.

ESCOLADEPROGRAMADORES

DATA DIVISION

```
  64KB ----+----1----+----2----+----3----+----4----+----5----
****** ****ZAP****AUTOSAVE********** TOP OF DATA ************
000001 123456*89012
000002         IDENTIFICATION DIVISION.
000003         PROGRAM-ID. PROG1.
000004        *
000005         ENVIRONMENT DIVISION.
000006         CONFIGURATION SECTION.
000007         INPUT-OUTPUT SECTION.
000008         FILE-CONTROL.
000009             SELECT ARQ-OUT ASSIGN TO UT-S-SYSPRINT.
000010        *
000011         DATA DIVISION.
000012         FILE SECTION.
000013         FD  ARQ-OUT LABEL RECORDS OMITED.
000014         01  OUTREC.
000015             05 FIRST-NAME PIC X(20).
000016             05 FILLER     PIC X.
000017             05 LAST-NAME  PIC X(20).
000018             05 FILLER     PIC X.
000019        *
```

A DATA DIVISION tem três seções.

- FILE SECTION.
- LINKAGE SECTION.
- WORKING-STORAGE SECTION

Neste momento vamos cobrir FILE SECTION e WORKING-STORAGE SECTION.
LINKAGE SECTION será coberto quando escrevermos um sub-programa.

FILE SECTION é usado para descrever os arquivos que são definidos no FILE-CONTROL no Environment Division. Isso é feito por ter uma entrada FD.
A linha começa com 'FD' na coluna 8 seguido por dois espaços seguidos pelo arquivo definido noSELECT no FILE-CONTROL.
Isso é normalmente seguido por uma definação do LABEL RECORDS. Neste exemplo, não temos nenhum LABEL RECORDS por isso dizemos LABEL RECORDS OMITTED.

Em seguida, definimos o registro ou registros que existem no arquivo. Este disco é feito de itens de grupo. Neste exemplo, temos apenas um item de grupo e que é OUTREC. Outrec tem uma série de subitens que desde outrec é um nível 01 e é seguido por quatro 05 itens que são subitens. Cada item de grupo pode ser subdividido em subitens quando é vantajoso. Por exemplo, poderíamos ter o seguinte:

```
05 BIRTH-DATE.
   10 BIRTH-MONTH    PIC 99.
   10 BIRTH-DAY      PIC 99.
   10 BIRTH-YEAR     PIC 9(4).
```

ESCOLADEPROGRAM

Observe que o item do grupo não tem uma cláusula de imagem enquanto seus subitens devem ter uma cláusula de imagem. Em código, poderíamos referenciar BIRTH-DATE que seria um campo de oito dígitos. Nós também poderíamos referenciar BIRTH-MONTH, BIRTH-DAY, e BIRTH-YEAR como campos individuais.

WORKING-STORAGE SECTION

A seção WORKING-STORAGE é onde definimos outros campos e grupos de campos que precisamos no processamento do programa. Podemos definir campos individuais que não são subgrupados com um nível de identidade de '77'. Um nível 77 também não é um subgruão. Há também campos que têm um nível de nível de 66 e 88. 66 é usado em campos de redefinição e não será coberto. 88 é usado para um campo de condição e será coberto em um dos programas.

PROCEDURE DIVISION

```
000020          PROCEDURE DIVISION.
000021          MAIN-PART.
000022              OPEN OUTPUT ARQ-OUT.
000023              MOVE SPACES TO OUTREC.
000024              MOVE 'SILVIO' TO FIRST-NAME.
000025              MOVE 'SANTOS' TO LAST-NAME.
000026              WRITE OUTREC.
000027              CLOSE ARQ-OUT.
000028              GOBACK.
****** ****ZAP****AUTOSAVE********** BOTTOM OF DATA **
```

A PROCEDURE DIVISION é onde a lógica do programa é escrita.
As linhas seguintes consistem em parágrafos e frases. Um parágrafo começa na seção A da linha (colums 8-11) e termina com um período. Seguindo um nome de parágrafo, as frases são escritas. A largada na seção B (12-72). Estes normalmente começam com um verbo seguido por nomes de variáveis e outras palavras-chave.

Na imagem acima você pode ver um parágrafo chamado MAIN-PART. Esse parágrafo tem sete frases.
Na linha 22 diz OPEN OUTPUT ARQ-OUT. Ele abre o arquivo ARQ-OUT para saída. A próxima declaração, linha 23, move espaços para a variável OUTREC.
Linhas 24 e 25 movem SILVIO SANTOS para as variáveis FIRST-NAME e LAST-NAME.
A linha 26 escreve um registro, mas não um arquivo.
A linha 27 fecha o arquivo e torna inelegível para qualquer mais escrita.
A próxima linha de declaração 28 diz GOBACK.
Também pode dizer STOP RUN e ambos terminarão o programa. A diferença é que se este programa fosse chamado por outro programa COBOL GOBACK encerraria o subprograma e retornaria ao programa de chamada. STOP RUN encerra todo o programa COBOL.

Agora que temos um membro do conjunto de dados contendo nosso programa COBOL, não terminamos. Precisamos de uma maneira de compilar a fonte, vinculá-la a quaisquer módulos de link que sejam necessários e executar o programa resultante.

Exemplo de JCL com programa FONTE PROG1.

EXTERNAL SOURCE

```
 REVEDIT  HERC01.PRIVLIB.COBOL(JCLPROG1) - 1.01            COLUMNS 00001
1COMMAND ===> _                                              SCROLL =
   64KB ----+----1----+----2----+----3----+----4----+----5----+----6----+
****** ****ZAP****AUTOSAVE********** TOP OF DATA ************************
000001 //HERC01XX      JOB CLASS=A,MSGCLASS=H,NOTIFY=HERC01,MSGLEVEL=(1,1
000002 //STEP1         EXEC COBUCG
000003 //COB.SYSPUNCH DD DUMMY
000004 //COB.SYSIN     DD DSN=HERC01.PRIVLIB.COBOL(PROG1),DISP=SHR
000005 //COB.SYSLIB    DD DSNAME=SYS1.COBLIB,DISP=SHR
000006 //GO.SYSPRINT   DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=80,BLKSIZE=80)
000007 //
****** ****ZAP****AUTOSAVE********* BOTTOM OF DATA *********************
```

Quando o JCL e a fonte COBOL estiverem em conjuntos de dados separados, o COB. SYSIN DD é semelhante ao seguinte:

//COB.SYSIN DD DSN=HERC01.PRIVLIB.COBOL(PROG1),DISP=SHR

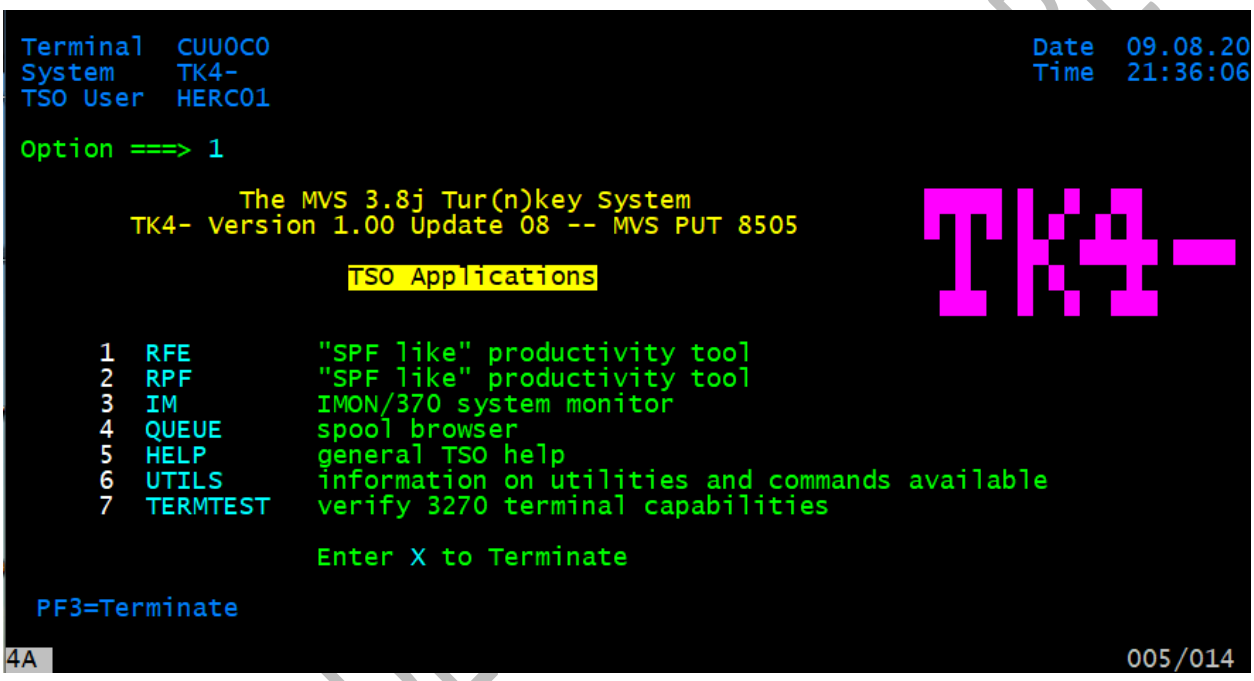
A vantagem de ter o JCL separado da fonte COBOL é que há apenas uma cópia do JCL e é reutilizável com a alteração do COB.SYSIN DSN .
Além disso, se o programa COBOL era grande, como muitos deles são é mais fácil ter o programa COBOL separado e os passos para construir seriam padrão para todos os programas.

ESCOLADEPROGRAMADORES

EXTRA - Adicionando um usuário ao sistema

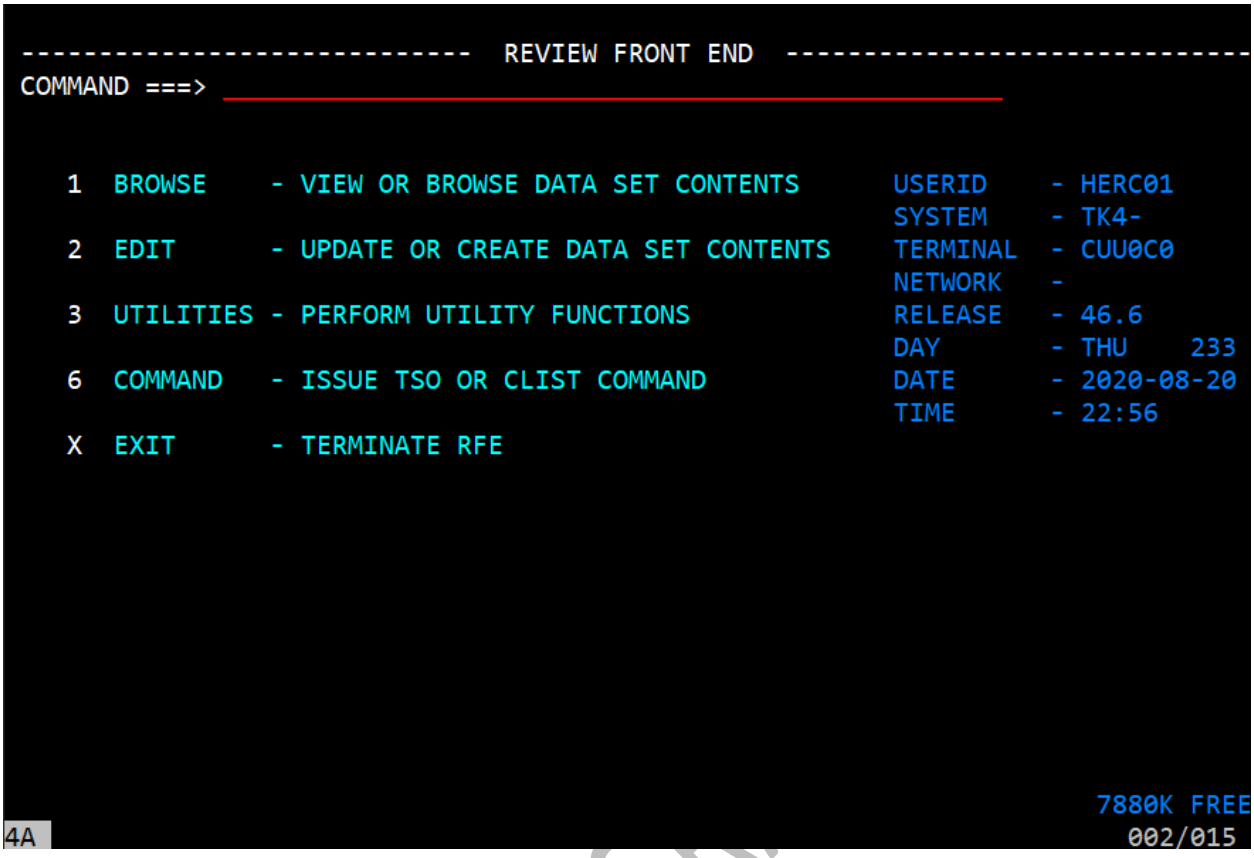
Você está fortemente desencorajado de usar os ids HERC01 ou HERC02 como seu id de usuário. Usar essas ids permite que você estrague seu sistema. É uma boa ideia criar uma identificação de usuário para o seu uso. Isso também cria uma série de conjuntos de dados para você usar.

O TK4- lançamento do MVS 3.8J tem um sistema de segurança chamado RAKF. É uma versão simplificada do sistema de segurança RACF da IBM. Para que um usuário seja adicionado ao sistema, os arquivos de segurança precisam ser modificados. Para fazer este sinal como HERC01 e quando você chegar ao menu principal inicie rfe.

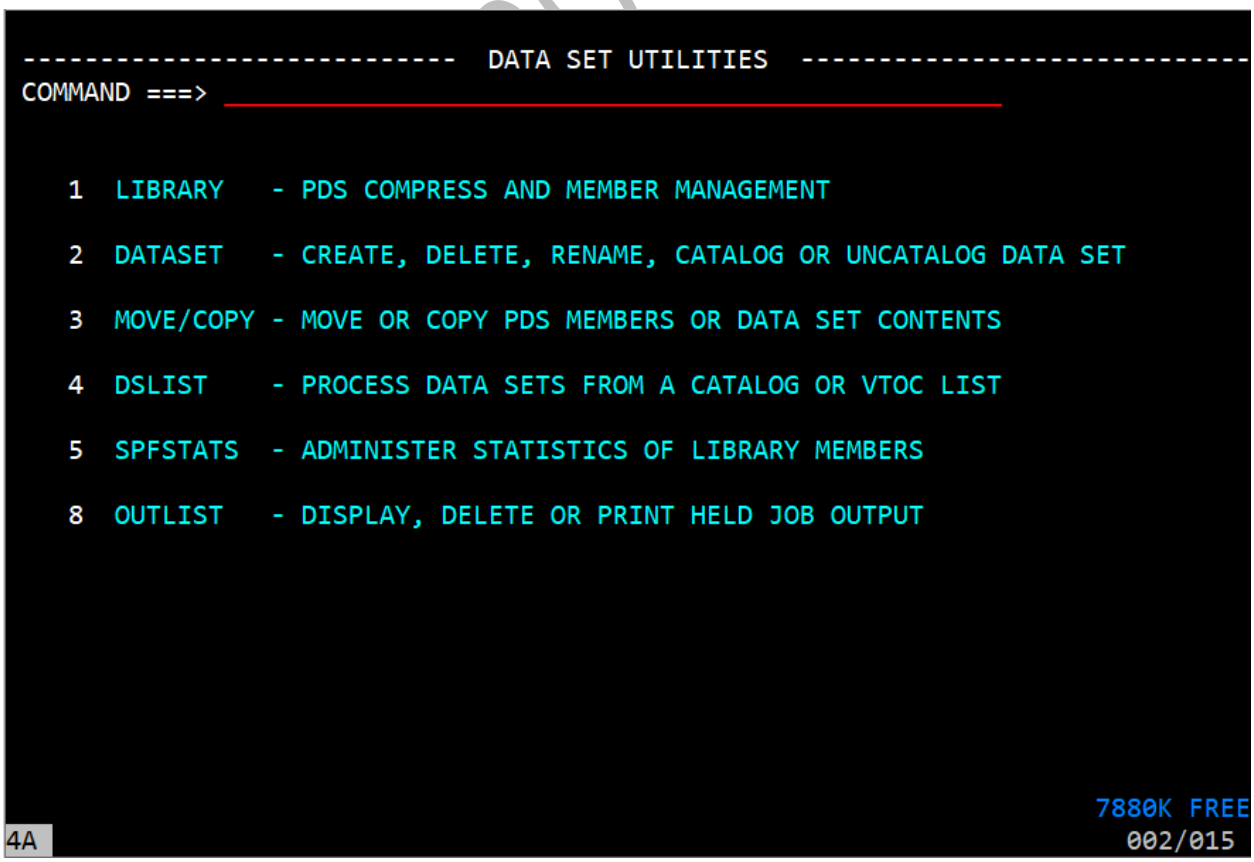


Selecione a opção 1 e pressione a tecla Enter. Isso vai trazer o menu RFE.

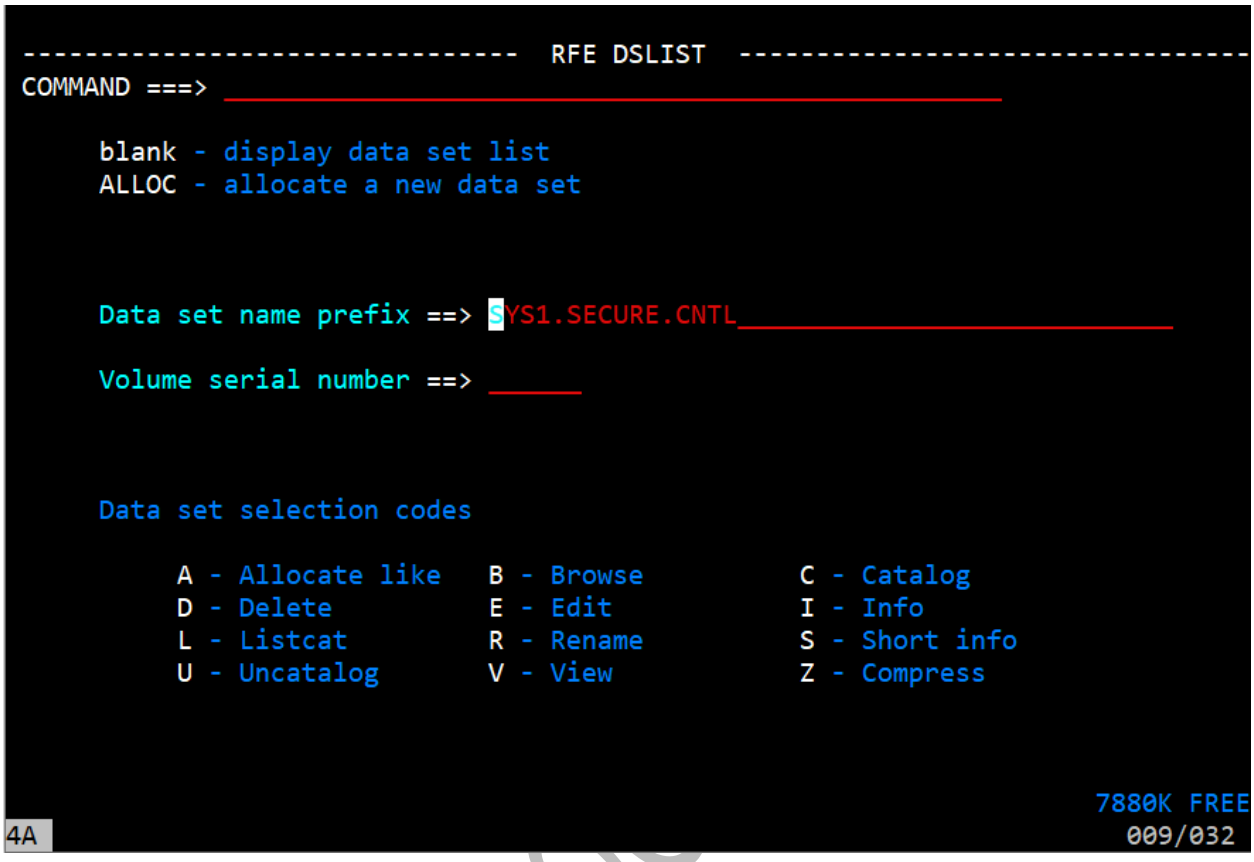
ESCOLADEPROGRAMADORES



Selecione a opção 3 para trazer o menu Data Set Utilities.

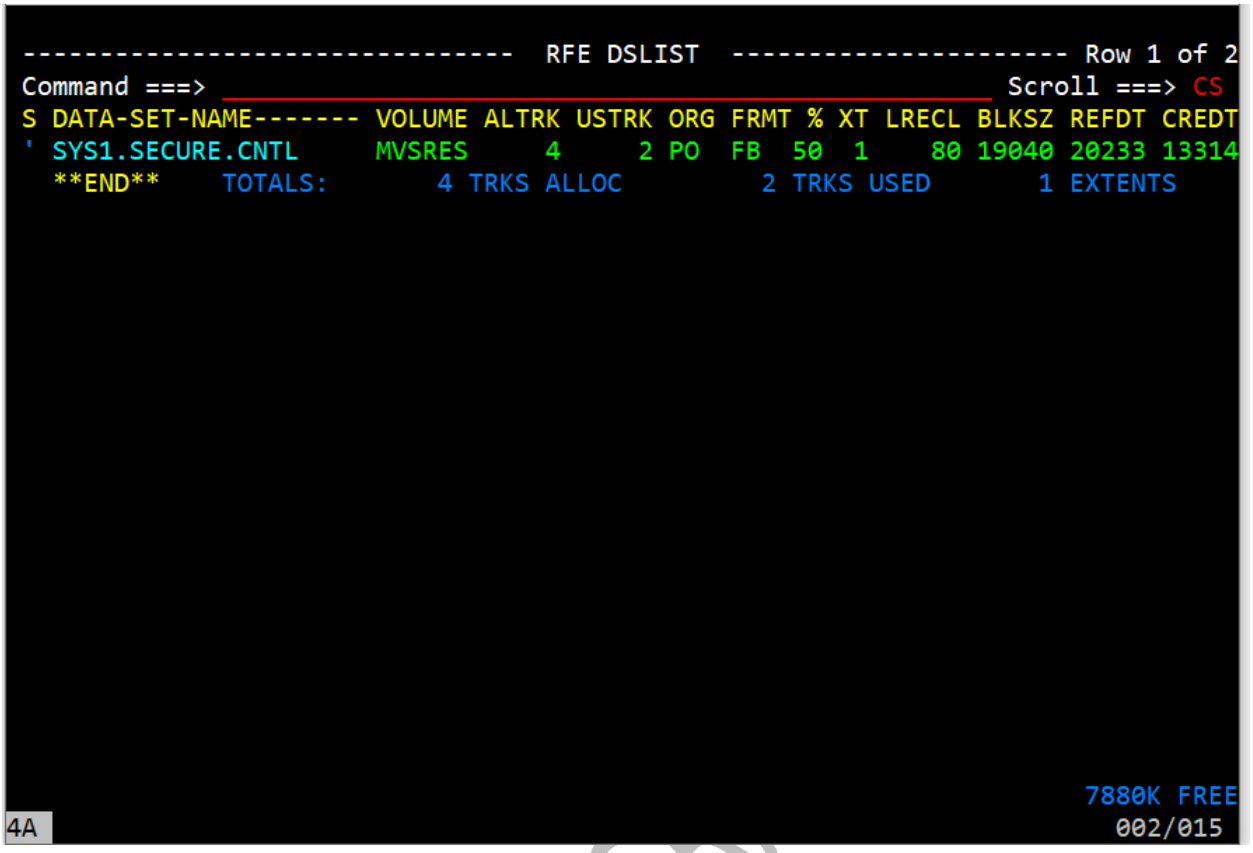


Selecione a opção 4 DSLIST e pressione a tecla enter. Isso vai trazer à tona o RFE DSLIST



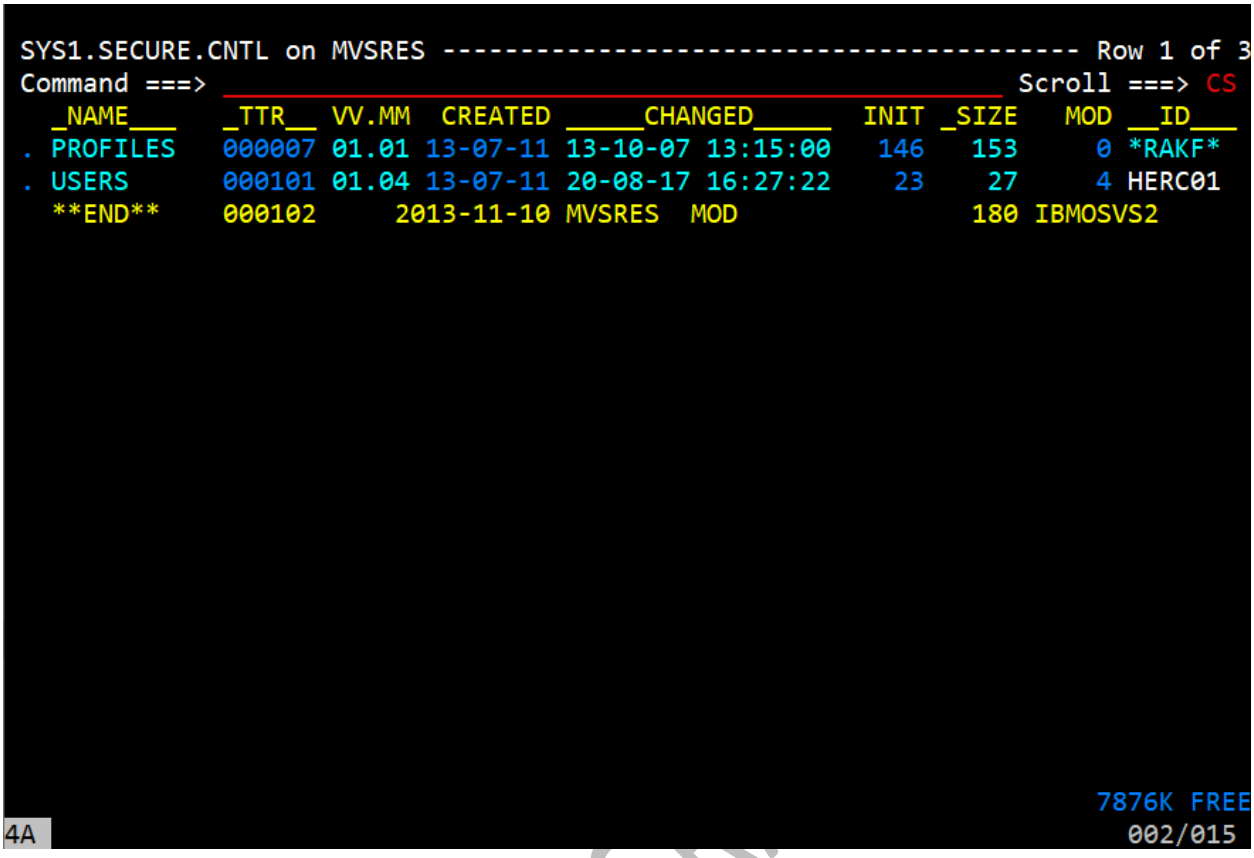
No campo prefixo de nome do conjunto Data, digite SYS1.SECURE.CNTL e pressione a tecla enter. Isso vai causar o RFE DSLIST mudar para uma visão de tudo que corresponde à entrada.

ESCOLADEPROG



Como o conjunto de dados específico é o nome, essa é a única entrada na lista. Posicione o cursor para o campo de seleção na frente do nome e digite um 'E'. Em seguida, pressione a tecla enter.

ESCOLADEPROGR



Isso causará uma tela que mostra os membros que estão no conjunto de dados. Selecione o membro dos usuários e pressione a tecla Enter.

ESCOLADEPROGR

```
REVEDIT  SYS1.SECURE.CNTL(USERS) - 1.04                 Columns 00001 00072
Command ===>                                              Scroll ===> CS
****** ****Zap****Autosave********** Top of Data ******************************
000001 ********************************************
000002 *                                            *
000003 * RAKF user definitions for TK4-             *
000004 *                                            *
000005 ********************************************
000006 *
000007 * HERC01 is authorized for everything including RAKF update
000008 *
000009 HERC01   ADMIN   *CUL8TR   Y
000010 HERC01   RAKFADM  CUL8TR   Y
000011 *
000012 * HERC02 is authorized for everything except RAKF update
000013 *
000014 HERC02   ADMIN    CUL8TR   Y
000015 *
000016 * herc03 and HERC04 are regular users
000017 *
000018 HERC03   USER     PASS4U   N
000019 HERC04   USER     PASS4U   N
000020 *
000021 * IBMUSER is authorized for everything except RAKF update
4A                                                          002/015
```

O arquivo de usuários é um arquivo de formato fixo onde as linhas devem estar em ordem crescente. Linhas de comentários implorando com um asterisco '*' na coluna 1. Aqui está o formato:

1 - 8 USERID
10 - 17 User Group
18 Multiple user Groups
19 - 26 Password
28 Operations Authority
31 - 80 Unused by RAKF

Observe que o HERC01 tem duas linhas no arquivo. A primeira linha tem um asterisco na coluna 18. Isso significa que há outra linha que adiciona e adiciona grupo de usuários ao HERC01. Isso significa que o HERC01 pode atuar em uma posição ADMIN e RAKFADM.

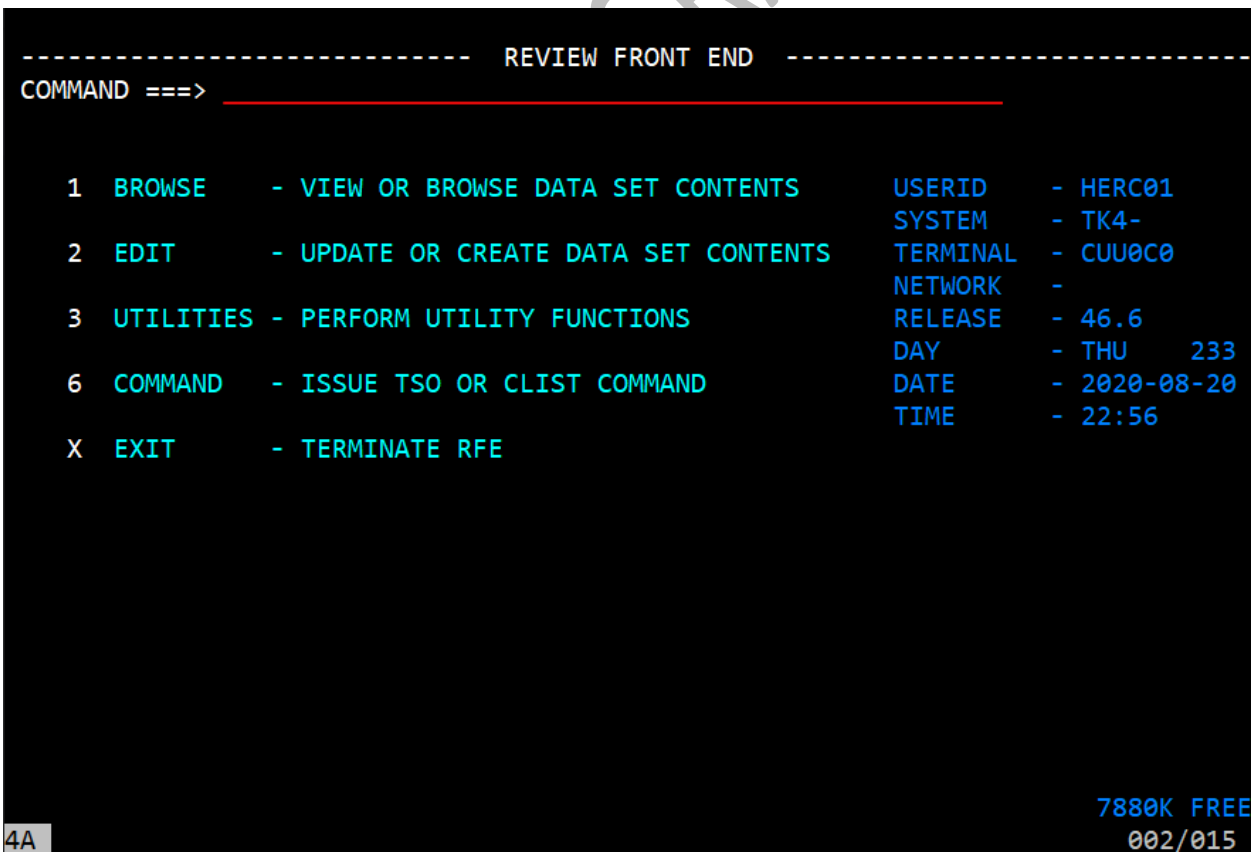
Olhando para HERC03 e HERC04, seu grupo de usuários é USUÁRIO. O USUÁRIO pode trabalhar com seus próprios arquivos, mas não olhar para o conteúdo dos arquivos de administração.

ESCOLADEPROGRAMADORES.COM.BR

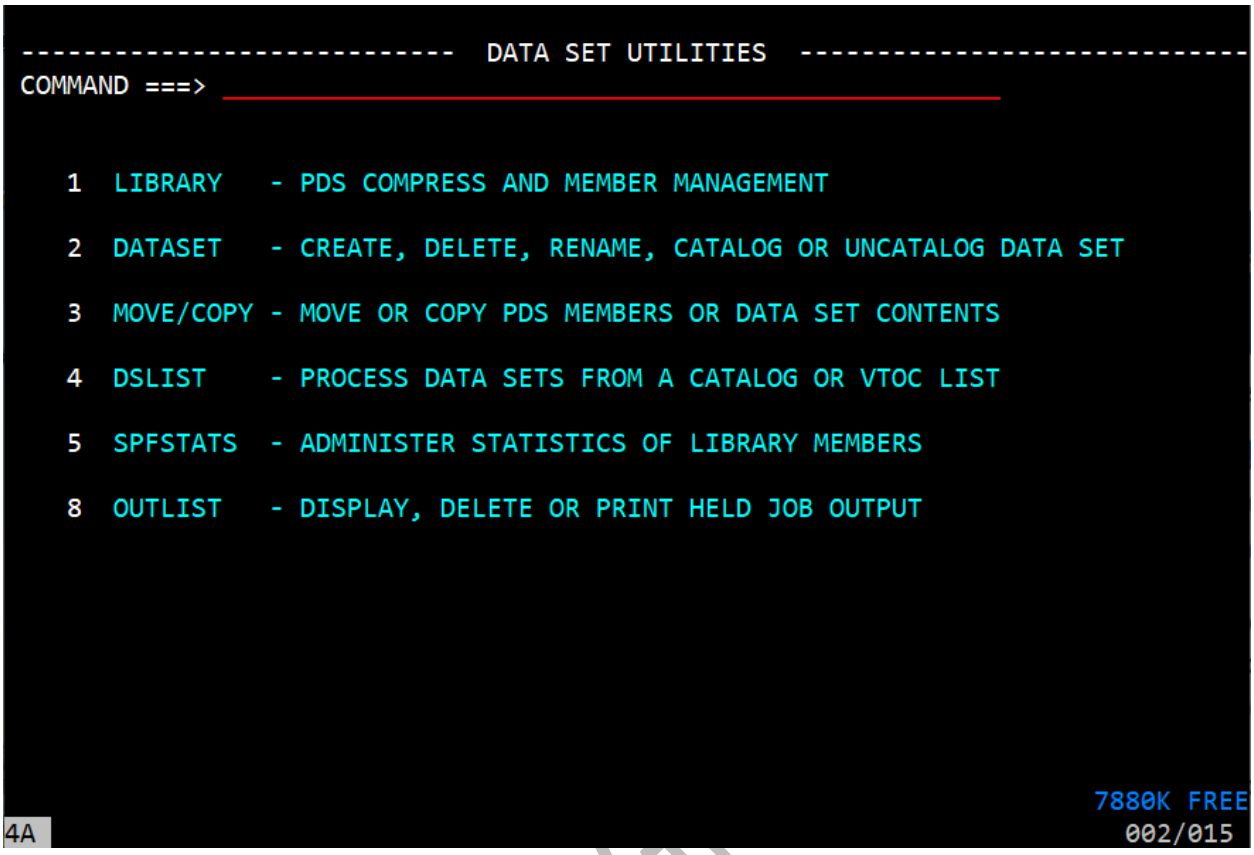
Para adicionar um novo usuário no arquivo do usuário RAKF, execute as seguintes etapas.

1. Posicione seu cursor para a linha do id do usuário que é menor alfabeticamente do que o usuário que você está adicionando e antes do id do usuário que é mais alfabeticamente do que o usuário que você está adicionando. Por exemplo, se um usuário GEORGE estava sendo adicionado você colocaria o cursor em a linha antes HERC01. Se o usuário MICHELLE estivesse sendo adicionado, você o colocaria na linha onde o IBMUSER existe.
2. Digite um 'i' e pressione enter. Uma nova linha será aberta.
3. Nas colunas 1-8 digite o nome de usuário.
4. Nas colunas 10-17 entre no grupo de usuários. Neste ponto, atribua a todos os usuários o grupo de usuários User.
5. Nas colunas 19-26 digite a senha.
6. Na coluna 28 digite 'N'.
7. Posicione o cursor na linha Comando e digite 'salvar'.
8. Pressione f3 até voltar ao menu principal.
9. Vá para o console do Hércules(TK4) e digite /stop rakf.
10. Depois que a parada terminar de executar, digite /start rakf.

Depois de inserir o novo id de usuário no sistema RAKF, parado e reiniciado RAKF, você ainda precisa adicionar o usuário ao MVS. Isso é feito pelo seguinte:

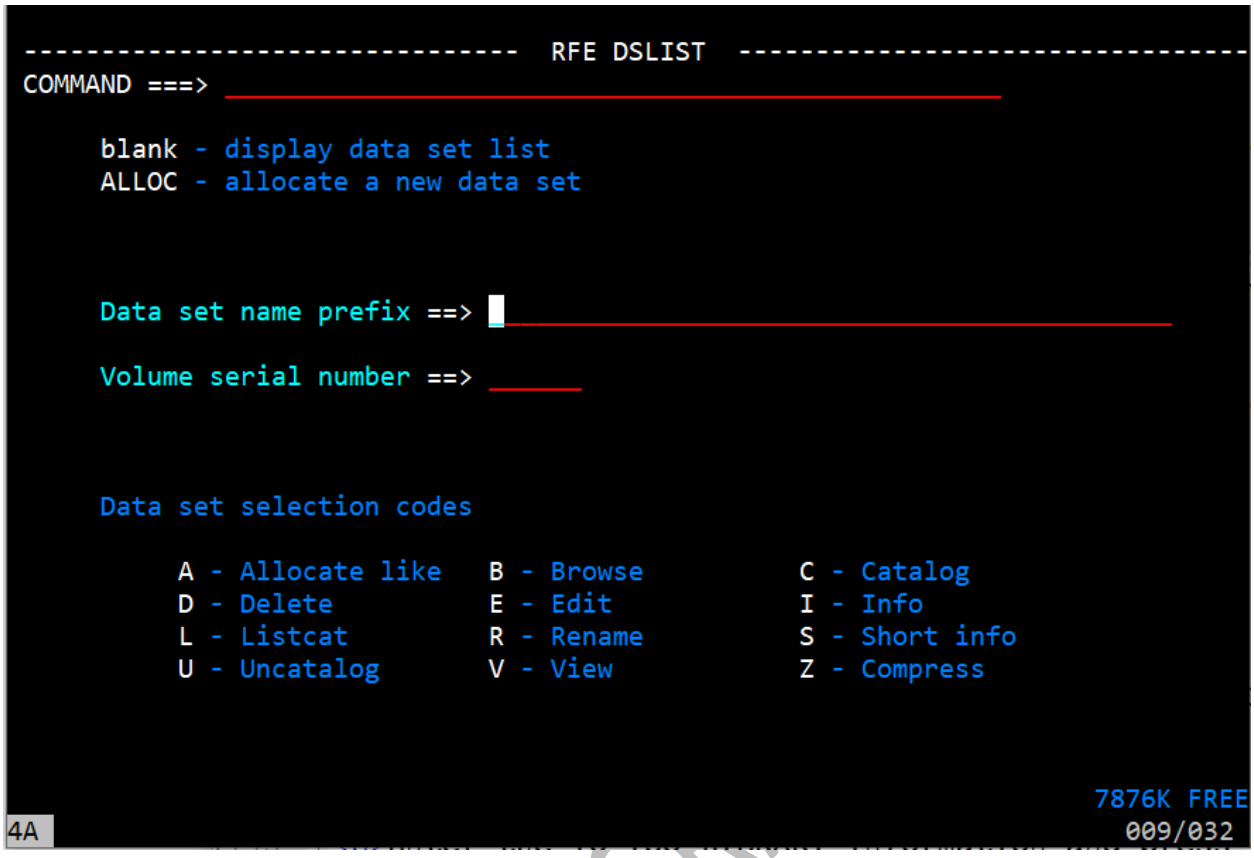


Traga o menu principal RFE e selecione UTILITÁRIOS (3) e pressione enter. Isso vai trazer o menu UTILITIES.



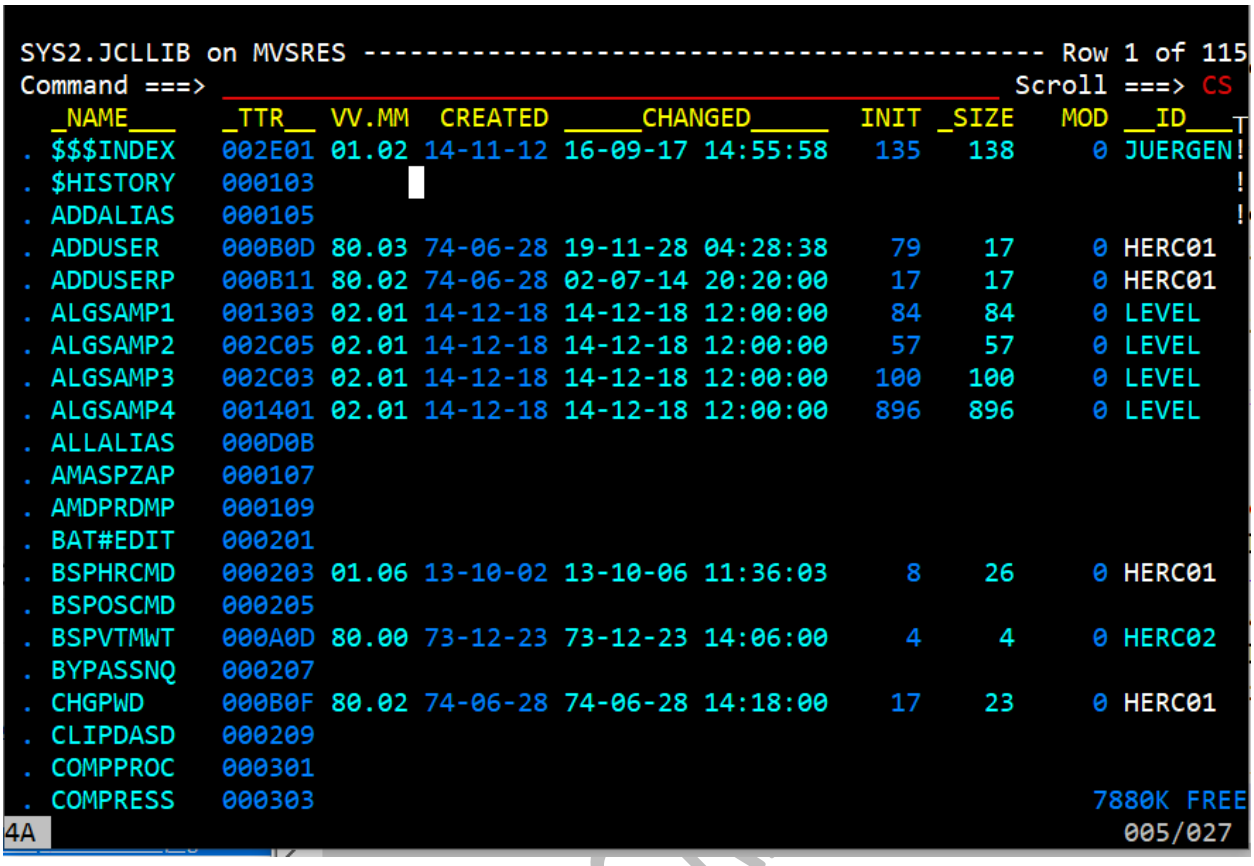
Selecione DSLIST (4) e pressione enter. Isso vai trazer à tona o painel RFE DSLIST.

ESCOLADEPROG



Digite 'sys2.jcllib' no campo prefixo de conjunto data e pressione enter. Isso dará um painel de resultados mostrando sys2. JCLLIB e que está no volume MVSRES. Posicione o cursor para essa linha e digitar 'E'. Isso trará um painel mostrando os membros no SYS2. Lista de membros da JCLLIB.

ESCOLADEPROGRAMADORES



Posicione o cursor para a linha ADDUSER e em seu campo de seleção digite 'E' e pressione enter. Isso trará a fonte para ADDUSER.

ESCOLADEPROGRAMADORES.COM.BR

```
REVEDIT  SYS2.JCLLIB(ADDUSER) - 80.03                  Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CS
****** ****Zap****Autosave********** Top of Data ******************************
000001 //ADDUSER JOB  (TSO),
000002 //             'Add TSO users',
000003 //             CLASS=A,
000004 //             MSGCLASS=H,
000005 //             COND=(0,NE),
000006 //             MSGLEVEL=(1,1)
000007 //*********************************************************************
000008 //*
000009 //* Name: SYS2.JCLLIB(ADDUSER)
000010 //*
000011 //* Desc: Add new TSO users
000012 //*
000013 //* Note: The newly created userids do not have a password set
000014 //*
000015 //*********************************************************************
000016 //*
000017 //ADDUSER EXEC ADDUSER,HLQ=RLAW,UTYPE=USER or SYSP
****** ****Zap****Autosave******** Bottom of Data ****************************


                                                          7788K FREE
4A                                                         008/047
```

Na linha 17 (//ADDUSER EXEC ADDUSER) modifique o campo após "HLQ=" para ser o nome de usuário adicionado no RAKF. Certifique-se de que o campo UTYPE seja USER.

Posicione-se em COMMAND => e digite SAVE, pressione enter. Em seguida, digite 'SUB' para submeter este trabalho. Isso submeterá o trabalho e executará

Agora você deve ser capaz de fazer logon no HERC01 e fazer login como usuário que acabou de adicionar.

ESCOLADEPROGRAMADORES.COM.BR

EXTRA - Programa FORTRAN

Fortran foi um dos primeiros compiladores. Foi criado em 1957 pela IBM e a versão foi feita por praticamente todos os fornecedores de sistemas operacionais. A primeira versão padronizada do Fortran foi em 1966 e batizada de FORTRAN IV. A IBM criou algumas versões do Fortran IV chamada Fortran G e Fortran H. Eram para máquinas com memória diferente. MVS 3.8J tem ambos os compiladores. Por muitos anos, fortran foi fortemente usado em ciências, engenharia e matemática. Não é mais usado tão fortemente como no passado.

Vamos olhar um código Fortan:

Uma linha de cobol consiste em três seções diferentes. As posições 1 - 6 são para um id de linha. Se a posição 7 não estiver em branco, então a linha é uma continuação da linha anterior. As posições 8 - 72 são para código.

```
000001 110    FORMAT( ' NAME IS ' 20A1)
000002 120    FORMAT(20A1)
000003        LOGICAL*1 NAME(20)
000004        READ(5,120) (NAME(I),I=1,20)
000005        WRITE(6,110) NAME
000006        STOP
000007        END
```

1	Esta linha tem um número de identificação que permite que a linha seja referenciada em determinadas declarações. A parte de código é a instrução Formato que está definindo uma linha de impressão. O ' NOME É ' é uma constante e é seguido pelo 20A1, o que significa que ele exibirá 20 campos alpanumericos de 1 personagem.
2	Esta linha também tinha um número de identificação e é uma declaração de formato que define 20 campos alfanuméricos com 1 caractere de comprimento.
3	Isso define um campo LÓGICO que tem 1 caractere de comprimento e é uma matriz de 20.
4	Esta instrução é lida a partir do dispositivo 5 usando o formato definido em id 120. Ele lê no nome da matriz começando no índice 1 e passando pelo índice 20. Isso é feito desta forma para que ele possa ler um nome de 20 caracteres a partir da entrada sysin (dispositivo 5). Os Compiladores Fortran IV não tinham um bom suporte para campos de corda e a definição do nome como 20 1 campos de charcter permite que o compilador essencialmente false uma string.
5	Esta instrução é adicionada no dispositivo 6 usando o formato do id 110 e saídas da matriz de nomes. O dispositivo 6 é definido no JCL que executa este código.
6	Diz ao programa para parar a execução.

ESCOLADEPROGRAMADORES

JCL com programa FONTE incorporado

```
//HERC01XX JOB (FORTRAN),CLASS=A,MSGCLASS=H,NOTIFY=HERC01
//FIRSTPGM  EXEC FORTHCLD,REGION.FORT=384K,PARM.FORT='LIST'
//FORT.SYSIN DD *
  110   FORMAT( ' NAME IS ' 20A1)
  120   FORMAT(20A1)
        LOGICAL*1 NAME(20)
        READ(5,120) (NAME(I),I=1,20)
        WRITE(6,110) NAME
        STOP
        END
/*
//GO.FT06F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=13200)
//GO.SYSIN DD *
SILVIO SANTOS
//
```

JCL com programa FONTE no DSN LEARN.FORTRAN.SRC (precisa criar o dataset e membro)

```
//RLAWFRT2 JOB (FORTRAN),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1)
//FIRSTPGM  EXEC FORTHCLD,REGION.FORT=384K,PARM.FORT='LIST'
//FORT.SYSIN DD DSN=LEARN.FORTRAN.SRC(FIRSTPGM)
//GO.FT06F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=13200)
//GO.SYSIN DD *
SILVIO SANTOS
//
```

GO. FT06F001, o 6 indica que esta é a linha DD para a saida do programa.

ESCOLADEPROGRAMADORES