JCL: Fazendo as coisas acontecerem

Código de Condição: Herói

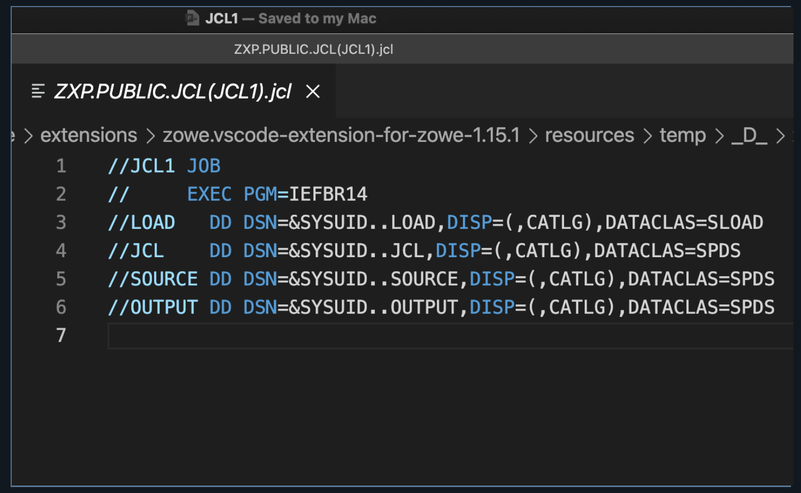
15 passos 90 minutos

O DESAFIO

Você já viu algum JCL, mas não nos aprofundamos muito. Nesses desafios, aprenderemos um pouco mais sobre para que o JCL é utilizado, por que ele é importante em um ambiente Z e como você pode levar essas habilidades ainda mais longe. JCL é uma parte essencial para fazer as coisas acontecerem no z/OS e ficar confortável com os conceitos e sintaxe permitirá que você supere muitos desafios em sua jornada.

ANTES DE VOCÊ COMEÇAR

Você deve ter concluído o desafio DSM1, tudo sobre Data Sets e Membros. Se você entendeu esses conceitos, está tudo pronto para continuar com as etapas deste desafio.

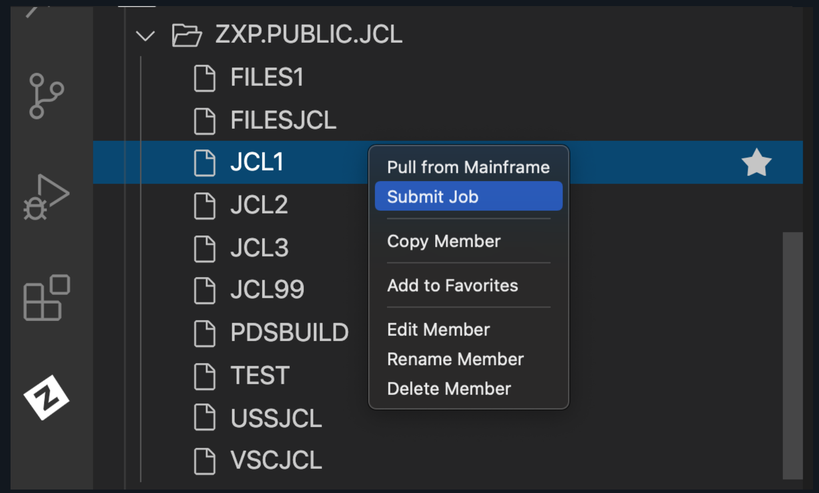


1. CARREGUE-O

Procure em ZXP.PUBLIC.JCL por um membro denominado JCL1. Este é um trabalho bastante simples que alocará alguns novos data sets necessários para este e outros desafios.

Você pode ver na linha #3, ela menciona & SYSUID..LOAD. O "e" comercial (&) com SYSUID depois dele é o que é conhecido como um *Symbolic* (Simbólico) e, quando o sistema vê isso, ele substitui &SYSUID com seu ID de usuário.

Isso significa que todos podem usar o mesmo trabalho, e ele substituirá automaticamente &SYSUID por sua ID de usuário. Que conveniente!

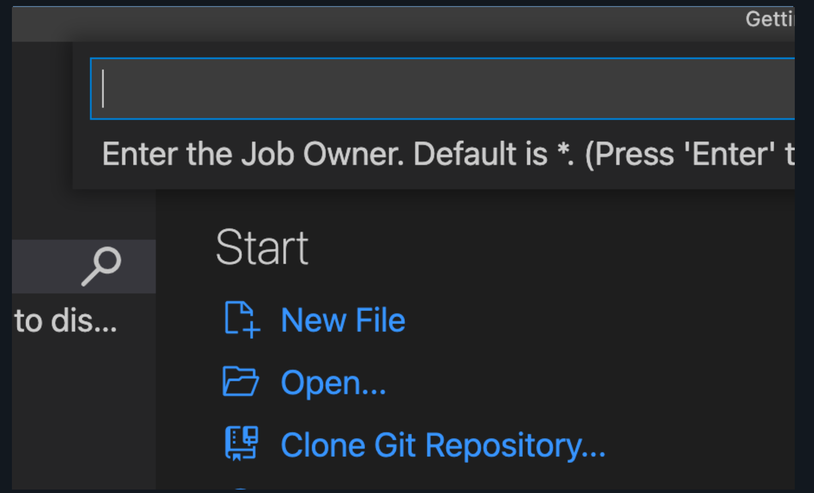


2. ENVIE-O

Clique com o botão direito no job e selecione *Submit Job*.

Vamos falar um pouquinho sobre *"Job"* (trabalho, tarefa...): o JCL é usado para descrever para o sistema exatamente o que você gostaria que ele fizesse. A tarefa que entregamos ao sistema é conhecida como Job, e o componente do z/OS que aceita, processa e gerencia a saída desses jobs é conhecido como JES, o *Job Entry Subsystem* (Subsistema de entrada de trabalho).

Portanto, para este desafio, enviamos um trabalho para o JES para que ele processe as tarefas que examinamos no 1º passo.

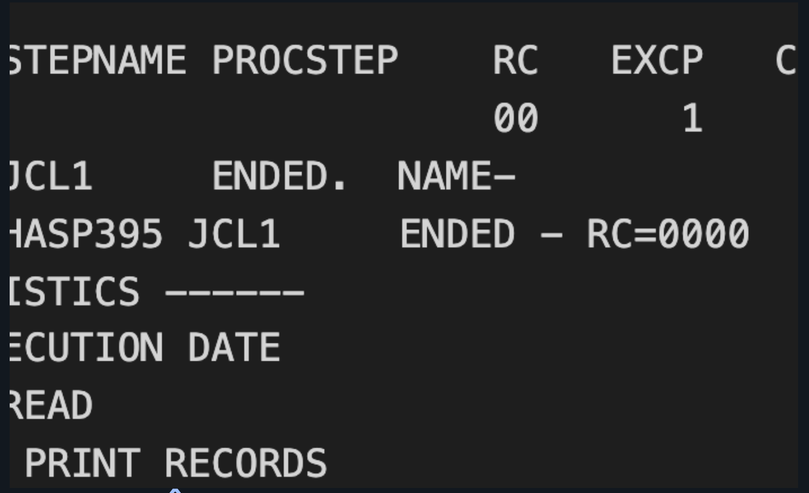


3. FILTRE E O ENCONTRE

Você já deve ter um perfil em JOBS no lado esquerdo do VS Code. Clique na lupa (🔎) à direita dele.

Insira seu ID de usuário para o *Job Owner* (Proprietário do Job)
Digite um asterisco (*) para o *Job Prefix* (Prefixo do Job)
Pressione Enter (em branco, sem dados) para *Job Id Search* (Busca do id do job).

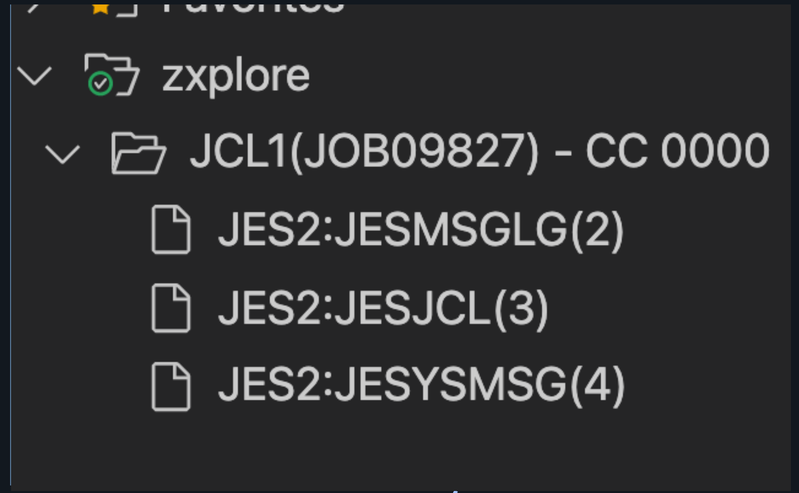
Você pode refazer este filtro clicando na lupa novamente e selecionando *Owner/Prefix* ou *Job Id*. Você deve conseguir encontrar o job que acabou de enviar aqui. Vamos nos aprofundar nisso a seguir.



4. VOCÊ TIROU ZERO. PERFEITO!

Clique na seta para expandir *"twistie"* que está junto ao job JCL1 que você acabou de enviar. Provavelmente haverá outros jobs lá também, mas estamos procurando especificamente por JCL1. Se você enviou mais de uma vez, encontre aquele com **CC 0000** à direita.

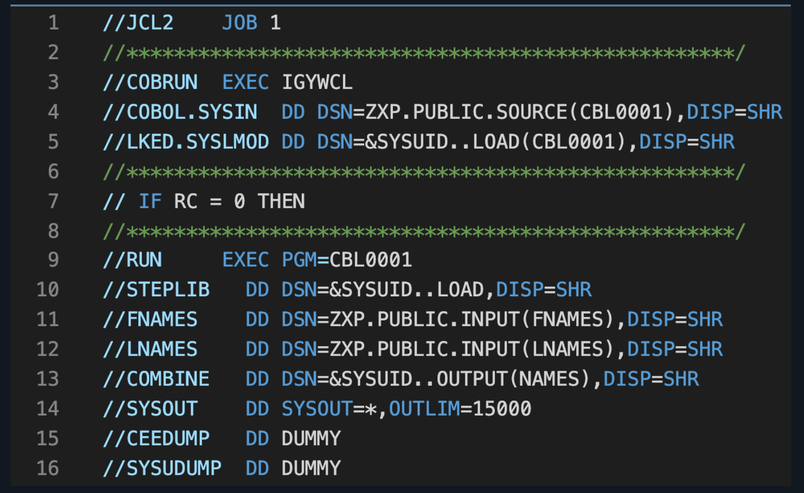
Você também verá esse número no membro JESMSGLG assim que abrir o twistie. Um *condition code* (CC – "código de condição") igual a zero significa que tudo foi executado conforme o esperado, sem erros, o que é bom! Se você tiver qualquer outro número, provavelmente há algo que vale a pena investigar e possivelmente consertar.



5. PULE DIRETO ATÉ ELE

Você deve ter notado que, após enviar o JCL, uma pequena mensagem aparece no canto inferior direito do VS Code. Em vez de vasculhar o *output* (saída) do JOBS, geralmente você pode apenas clicar na mensagem e ela o levará direto para o output.

Um job começará em ATIVO enquanto está sendo executado. Você pode atualizar o status de um trabalho fechando e reabrindo o twistie à esquerda do nome do job.



```
1  //JCL2    JOB 1
2  //*************************************************/
3  //COBRUN  EXEC IGYWCL
4  //COBOL.SYSIN  DD DSN=ZXP.PUBLIC.SOURCE(CBL0001),DISP=SHR
5  //LKED.SYSLMOD DD DSN=&SYSUID..LOAD(CBL0001),DISP=SHR
6  //*************************************************/
7  // IF RC = 0 THEN
8  //*************************************************/
9  //RUN     EXEC PGM=CBL0001
10 //STEPLIB   DD DSN=&SYSUID..LOAD,DISP=SHR
11 //FNAMES    DD DSN=ZXP.PUBLIC.INPUT(FNAMES),DISP=SHR
12 //LNAMES    DD DSN=ZXP.PUBLIC.INPUT(LNAMES),DISP=SHR
13 //COMBINE   DD DSN=&SYSUID..OUTPUT(NAMES),DISP=SHR
14 //SYSOUT    DD SYSOUT=*,OUTLIM=15000
15 //CEEDUMP   DD DUMMY
16 //SYSUDUMP  DD DUMMY
```

6. INICIANDO ALGUM COBOL

Copie o JCL2 de ZXP.PUBLIC.JCL para o seu data set ZXXXXX.JCL e, em seguida, abra-o a partir de seu data set JCL. Pode ser necessário ocultar e expandir na seta ao lado de DATA SETS para atualizar a visualização, então seu JCL aparecerá. Este JCL é usado para compilar e executar algum código COBOL. Após a compilação, ele colocará o programa resultante em seu data set LOAD.

Em JCL, as instruções que definem de onde os dados vêm ou vão são conhecidas como *Data Definition Statements* (Instruções de Definição de Dados) ou simplesmente *DD Statements*.

Você pode ver o data set de *input* (entrada) na Linha 4 e onde está sendo colocado o *output* na Linha 5.

Lendo um pouco mais, se obtiver um Código de Retorno de 0 (porque tudo correu sem problemas), ele executará o programa. Faz sentido até agora? Estamos usando JCL para compilar e, em seguida, executar alguns códigos.

"POR QUE O JES E O JCL SÃO IMPORTANTES? NÃO POSSO APENAS RODAR PROGRAMAS?"

Quando você submete o JCL, ele vai para o *Job Entry Subsystem* (abreviado para JES). O JES examina o JCL que você enviou e reúne todos os recursos necessários para realizar a tarefa. Em um sistema muito carregado, pode ser necessário atribuir uma prioridade alguns jobs abaixo de outros para que jobs importantes sejam realizados mais rapidamente.

Pense em JCL como o pedido que um garçom escreve, e JES como o pessoal da cozinha que olha para o pedido e decide como vai lidar com ele. O L em JCL significa Linguagem, mas realmente não é uma linguagem de programação, mas uma maneira de descrevermos com eficácia as tarefas do sistema.

Todo o resto que aparece na saída do trabalho (do passo 4) são informações sobre como o trabalho foi executado. Como você pode ver, há uma tonelada de informações aqui.

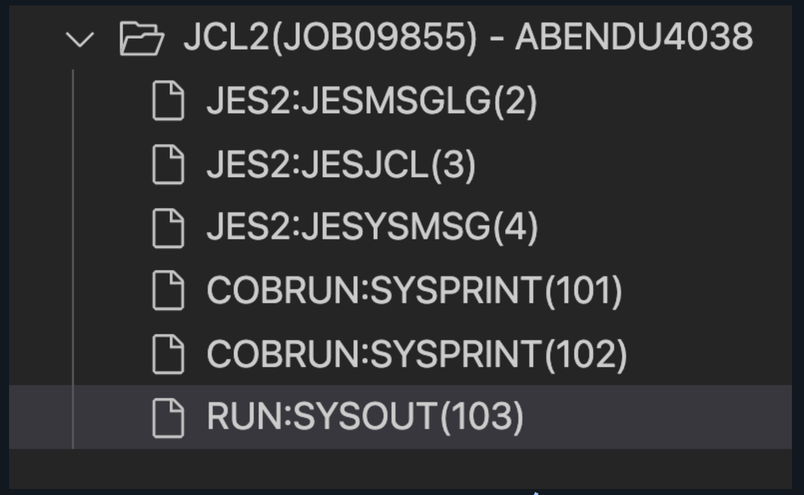
```
1  //JCL2    JOB 1
2  //*************************************************/
3  //COBRUN  EXEC IGYWCL
4  //COBOL.SYSIN  DD DSN=ZXP.PUBLIC.SOURCE(CBL0001),DISP=SHR
5  //LKED.SYSLMOD DD DSN=&SYSUID..LOAD(CBL0001),DISP=SHR
6  //*************************************************/
7  // IF RC = 0 THEN
8  //*************************************************/
9  //RUN     EXEC PGM=CBL0001
10 //STEPLIB   DD DSN=&SYSUID..LOAD,DISP=SHR
11 //FNAMES    DD DSN=ZXP.PUBLIC.INPUT(FNAMES),DISP=SHR
12 //LNAMES    DD DSN=ZXP.PUBLIC.INPUT(LNAMES),DISP=SHR
13 //COMBINE   DD DSN=&SYSUID..OUTPUT(NAMES),DISP=SHR
14 //SYSOUT    DD SYSOUT=*,OUTLIM=15000
15 //CEEDUMP   DD DUMMY
16 //SYSUDUMP  DD DUMMY
```

7. RODA, CÓDIGO!

Depois de compilar o código COBOL, ele executará o código (Linha 9) e dirá onde encontrar os data sets de input (Linhas 11-12), bem como onde colocar a saída (Linha 13).

O nome do DD statement é o que vem logo após as barras duplas, como FNAMES e LNAMES, por exemplo.

Todas as linhas que começam com //* são comentários e são ignoradas pelo JES. As linhas comentadas são úteis para fornecer informações ou conter linhas de código que podemos usar mais tarde, mas não precisamos agora.



8. NEM SEMPRE VEM SÓ ZERO

Envie o JCL2 de seu data set ZXXXXX.JCL e, em seguida, observe o output, usando o que você aprendeu nas etapas 1-5. Temos um ABEND (abreviação de *Abnormal End* – finalização anormal), então algo não deu certo.

Mas não se preocupe! Chegaremos ao fundo disso.

Na próxima etapa, veremos o código COBOL e veremos como o código COBOL real se compara ao código JCL que estamos usando para compilá-lo e iniciá-lo.

```
*-----------------------
 IDENTIFICATION DIVISION.
*-----------------------
 PROGRAM-ID.    NAMES
 AUTHOR.        Otto B. Named
*--------------------
 ENVIRONMENT DIVISION.
*--------------------
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT FIRST-NAME ASSIGN TO FNAMES.
     SELECT LAST-NAME  ASSIGN TO LNAMES.
     SELECT FIRST-LAST ASSIGN TO COMBINED.
```

9. COMPARE O CÓDIGO

A instrução no JCL que começa com //COBOL.SYSIN aponta para o código COBOL que desejamos compilar, portanto, vamos começar por aí. Abra esse código e comece a olhar para a área FILE-CONTROL.

É aqui que obtemos os nomes para o programa que precisamos fazer referência em nosso JCL. Por exemplo, FIRST-NAME é o que referimos no código COBOL e que é atribuído (ou conectado) à instrução FNAMES DD no JCL.

Abra o JCL e o código COBOL, observe o output e você deverá ser capaz de descobrir quais mudanças simples precisam ser feitas para que tudo fique alinhado. Na verdade, desenhar tudo em um pedaço de papel poderá ajudar.

Depois de corrigir o problema no JCL2, ele deve ser executado com CC=0 e você encontrará o output correto no data set OUTPUT. Esta tarefa pode te dar um trabalho de detetive!

"O QUE SIGNIFICA COMPILAR? O QUE É COBOL?"

COBOL é uma linguagem de programação usada em muitas instituições financeiras, de saúde e governamentais. Seu alto grau de precisão matemática e métodos de codificação simples o tornam um ajuste natural quando os programas precisam ser rápidos, precisos e fáceis de entender.

O código que é escrito por humanos precisa ser transformado em Código de Máquina *(Machine Code)* para ser executado como um programa. A compilação é uma etapa que realiza essa transformação. Nosso JCL tem duas etapas principais, compilar o código-fonte em código de máquina e, em seguida, executar o programa (código de máquina).

Este programa em particular também requer dois arquivos de entrada e grava em um arquivo de saída, portanto, especificaremos esses arquivos (data sets) no JCL também.

```
//*
//PEEKSKL EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
Peekskill - 41mi
//SYSUT2   DD DSN=&SYSUID..JCL3OUT,DISP=(MOD,PASS,DELETE),
//            SPACE=(TRK,(1,1)),UNIT=SYSDA,
//            DCB=(DSORG=PS,RECFM=FB,LRECL=80)
//*
//CORTLNDT EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
Cortlandt - 38mi
```

10. TODOS A BORDO

Copie o JCL3 de ZXP.PUBLIC.JCL em seu próprio data set JCL. Carregue-o e dê uma olhada no que há dentro. Você verá que este JCL contém várias etapas, com cada uma usando o programa IEBGENER para direcionar um ingrediente a um data set sequencial.
Há um cabeçalho, algumas informações (em inglês) da estação para as paradas entre Poughkeepsie, Nova Iorque e Grand Central Terminal na cidade de Nova Iorque, e algum texto sobre o horário de funcionamento.
Parece bastante simples... Vamos ver qual é a parte complicada.

“JOB OUTPUT? DD STATEMENTS? ME AJUDA A ENTENDER, POR FAVOR...”

Quando um job roda, ele produz output (saída). Isso pode incluir os dados que você está procurando, os motivos pelos quais um job não foi executado com sucesso ou talvez apenas informações sobre o sistema e as etapas necessárias para fazer o trabalho acontecer. Há muitas informações aqui que você provavelmente *não* precisa, mas é sempre melhor tê-las do que ficar confuso sobre o estado de um job importante.

Uma grande parte de seu JCL são as instruções DD, que especificam quais data sets (e membros) usar para o input e output da tarefa que você está enviando ao sistema. O DD na instrução DD significa Data Definition e eles começam com //DD. Eles irão especificar o nome do data set (ou membro), se já existe ou precisa ser criado (conhecido como disposition), onde a saída deve ir, quanto espaço deve ocupar e uma série de outras variáveis.

```
//JCL3     JOB
//*
//* IEBGENER is a system utility program to copy data
//*  where the default input  filename is SYSUT1
//*  and   the default output filename is SYSUT2
//*
//HEADER EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
********************************************
METRO NORTH POUGHKEEPSIE -> NYC M-F SCHEDULE
PEAK HOUR OPERATION
********************************************
```

11. VOCÊ NÃO É DUMMY

Você também deve ter notado muitas menções a DUMMY. Não se preocupe, este JCL não está ofendendo ninguém, é apenas uma forma de dizer "Este parâmetro é obrigatório, mas não faremos nada com ele, então não importa".

Nós o usamos aqui porque o programa que estamos executando em cada etapa (IEBGENER) requer uma instrução de entrada e exige uma instrução SYSPRINT, mas não faremos uso dela, então DUMMY é uma maneira de dizer "Este não importa, não perca seu tempo configurando isso".

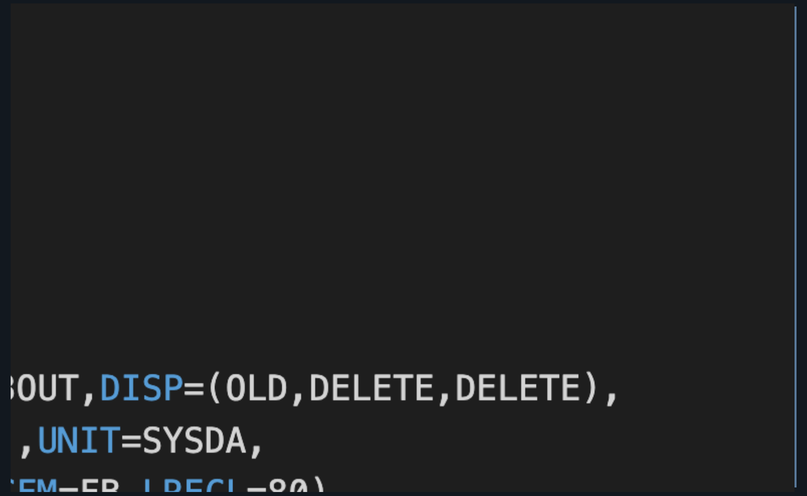
```
T,DISP=(MOD,PASS,DELETE),
NIT=SYSDA,
=FB,LRECL=80)
```

12. UMA DISPOSIÇÃO AMIGÁVEL

Em cada parte da JCL que usamos até agora, encontramos algum tipo de parâmetro DISP (*disposition*, ou disposição). Os parâmetros DISP são usados para descrever como o JCL deve usar ou criar um data set e o que fazer com ele após a conclusão do job.

Um parâmetro DISP padrão tem três partes. O primeiro parâmetro é o status, que pode ser qualquer um dos seguintes:

NEW Cria um novo data set
SHR Reutiliza um data set existente e permite que outras pessoas o usem, se quiserem
OLD Reutiliza um data set existente, mas não deixa que outros o usem enquanto o estamos usando
MOD Apenas para conjuntos de dados sequenciais. Reutiliza um data set existente, mas apenas anexa novos registros na parte inferior dele. Se nenhum data set existir, cria um novo.

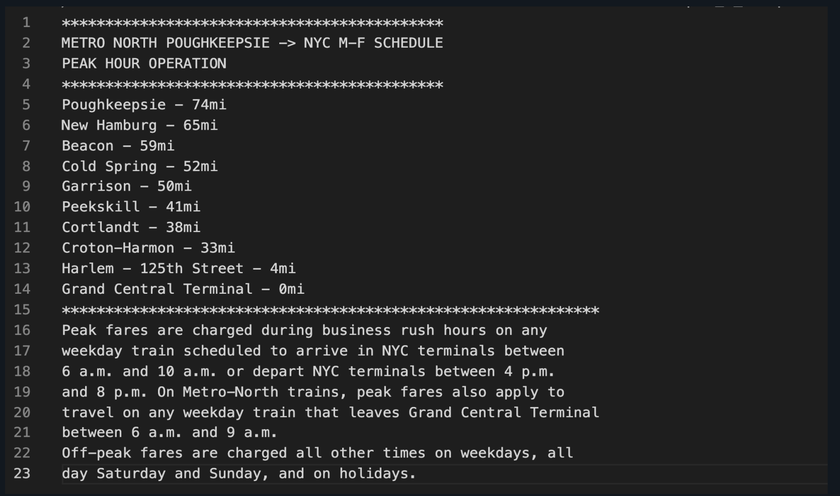


13. QUAL O SEU STATUS

O campo 2 do parâmetro DISP descreve o que deve acontecer com o data set no caso de um preenchimento normal, e o terceiro campo é o que deve acontecer com ele em caso de falha.

Existem vários valores que podemos usar aqui, mas para este desafio, você só precisa saber o seguinte:

DELETE Apaga ele completamente do armazenamento
CATLG Registra o data set para que possamos usá-lo mais tarde
PASS Depois que esta etapa for concluída, segure-o para que as etapas posteriores a esta possam usá-lo



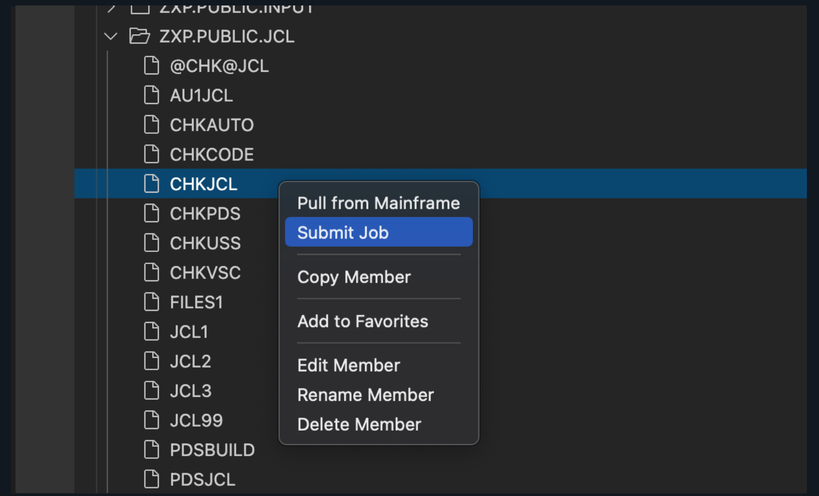
```
1  ********************************************
2  METRO NORTH POUGHKEEPSIE -> NYC M-F SCHEDULE
3  PEAK HOUR OPERATION
4  ********************************************
5  Poughkeepsie - 74mi
6  New Hamburg - 65mi
7  Beacon - 59mi
8  Cold Spring - 52mi
9  Garrison - 50mi
10 Peekskill - 41mi
11 Cortlandt - 38mi
12 Croton-Harmon - 33mi
13 Harlem - 125th Street - 4mi
14 Grand Central Terminal - 0mi
15 ************************************************************
16 Peak fares are charged during business rush hours on any
17 weekday train scheduled to arrive in NYC terminals between
18 6 a.m. and 10 a.m. or depart NYC terminals between 4 p.m.
19 and 8 p.m. On Metro-North trains, peak fares also apply to
20 travel on any weekday train that leaves Grand Central Terminal
21 between 6 a.m. and 9 a.m.
22 Off-peak fares are charged all other times on weekdays, all
23 day Saturday and Sunday, and on holidays.
```

14. BEM NA HORA

Envie o job JCL3 e observe o output. Há duas edições que você precisa fazer para que este job seja executado 100% corretamente e produza uma lista completa de 10 paradas, de Poughkeepsie ao Grand Central Terminal, mais as informações na parte superior e inferior do data set.

Você também não deve ter paradas repetidas. Se Beacon ou Cortlandt estiverem listados duas vezes, algo precisa ser consertado.

Quando concluído, você deve ter a saída completa em seu data set sequencial JCL3OUT, totalizando 23 linhas (registros).



15. QUEM DIRIA?

Agora envie o CHKJCL em ZXP.PUBLIC.JCL para validar o output correto do JCL2 e JCL3. Você quer ver o *completion code* (CC) 0.

Erro de verificação de desafio? Você pode querer excluir o data set de saída antes de reenviar o JCL, pois isso afetará o parâmetro DISP correto necessário para concluir a tarefa. Consulte os passos anteriores para encontrar jobs que esperam que novos conjuntos de dados sejam criados.

Você conquistou muito. Verifique novamente seu trabalho e envie-o rodando CHKJCL em ZXP.PUBLIC.JCL. Não se preocupe, nada que você precise editar lá, para variar.

BOM TRABALHO! VAMOS RECAPITULAR

JCL pode parecer um pouco complicado e talvez até um pouco desnecessário no início. Não estamos acostumados a usar código para iniciar programas, geralmente apenas clicamos duas vezes neles e eles são executados!

No entanto, quando você começa a entrar nos tipos de aplicativos que mantêm os sistemas Z ocupados 24 horas por dia, 7 dias por semana, começa a apreciar a precisão e o poder que a estrutura oferece. Basta dizer que JCL é uma habilidade necessária para qualquer verdadeiro profissional Z.

A SEGUIR...

Unix System Services (USS)