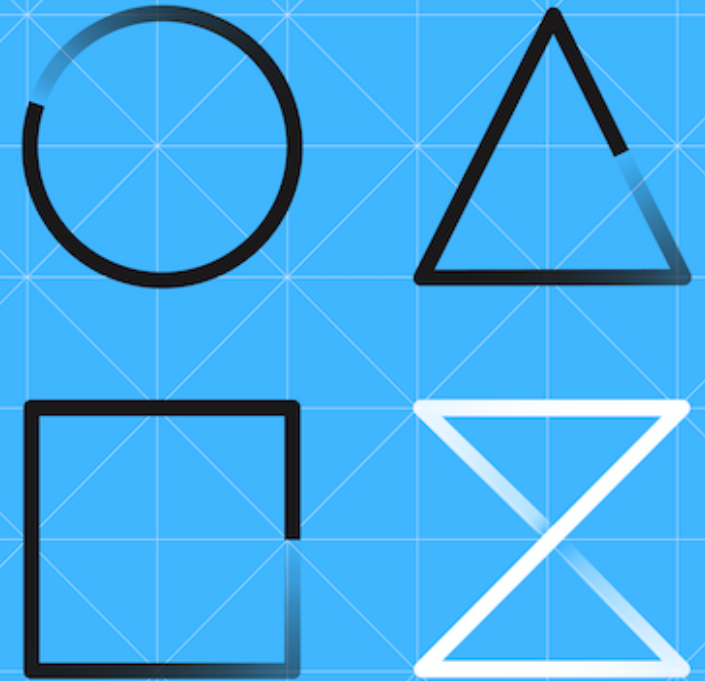


JCL1

Orchestrating the Enterprise

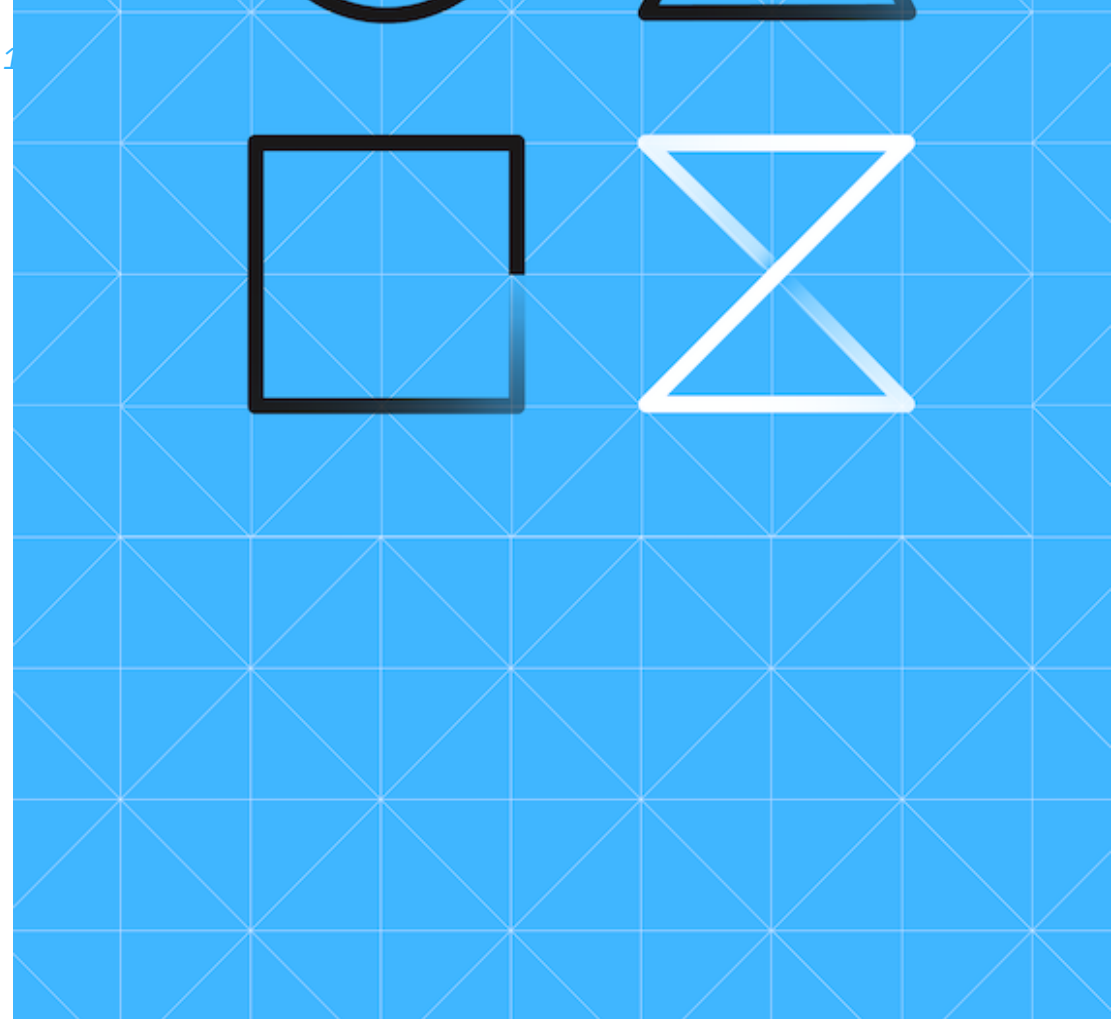
Automated Translation by Watson Language Translator



JCL1

Orquestrando a empresa

- JCL1-Fazendo as coisas acontecerem
 - 0 desafio
 - Antes de começar
 - Investimento
- 1 LOAD IT UP
- 2 SUBMETA-0
- 3 FILTRAR E ENCONTRAR
- 4 Você tem um zero. PERFEITO!
- 5 SALTE PARA A DIREITA
- 6 Meu primeiro trabalho de cópia
- 7 primeiro erro
- 8 INICIANDO ALGUM COBOL
- 9 EXECUTAR, CÓDIGO, EXECUTAR
- 10 ELES NÃO PODEM SER ZEROS
- 11 COMPARAR O CÓDIGO
- 12 TODOS A BORDO
- 13 UMA DISPOSIÇÃO AMIGÁVEL
- 14 QUAL O SEU STATUS?
- 15 VOCÊ ' RE NO DUMMY
- 16 DIREITO NA HORA
- 17 QUEM É NOVO?



JCL1/231110-1646

JCL1-FAZENDO AS COISAS ACONTECEREM

0 Desafio

Você já viu alguma JCL, mas nós não fomos realmente em profundidade. Nesses desafios, vamos aprender um pouco mais sobre para que JCL é usado, por que isso é importante em um ambiente Z e como você pode levar essas habilidades ainda mais longe. A JCL é uma parte essencial para fazer as coisas acontecerem no z/OS e ficar confortável com os conceitos e a sintaxe permitirá que você possa superar muitos desafios que você pode enfrentar enquanto explora.

Antes De Começar

Você deveria ter concluído o desafio FILES1, tudo sobre Conjuntos de Dados e Membros.. Se você tiver esses conceitos entendidos, você está pronto para continuar com as etapas neste desafio.

Investimento

Etapas	Duração
17	60 minutos

JCL1/231110-1646

1 LOAD IT UP

```
≡ ZXP.PUBLIC.JCL(JCLSETUP).jcl
1  //JCLSETUP JOB
2  //      EXEC PGM=IEFBR14
3  //LOAD   DD DSN=&SYSUID..LOAD,DISP=(,CATLG),DATACLAS=SLOAD
4  //JCL    DD DSN=&SYSUID..JCL,DISP=(,CATLG),DATACLAS=SPDS
5  //SOURCE DD DSN=&SYSUID..SOURCE,DISP=(,CATLG),DATACLAS=SPDS
6  //OUTPUT DD DSN=&SYSUID..OUTPUT,DISP=(,CATLG),DATACLAS=SPDS
7
```

Olhe em **ZXP.PUBLIC.JCL** para um membro nomeado **JCLSETUP** .

Esta é uma tarefa bastante simples que alocará alguns novos conjuntos de dados que você precisa para isso e outros desafios.

(A linha #4 na captura de tela cria seu próprio conjunto de dados JCL.)

É possível ver na Linha #3, ele menciona &SYSUID .. LOAD.

O Ampersand (&) com SYSUID depois que é o que é conhecido como um "Simbólico", e quando o sistema vê isso, ele substituirá automaticamente **&SYSUID.** com o seu ID do usuário.

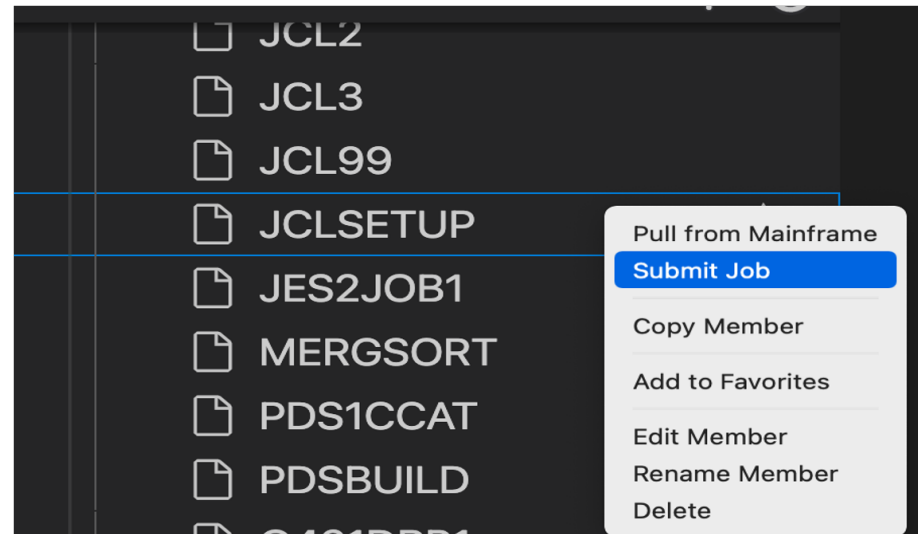
Observe o "." no final-isto marca o fim do nome simbólico.

Você não precisa fazer nenhuma chance para este trabalho para que ele funcione.

Isso significa que todos podem usar a mesma tarefa e ela substituirá automaticamente o &SYSUID. por seu ID do usuário.. Como conveniente!

JCL1/23110-1646

2 SUBMETA-0



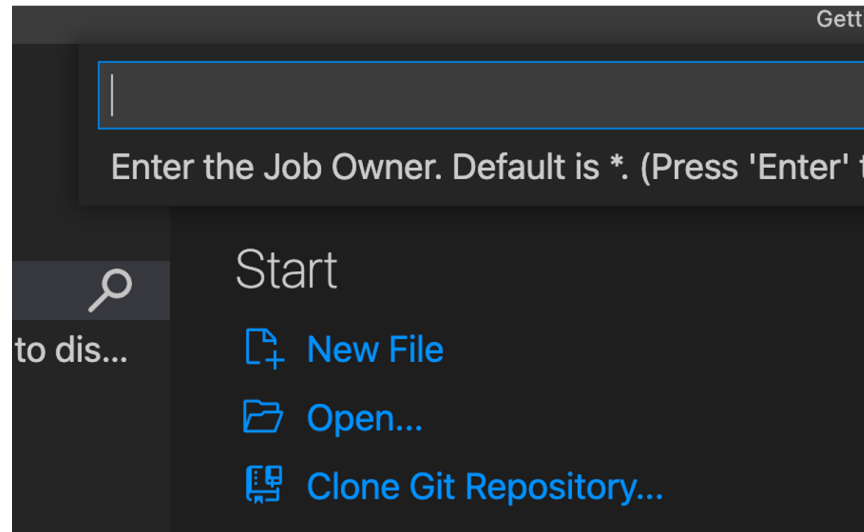
Clique com o botão direito do mouse nessa tarefa e selecione **Submit Job**.

Uma nota sobre a palavra "Job": JCL é usado para descrever para o sistema exatamente o que você deseja que ele faça. A tarefa que entregamos ao sistema é conhecida como uma "Tarefa", e o componente do z/OS que aceita, processa e gerencia a saída dessas tarefas é conhecido como o **Subsistema de Entrada de Job (JES)**.

Então, para esse desafio, você enviou um trabalho para o JES para que ele processasse a tarefa que você acabou de olhar.

Nota : esta tarefa é destinada a criar conjuntos de dados para você e supõe que você ainda não tenha esses conjuntos de dados; se você executá-la mais de uma vez, é provável que você veja erros sobre **DUPLICAÇÃO** Nomes do conjunto de dados

3 FILTRAR E ENCONTRAR



Você já deve ter um perfil em **JOBS** no lado esquerdo do VSCode.

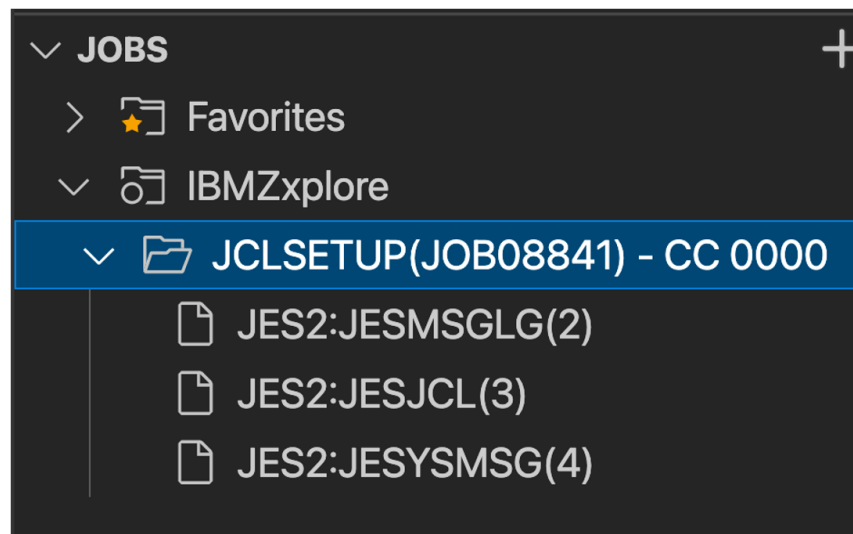
Clique na lupa () à direita dele.

- Insira seu ID do usuário para o Proprietário da Tarefa
- Insira um asterisco (*) para o Prefixo da tarefa
- Pressione Enter (em branco, sem dados) para a Procura de ID da Tarefa.

É possível refazer esse filtro clicando na lupa novamente e selecionando Proprietário / Prefixo ou ID da Tarefa. Você deve ser capaz de encontrar o trabalho que acabou de enviar aqui. Procurar por **JCLSETUP**.

O próximo passo será mais detalhado.

4 VOCÊ TEM UM ZERO. PERFEITO!



Abra o triângulo "twistie" ao lado da tarefa JCLSETUP que você acabou de enviar. Provavelmente haverá outros empregos lá também, mas você está especificamente à procura de **JCLSETUP** . Se você enviou mais de uma vez, encontre o que tem CC 0000 à direita.

Você também verá esse número no **JESMSG LG** depois de abrir a seta. Um código de condição (CC) de zero significa que tudo foi executado como esperado, sem erros, então isso é bom!

Se você tem qualquer outro número para o código de conclusão, então geralmente há algo que vale a pena investigar-e provavelmente corrigir.

JCL1/23110-1646

5 SALTE PARA A DIREITA

```
JOB08841  -STEPNAME PROCSTEP    RC    EXCP
JOB08841  -                      00      1
JOB08841  -JCLSETUP ENDED.  NAME-
JOB08841  $HASP395 JCLSETUP ENDED - RC=0000
ES2 JOB STATISTICS -----
2022 JOB EXECUTION DATE
        6 CARDS READ
```

Você pode ter notado que depois de enviar JCL no VSCode, uma pequena mensagem aparece no canto inferior direito da janela do VSCode.

Em vez de cavar através de sua saída JOBS, você geralmente pode apenas clicar nessa mensagem, e ele irá levá-lo direito para a saída.

Uma tarefa será iniciada em **ACTIVE** enquanto ele está sendo executado

É possível atualizar o status de uma tarefa fechando e reabrindo a seta à esquerda do nome da tarefa.

JCL1/23110-1646

O QUE É JES E JCL IMPORTANTE? POR QUE NÃO POSSO SIMPLEMENTE EXECUTAR PROGRAMAS?

Quando você envia JCL, ele vai para o Job Entry Subsystem (encurtado para JES).

O JES examina a JCL enviada e reúne todos os recursos necessários para realizar a tarefa. Em um sistema pesadamente carregado, pode ser necessário priorizar algumas tarefas mais baixas ou mais altas do que outras para que o trabalho importante seja feito mais rapidamente.

Pense em JCL como a ordem que um garçom escreve, e JES como a equipe de cozinha que olha para a ordem e decide como eles vão lidar com isso.

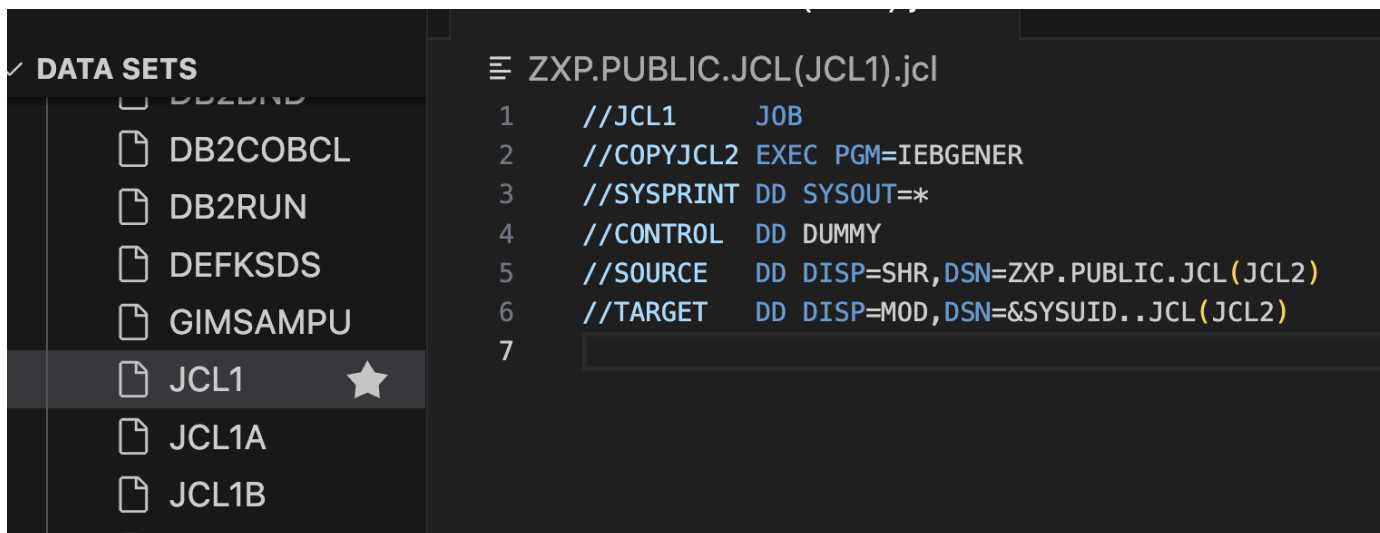
O L em JCL significa Linguagem, mas não é uma linguagem de programação tanto quanto é uma maneira de descrever efetivamente as tarefas para o sistema.

Todo o resto que aparece na saída da tarefa (o "log da tarefa") é informação sobre como a tarefa foi executada.. Como podem ver, há muita informação aqui.

JCL1/231110-1646

6 MEU PRIMEIRO TRABALHO DE CÓPIA

Agora você tem mais alguns conjuntos de dados com os quais trabalhar, e com isso sendo um desafio JCL, você precisa obter alguns de seus próprios JCL para trabalhar.



The screenshot shows the IBM Z Explorer interface. On the left, under the 'DATA SETS' section, a list of files is displayed: DB2COBCL, DB2RUN, DEFKSDS, GIMSAMPU, JCL1 (highlighted with a star), JCL1A, and JCL1B. On the right, the content of the selected file 'ZXP.PUBLIC.JCL(JCL1).jcl' is shown. The JCL code is as follows:

```
1 //JCL1 JOB
2 //COPYJCL2 EXEC PGM=IEBGENER
3 //SYSPRINT DD SYSOUT=*
4 //CONTROL DD DUMMY
5 //SOURCE DD DISP=SHR,DSN=ZXP.PUBLIC.JCL(JCL2)
6 //TARGET DD DISP=MOD,DSN=&SYSUID..JCL(JCL2)
7
```

Copie o arquivo **JCL1** membro de **ZXP.PUBLIC.JCL** para o seu próprio **JCL** e, em seguida, abra sua cópia. Você *necessidade de* para ter isso em seu próprio conjunto de dados JCL, pois você o modificará nas próximas etapas.

Pode ser necessário fechar e reabrir o seu **DATA SETS** triângulo para atualizar a visualização, para que o JCL1 seja exibido.

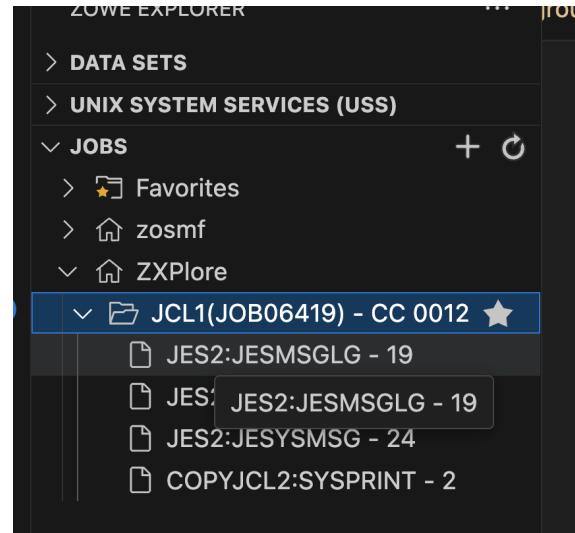
Essa tarefa é fornecida para que seja possível copiar o JCL2 membro de ZXP.PUBLIC.JCL em seu conjunto de dados JCL, mas usando o programa utilitário IEBGENER em vez de VSCode

Envie JCL1 da mesma maneira que você enviou o JCLSETUP e examine o log da tarefa.

JCL1/231110-1646

7 PRIMEIRO ERRO

Quando você olha para a seção JOBS e vê sua tarefa JCL1, você deve ver imediatamente que o código de conclusão é **0012** ; assumir que isso significa que algo deu errado.



Abra o arquivo **JES2 :JESMSG LG** da tarefa e verifique se há uma indicação do que causou a falha.

```

1      1                JES2 JOB LOG -- SYSTEM
2      ✓ 0
3      08.13.04 JOB06419 ---- TUESDAY, 07 NOV 2023 ----
4      08.13.04 JOB06419 IRR010I USERID Z##### IS ASSIGNED TO
5      08.13.05 JOB06419 ICH70001I Z##### LAST ACCESS AT 08:02:
6      08.13.05 JOB06419 $HASP373 JCL1 STARTED - INIT 1 -
7      08.13.05 JOB06419 IEC130I SYSIN DD STATEMENT MISSING
8      08.13.05 JOB06419 -
9      08.13.05 JOB06419 -STEPNAME PROCSTEP RC EXCP CONN
10     08.13.05 JOB06419 -COPYJCL2 12 10 0
11     08.13.05 JOB06419 -JCL1 ENDED. NAME-
12     08.13.05 JOB06419 $HASP395 JCL1 ENDED - RC=0012
13     0----- JES2 JOB STATISTICS -----
14     - 07 NOV 2023 JOB EXECUTION DATE
15     - 6 CARDS READ
16     - 53 SYSOUT PRINT RECORDS
17     - 0 SYSOUT PUNCH RECORDS
18     - 3 SYSOUT SPOOL KBYTES
19     - 0.00 MINUTES EXECUTION TIME
20

```

Nesse caso, é possível ver que o problema é devido a uma instrução DD ausente para **SYSIN**

Na JCL, as instruções que definem de onde os dados estão vindo, ou indo para, são conhecidas como **Definição de dados** ou simplesmente, "instruções DD".

Nesta tarefa, há apenas duas etapas-ambas executam o **IEBGENER** programa; se você pesquisar on-line, você eventualmente encontrará as seguintes informações sobre a configuração para IEBGENER:

<https://www.ibm.com/docs/en/zos/3.1.0?topic=c-job-control-instruções-4>

Arquivo / DD	Finalidade
SYSPRINT	Define um conjunto de dados sequenciais para mensagens O conjunto de dados pode ser gravado em um dispositivo de saída do sistema, um volume de fita ou um volume DASD
SYSUT1	Define o conjunto de dados de entrada Ele pode definir um conjunto de dados sequenciais, um membro de um conjunto de dados particionados ou PDSE ou um arquivo z/OS UNIX Um conjunto de dados

Arquivo / DD	Finalidade
	sequenciais pode ser formato básico, formato grande, formato estendido, formato compactado, entrada em spool, fita, leitor de cartão, terminal TSO ou DUMMY
SYSUT2	Define o conjunto de dados de saída Ele pode definir um conjunto de dados sequenciais, um membro de um conjunto de dados particionados ou PDSE, um conjunto de dados particionados ou PDSE ou um arquivo UNIX do z/OS Um conjunto de dados sequenciais pode ser formato básico, formato grande, formato estendido, formato compactado, saída em spool, fita, perfuração de cartão, impressora, terminal TSO ou DUMMY
SYSIN	Define o conjunto de dados de controle, ou especifica DUMMY quando a saída é sequencial e nenhuma edição é especificada. O conjunto de dados de controle normalmente reside no fluxo de entrada; no entanto, ele pode ser definido como um membro em um conjunto de dados particionados ou PDSE

JCL1/231110-1646

Esperamos que você possa ver que os arquivos designados para IEBGENER na tarefa JCL1, usando as instruções DD, não correspondem aos arquivos necessários pelo programa.

Faça as mudanças apropriadas nas instruções DD em seu membro JCL1 (**Lembre-se de salvar!**) e submeta novamente.

Se você fez as mudanças corretas, a tarefa deverá concluir com CC=0000.

Caso contrário, verifique o log da tarefa novamente para mensagens que indiquem a causa dos novos erros; faça ajustes e tente novamente ...

Com este tipo de programa, trabalhar o que deu errado com coisas como declarações DD incorretas ou ausentes é fácil de identificar e corrigir-está no livro!

Depois de ter uma conclusão bem-sucedida da tarefa JCL1, você deve ter um novo **JCL2** no conjunto de dados JCL.

(Você também terá um **JCL3** que é criado pela segunda etapa da tarefa JCL1)

8 INICIANDO ALGUM COBOL

```
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
```

Edite sua cópia pessoal do **JCL2** definição de tarefa..

Pode ser necessário fechar e reabrir o seu **DATA SETS** triângulo para atualizar a visualização, para que o JCL2 seja exibido.

Essa JCL é usada para compilar e executar algum código COBOL Depois de compilar, ele colocará o programa resultante em seu **LOAD** conjunto de dados.

Nota : os programas no conjunto de dados LOAD são binários-você não será capaz de ver nada aqui com VSCode.

Procure a linha que começa com **// COBRUN** -este é o início de uma "etapa" da tarefa que executará o compilador COBOL

Na próxima linha, é possível ver o conjunto de dados de entrada (a origem) na Linha 18 (**// COBOL.SYSIN**), e onde colocará a saída na linha a seguir (**//LKED.SYSLMOD**).

As linhas após o início do **// RUN** A etapa tem um formato semelhante:

```
//ddname DD DSN=dataset,DISP=access
```

- "ddname" também é conhecido como o nome do arquivo-o nome usado por programas para acessar dados em conjuntos de dados
- "dataset" é a localização real dos dados-isso pode mudar, mas o programa não precisa estar ciente
- "access" (ou "disposition") indica como o programa pode usar o conjunto de dados

Lendo ainda mais, se a etapa da tarefa for concluída com um Código de Conclusão de 0 (porque não houve problemas) da etapa de compilação, ela então executará o **CBL0001** programa.

Tudo fazendo sentido até agora? Você estará usando a JCL para compilar o código-fonte COBOL e, em seguida, executar o programa resultante

O QUE SIGNIFICA COMPILAÇÃO? 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 para que ele seja executado como um programa. Compilar é uma etapa que executa esta transformação. Nossa JCL tem duas etapas principais, compilando o código fonte em código de máquina e, em seguida, executando o programa.

Esse programa específico também requer dois arquivos de entrada e grava em um arquivo de saída, portanto, esses arquivos (conjuntos de dados) também serão especificados na JCL.

9 EXECUTAR, CÓDIGO, EXECUTAR

Após compilar com êxito o código COBOL, o JES executará o programa (o **// RUN EXEC PGM=CBL0001** e informá-lo onde localizar os conjuntos de dados de entrada, bem como onde a saída será armazenada em seu conjunto de dados OUTPUT-

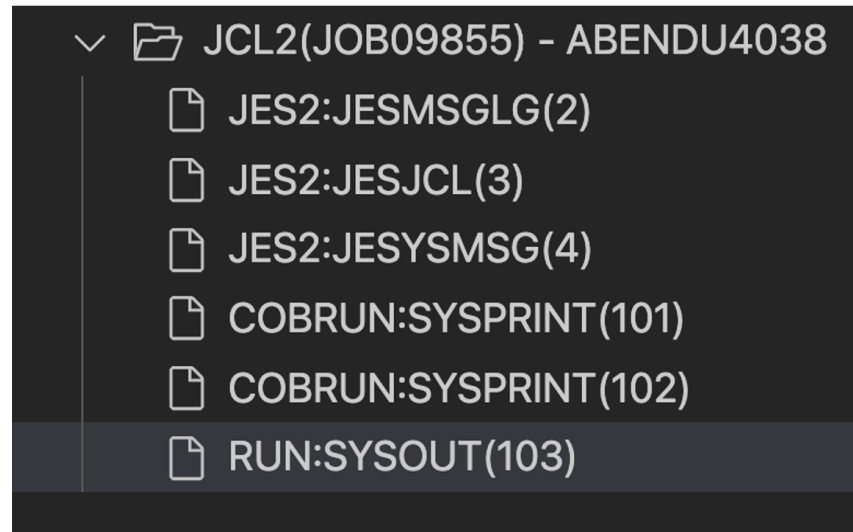
```
//COMBINE DD DSN=&SYSUID..OUTPUT(NAMES),DISP=SHR
```

O nome da instrução DD é o que vem diretamente após as barras duplas, por exemplo, "FNAMES" e "LNAMES".

Quaisquer linhas que comecem com **// *** são comentários e são ignorados pelo JES. As linhas comentadas são úteis para fornecer informações informativas ou conter linhas de código que podemos usar mais tarde, mas não precisam agora.

JCL1/231110-1646

10 ELES NÃO PODEM SER ZEROS



Enviar **JCL2** de seu conjunto de dados JCL e, em seguida, examine a saída, usando o que você aprendeu com as etapas anteriores neste desafio.

NOTA: Você *will* obter um **ABEND** (abreviação de Abnormal End), então algo ainda não está certo.

Mas não se preocupe-com suas novas habilidades, você vai chegar ao fundo disso!

Nas etapas anteriores para a tarefa JCL1, você poderia usar a documentação para IEBGENER para determinar quais instruções DD eram necessárias para o programa funcionar; neste caso, não há documentação- apenas o próprio código COBOL.

Na próxima etapa, você examinará o código COBOL e verá como o código real corresponde com o código JCL que está sendo usado para compilar e executá-lo.

E de novo, não se preocupe! Não é necessário se tornar um especialista COBOL para resolver isso- lembre-se de que isso é um **JCL** desafio, não um desafio COBOL.

JCL1/23110-1646

Para que os programas zOS trabalhem com conjuntos de dados em uma tarefa, eles precisam de 4 coisas para estar no lugar:

1. a definição interna do arquivo que representa a estrutura de dados e o conteúdo que o programa criará, lerá, modificará ou excluirá-isto é definido dentro do programa
2. o conjunto de dados atual que uma execução específica do programa pode usar-cada vez que o programa é executado, ele pode usar conjuntos de dados diferentes, desde que eles tenham o formato correto que o programa esperava; este é o nome usado no **DSN=** parâmetro de uma instrução DD
3. uma opção correta de disposição para o conjunto de dados para indicar se ele deve ser criado, excluído, transmitido-este é o **DISP=** parâmetro de uma instrução DD
4. uma instrução DD que vincula o arquivo interno do programa ao conjunto de dados específico-isto deve corresponder ao nome da definição de arquivo do proram e especificar o conjunto de dados necessário e a disposição

11 COMPARAR O CÓDIGO

```
*-----  
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.
```

A instrução JCL começando **//COBOL.SYSIN** aponta para o conjunto de dados de código de origem COBOL que ele compilará, portanto, comece por lá Abra esse código-fonte do programa no VSCode e comece examinando o **ARQUIVO-CONTROLE** área.

É aqui que você obtém os nomes usados pelo programa que você precisa para corresponder na JCL. Por exemplo, **PRIMEIRO-NOME** é uma referência de registro no código COBOL para um arquivo e que é designado (ou vinculado) ao **FNAMES** Instrução DD na JCL..

Abra a JCL, o código COBOL e o joblog-examine a saída e você deverá ser capaz de ver o que

*muito simples **único** alteração*

precisa ser feito para o **JCL2** tarefa para que tudo seja vinculado entre o programa COBOL, os conjuntos de dados e a JCL (linguagem de controle de tarefas)

Quando você tiver corrigido o problema no JCL2, ele deverá ser executado com um CC=0000 e você *localizar a saída correta no membro correto* do conjunto de dados OUTPUT.

12 TODOS A BORDO

```
//*  
//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
```

Abra o arquivo **JCL3** membro do conjunto de dados JCL e dê uma olhada no que está dentro. Você verá que essa JCL contém um número de etapas, com cada uma usando o **IEBGENER** para direcionar um ou mais registros em um conjunto de dados sequenciais.

Esta é uma extensão muito simples da tarefa JCL1 que você trabalhou anteriormente-uma grande diferença aqui é como os dados de "origem" são definidos.

Nesta tarefa, os dados para todos **SYSUT1** DDs é definido como "in-stream"-ele está bem ali na tarefa após a instrução DD. Isso é especificado pelo **DD *** em vez de uma **DSN=** parâmetro. O programa IEBGENER copiará a entrada de SYSUT1 até que a próxima instrução JCL seja atingida-começando com **//** ou **/ ***

Você verá que há um cabeçalho, algumas informações da estação para as paradas de trem de pico entre Poughkeepsie, NY e Grand Central Terminal em NYC, seguido por algum texto sobre horas de operação.

Parece muito simples ... qual será a parte complicada?

13 UMA DISPOSIÇÃO AMIGÁVEL

```
DISP=(MOD,PASS,DELETE),  
UNIT=SYSDA,  
LRECL=80)
```

Em cada pedaço de JCL que você usou até agora, você terá encontrado algum tipo de **DISP** (disposição) parâmetro.

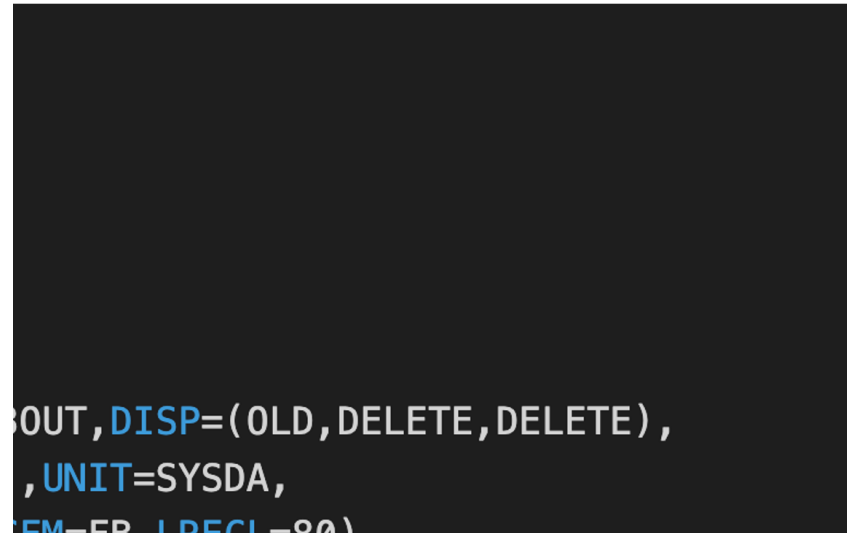
Os parâmetros DISP são usados para descrever como a JCL deve usar ou criar um conjunto de dados e o que fazer com ele após a conclusão da tarefa, ou da etapa da tarefa,.

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

Parâmetro	Significado de
NOVO	Criar um novo conjunto de dados.
SHR	Reutilizar um conjunto de dados existente e permitir que outras pessoas o usem se quiserem
OLD	

Parâmetro	Significado de
	Reutilizar um conjunto de dados existente, mas não deixe que outros o usem enquanto o estamos usando
MOD	Somente para conjuntos de dados sequenciais Reutilizar um conjunto de dados existente, mas apenas anexar novos registros à parte inferior dele Se nenhum conjunto de dados existir, crie um novo.

14 QUAL O SEU STATUS?



O campo 2 do parâmetro DISP descreve o que deve acontecer com o conjunto de dados no caso de uma conclusão normal da etapa da tarefa, e o terceiro campo é o que deve acontecer com o conjunto de dados no caso de uma falha da etapa da tarefa

Há uma série de valores que podem ser usados aqui, mas para esse desafio, você só precisa saber o seguinte:

Campo 2	Significado de
DELETE	Apagar completamente do armazenamento
CATLG	Registre o conjunto de dados para que seja possível usá-lo após a conclusão da tarefa
PASS	Após essa etapa ser concluída, mantenha-a pressionada para que as etapas de tarefa que vêm depois disso (na mesma tarefa) possam usá-la

15 VOCÊ 'RE NO DUMMY

```
//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
*****
```

Você deve notar muitas menções de **DUMMY** nas instruções DD.

Não se preocupe, essa JCL não está chamando ninguém de nomes; é apenas uma maneira de dizer "Essa alocação de arquivo é necessária, mas dessa vez não será usada para nada, portanto, não importa".

Como você viu anteriormente, o **IEBGENER** programa em execução em cada etapa requer:

- **SYSIN** uma instrução de arquivo de entrada (DD) para comandos de controle
- **SYSPRINT** uma instrução DD de saída para o programa relatar sucesso, falha, progresso
- **SYSUT1** uma instrução DD "source" da qual os dados devem ser copiados de
- **SYSUT2** uma instrução DD "target" na qual os dados de SYSUT1 devem ser copiados para

Mas nesta tarefa, a saída do programa não é necessária, e não há instruções de controle necessárias, então DUMMY é uma maneira de dizer "Não importa, não desperdice seu tempo configurando isso"-basta tomar a ação padrão e copiar o conjunto de dados associado com **SYSUT1** no conjunto de dados associado a **SYSUT2** .

16 DIREITO NA HORA

Envie sua cópia do **JCL3** e veja a saída.

NOTA: você deve esperar que essa tarefa seja concluída com o código 0000-isso significa que nada está quebrado-isso não significa que a saída está correta!

Há duas edições que você precisa fazer para que essa tarefa seja executada 100% corretamente:

- uma lista completa de **todas as 10 estações paragens** , de Poughkeepsie a Grand Central Terminal
- as informações na parte superior e inferior do conjunto de dados.
- Você deveria ter **não há paradas de repetição** . Se Beacon ou Cortlandt estiver listado lá duas vezes, algo ainda precisa ser corrigido.

NOTA: *Será necessário excluir o **JCL3OUT** conjunto de dados de saída toda vez antes de reenviar **JCL3***

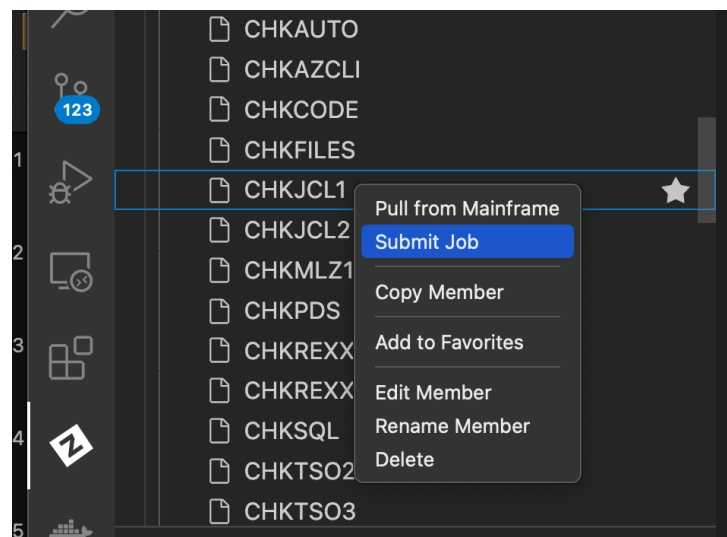
Opcionalmente, verifique novamente por meio da JCL para as tarefas JCLSETUP e PDSBUILD para ver como você pode excluir JCL3OUT automaticamente, incluindo uma etapa extra no início de JCL3

Envie JCL3 novamente e verifique se você tem a saída correta.

Quando concluído, você deve ter a saída completa em seu **JCL3OUT** conjunto de dados sequenciais, totalizando 23 linhas (registros)-o mesmo que a visualização a seguir:

```
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.
```

17 QUEM É NOVO?



Agora envie a tarefa **CHKJCL1** a partir de **ZXP.PUBLIC.JCL** para validar a saída correta de JCL2 e JCL3 e esperar ver o código de conclusão (CC) de 0000.

Se CHKJCL1 retornar CC=0127, volte e verifique novamente e ajuste seu trabalho para JCL2 e JCL3 e envie essas tarefas novamente, se necessário.

Verifique novamente a saída correta enviando **CHKJCL1** novamente até obter CC=0000.

AGAIN: Você precisará certificar-se de que o **JCL3OUT** o conjunto de dados de saída é excluído antes do reenvio **JCL3**

Você conseguiu muito, e espero entender o requisito para seguir os detalhes ... Bem feito!

Bom trabalho-vamos recapitular	Em seguida ...
<p>JCL pode parecer um pouco complicado, e talvez até mesmo um pouco desnecessário no início. Nós não estamos acostumados a usar código para iniciar programas, nós geralmente apenas clique duas vezes sobre eles e eles são executados! No entanto, uma vez que você começa a entrar nos tipos de aplicativos que mantêm os sistemas Z ocupados 24/7, você começa a construir uma apreciação para a precisão e poder que a estrutura lhe dá. Basta dizer que a JCL é uma habilidade necessária para qualquer verdadeiro profissional Z.</p>	<p>Efetuar o registro de saída Utilitários DfSMS para outros "utilitários" comuns usados na JCL para gerenciar conjuntos e outros sistemas de armazenamento.</p> <p>Descubra o ambiente Unix dentro do zOS-Unix System Services (USS)</p>