REXX

Fita adesiva corporativa de TI

12 passos 60 minutos

O DESAFIO

Neste desafio, você conhecerá o Rexx, uma linguagem de programação conhecida por sua simplicidade, poder e relativa facilidade de uso. Veremos como você executa um programa Rexx a partir de uma linha de comando, bem como como iniciar um TSO Address Space para executar um programa Rexx interativo. Para interagir com o TSO, você precisará do ZOWE Command Line Interface (CLI) instalado.

ANTES DE VOCÊ COMEÇAR

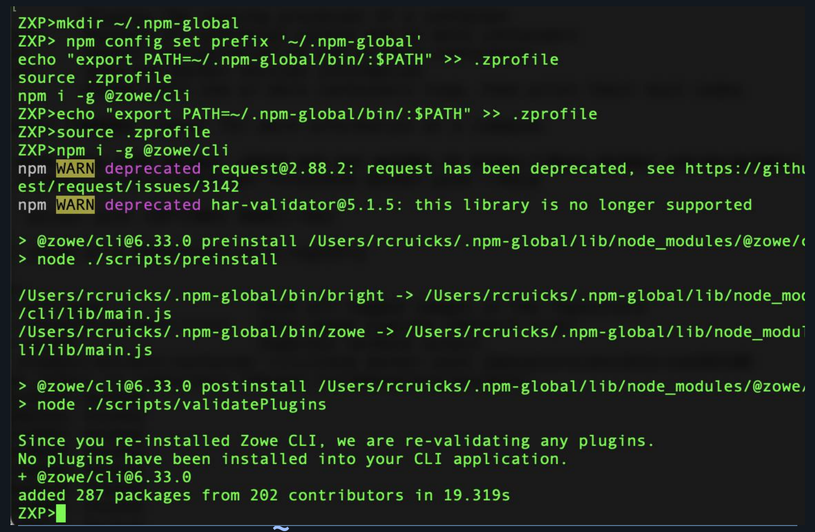
Este desafio requer alguma configuração para seu ambiente de terminal. Certifique-se de executar as etapas de instalação do ZOWE CLI e, em seguida, vá para o Rexx

1. INSTALANDO O ZOWE CLI

Estamos usando o plug-in Zowe Explorer para VS Code ao longo deste concurso, mas o Zowe faz muito mais e é responsável por trazer muito mais para o mainframe.

Para ser claro, **você está instalando o Zowe CLI em seu próprio computador, não no mainframe.** Você usará a Interface de Linha de Comando do Zowe para trabalhar com o Zowe e com o z/OSMF que está sendo executado no mainframe, mas estará conduzindo a maior parte desse desafio em seu próprio computador.

Os usuários do Linux podem precisar explorar um pouco para encontrar o que funciona em seu sistema específico, mas deve se aproximar das etapas do Mac, apenas substituindo o arquivo de perfil do shell correto.

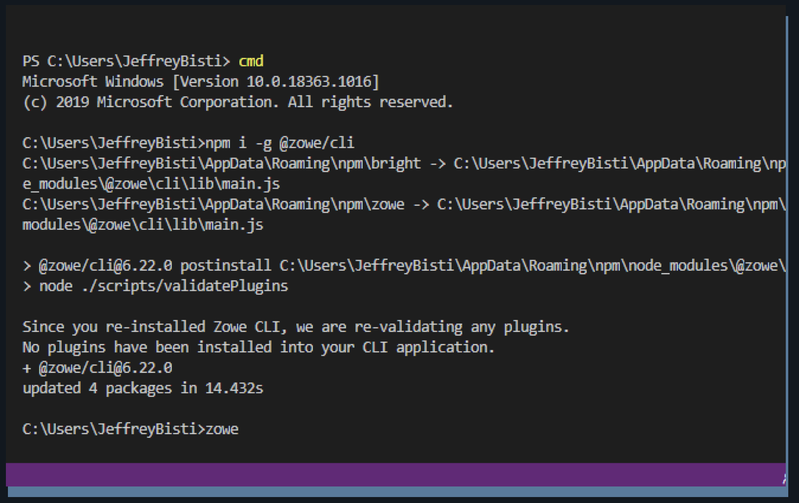


2. INSTALAÇÃO DO ZOWE CLI NO MacOS

Para usar pacotes Node no sistema operacional, precisamos carregá-los em um diretório .npm-global que pode ser acessado por usuários comuns. Essas etapas vão configurar isso, informarão ao npm (Node Package Manager) para usá-lo e incluirão isso na lista normal de locais em que os programas serão executados. Para usuários do MacOS, isso deve funcionar.

1: **mkdir ~/.npm-global**
2**: npm config set prefix '~/.npm-global'**
3: **echo "export PATH=~/.npm-global/bin/:$PATH" >> .zprofile**
4: **source .zprofile**
5: **npm i -g @zowe/cli**

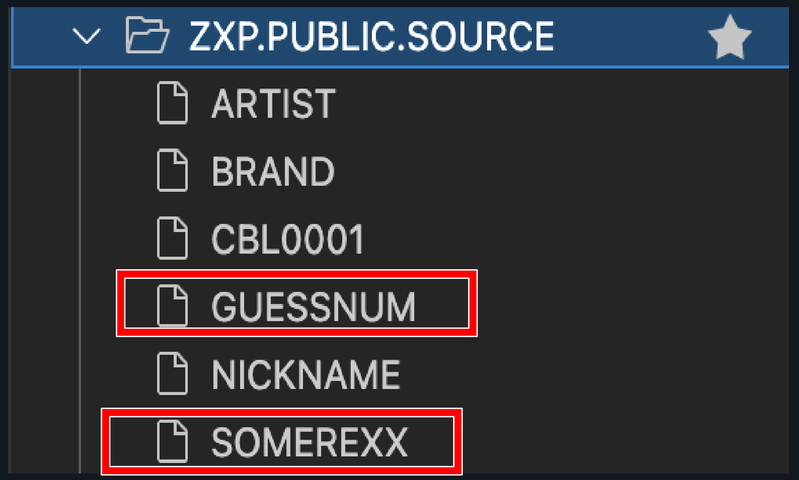
.



3. CONFIGURANDO O NPM NO WINDOWS

No Windows, primeiro vamos mudar do PowerShell para o CMD e, em seguida, instalar o zowe cli usando npm, o gerenciador de pacotes do Node. Isso deve funcionar para a maioria dos usuários, embora sua saída possa parecer um pouco diferente do que você vê na imagem de captura de tela.

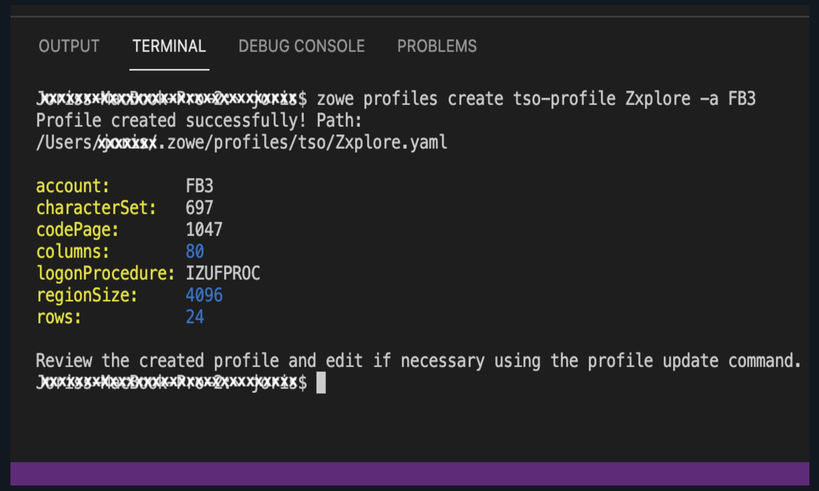
1: digite **cmd** (isso vai mudar o shell de PowerShell para cmd)
1: **npm i -g @zowe/cli**
2: **zowe**



4. ABRA UM TERMINAL

Comece copiando os dois arquivos de ZXP.PUBLIC.SOURCE em seu próprio data set SOURCE. Especificamente, estamos procurando por SOMEREXX e GUESSNUM.

Em seguida, abra um Terminal, assim como você fez para os comandos USS, mas não faça SSH em nada. Você deve ser capaz de digitar o comando **zowe** daqui e obter alguma saída do programa

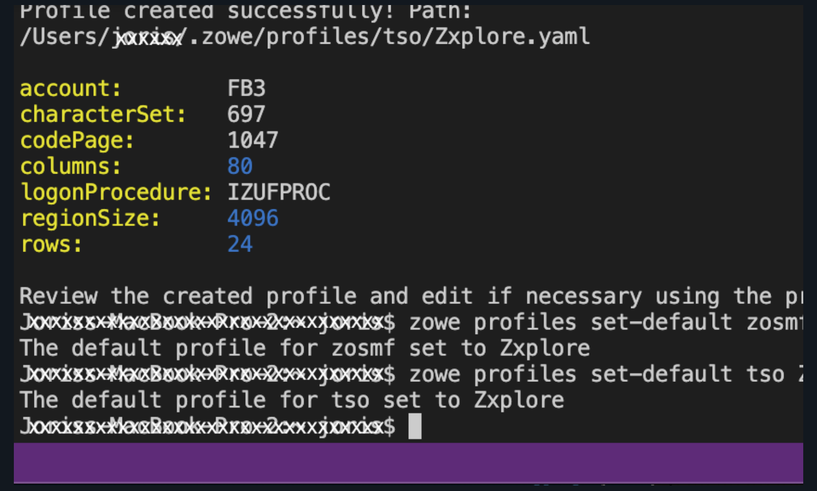


5. CRIE UM PERFIL TSO

Até agora, usamos um perfil z/OSMF para conectar. Precisamos criar um perfil TSO para emitir comandos TSO (mais informações na próxima página). Digite este comando:

zowe profiles create tso-profile zxplore -a FB3

se o nome de perfil que você quiser, zxplore é apenas um exemplo, e se você está seguindo nossos exemplos, é bom manter a consistência.



6. DEFINA PERFIS PADRÃO

Nesse ponto, também é uma boa ideia garantir que os perfis corretos estejam definidos como padrão. Eles provavelmente são, mas caso precise:
zowe profiles set-default zosmf zxplore
zowe profiles set-default tso zxplore

(Claro, use os nomes de perfil que você escolheu)

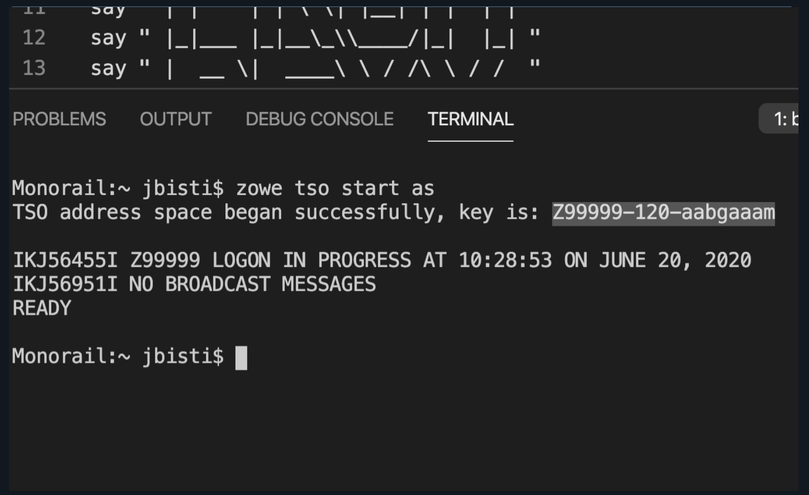
Se você quiser começar de novo, use o comando **zowe profiles delete** e siga as instruções. Se você tiver outros perfis de outras atividades de mainframe, às vezes é necessário reiniciar o VS Code para que uma mudança de perfil tenha efeito.

7. RODE ALGUM REXX

Digite o seguinte comando:

zowe tso issue command "exec 'zXXXXX.source(somerexx)'" --ssm

Certifique-se de incluir todas as aspas simples e duplas. Além disso, esta pode ser a última vez que apontamos que sempre que você vê Zxxxxx ou Z99999, você precisa inserir seu próprio ID de usuário. Sabemos que a maioria de vocês está cansada de ver isso escrito todas as vezes, mas você não acreditaria em quantas pessoas ficam confusas por não substituir o texto de exemplo.

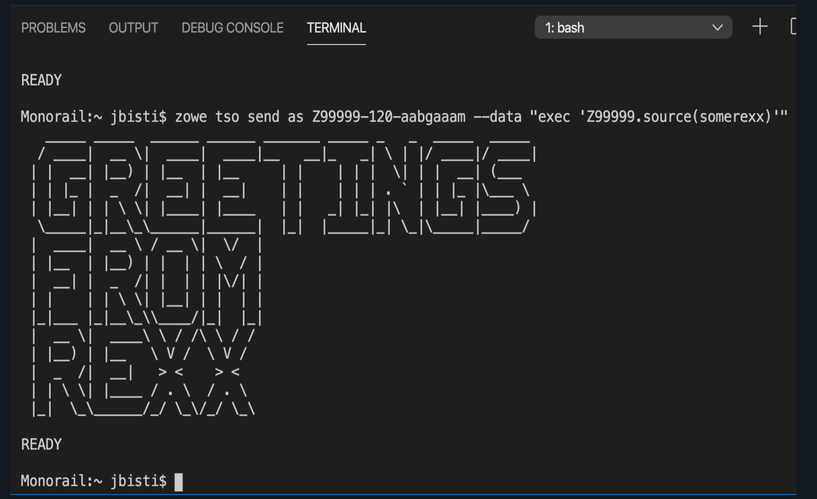


8. INICIE UM ADDRESS SPACE

Inicie um address space com o seguinte comando:
zowe tso start as

Isso criará um espaço de endereço para você e informará sua chave, que começará com seu ID de usuário (como você pode ver acima). Você usará essa chave para as próximas etapas. Às vezes, depois de não ser usado por um tempo, uma sessão TSO desaparece. Basta fazer outro usando o mesmo comando.

Você pode parar um espaço de endereçamento com a opção ***stop*** (parar).



9. RODE O MESMO REXX

Execute o mesmo programa Rexx que fizemos na etapa 7, mas desta vez, direcione a entrada para o address space (e lembrese, você provavelmente pode pressionar a seta para cima para recuperar comandos anteriores)

zowe tso send as your-as-key --data "exec 'Zxxxxx.source(somerexx)'"

Nota: (1) Isso é tudo em uma linha
(2) sua chave será diferente
(3) desta vez deixamos de fora o –ssm

Você deve obter exatamente a mesma resposta de antes. A diferença é que agora você está emitindo esses comandos por meio de um Espaço de Endereço TSO semi-persistente, o que fará mais sentido no próximo passo.

“TSO? ADDRESS SPACE?”

TSO (Time Sharing Option) é outra maneira que o z/OS permite que muitos e muitos usuários obtenham acesso a data sets, executem programas e examinem a saída. É essencialmente a interface de linha de comando para o z/OS (quando você não está usando USS para acessar o lado UNIX das coisas).

Pense em um Address Space como um tíquete que permite que você comece a usar a memória do sistema. Um espaço de endereço representa uma enorme quantidade de memória, embora o sistema ainda controle o que vive na memória real do chip, versus o que é movido (ou paginado) para o disco. Para onde seu programa aloca sua memória depende de quão importante é, como deve ser usado e se será compartilhado com outros programas. Os Espaços de Endereço são uma parte essencial do z/OS e você deve ler mais sobre eles quando tiver um tempinho:
https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconcepts_82.htm

```
READY

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "exec 'Z99999.source(guessnum)'"
I'm thinking of a number between 1 and 10.
What is your guess?
```

10. ADIVINHA? OUTRO EXEC

Faça o mesmo, só que agora com o programa **GUESSNUM**

Este é um programa que gera um número aleatório entre 1 e 10, e você tem que adivinhar esse número. Pode haver outros jogos que você gostaria de jogar agora, mas nem todos eles vão te ensinar sobre Rexx e TSO, então tenha isso em mente ao olhar para o seu controle de vídeo-game.


```
Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "exec 'Z99999.source(guessnum)'"
I'm thinking of a number between 1 and 10.
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "1"
That's not it. Try again
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "2"
That's not it. Try again
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "3"
That's not it. Try again
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "4"
You got it! And it only took you 4 tries!
READY
```

11. REXX AO VIVO!

Envie seus palpites para o programa substituindo tudo entre aspas duplas por um número. Você pode ver aqui que estamos usando a chave do address space persistente para garantir que nossa entrada vá para o espaço de endereço TSO, que está esperando por nossa entrada.

Se acertar na primeira tentativa, parabéns! Sinta-se à vontade para iniciar o programa novamente e certifique-se de vê-lo passar pelas etapas *"Try again"* (Tente novamente).

Agora envie sua verificação de conclusão – CHKREXX1

```
1  /* REXX */
2  say "I'm thinking of a number between 1 and 10."
3  secret = RANDOM(1,10)
4  tries = 1
5
6  do while (guess \= secret)
7      say "What is your guess?"
8      pull guess
9      if (guess \= secret) then
10     do
11         say "That's not it. Try again"
12         tries = tries + 1
13     end
14 end
15 say "You got it! And it only took you" tries "tries!"
16 exit
```

12. OLHA QUE CÓDIGO FOFO...

Se você ainda não o fez, carregue o código desse programa em seu editor de código VS. Não é um programa complicado de forma alguma, mas você pode estar percebendo o quão simples esse código Rexx realmente é. 16 linhas de código, incluindo um comentário e um espaço em branco, e sim, esses comandos "say" (diga) e "pull" (puxe) realmente fazem o que você acha que eles fazem.

Você pode ver por que as pessoas amam essa linguagem. Ele também tem o que considero o melhor logotipo do mundo para uma linguagem de programação. Olha pra ele. Só olha...

BOM TRABALHO! VAMOS RECAPITULAR

Agora você está entrando no ritmo das coisas. Você está interagindo com um programa Rexx, executado em um espaço de endereço TSO, e está fazendo isso por meio do comando zowe.

Você provavelmente também aprendeu um pouco sobre a linguagem Rexx e pode até ter feito algumas leituras extras sobre Espaços de Endereço. Tudo aqui irá ajudá-lo a se tornar um profissional qualificado de Mainframe.

A SEGUIR...

O ferro está definitivamente fumegando, e você provavelmente está se tornando um fã de Rexx. No canal avançado, você encontrará o REXX2, onde escreverá seu próprio código do zero e implementará algumas leituras/gravações de arquivos. Por enquanto, porém, vamos continuar para o próximo desafio em Fundamentos.