01

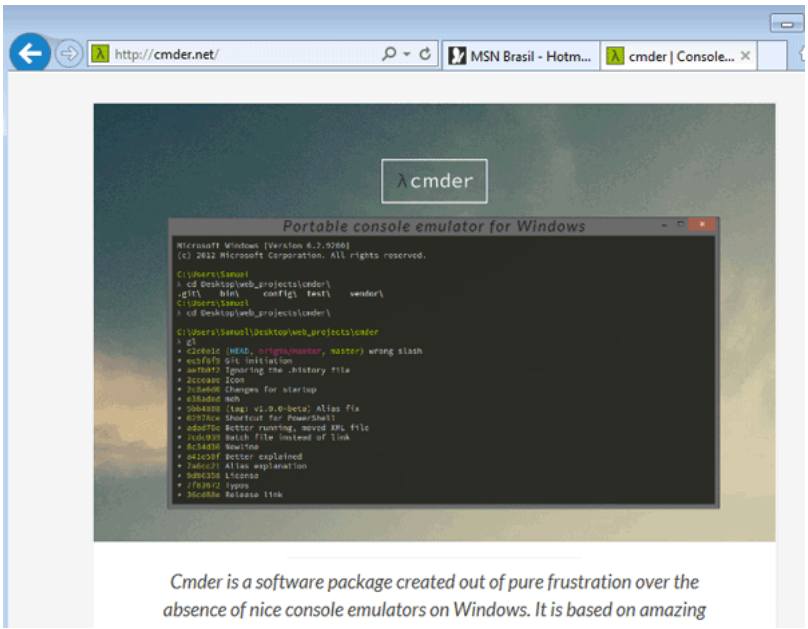
Um novo prompt e executando scripts

Transcrição

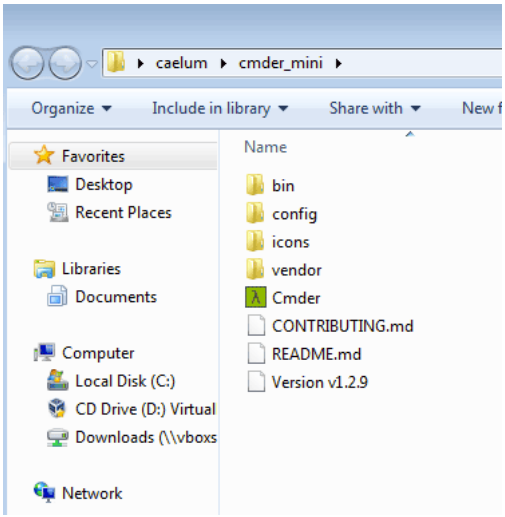
No último capítulo vimos como usar o terminal/prompt e os comandos principais para lidar com arquivos e diretórios. Vimos comandos como `dir`, `del` ou `cd`. Vamos continuar usando o prompt, mas antes de aprender novos comandos, instalaremos uma versão um pouco mais amigável para nós, humanos. Há uma variação do terminal que se chama `cmder`, a qual deixa a linha de comando colorida e com algumas facilidades. Nada te impede de continuar com o terminal antigo, no entanto vamos instalar o `cmder` para melhorar a visualização dos comandos.

Instalação do cmder

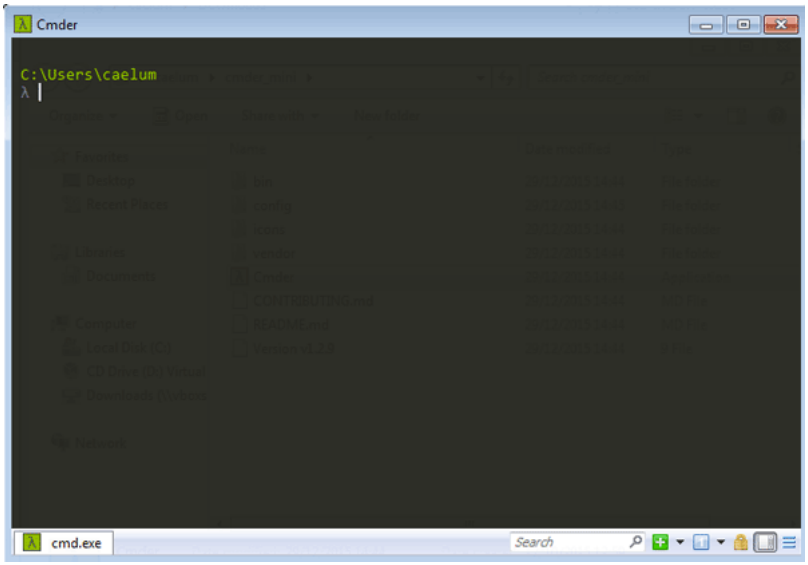
Para instalar o `cmder` devemos primeiro baixar o ZIP no site: http://cmder.net (http://cmder.net)



Escolha a versão mini e depois extraia o ZIP.



Para abrir o novo terminal do Cmder basta entrar na pasta do *cmder* e dar um duplo-clique no executável. Isso faz com que abra um novo terminal, e veja que já está colorido! A diferença é sutil, mas vai nos ajudar mais para frente.



Para saber mais sobre Git: Aqui usaremos a versão mínima do *cmder* mas para quem está interessado em usar o GIT já pode baixar a versão full também. O Git é um sistema de versão para código fonte. Ele ajuda gerenciar esse código fonte. Isso significa, sincronizar o trabalho de vários desenvolvedores no mesmo projeto. O Git mantém um histórico das alterações e cria automaticamente backups. Na Alura temos um treinamento focado nessa ferramenta poderosa: https://cursos.alura.com.br/course/git-github-controle-de-versao (https://cursos.alura.com.br/course/git-github-controle-de-versao)

Meu primeiro script

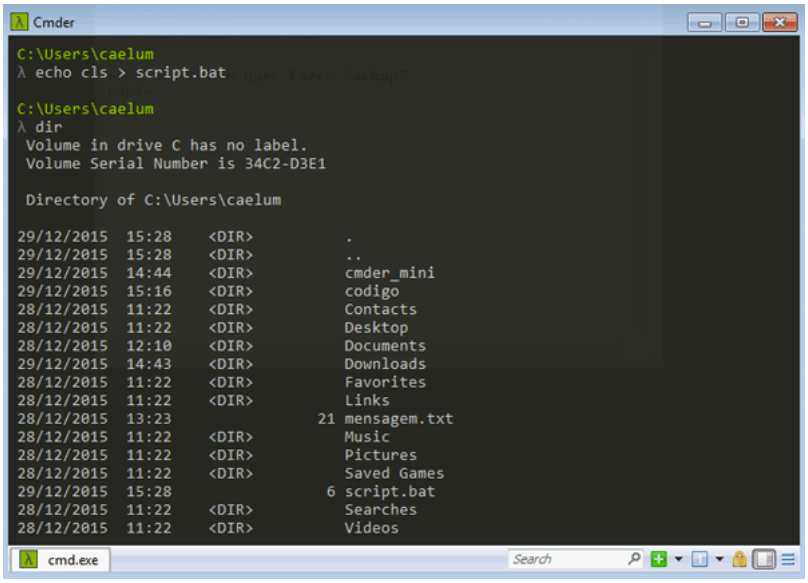
Uma das motivações para aprender os comandos no terminal é a automação de tarefas, isto é, fazer várias coisas repetitivas mais facilmente. Uma das tarefas de um desenvolvedor é criar um backup de arquivos. Vamos, então, criar um pequeno script para "backupear" os arquivos. Tudo executado na linha de comando, ok?

Os scripts no mundo Windows não são nada mais do que arquivos de texto com a extensão `bat`. `bat` significa *batch*, em português "lote". Ou seja, com esse script é possível executar vários comandos em lote.

Vamos criar um arquivo *batch*, na linha de comando:

```
echo cls > script.bat
```

Repare que o comando não foi executado ainda, gravamos apenas o comando `cls` no arquivo `script.bat`.



E como posso executar o script? Basta digitar:

```
script.bat
```

O script foi chamado e todos os comandos dentro dele foram executados sequencialmente! Como resultado, a tela está limpa! Fantástico :)

Processamento em lote

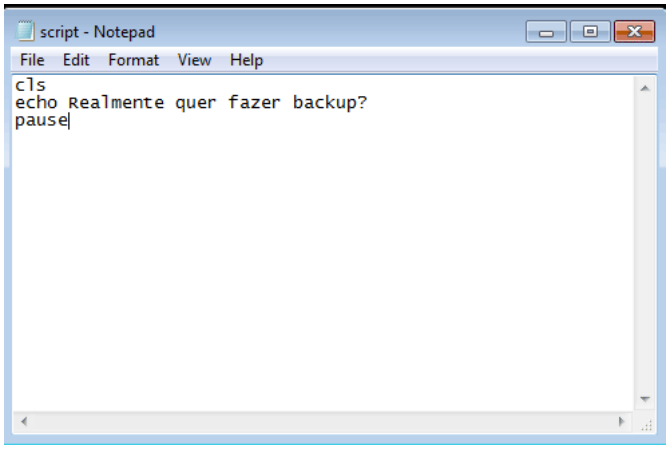
O arquivo tem a extensão `.bat` (batch, lote) não por acaso, pois podemos executar mais que um comando em sequência. Vamos abrir o arquivo `script.bat` em um editor de texto. Vou usar um bem simples.

Já que podemos executar qualquer comando, adicionaremos depois do `cls` um `echo` para perguntar se ele realmente quer continuar:

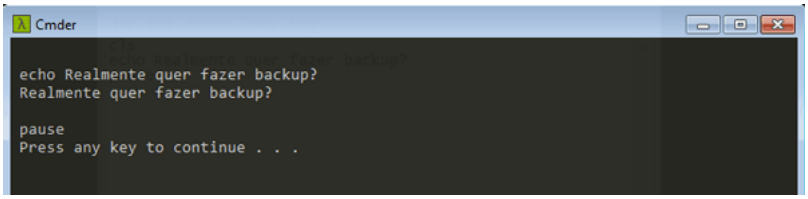
```
cls
echo Realmente quer fazer backup?
```

E para parar a execução podemos adicionar o comando `pause` :

```
cls
echo Realmente quer fazer backup?
pause
```



O comando `pause` faz com que o usuário do script precise confirmar a continuação da execução do script. Vamos testar uma vez para deixar mais claro:



Isto nos permite dar a opção do usuário abortar a execução do script caso ele não deseje continuar, através do atalho `CTRL + C` ou simplesmente fechando o terminal.

Ótimo, já é um bom início. Agora vamos limpar a tela com `cls`, adicionar uma mensagem para o usuário saber que vamos iniciar o backup e preparar a pasta do backup através do script, indo para a pasta `home` do nosso usuário e criando o diretório de backup:

```
cls
echo Realmente quer fazer backup?
pause

cls
echo ok, fazendo backup...
cd C:\Users\caelum
mkdir backup
```

Todos esses comandos já conhecemos e agora podemos copiar os arquivos da pasta `codigo` para a pasta `backup`. O problema ainda é que o comando `copy` é muito simples e não consegue copiar pastas e sub-pastas. Ainda bem que existe o irmão gêmeo dele, o `xcopy`. O `xcopy` copia pastas e sub-pastas desde que usamos os parâmetros `/E` (para copiar subpastas) e `/Y` (para confirmar automaticamente a sobrescrita de arquivos). Sabendo disso podemos escrever:

```
cls
echo Realmente quer fazer backup?
pause

cls
echo ok, fazendo backup...
cd C:\Users\caelum
mkdir backup

xcopy /E /Y "C:\Users\caelum\codigo"  "C:\Users\caelum\backup"
```

Agora no fim do script podemos listar os arquivo da pasta backup:

```
cls
echo Realmente quer fazer backup?
pause

cls
echo ok, fazendo backup...
cd C:\Users\caelum
mkdir backup

xcopy /E /Y "C:\Users\caelum\codigo" "C:\Users\caelum\backup"

echo Listando os arquivos do backup
dir C:\Users\caelum\backup
```

Pronto! Agora nosso script realiza o backup da pasta `codigo` como desejávamos!

Repare que o script é nada mais de um conjunto de comandos pequenos para realizar uma tarefa maior.