**Winodws – Prompt de Comando e Powershell**

**Prompt de Comando**

1. **Aula 1 – Os Primeiros Comandos No Prompt:**

- echo: Printa na tela tudo o que for escrito a seguir

- >: indica onde o texto que foi escrito será salvo, caso queira. Colocar nome do arquivo e sua extensão para que seja criado.



- dir: Lista tudo o que está em uma pasta.

- /O: Uma opção de visualização do dir. Ele ordena a saída de acordo com o que for escolhido depois:

- D: Data;

- S: size.

Podemos ainda escolher qual pasta queremos que ele mostre o conteúdo, organizado ou não, apenas colocando o caminho dela:



- cd (change diretor) + nome da pasta: Entra na pasta desejada que foi listada através do dir.

- cd + ..: Volta para diretório anterior;

- cd + /: Volta para a raiz.

- mkdir + nome\_pasta: Cria um diretório.

- move + nome do que vai ser movido + para onde será movido: Move itens, pastas e afins para outros lugares.

- Seta pra cima: Navega pelo histórico de comandos que já usamos nessa sessão.

- type: Lê o que tá dentro de arquivos txt.

- copy: Copia o item que quiser. Sempre lembrar de colocar a extensão que quer que a cópia tenha.

- rename: renomeia o item. Sempre colocar a extensão do arquivo.

- del + nome arquivo: deleta arquivo.

- Tab: completa os nomes dos comandos ou arquivos que tenhamos começado a digitar. Pode apertar várias vezes até aparecer o que está procurando.

- cls: limpa a tela do prompt.

- help ou help + nome do comando: Exibe uma lista com todos os comandos ou a descrição do comando escolhido.

Para saber mais:

1. Adicionando conteúdo a um arquivo:

Vimos neste capítulo do treinamento que podemos criar um arquivo utilizando o sinal **>**, mas e se quisermos adicionar mais linhas a um arquivo de texto já existente, por exemplo para guardar o resultado de diversas execuções de um programa em um único arquivo?

Para isto existe o **>>**, quando colocamos o sinal de maior, duas vezes, o Prompt entende que só deve criar um arquivo novo quando o arquivo que pedimos não existir! Caso ele já exista, ele adiciona o novo conteúdo ao final do arquivo, sem sobrescrevê-lo!

Veja só como funciona:

Digamos que primeiro criamos um novo arquivo de texto com o nosso conhecido comando echo :

echo Ola mundo > arquivo.txt

Ao abrirmos nosso arquivo.txt , vemos o texto que esperávamos:

type arquivo.txt

//arquivo.txt

Ola MundoCOPIAR CÓDIGO

Porém se agora desejarmos adicionar um novo texto abaixo de "Ola Mundo"? Utilizamos o '>>'!

Desta forma:

echo Novos dados! >> arquivo.txt

Quando abrimos nosso arquivo.txt, vemos que o texto foi adicionado corretamente:

type arquivo.txt

//arquivo.txt

Ola Mundo

Novos dados!

1. Pasta Home do usuário:
   1. Você já reparou que quando abrimos o prompt ele sempre abre na mesma pasta? Essa é a pasta do seu usuário, onde a maioria das pessoas salvam os Documentos, Fotos e Videos. Lá também ficam os arquivos particulares do seu usuário, como de configuração dos programas específicos e favoritos do seu navegador. A grande vantagem de cada usuário ter sua pasta é que temos uma separação dos dados do sistema e dos dados de cada usuário, facilitando assim o backup dos dados dos usuários caso seja necessário e aumentando a segurança também. A pasta do seu usuário é conhecida como pasta Home, e essa nomenclatura é adotada por muitos desenvolvedores quando querem se referir a pasta do usuário, independentemente do sistema operacional.
2. O comando tree:
   1. Dependendo do diretório atual podem aparecer muitas informações mas repare que o comando tree mostra as pastas e subpastas organizadas em uma árvore.
   2. O tree pode ser útil para entender a estrutura de um projeto. Muitas vezes você precisa baixar um projeto na Alura para importar alguns arquivos iniciais. Com o comando tree você já pode ver facilmente como o projeto está organizado! Muito útil :).
3. Comando more:
   1. O comando more funciona de um jeito semelhante ao comando type, com a diferença de exibir página por página do arquivo no terminal, em vez de mostra-lo todo de uma vez.
   2. O seu uso é análogo ao comando type, podendo ser chamado assim:
   3. **more arquivo.txt**
   4. Ele exibirá uma página de cada vez do arquivo, sendo muito útil quando queremos exibir arquivos de texto com várias linhas para ir lendo lentamente, em vez de abri-lo todo no terminal de uma vez. Um exemplo disso é quando queremos ler os logs de uma aplicação que está em um servidor na nuvem, neste caso é preciso ler grandes arquivos de texto linha a linha, para identificar um bug ou realizar algum teste.
   5. Agora que você já conhece o comando more e um exemplo prático de sua aplicação, vou lhe mostrar como controlar a exibição das páginas:
      1. Para passar de página em página, utilize a tecla espaço do seu teclado.
      2. Para passar de linha a linha, utilize a tecla enter do seu teclado.
      3. Para sair da exibição do comando, sem chegar ao final do arquivo utilize a tecla q do seu teclado.
   6. Como você já sabe, sempre que quiser aprender mais sobre um comando, pode utilizar o help, então caso tenha a curiosidade de aprender ainda mais sobre o comando more, é só digitar help more no terminal!

O que aprendemos:

* para que serve o Prompt
* criar (mkdir) e remover (rmdir) diretórios
* copiar (copy) e apagar arquivos (del)
* imprimir no prompt com o comando echo
* saber a diferença entre . e ..
* criar arquivos com >
* ver o conteúdo do arquivo (type)
* mover pastas e arquivos (move)
* renomear diretórios e arquivos (rename)

1. **Aula 2 – Um Novo Prompt e Executando Scripts:**
   1. Baixamos o cmder para executar os códigos com maior facilidade que o cmd do windows.
   2. Criamos nosso primeiro script de backup. Ele basicamente pega todos os arquivos e subpastas de uma pasta e copia com o xcopy /e /y para a basta de backup.
   3. Pause: Comando que espera o usuário fazer alguma interação com o terminal antes de continuar.
   4. Scripts são feitos em documentos .bat e podem ser editados pelo bloco de notas.:
      1. O batch script é um arquivo simples que agrupa uma série de comandos, normalmente utilizado para automatizar tarefas cotidianas. Um arquivo batch (.bat) é um script que é executado sequencialmente pelo prompt (um comando depois do outro).
      2. No desenvolvimento usamos scripts de diversas formas. Por exemplo, para limpar uma máquina antes de instalar arquivos novos, guiar o usuário pela instalação, preparar arquivos de configuração e importar dados dentro da aplicação ou de um banco de dados. E é claro, o exemplo clássico de um script para automatizar o backup!
   5. @echo off:
      1. Você pode notar que é exibido duas vezes cada mensagem, uma vez exibindo o comando e outra na saída dele, pois quando executamos um script no prompt, ele exibe o nome do comando que está dentro do script e o seu resultado. Para desabilitar essa exibição dos comandos na hora de executar um script, devemos começar o script com o comando @echo off, ficando desta maneira:

@echo off

cls

echo Dia de hoje:

echo %date%

echo Hora atual:

echo %time%

* + 1. O comando @echo off faz com que os comandos que estão nos scripts não apareçam durante sua execução, mostrando apenas seus resultados.
    2. É útil quando queremos deixar a execução dos scripts mais limpas, ou quando temos alguma informação sensível no script que não desejamos que usuário veja.