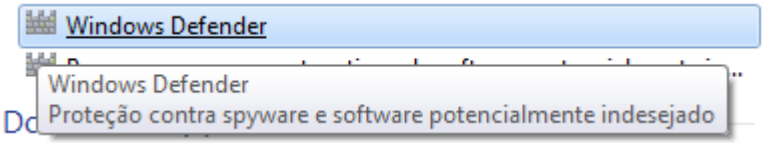
S.O. – Windows

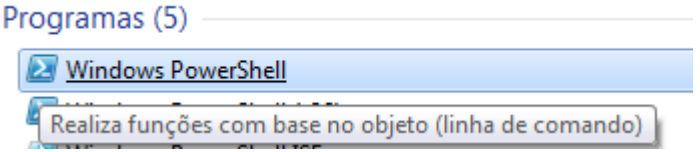
Aula 07

Windows Defender



- **Proteção em tempo real.** O Windows Defender emite um alerta quando algum spyware tenta se executar ou se instalar no computador. Ele também emite um alerta quando alguns programas tentam alterar configurações importantes do Windows.
- **Opções de verificação.** É possível usar o Windows Defender para verificar se há spyware instalado no computador, agendar verificações regulares e remover automaticamente tudo o que for detectado em uma verificação.

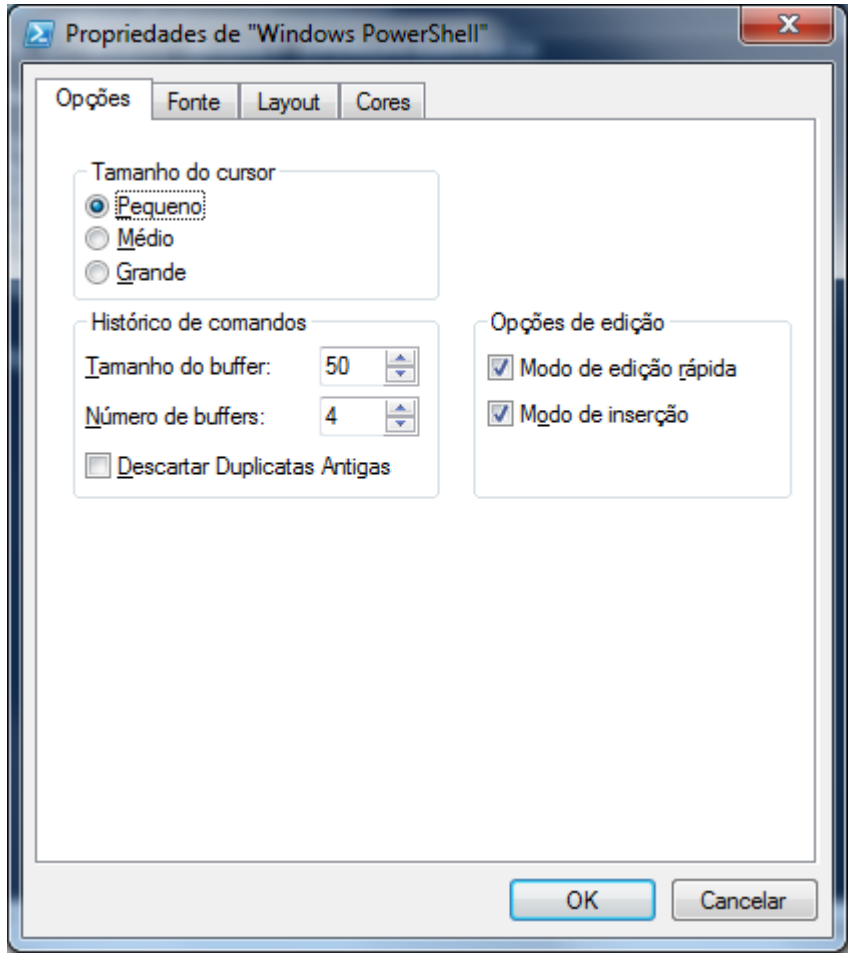
Windows PowerShell



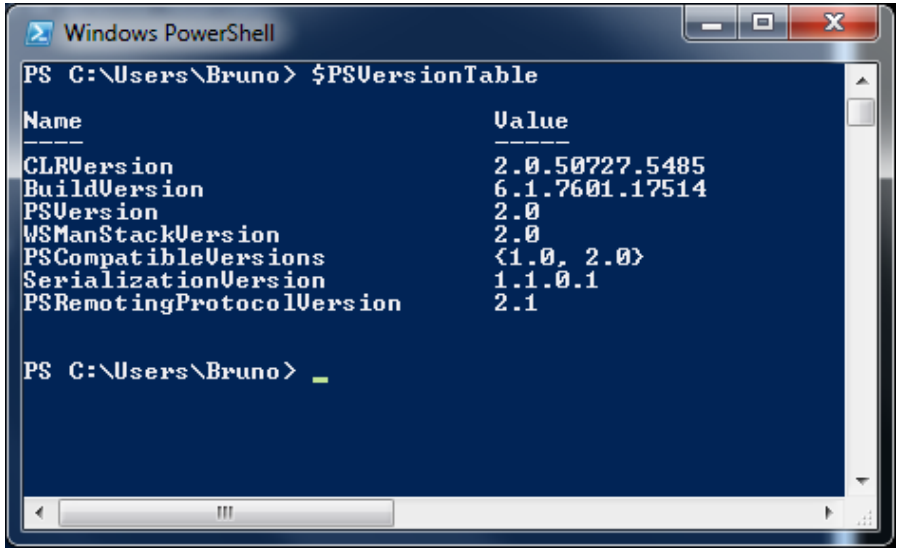
- Microsoft **Windows PowerShell** é um novo prompt de comando do **Windows**, muito mais poderoso que o cmd.exe, voltado à automatização, via scripts e canalização de objetos por uma sequência de comandos, para manutenção de sistemas por parte de administradores, além de um controle maior do sistema.

Windows PowerShell

Alterando algumas configurações da janela:

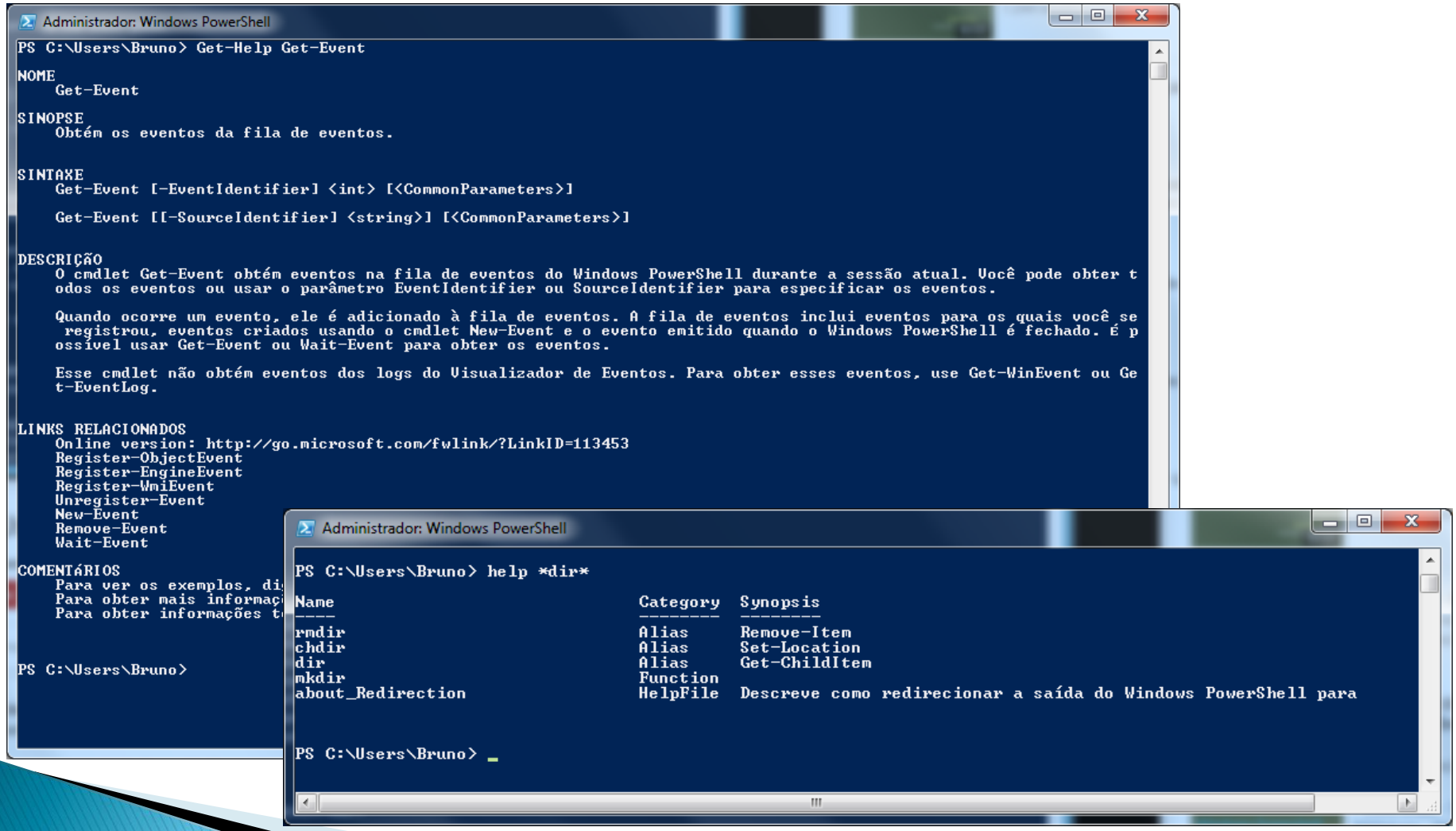


Verificando Versão:



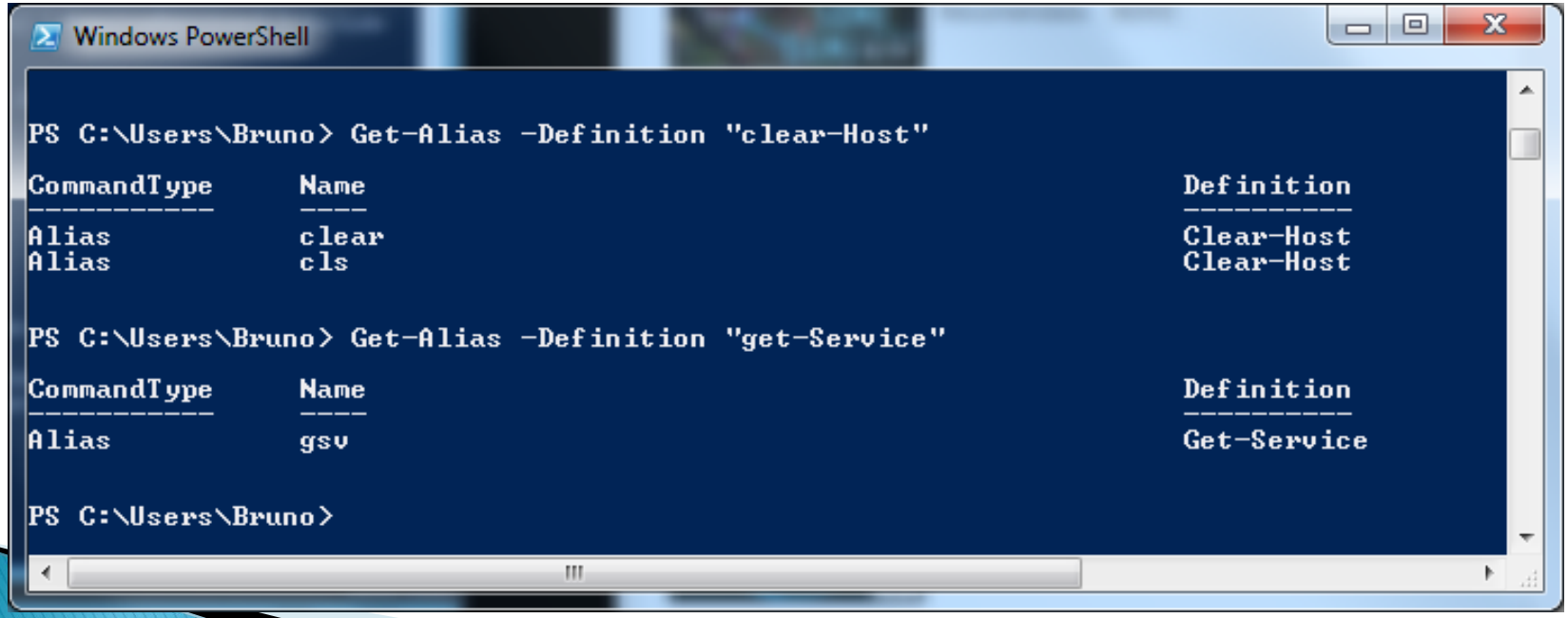
Windows PowerShell

Comando HELP



Windows PowerShell

- Comandos Legados (aliases):
  - cls = Clear-Host
  - dir = Get-ChildItem
    - Get-Alias (para verificar demais comandos)



Windows PowerShell

- **cmdlets** do PowerShell
  - Sintaxe: Estilo Verbo-Substantivo
- Anatomia de um comando:

Comando	Parâmetro 01	Parâmetro 02
Get-EventLog	-LogName Security	-ComputerName Bruno

- Parâmetros podem se apresentar de várias formas:
  - Parâmetro / Valor
  - Parâmetro / Valores múltiplos
  - Parâmetro sem valores

Windows PowerShell

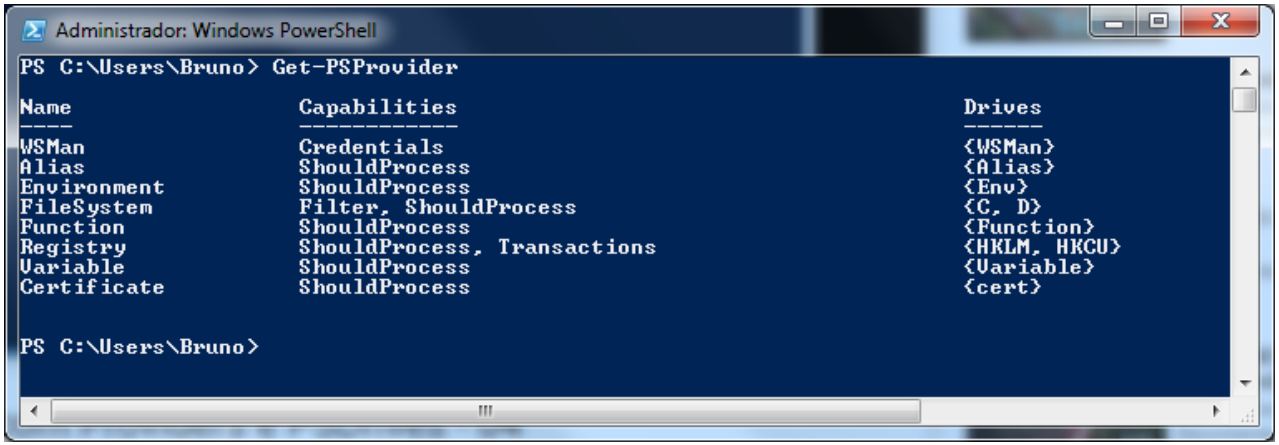
- Visualizando a lista de verbos aprovados
  - Get-Verb
- Usando Aliases
  - Alias = Nome alternativo de um cmdlet
- Exemplo:
  - cmdlet = Get-ChildItem
  - Alias = dir
  - Alias = ls

Windows PowerShell

- Criando seu próprio Alias
  - Exemplo:
  - New-Alias -Name serv -Value Get-Service
- Todos comandos criados com New-Alias são perdidos quando se fecha o Shell, para se manter tem que adicionar os Alias no script do perfil do usuário que está executando o PowerShell.

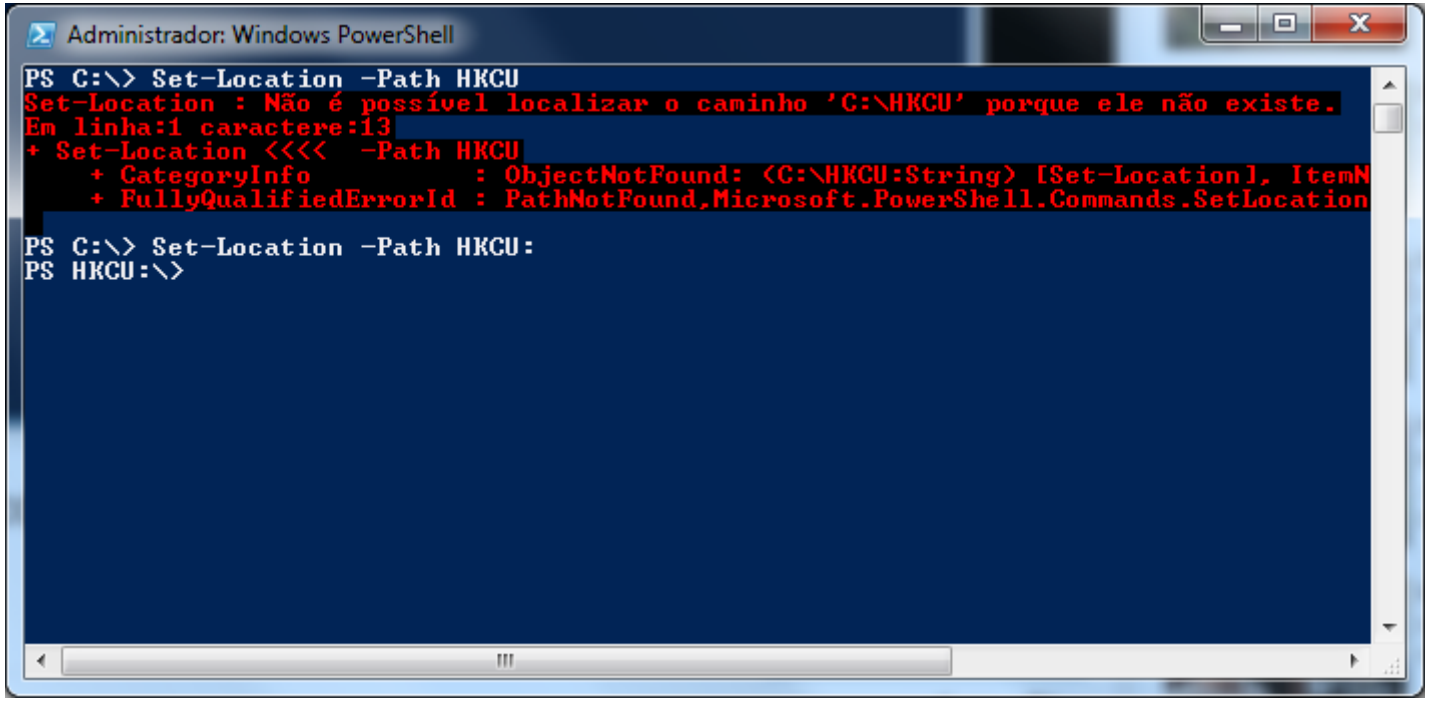
Windows PowerShell

- Providers do PowerShell
  - PSProvider = adaptador.
- Get-PSProvider



- PSDrive – Conexões a vários tipos de storage
- Get-PSDrive = mostra os drives conectados atualmente ao PowerShell.

Windows PowerShell

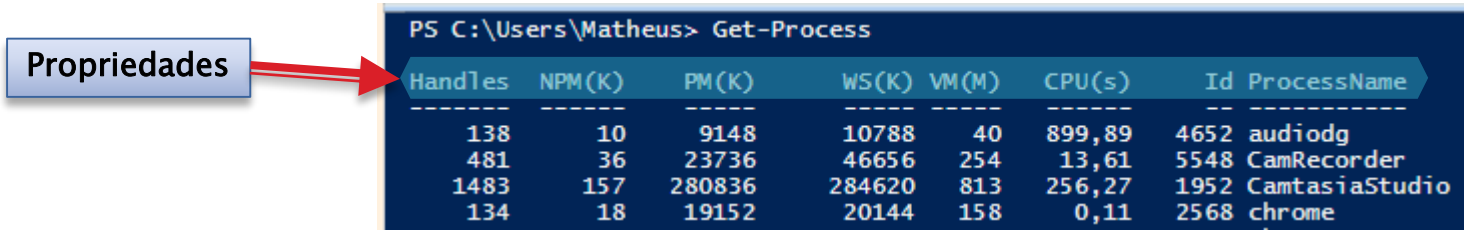


Windows PowerShell

- **Variáveis**
  - $Nome = “Bruno“
  - $Idade = 32
    - Se quiser especificar o tipo de variável:
      - [String]$Nome = “Bruno“
- **Vetores**
  - $Nomes = “Bruno”,“Vinícius”,“Roberto”
  - $Nomes[2] (Exibirá o nome Roberto)
    - $Nomes += “Adão” (aumentando um vetor existente)

Windows PowerShell

- **Objetos**
  - São entidades que possuem propriedades e métodos.



  - Em cima dessas propriedades podemos aplicar os operadores comparativos e condicionais.
    - $Process = Get-Process
    - $Process[0]
  - Podemos listar todas propriedades e métodos de um objeto utilizando o comando: **Get-Process | Get-Member** que nos retorna a lista das propriedades e métodos desse objeto, se quisermos acessar uma propriedade ou um método do objeto pode fazer isso dessa maneira.
    - $Process[0].Name

Windows PowerShell

- **Operadores Comparativos**
  - **-eq**, Igual a, utilizado para comparar números e textos.
  - **-lt**, Menor que, Utilizado para comparar números.
  - **-gt**, Maior que, utilizado para comparar números.
  - **-ge**, Maior ou igual a, utilizado para comparar números.
  - **-le**, Menor ou igual a, utilizado para comparar números.
  - **-ne**, Não igual a, utilizado para comparar números e textos.
  - **-contains**, Contém, utilizado para comparar vetores.
  - **-like**, Igual, utilizado para comparar textos ou números.
  - **-notlike**, Não igual a.
  - **-is**, é, utilizado para comparar tipos de objetos.
  - **-isnot**, não é, utilizado para comparar tipos de objetos

Windows PowerShell

- **Operadores Condicionais**
  - **if**, verifica se a condição for verdadeira.
  - **else**, utilizado após o **if,** verifica se a condição não foi verdadeira.
  - **elseif**,utilizado após um **if** ou outro **elseif,** verifica se a condição anterior não for verdadeira e uma nova condição for.
  - **while,** utilizado para repetir um bloco enquanto uma condição for verdadeira.
  - **for,** utilizado para repetir um bloco um predeterminado numero de vezes.
  - **foreach**, utilizado para repetir um bloco uma vez para cada item em uma lista.
  - **where,** utilizado para filtrar itens em uma lista.
  - **switch,** utilizado para executar multiplas verificações substituindo uma cadeia de ifs.

Windows PowerShell

- **Exercício**
  - Se eu quisesse exibir em tela de 0 a 19 eu faria o seguinte script:

  - Escreva um script no PowerShell que exiba os números pares de 2 a 20 na tela. Dica: **$i+=2**

Windows PowerShell

- **Redirecionando Saídas dos cmdlets**
  - Por exemplo, queremos matar o processo do Notepad que esteja rodando em nosso computador, podemos executar:
    - **Get-Process -Name "Notepad" | Stop-Process**
  - Quanto utilizamos a pipeline para redirecionar a saída de um comando para outro, ele passa um objeto por vez, sendo o objeto passa pela pipeline referenciado como **$_** do outro lado, sendo assim se você quiser aplicar um filtro utilizando o **Where-Object,** você pode fazer dessa maneira.
    - **Get-Process | Where-Object { $_.Name -like "Notepad" }**

Windows PowerShell

- **Redirecionando Saídas dos cmdlets**
  - Outro comando muito utilizado após a pipeline é o **ForEach-Object**, que serve caso nos queiramos executar uma ação para cada um dos objetos passados na pipeline, por exemplo.
    - **Get-Process | Where-Object { $_.Name -like "N*"} | ForEach-Object {Write-Host $_.Name" começa com N"}**

Windows PowerShell

- **Encontrando os cmdlets**
  - Se eu quero pegar alguma informação já sei que devo procurar um comando que comece com **Get**, se quero alterar alguma coisa procuro por **Set**, se quero criar algo novo procuro por **New** ou **Add**, se quero remover **Remove.**
  - Queremos encontrar algum comando para acharmos algumas informações do disco, então executamos o comando:
    - **Get-Command "Get-*Drive*"**