Perfmon e Power Shell

Paulo Romero Martins Maciel
prmm@cin.ufpe.br
{rsm4, casm3, jrd, prps}@cin.ufpe.br

Agenda

- Introdução
- Conclusão
- Referências

Introdução

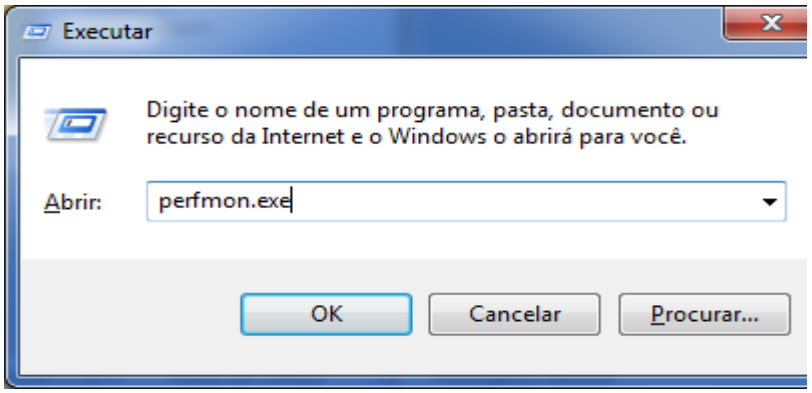
- O **Monitor de Desempenho** (*perfmon.exe*) é a versão melhorada do *System Monitor* (*sysmon.exe)*, que está presente no Windows desde o Windows 95
- Ferramenta útil para detectar gargalos em um servidor de aplicação Windows
- Permite monitorar CPU, disco, memória, rede e processos

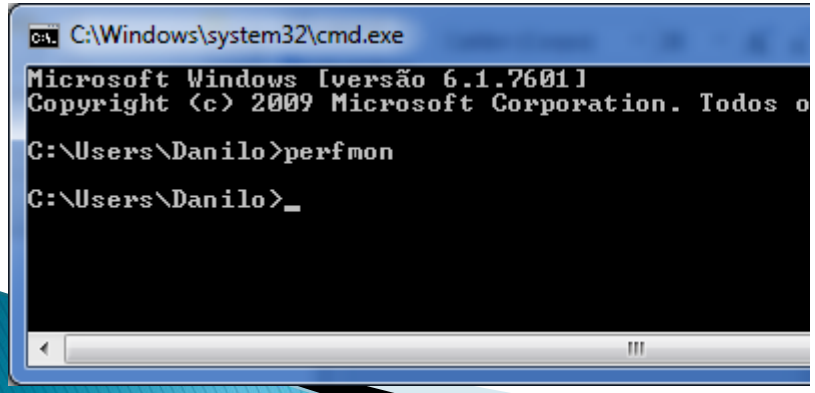
Introdução

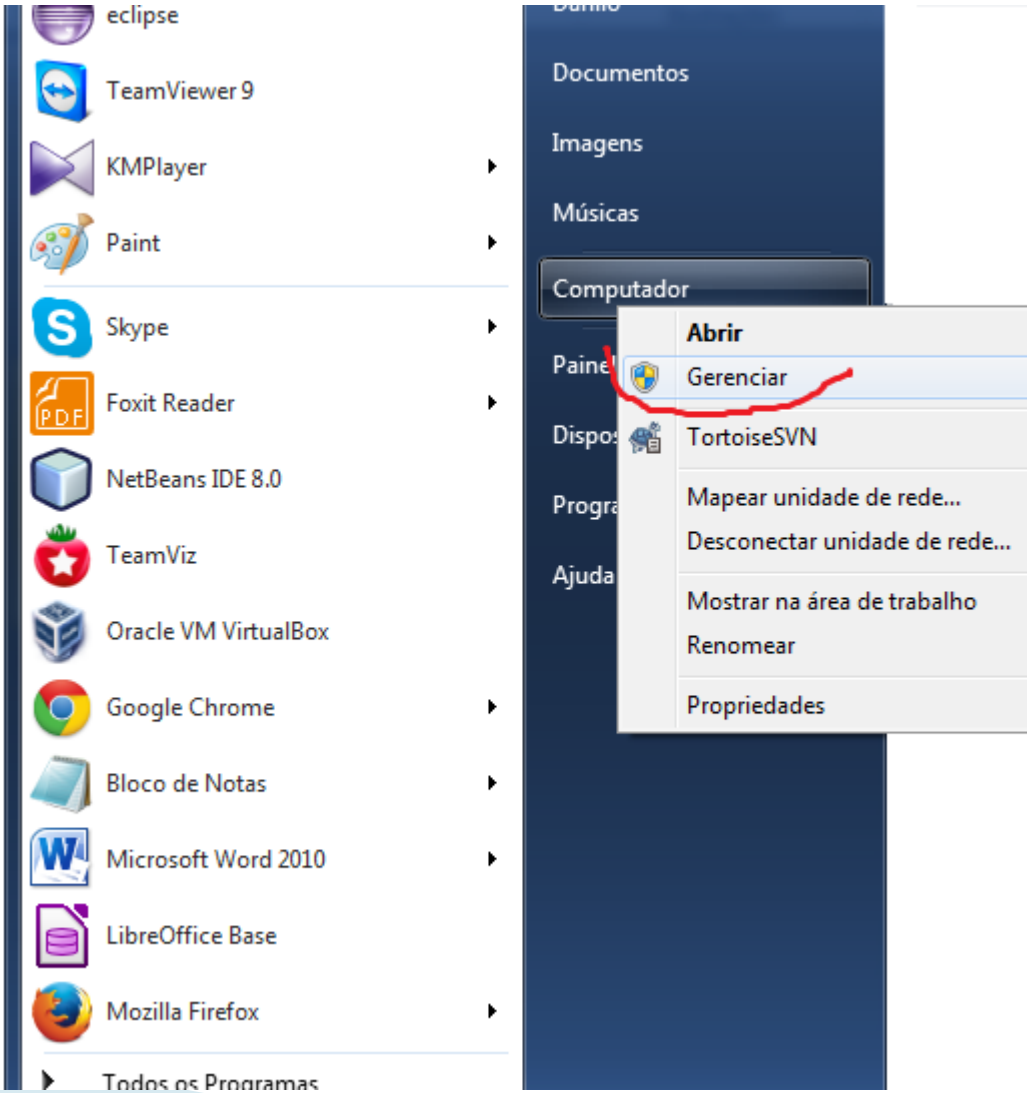
- Funciona por amostragem ou baseado em eventos
- Funciona em tempo real, ou armazenando resultados em um arquivo de log para análise posterior
- Funciona de forma local ou remota

Introdução

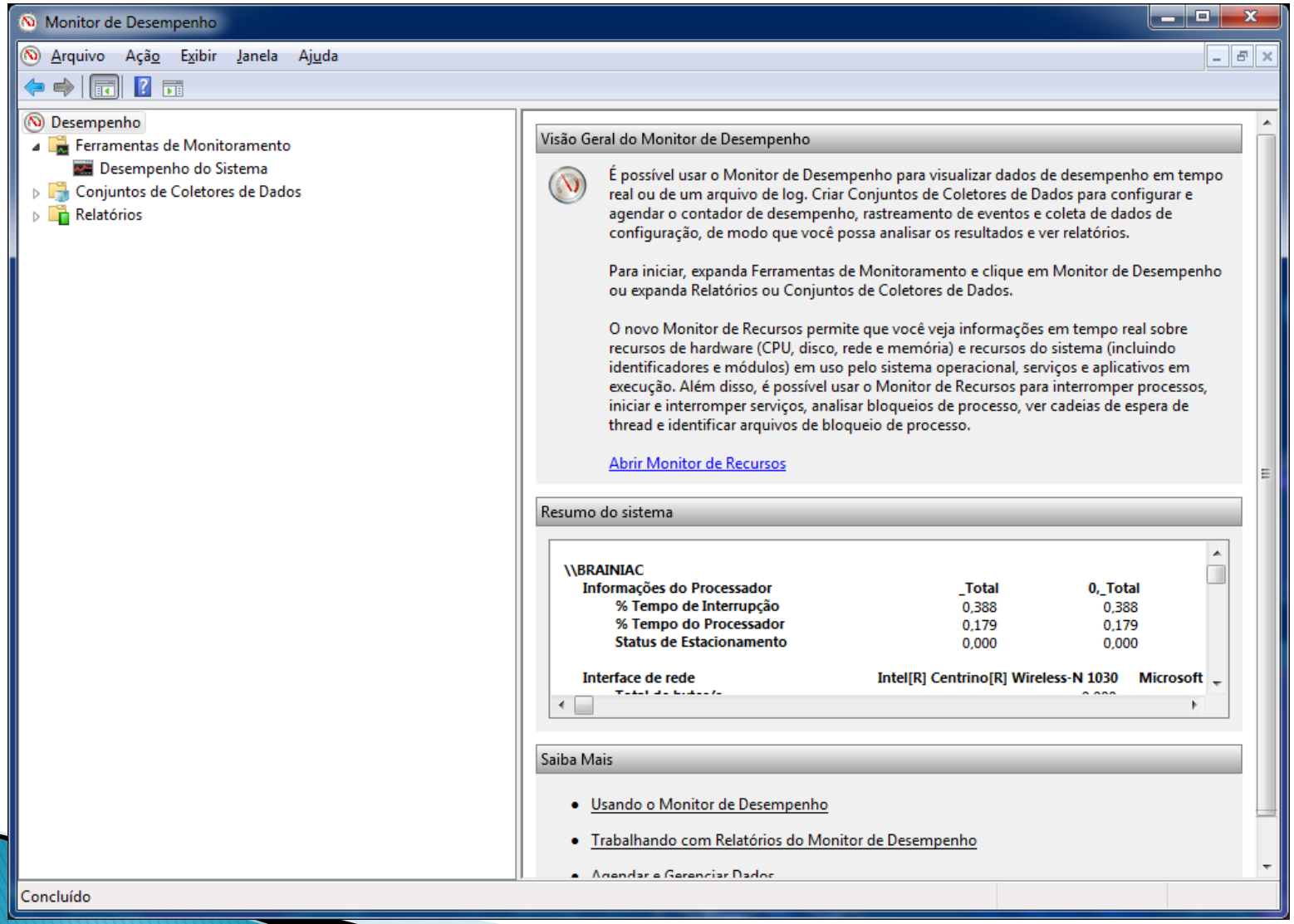
- Executando o Perfmon



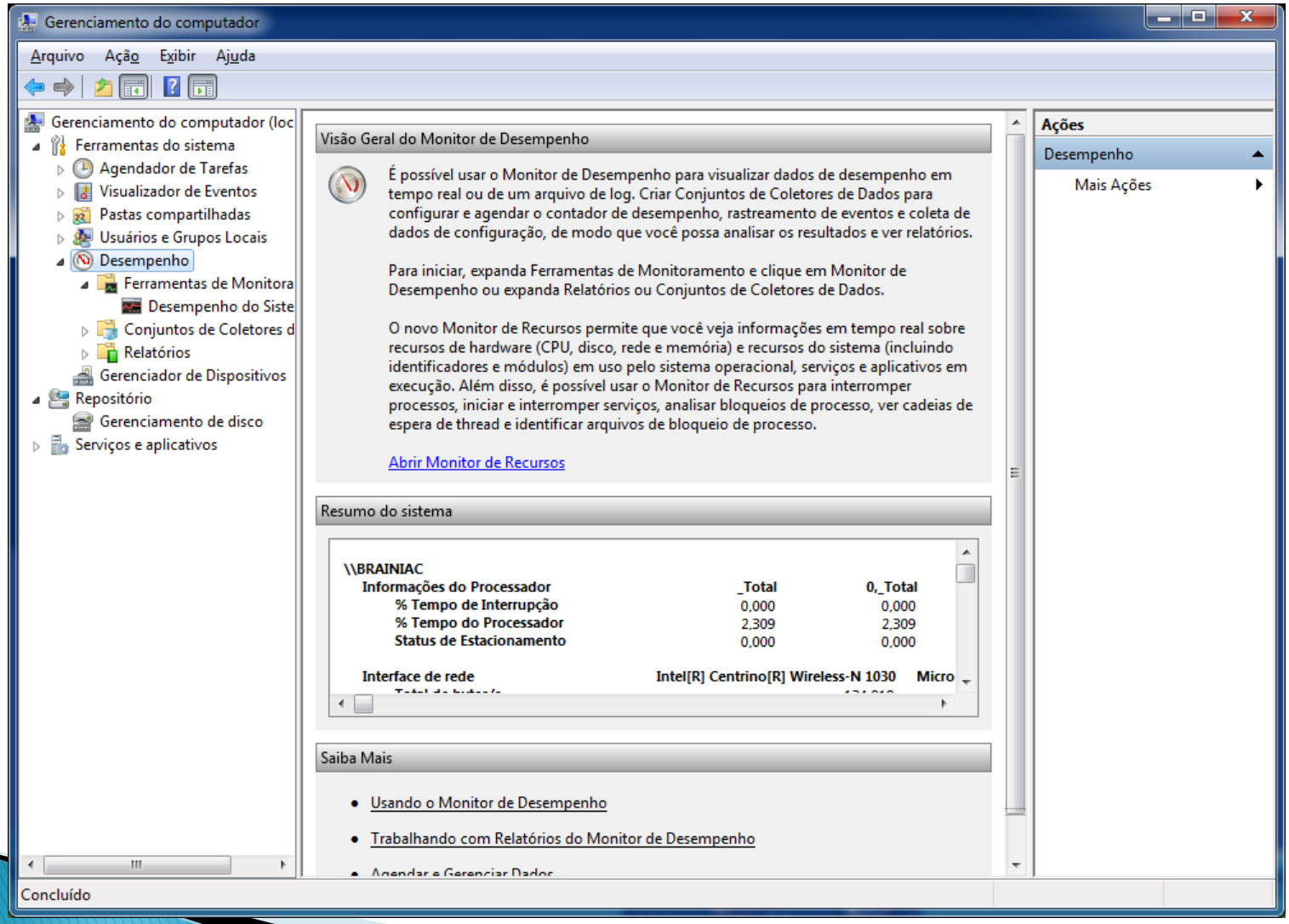




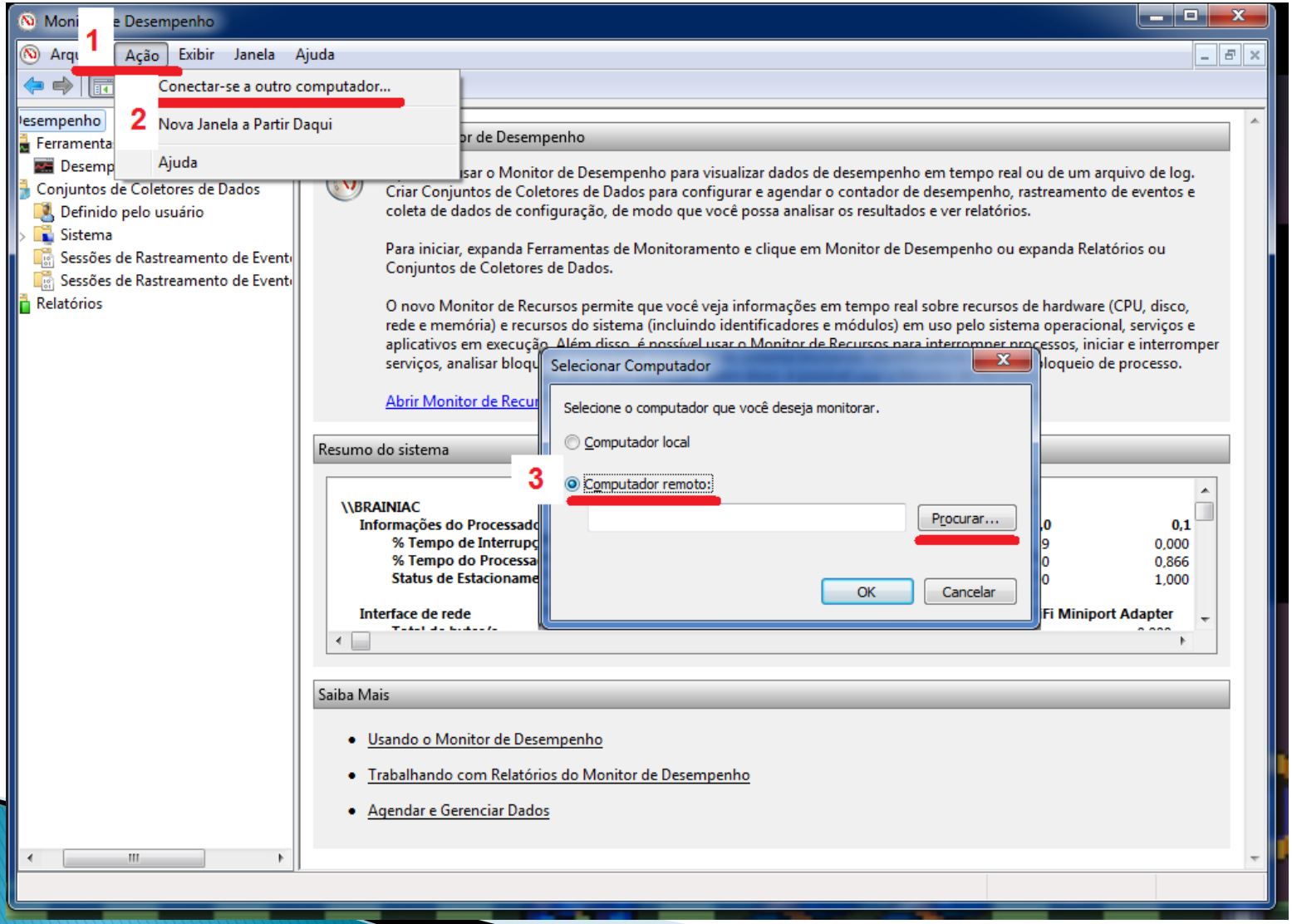
Introdução


Monitor de Desempenho
Arquivo Ação Exibir Janela Ajuda
Desempenho
Ferramentas de Monitoramento
Desempenho do Sistema
Conjuntos de Coletores de Dados
Relatórios
Visão Geral do Monitor de Desempenho
É possível usar o Monitor de Desempenho para visualizar dados de desempenho em tempo real ou de um arquivo de log. Criar Conjuntos de Coletores de Dados para configurar e agendar o contador de desempenho, rastreamento de eventos e coleta de dados de configuração, de modo que você possa analisar os resultados e ver relatórios.
Para iniciar, expanda Ferramentas de Monitoramento e clique em Monitor de Desempenho ou expanda Relatórios ou Conjuntos de Coletores de Dados.
O novo Monitor de Recursos permite que você veja informações em tempo real sobre recursos de hardware (CPU, disco, rede e memória) e recursos do sistema (incluindo identificadores e módulos) em uso pelo sistema operacional, serviços e aplicativos em execução. Além disso, é possível usar o Monitor de Recursos para interromper processos, iniciar e interromper serviços, analisar bloqueios de processo, ver cadeias de espera de thread e identificar arquivos de bloqueio de processo.
Abrir Monitor de Recursos
Resumo do sistema
\\BRAINIAC
Informações do Processador _Total 0,_Total
% Tempo de Interrupção 0,388 0,388
% Tempo do Processador 0,179 0,179
Status de Estacionamento 0,000 0,000
Interface de rede Intel[R] Centrino[R] Wireless-N 1030 Microsoft
Saiba Mais
Usando o Monitor de Desempenho
Trabalhando com Relatórios do Monitor de Desempenho
Concluído

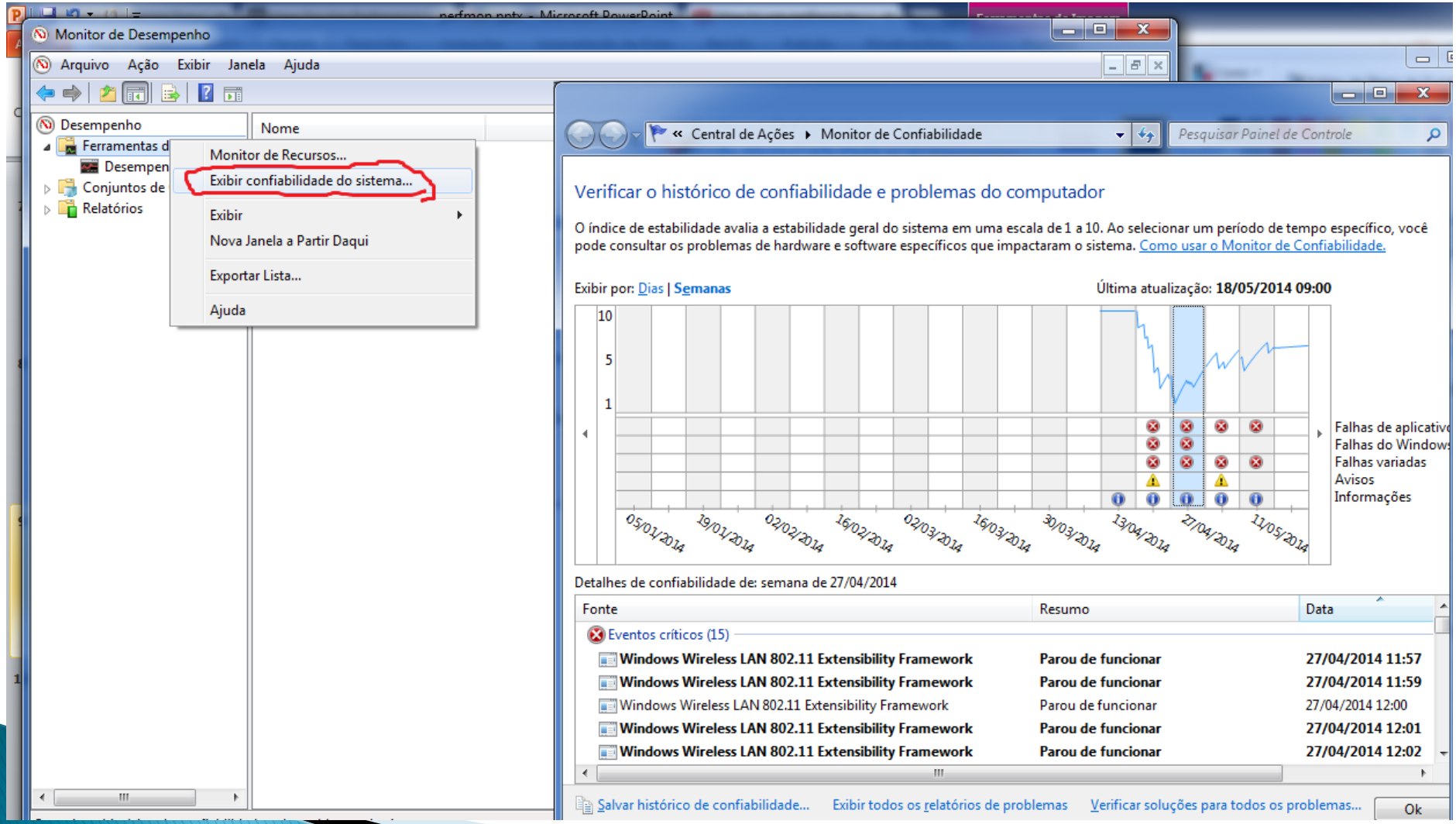
Gerenciamento de computador



Conectando a um computador remoto



Monitor de confiabilidade



Introdução

- O Monitor de Desempenho coleta dados a partir de três fontes
  - **Contador de desempenho**: refletem parte do estado do sistema ou atividade
  - **Rastreamento de eventos**: permitem escutar determinados eventos de um sistema ou aplicação
  - **Informação de configuração**: coletado a partir de informações do registro do Windows
- O Monitor de Desempenho agrupa várias métricas coletadas a partir das fontes acima em uma unidade chamada **Conjunto de Coletores de Dados**

Contadores de desempenho

- Contadores de desempenho de processador
  - Processador\% Tempo do Processador
    - Intervalo aceitável*: 0 – 85%
  - Processador\% Tempo de Usuário
  - Processador\% Tempo de Interrupção
    - Intervalo aceitável: 0 – 15%
  - Sistema\Comprimento da Fila de Processador
    - Intervalo aceitável: 0 – duas vezes o número de cpus
- Soluções
  - Otimizar aplicativo
  - Upgrade da CPU

* Apenas uma sugestão

Contadores de desempenho

- Contadores de desempenho da memória
  - Memória\% Bytes Confirmados em Uso
    - Intervalo aceitável: 0 – 80%
  - Memória\% Mbytes Disponíveis
    - Intervalo aceitável: 5% do total da Ram – 100%
  - Memória\Entradas Livres de Tabela de Paginação do Sistema
    - Intervalo aceitável: 5000 – inf
  - Memória\Bytes de Pool Não-Paginável
  - Memória\Bytes de Pool Paginável

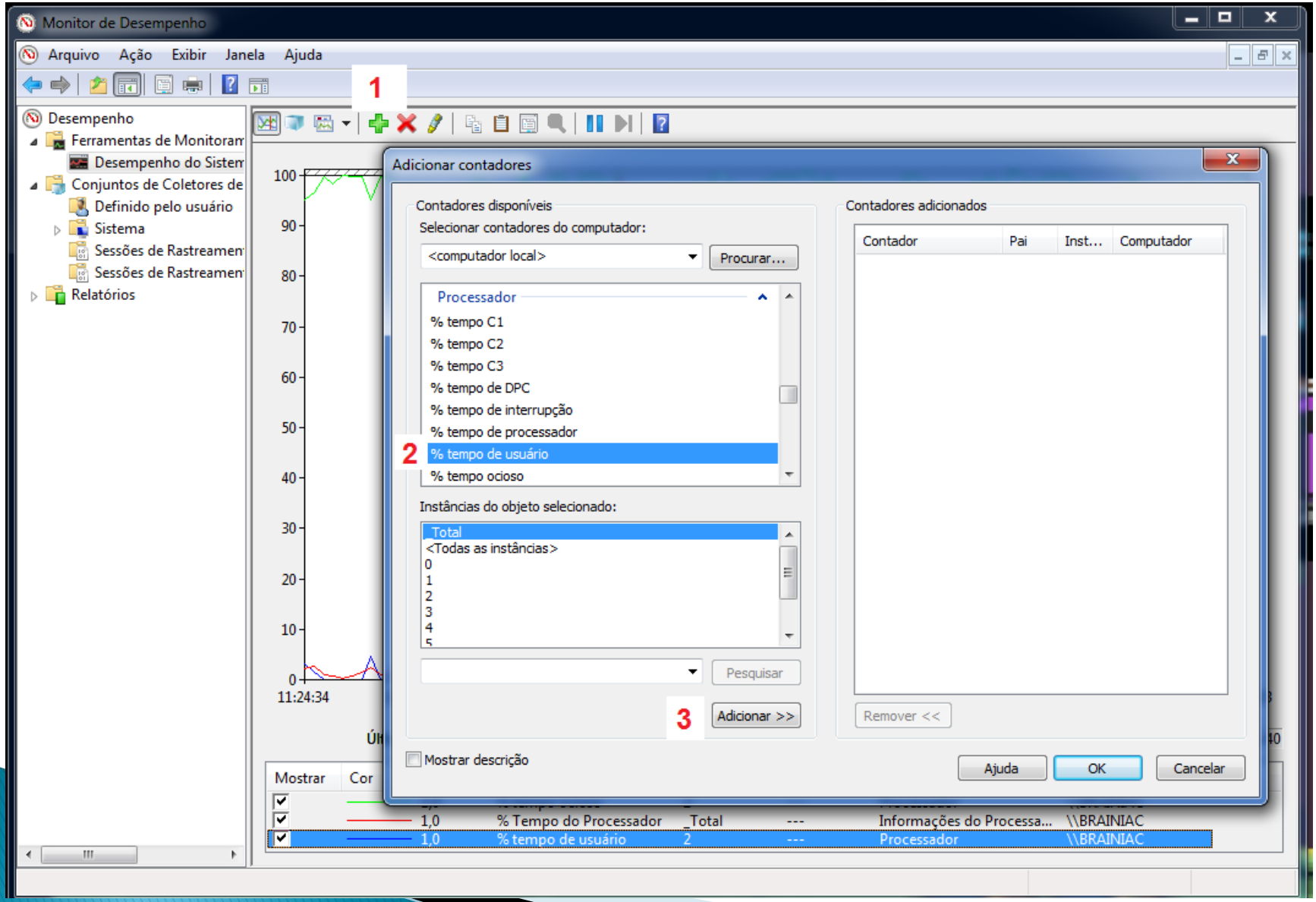
Contadores de desempenho

- Contadores de desempenho do disco
  - LogicalDisk\% Espaço Livre
    - Intervalo aceitável: 15% – 100%
  - PhysicalDisk\% Tempo Ocioso
    - Intervalo aceitável: 20% – 100%
  - PhysicalDisk\Média de Disco s/Leitura
    - Intervalo aceitável: 0 – 25ms
  - PhysicalDisk\Média de Disco s/Gravação
    - Intervalo aceitável: 0 – 25ms

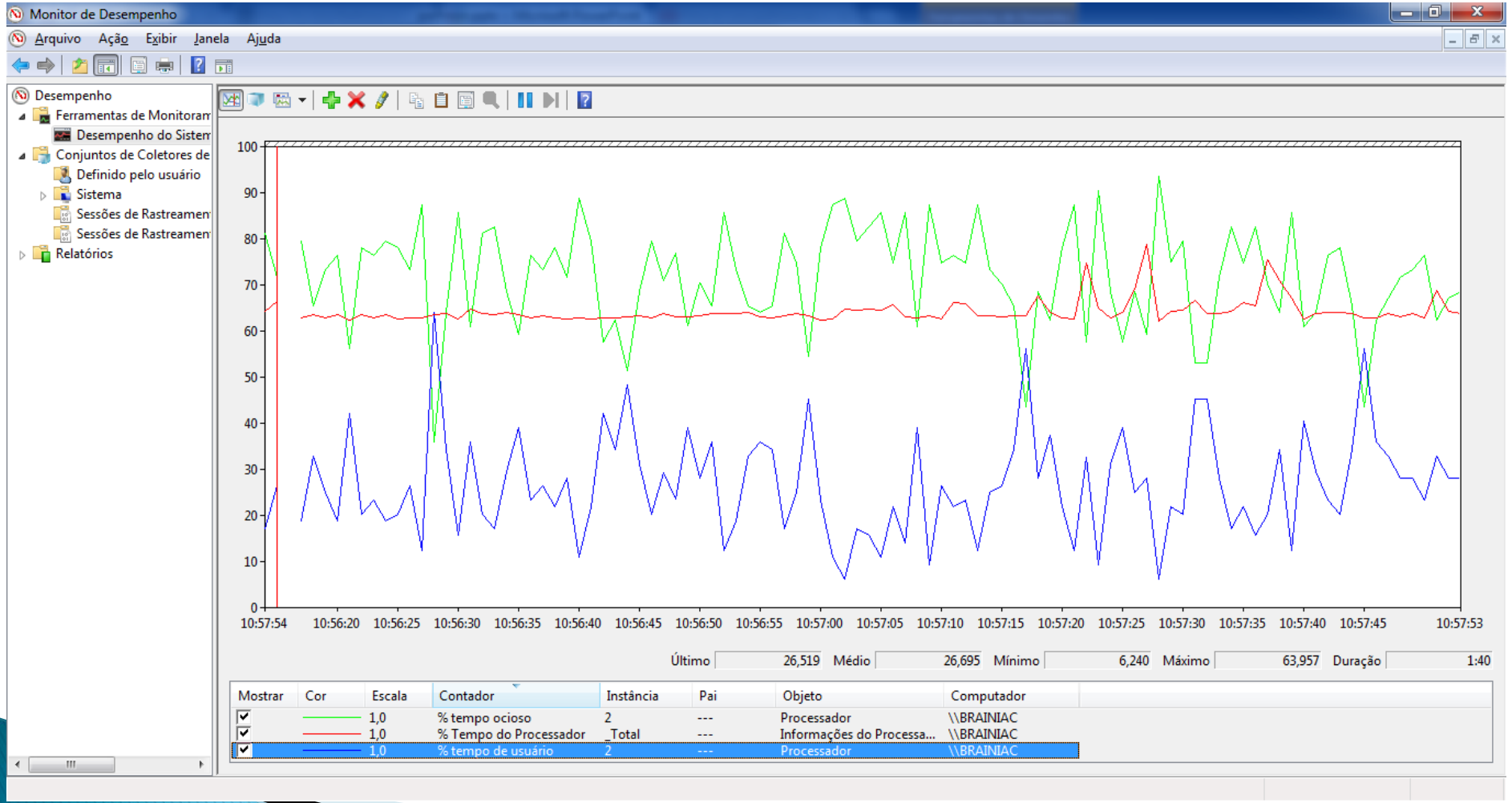
Contadores de desempenho

- Contadores de desempenho da rede
  - Interface de Rede\Total de Bytes/s
    - Intervalo aceitável: 0 – 75%
  - Interface de Rede\Comprimento da Fila de Saída
    - Intervalo aceitável: 0 – 2
- Soluções:
  - Segmentar a rede
  - Substituir a interface de rede

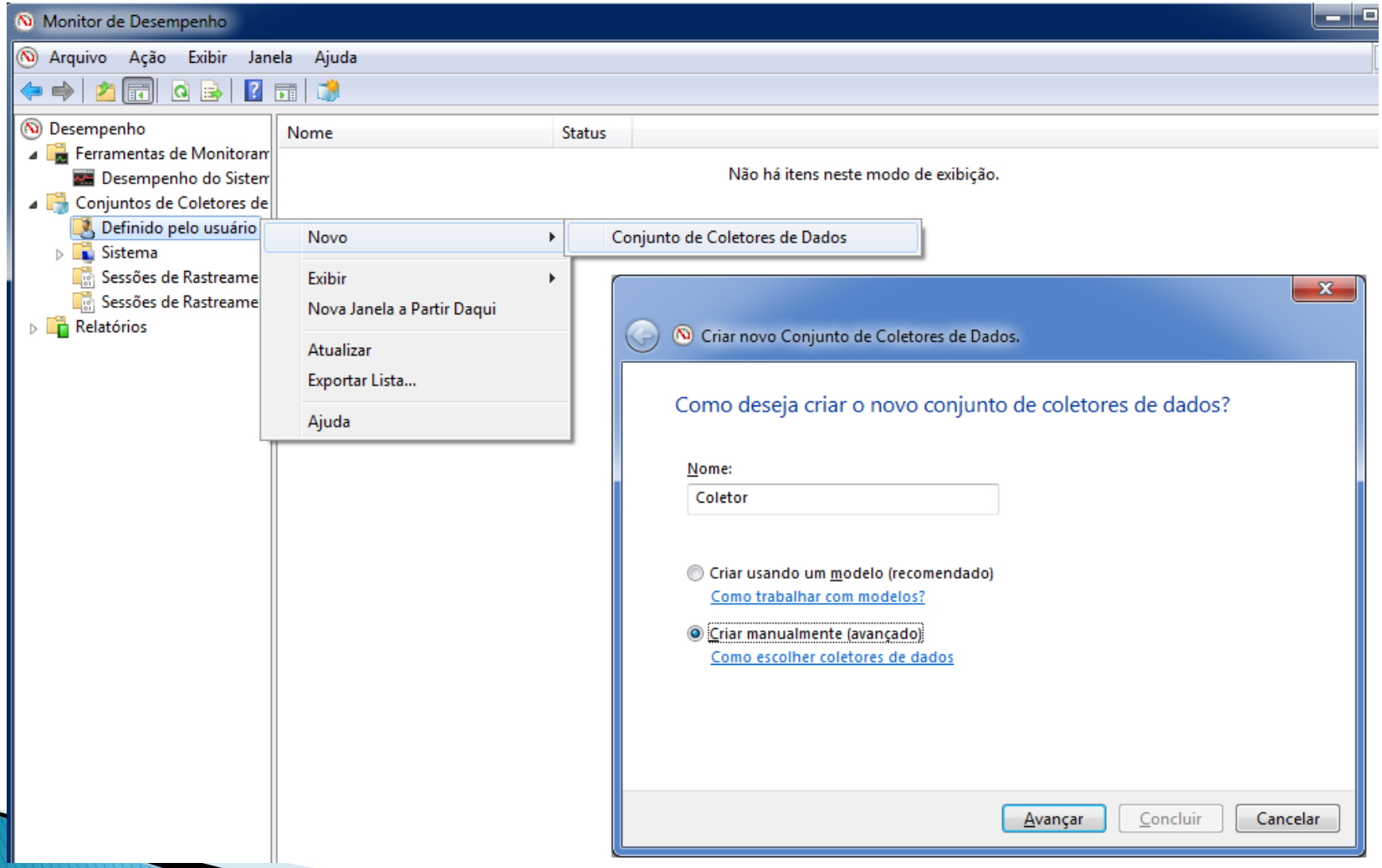
Monitorando em tempo real



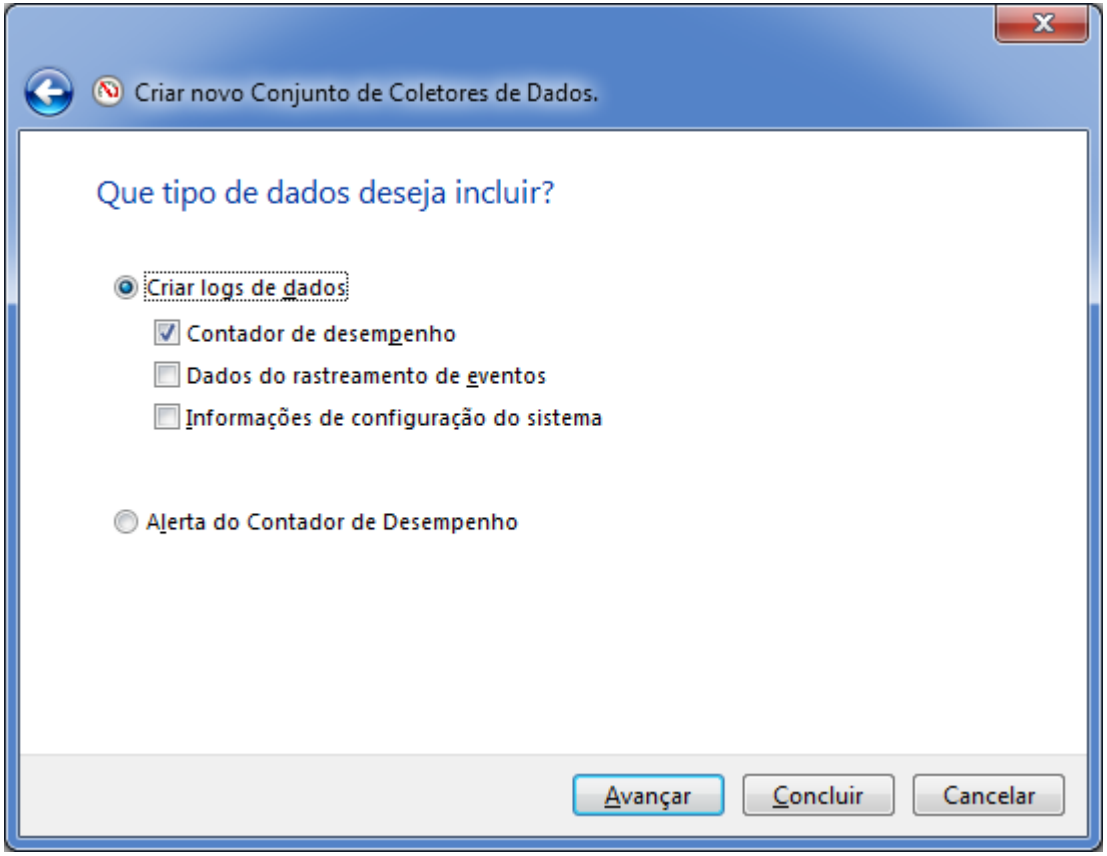
Monitorando em tempo real



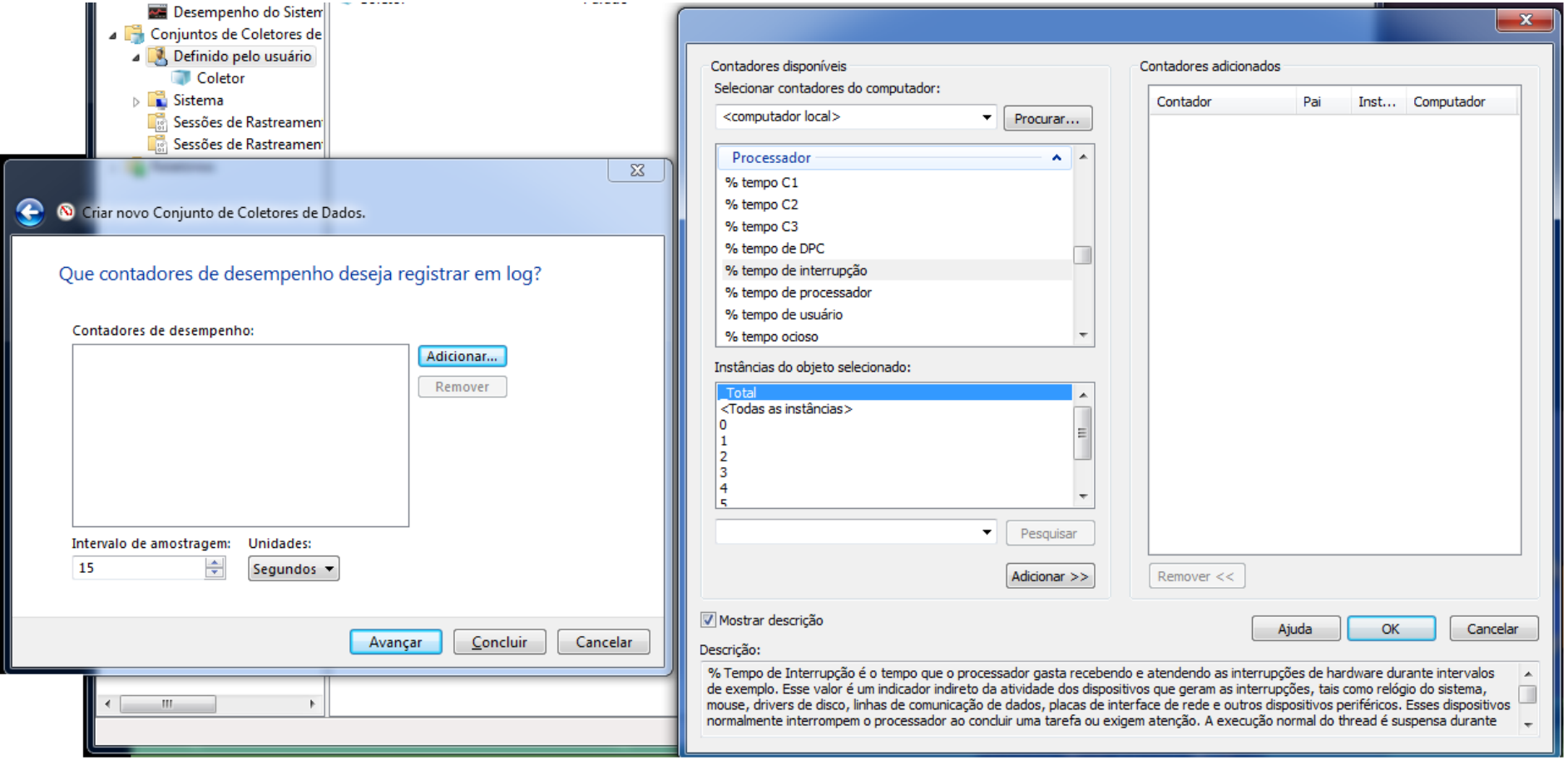
Criando Conjunto de Coletor de Dados



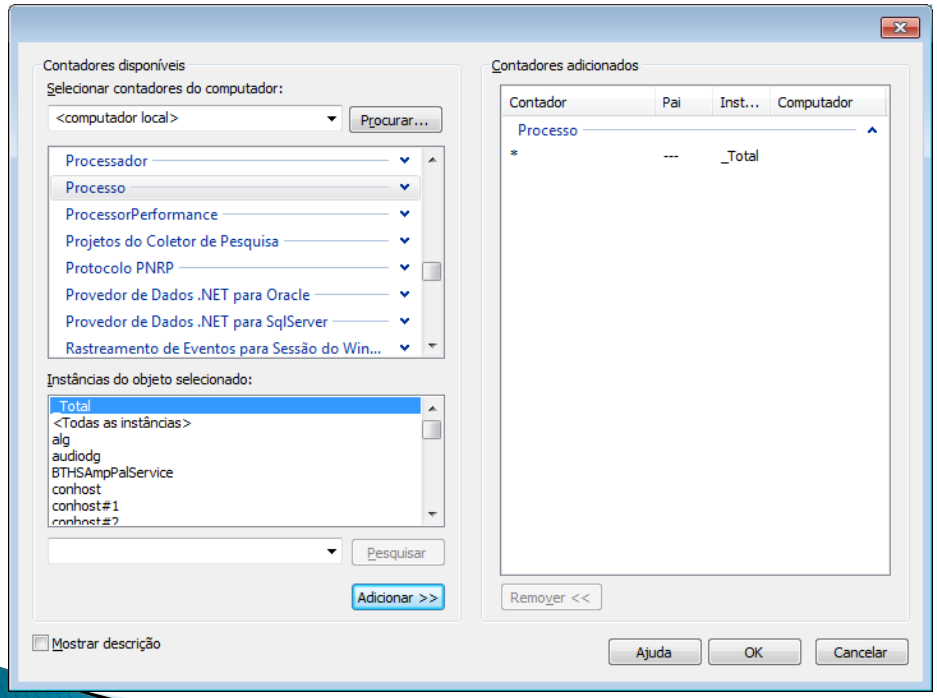
Criando Conjunto de Coletor de Dados

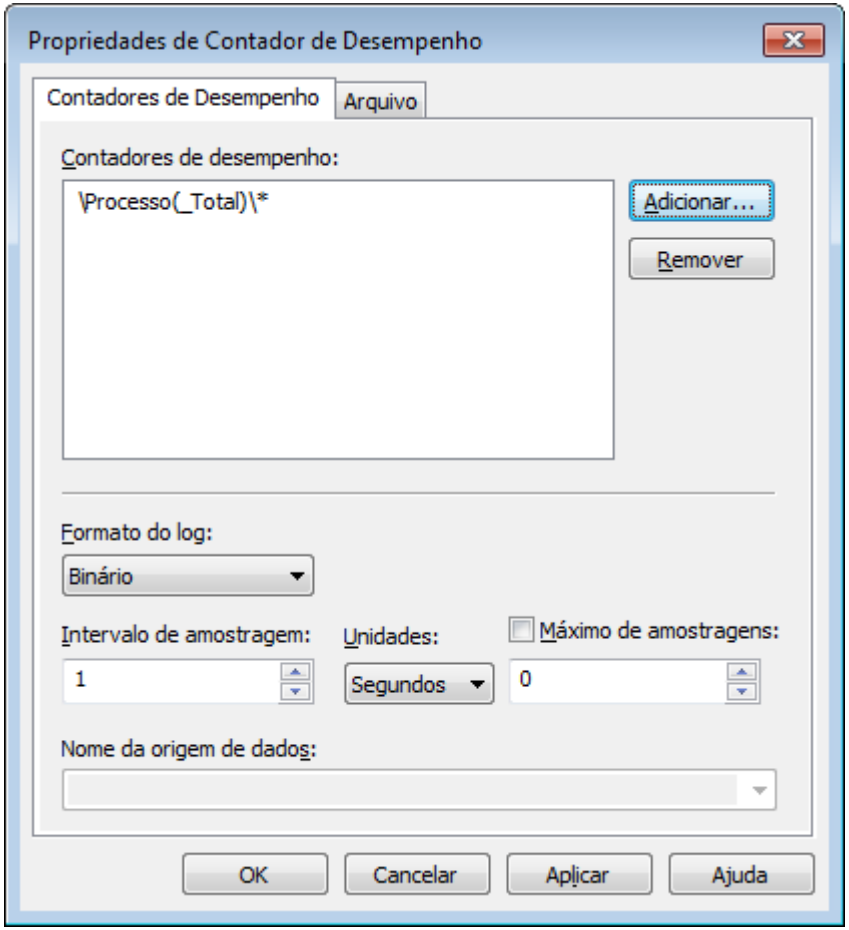


Criando Conjunto de Coletor de Dados

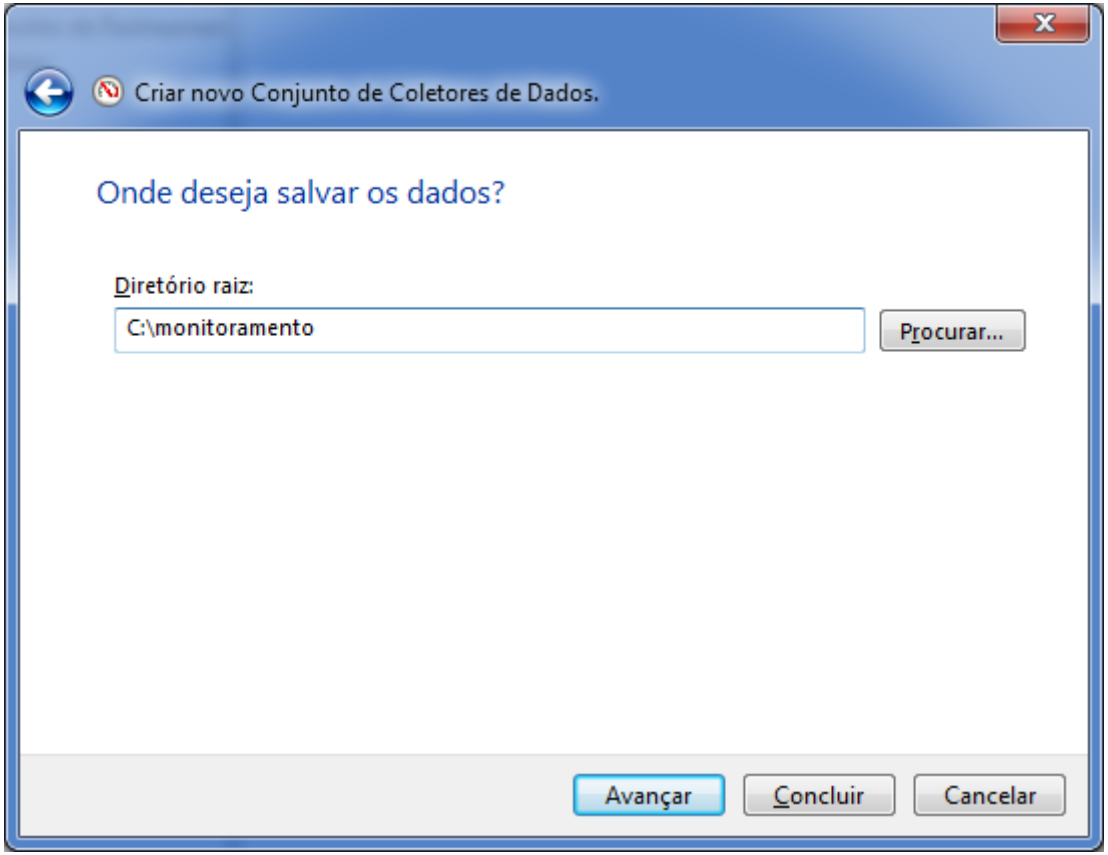


Adicionando todos os contadores de uma categoria





Criando Conjunto de Coletor de Dados

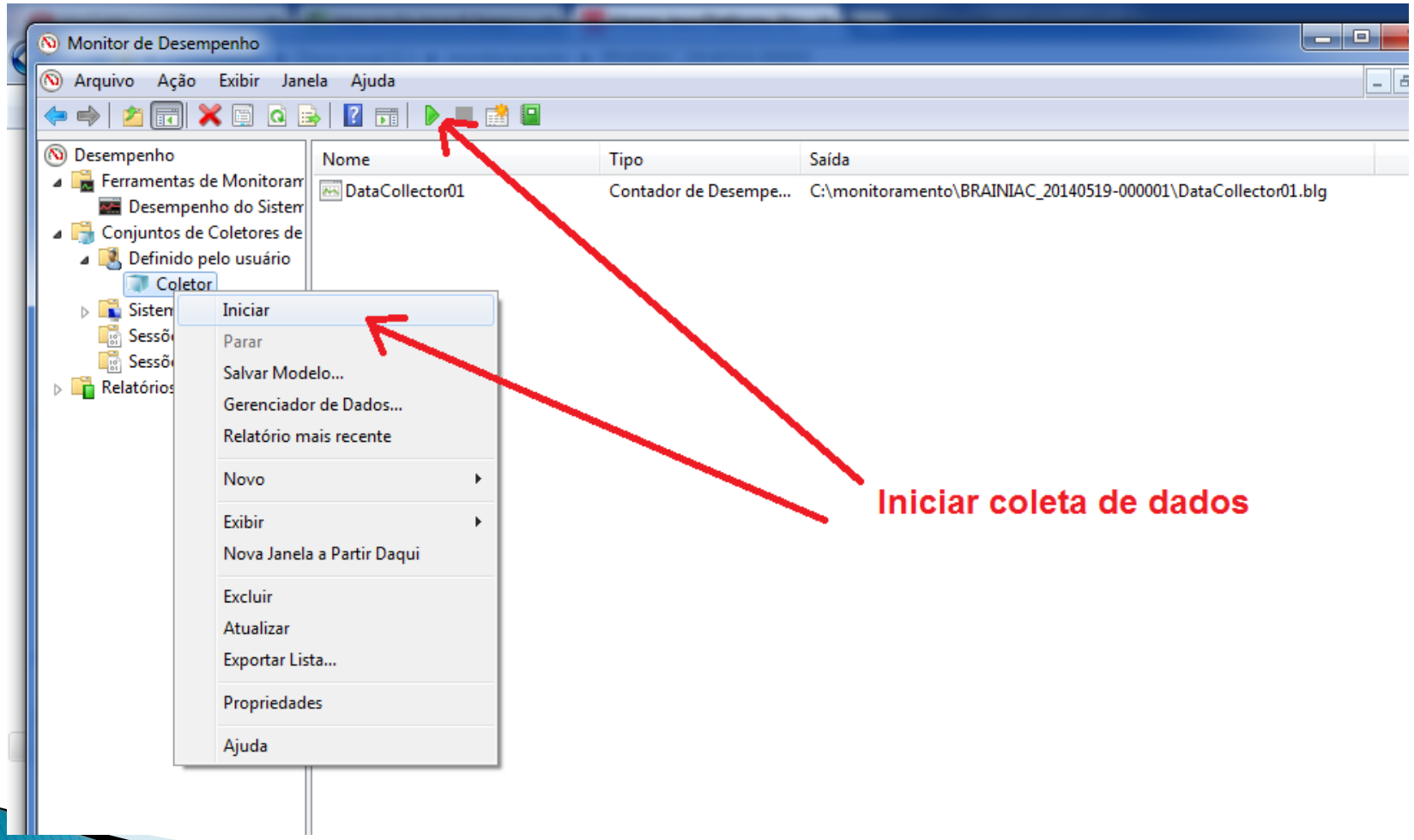


Criando Conjunto de Coletor de Dados

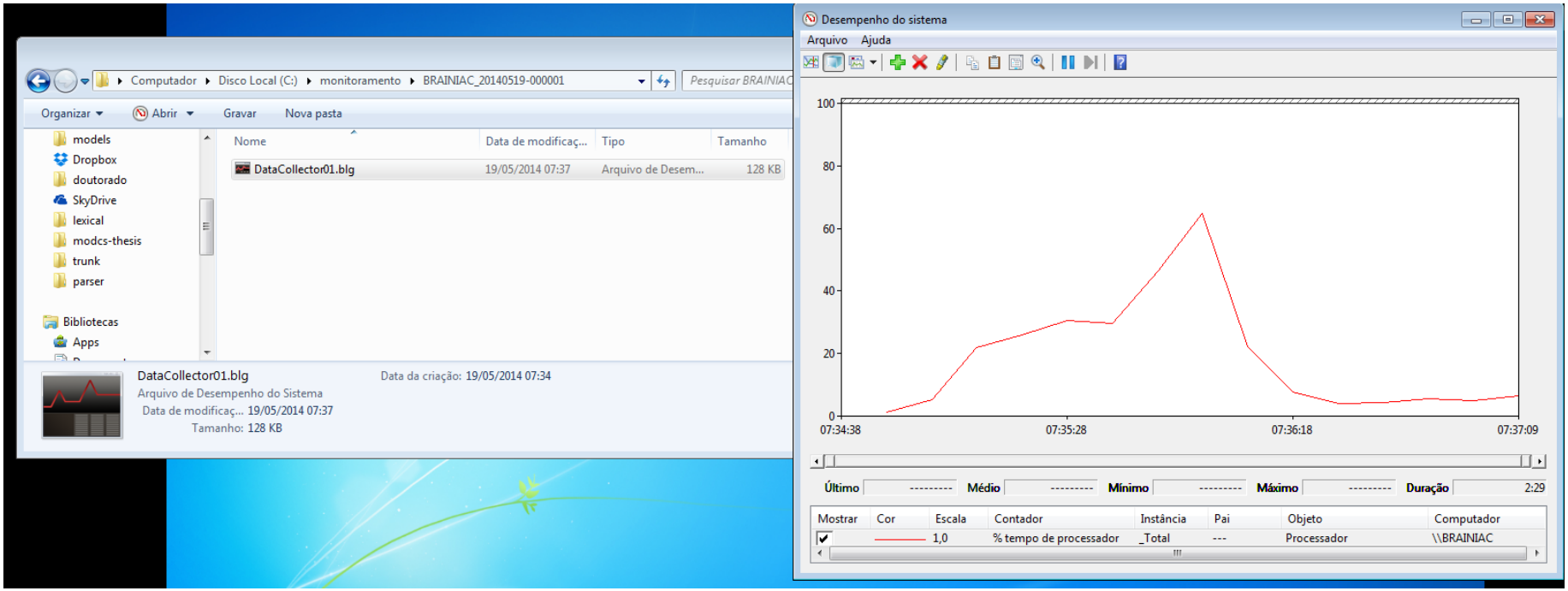
Criando Conjunto de Coletor de Dados

- Intervalo de amostragem depende do total de tempo que você vai monitorar o sistema
- Um intervalo de amostragem menor do que o necessário pode implicar em duas coisas:
  - Monitoramento causando overhead no sistema
  - Arquivo de log muito grande
- Um intervalo grande demais também é prejudicial:
  - Alguns eventos podem passar despercebidos
  - Afeta o tempo de reação, no caso de análise em tempo real

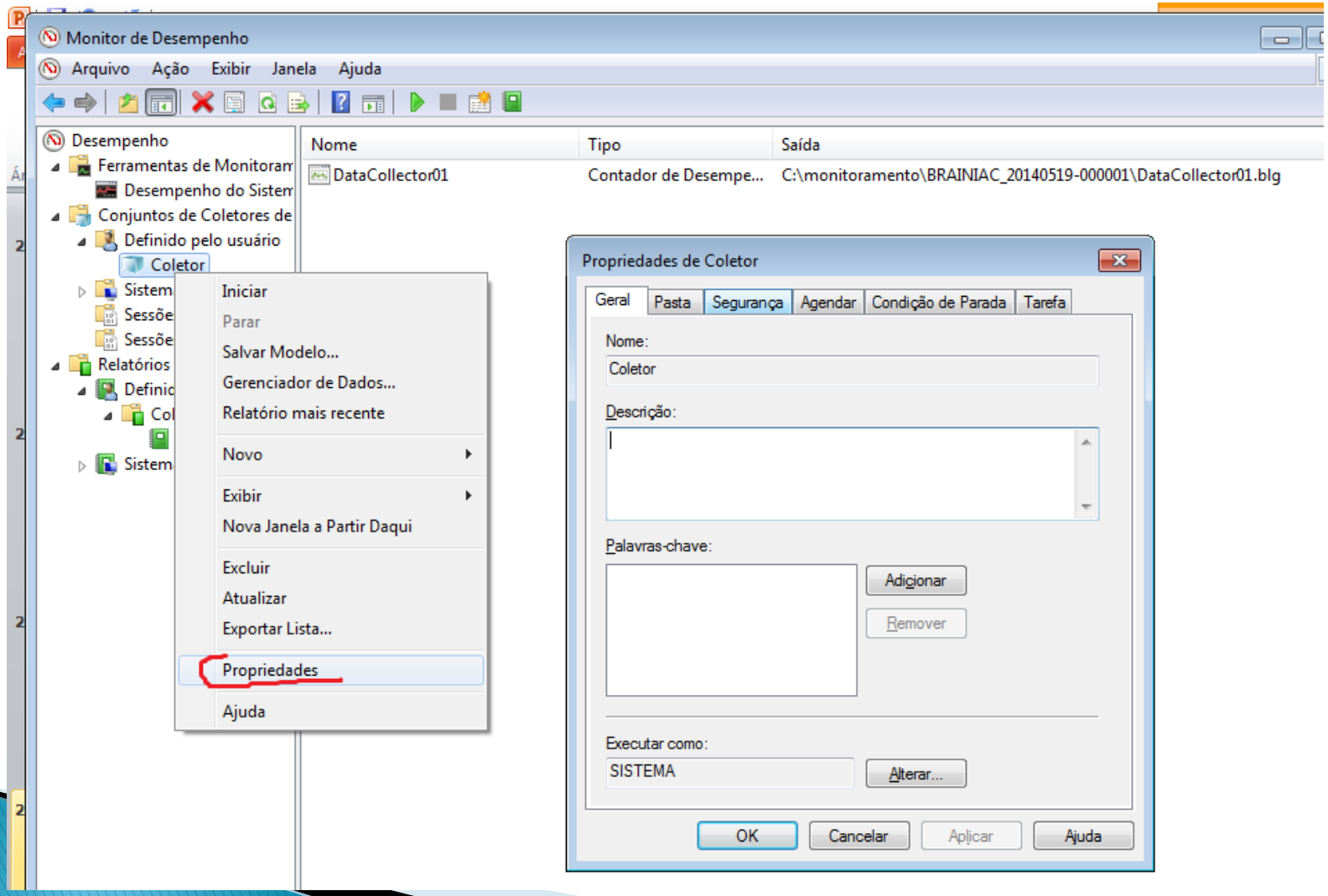
Iniciando captura dos dados



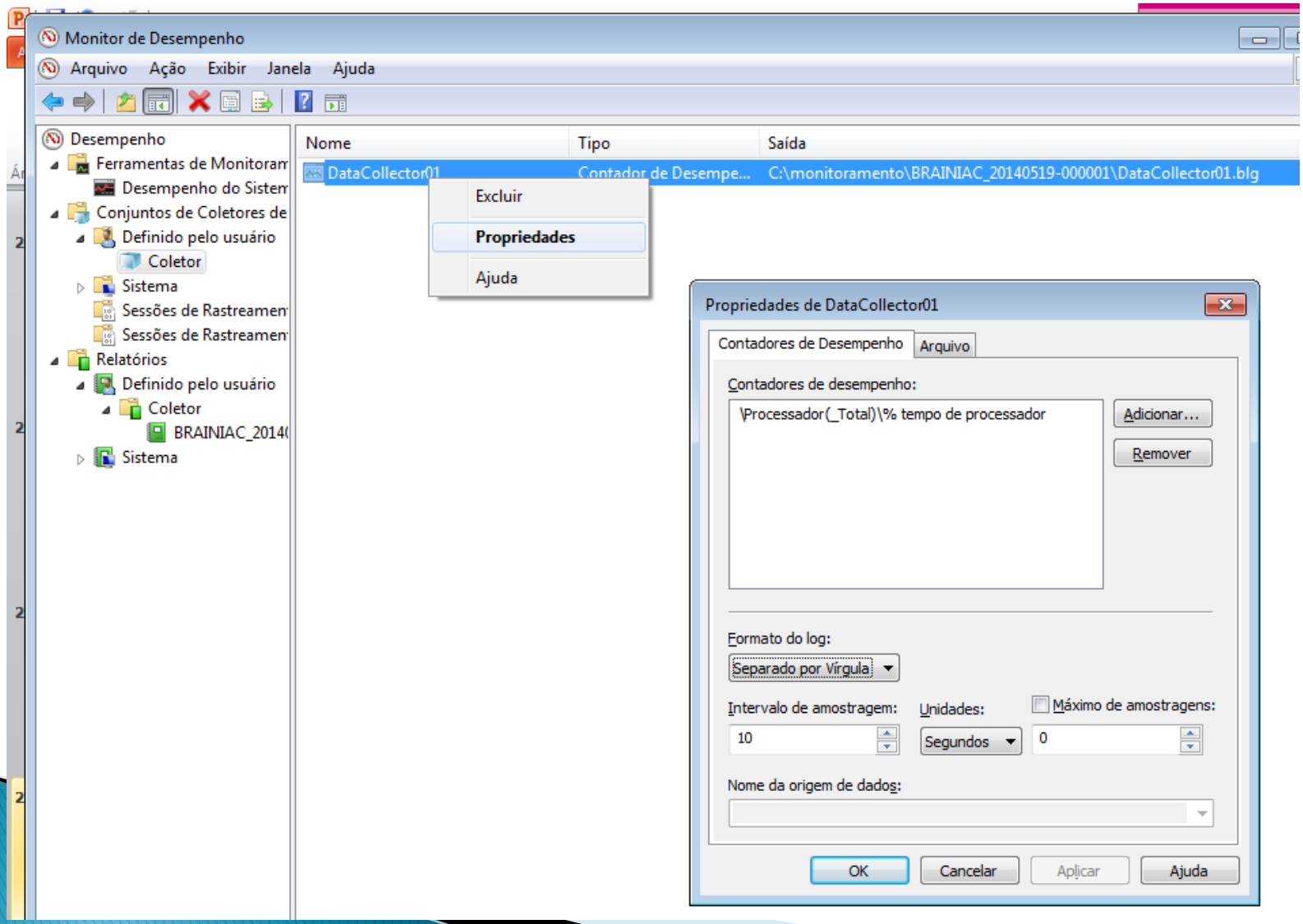
Visualizando arquivo de log



Propriedades gerais do coletor de dados



Propriedades do arquivo de log



Tópicos não explorados

- Alertas de contadores
- Rastreamento de eventos e *Event tracing for Windows* (EWT)
- Monitorando informação de configuração do sistema

Considerações finais

- O Monitor de Performance é uma poderosa ferramenta de monitoramento e traz como vantagem o fato de já vir incluída no Windows por padrão
- Tem como principal função ajudar administradores a descobrir gargalos em servidores Windows
- É necessário cuidado no dimensionamento da janela de monitoramento e intervalo de amostragem

Agenda

- Sobre o PowerShell
  - Introdução
  - Calculadora
  - Help
  - Funções
  - Comandos externos
- Instalação
- Comandos PowerShell
- Criando um Script
  - Executando

Introdução

- O Windows PowerShell é o novo shell de linha de comando do Windows. (PS)PowerShell é uma interface que permite aos usuários interagir com o sistema operacional e pode ser tanto no modo gráfico Graphical User Interface (**GUI**) quanto em modo texto Command-Line Interface (**CLI**).

Sobre o PowerShell

- Nova geração de Shell (Família Microsoft
  - Windows)
- Permite a execução remota (Versão 2.0)
- Integra com .NET Fremework

Sobre o PowerShell

Calculadora

- 5 – 4
- (5 + 9) * 4
- 5GB/120MB

O PowerShell suporta **valores de armazenamento computacional** como:

- **Kilobytes (KB)**
- **Megabytes (MB)**
- **Gigabytes (GB)**
- **Terabytes (TB)**
- **Petabytes (PB)**

Sobre o PowerShell

Comandos Externos

- – O PowerShell pode executar comandos do prompt de comandos Microsoft;

- Ipconfig
- cls
- ls
- clear
- ping
- dir

Inslatação

- Windows installer 3.1;
- .Net Fremework 2.0 SP 1;
- PowerShell **1.0/2.0/3.0** (ou **4.0**) **Beta 5.0**
  $PSVersionTable

- A **versão 1.0** foi lançada em 2006 para Windows XP SP2/SP3 e o Windows Vista;
- A **versão 2.0** está integrada com o Windows 7 possível instalação no Windows XP;
- A **versão 3.0/4.0** integrada no Windows 8 e 8.1

Comandos Power Shell

Os comandos do powershell são chamados de cmdlets. Os nomes dos comandos são compostos por um verbo seguido de um hífen (–) e uma ação.

- – **Digite no terminal:**
  - Get-Command
  - Get-Help
  - Get-Location
  - Get-History

Power Shell

Command-lets(CMDLETS)	Descrição
Add	Adiciona um recurso ou anexa um item em outro item. Exemplo: **Add-Computer** Assim como tem o **Add** existe o **Remove**
Clear	Remove um recurso. Exemplo: **Clear-Content**
Close	Altera o estado de um recurso. Assim como existe **Close** existe o **Open**
Format	Formata (arruma) objetos ou saídas em determinados layouts.
Get	Ação que recupera informações, por exemplo, uma lista de objetos. Exemplo: **Get-Command** .
Move	Move recursos de uma localização para outra.
Show	Exibe informações relacionadas ao “substantivo”

PowerShell ISE

PowerShell ISE (Integrated Scripting Environment), um ambiente de programação do PowerShell que facilita o desenvolvimento de scripts, pois você pode executar comandos, gravar, testar e depurar scripts em uma interface de usuário gráfica baseada no Windows.

Help

Um fator muito importante no uso de um programa ou linguagem de programação é ter uma base de conhecimento completa e atualizada. O PowerShell 3.0 tem um help atualizável e fácil de se usar. Atualizar **Update-Help**.

- **Get-Help <cmdlet>** – Exibe o help no console
- **Get-Help <cmdlet> -Online** –Exibe o help online na biblioteca do TechNet. Digamos que eu queira saber mais sobre "**Compare-Object**". A seguir...

Help

- **Get-Help Compare-Object -Online** – Acessa os recursos online
- **Get-Help Compare-Object -Examples** – Exibe exemplos do comando
- **Get-Help Compare-Object -Detailed** – Exibe um help datalhado
- **Get-Help Compare-Object -ShowWindow** – Exibe uma janela

Funções

O comando **Clear-Host** não é um cmdlet, porém possui o mesmo modelo de verbo-substantivo. O comando Clear-Host é, na verdade, uma função interna.

- Para listar as funções utilize o comando

- **get-command -commandtype function**

Alias

Alias são como apelidos para os cmdlets e funções:

- Por exemplo, podemos usar o comando clear-host para limpar a tela, porém existe o alias chamado clear que executa o clear-host. Liste todos os Alias com o seguinte comando:

- **get-command -commandtype alias**

Exemplos Alias

Um bom exemplo de Alias é para listagem de diretórios e você pode usar qualquer um dos Alias abaixo:

- **LS – UNIX**
- **DIR – MS-DOS**
- **Get-ChildItem – PowerShell**

Criar uma Alias

- Como criar um alias?
- **Set-Alias Dia Get-Date**
- O comando acima criar um alias chamado **Dia** para o cmdlets **Get-Date.**

Exibição

- As informações que você pode coletar através do Windows Power Shell pode ser formatada de modo que facilite a visualização das informações.
- Um dos cmdlets que nós administradores sempre precisamos executar é o **Get-Process**, pois lista os processos em execução em nosso servidor ou estação:
- **Get-Process**

Exibição

Usado o pipe (|) podemos passar a saída do comando para diversas opções. O pipe é um operador. Cada comando após o pipe recebe um objeto do comando anterior, realiza alguma operação no objeto, e depois passa adiante para o próximo comando no pipeline.

- **Get-Process | more**
- **Get-Process | Format-List**
- **Get-Process | Format-List | more**
- Get-Process | ConvertTo-HTML | Out-File "Processos.html"
- Get-Process | Export-CSV "Processos.csv"

cmdlets out

Alguns cmdlets existentes criam saídas legais, são os casos do cmdlets out. Para listar os cmdlets "**out**": **Get-command out***

- **Out-Default** – Envie a saída para o formatador padrão e o cmdlet de saída padrão.
- **Out-File** – Envia a saída para um arquivo.
- **Out-GridView** – Envia a saída para uma tabela interativa em uma janela separada...

- Exemplos: **Get-Process | Out-GridView**
- **Get-Process | out-file -filepath C:\Scripts\processos.txt**

Redirecionador

Também podemos usar o redirecionador que é o sinal de "**maior que**" > para criar e gravar no arquivo e usar duas vezes o comando >> para adicionar informações no fim do arquivo já existente.

- Exemplo: **Get-Process > teste1.txt Get-Alias >> teste1.txt**

Filtrar Resultados

O cmdlet Where-Object fornece a capacidade de criarmos filtros específicos no retorno de outros cmdlets. Como você já deve ter percebido, alguns cmdlets exibem na tela todos os dados de determinado objeto ou recurso, como por exemplo o cmdlet Get-Service trará na tela todos os serviços estando iniciados e parados. Com o **Where-Object** você pode criar um filtro e trazer apenas os serviços em execução.

- **get-service | where-object {$_.Status -eq "Running"}**

Filtrar Resultados

Operador	Descrição
-lt	Menor que
-le	Menor ou igual
-gt	Maior que
-ge	Maior ou igual
-eq	Igual
-ne	Não igual
-like	Usa wildcards para comparar padrões

Filtrar Resultados

Cada cmdlet exibe na tela diferentes resultados, portanto no momento de usar **Where-Object** você deve conhecer o resultado padrão e analisar quais são os nomes dos campos que deseja utilizar como campo. No exemplo abaixo foi executado o cmdlet **Get-ChildItem** e podemos notar que existem 15 campos.

Filtrar Resultados

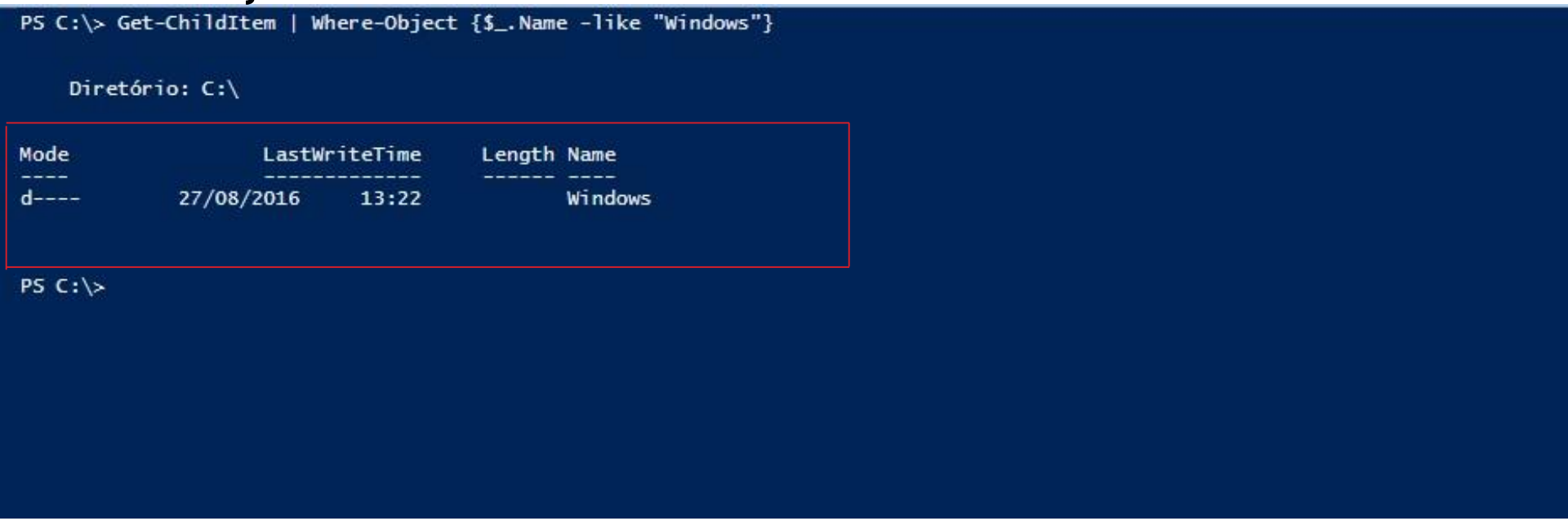
```
PS C:\> Get-ChildItem

    Diretório: C:\

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         08/09/2014     12:37            ADT-x86
d----         13/08/2014     21:52            apache-tomcat-7.0.55
d----         10/08/2014     02:10            Dev-Cpp
d----         24/08/2014     03:28            Eclipse Android
d----         12/09/2014     15:08            Eclipse Jee Kepler
d----         04/08/2014     01:43            inetpub
d----         26/07/2012     04:33            PerfLogs
d----         31/08/2016     02:45            Program Files
d-r--         30/08/2016     01:18            Program Files (x86)
d----         29/08/2016     01:10            Sharpe-Gui
d----         16/08/2014     03:37            SQLin
d-r--         10/08/2014     00:26            Users
d----         27/08/2016     13:22            Windows
d----         21/08/2014     05:10            workspace
-a---         14/08/2014     01:04   370612741 ADT-x86.zip
```

Filtrar Resultados

Podemos então fazer um filtro com where-object {$_.Name -like "Windows"}



Criando um Script

- É possível criar scripts PowerShell e sempre que necessário executa-lo como se fosse o velho ***batizinho*** (Batch Files).

- A extensão para execução de scripts no PowerShell é .PS1. Usando o editor de textos basta criar um arquivo e salvar como nomedesejado**.ps1**

- A vantagem de fazer uso de scripts PowerShell é criar ferramentas poderosas de administração ou de automação de tarefas cotidianas.

Criando um Script

O cmdlet Set-ExecutionPolicy permite determinar como os scripts serão permitidos para execução. Windows PowerShell tem quatro diferentes políticas de execução:

- **Restricted** – Nenhum script pode ser executados. Windows PowerShell pode ser usado apenas no modo interativo.
- **AllSigned** – Somente scripts assinados por um fornecedor confiável pode ser executado.
- **RemoteSigned** – os scripts baixados devem ser assinados por um fornecedor confiável antes que eles possam ser executados.
- **Unrestricted** – Sem restrições, todos os scripts do Windows PowerShell pode ser executado.

Criando um Script

- Alguns ambientes podem não permitir a execução de scripts por motivos de segurança. Para habilitar a execução de scripts você deve definir uma política de execução com o comando:
  - – Dar permissão
  - **Set-ExecutionPolicy RemoteSigned**
  - Set-ExecutionPolicy Unrestricted –Force

Criando um Script

Criar um arquivo texto simples na raiz com o nome qualquer (ex: test.ps1) e edite o script abaixo:

– "Massa D+"

```
19  # Comentário no PowerSehell
20  "Massa D+"
```

– Executar

.\test.ps1

Monitorando Processos

Get-Process

- **Handles**: O **número** de manipulações abertas pelo **processo**.
- **NPM(K)**: A **quantidade** de **memória** não paginada usada pelo processo, em kilobytes.
- **PM(K)**: A quantidade de **memória** paginada usada pelo processo, em **kilobytes**.
- **WS(K)**: O tamanho do conjunto de trabalho do **processo**, em **kilobytes**. O conjunto de trabalho consiste nas **páginas** de **memória** recentemente **referenciadas** pelo **processo**.
- **VM(M)**: A **quantidade** de **memória** virtual usada pelo **processo**, em **megabytes**. A memória virtual inclui o armazenamento em disco dos arquivos de paginação.
- **CPU(s)**: O **tempo** do **processador** que o processo usou em todos os processadores, em **segundos**.
- **ID**: O **ID** de processo (PID) do **processo**.
- **ProcessName**: O **nome** do **processo**.

Variáveis

$nome = “Pessoa"
dir variable:
remove-item variable:\nome

Tipos de Dados

- $num1=10
- $num2="20"
- $num1+$num2
- $num2+$num1
- [int]$num2+$num1
- $num1.GetType().Name
- $processos = get-process
- $processos -is [array]

Manipulação de Arquivos

- **Criar**
  - New-Item -Path 'C:\temp\New Folder' -ItemType "directory“
- **Editar**
  - Add-Content .\arquivo.txt "texto q“
  - Add-Content -Path "c:\sample.txt" -Value "`r`nThis is the last line“
- **Recuperar**
  - Get-Content .\arquivo.txt

Recuperação de Informações

- Get-Counter -ListSet *
- Get-Counter -ListSet * | Sort-Object CounterSetName | Format-Table CounterSetName
- Get-Counter -ListSet "Informações do Processador"
- Get-Counter "\Informações do Processador(*)\Frequência do Processador"

Recuperação de Informações

- Get-Process| where {$_.ProcessName -like "chrome"}
- Get-Process| where {($_.ProcessName -like "chrome") -AND ($_.WS -gt 300000000)}
- Get-Process | Get-Member

Recuperação de Informações

- $var = (Get-Process| where {($_.ProcessName -like "chrome") -AND ($_.WS -gt 300000000)}) |Select-Object processName,CPU

- Add-Content res.txt $var

- Start-sleep

Script

```
For ($i=0; $i -le 10; $i++) {
    "10 * $i = " + (10 * $i)
    }
-----------------------------------------------------------------------------
For ($i=0; $i -le 10; $i++) {

    Get-Process| where {($_.ProcessName -like "chrome") -and ($_.ws -gt 300000000)  }

    Start-Sleep 3

    }
```

Referências

- https://docs.microsoft.com/pt-br/previous-versions/technet-magazine/cc718984(v=msdn.10)
- http://en.wikipedia.org/wiki/System_Monitor
- https://docs.microsoft.com/pt-br/previous-versions/ee310108(v=msdn.10)
- Help da ferramenta