МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационный технологий и программирования

Лабораторная работа №6

По дисциплине «Администрирование в ОС Windows Server»

Выполнили студенты группы М33081:

Найман Егор

Кузнецова Алика

Мещеряков Никита

Проверил:

Папикян С. С.

САНКТ-ПЕТЕРБУРГ

2022

Ответы на вопросы:

1. В чем назначение каждого из разделов журнала событий?

Performance counter (счётчик производительности) — журнал, в который с определенной периодичностью заносятся значения Счетчиков – атрибутов программных объектов, представляющих или аппаратные или программные подсистемы (Процессор, UDP, Файл Подкачки и т.п.).

Event trace data (сборщик данных отслеживания событий) — журнал, куда заносятся все события, происходящие в подсистеме, которая выбрана в виде провайдера при создании сборщика.

System configuration information (оповещение счетчика производительности) — сведения о конфигурации. Регистрируют состояние параметров реестра и изменения, которые в них вносятся.

2. Зачем нужен раздел Перенаправленные события?

Перенаправленное событие — это событие, зарегистрированное в системе событий WPF, поддерживаемое экземпляром RoutedEvent класса и обрабатываемое системой событий WPF

Перенаправленные события имеют особое поведение, но это поведение в значительной степени невидимо, если вы обрабатываете событие в элементе, вызвавом его. Однако перенаправленные события имеют отношение, если требуется присоединить обработчик событий к родительскому элементу, чтобы обрабатывать события, создаваемые дочерними элементами, например в составном элементе управления.

Перенаправленные события поддерживают обмен информацией о событиях между элементами по маршруту событий, так как каждый прослушиватель имеет доступ к одному и тому же экземпляру данных события. Если один элемент изменяет что-то в данных события, это изменение отображается для последующих элементов в маршруте событий.

Перенаправленные события поддерживают обработчики событий класса , обрабатывающие событие перед обработчиками экземпляров для одного и того же события в любом экземпляре класса прослушивателя. Эта функция полезна в проектировании элементов управления, так как обработчик класса может принудительно применять поведение класса на основе событий, которое не может быть случайно подавлено обработчиком экземпляра.

3. Где находится журналы событий Windows в виде файлов?

По умолчанию файлы Просмотр событий используют расширение EVT и находятся в папке %SystemRoot%\System32\winevt\Logs. Имя файла журнала и сведения о расположении хранятся в реестре

4. Как с помощью графической оснастки журнала событий установить по известному VID коду, когда было подключено и настроено устройство?

Когда устройство подключается генерируется запись в журнале Microsoft-Windows-DriverFrameworks-UserMode/Operational event log, чтобы отфильтровать нужное нам событие, открываем просмотр событий, переходим в необходимый нам журнал и выбираем фильтр текущего журнала, выбираем XML фильтр и выбираем пункт “изменение фильтра вручную”

Затем применяем фильтр

<QueryList>

<Query Id="0" Path="Microsoft-Windows-DriverFrameworks-UserMode/Operational">

<Select Path="Microsoft-Windows-DriverFrameworks-UserMode/Operational">

Event [System [

EventID = SampleID

] and

EventData[

Data[@Name="VendoIDr"]=Sample ID]

]

</Select>

</Query>

</QueryList>

где в EventId мы выбираем событие которое нам необходимо: подключение, установка и тп. (в основном коды 1003, 1004, 2000, 2001, 2003, 2004, 2005, 2006, 2010, 2100, 2101, 2105, 2016)

А в VendorID известный нам VID

5. Почему были выбраны конкретные счетчики в Части 4 п.1? Обоснуйте выбор.

Сравнивали запуск и работы в прилжениях Internet Explorer и MozilaFirefox

% загруженности процессора

% использование выделенной памяти

% активности диска

Артефакты:

1. **Скрипт из Части 1.**

MonitoringLog.ps1 | ProcessesLog.ps1 | 2.ps1 | 3.ps1

```powershell
try
{
    New-EventLog -LogName ProcessMonitoringLog -Source MonitoringLog -ErrorAction Stop
}
catch
{
    Write-Error "EvenLog is already exists"
}
```

MonitoringLog.ps1 | ProcessesLog.ps1 | 2.ps1 | 3.ps1

```powershell
try
{
    $processList = Get-Process -IncludeUserName | Select-Object ID, ProcessName, Path, UserName, CPU, WS
    $processInfoList = @()
    ForEach ($process in $processList)
    {
        $processInfo =
        @{
            ID = $process.ID
            ProcessName = $process.ProcessName
            Path = $process.Path
            UserName = $process.UserName
            Cpu = $process.CPU
            Ws = $process.WS
            Time = Get-Date
        }
        $processInfoList += New-Object -TypeName PSObject -Property $processInfo
    }
    $processInfoList | Export-CSV "C:\Dev\ProcessInfo.csv" -ErrorAction Stop
    Write-EventLog -LogName ProcessMonitoringLog -Source MonitoringLog -EventId 1 -EntryType SuccessAudit -Message "Succesful"
}
catch
{
    Write-EventLog -LogName ProcessMonitoringLog -Source MonitoringLog -EventId 2 -EntryType FailureAudit -Message "Faild"
}
```

2. **Скрипт из Части 2**

MonitoringLog.ps1 | ProcessesLog.ps1 | 2.ps1 | 3.ps1

```powershell
$action = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument '-File C:\Dev\MonitoringLogCreater.ps1'
$time = New-TimeSpan -Minutes 3
$trigger = New-ScheduledTaskTrigger -Once -At (Get-Date).Date -RepetitionInterval $time -RepetitionDuration ([timeSpan]::MaxValue)
$settings = New-ScheduledTaskSettingsSet -DontStopIfGoingOnBatteries -AllowStartIfOnBatteries

try
{
    Register-ScheduledTask -TaskName "Task monitoring" -Action $action -Trigger $trigger -Settings $settings
}
catch
{
    Write-Error "Scheduled task is already exist"
}
```

3. **Скрипт из части 3. п. 3**

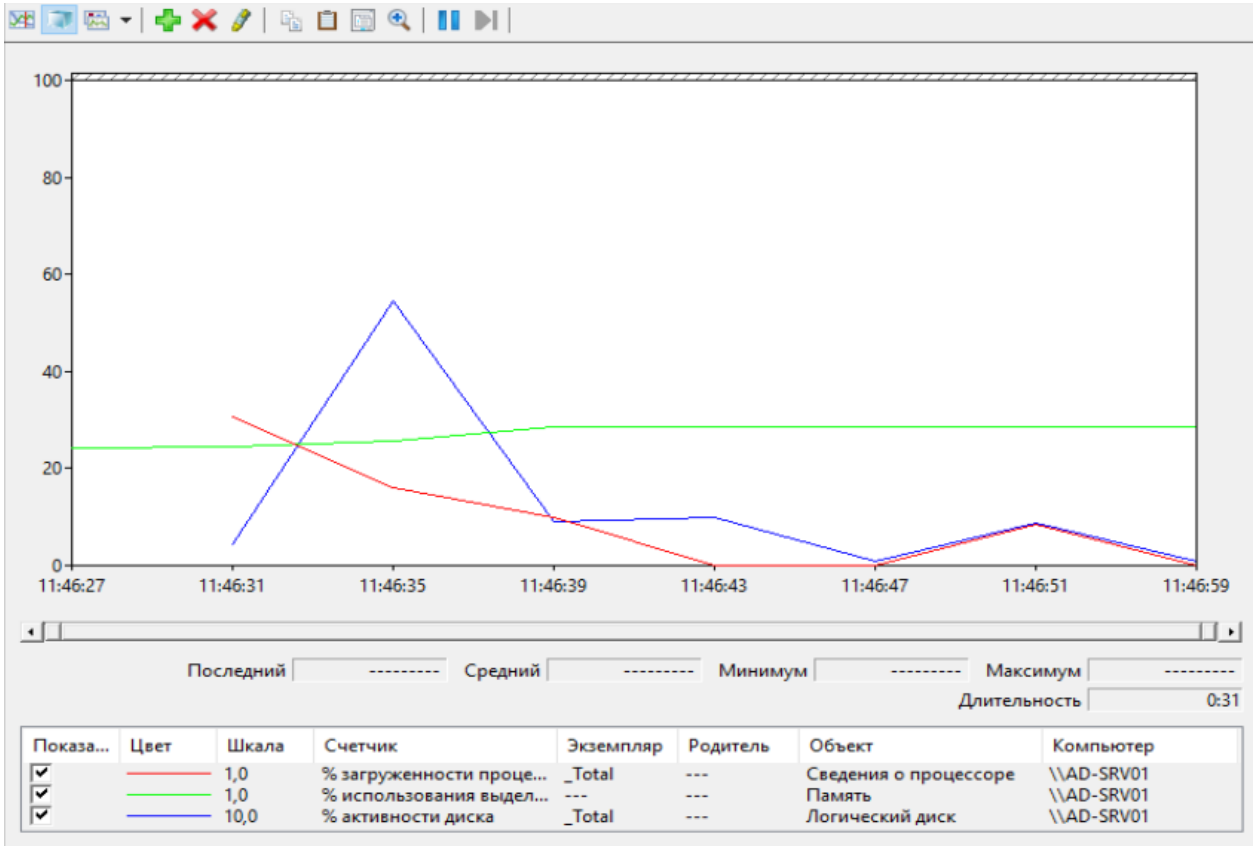
MonitoringLog.ps1 | ProcessesLog.ps1 | 2.ps1 | 3.ps1

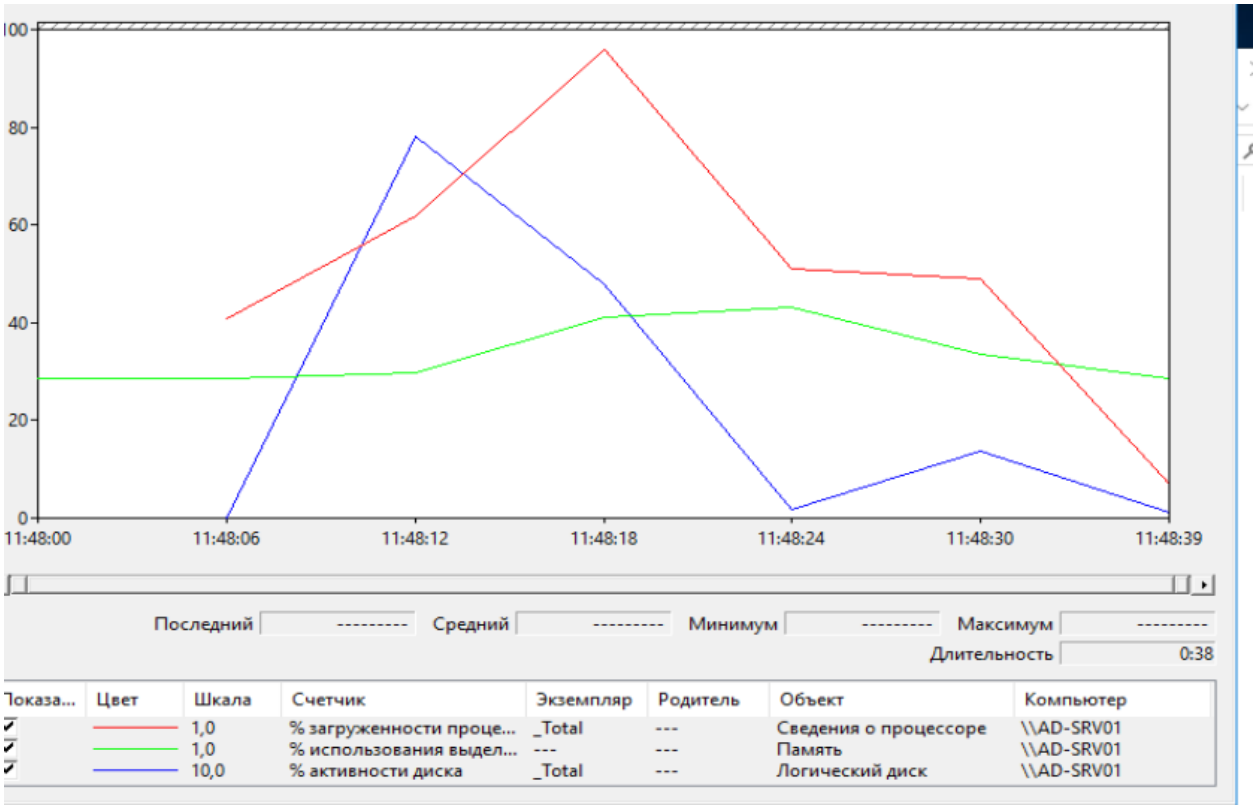
```powershell
Get-EventLog System | Where-Object {$_.EventID -contains "19"} | Select -First 10 > "C:\Dev\task.txt"

Get-HotFix | Sort InstalledOn -Descending | Select-Object HotFixID, InstalledOn -First 5 >> "C:\Dev\task.txt"

Get-WinEvent -Force -ListLog * | Where-Object {$_.RecordCount -AND $_.LastWriteTime -LE (Get-Date).AddDays(-1)} |
    ForEach-Object {Get-WinEvent -LogName $_.LogName} | Group-Object -Property LevelDisplayName |
    Where-Object {"Ошибка","Предупреждение" -contains $_.Name} | Select-Object Name, Count >> "C:\Dev\task.txt"
```

4. **Материалы и результаты анализа из Части 4 п. 3-4**





5. **Скрипты из части 5.**

Clean.ps1 | 5.2.ps1

```powershell
Remove-Item "E:\*"
```

Clean.ps1 | 5.2.ps1

```powershell
$count = 0
try
{
    while ($true)
    {
        $count++
        $objFile = [io.file]::Create("E:\$count.txt")
        $objFile.SetLength(1mb)
        $objFile.Close()
    }
}
catch
{
        $objFile.Close() | Write-Error "Disk is full"
}
```