Resource Usage

SYSTEMINFO

You can cheat and use good old SYSTEMINFO from any command line. This will give you fairly comprehensive system information.

```
systeminfo
```

Memory Usage

Again you can cheat, use SYSTEMINFO and filter the output:

```
systeminfo | Select-String 'Memory:'
```

The code snippet below will work with PowerShell 3.0 and newer

```
if ([Environment]::Is64BitOperatingSystem) {
  #64 bit logic here
  get-process | Group-Object -Property ProcessName |
  % {
      [PSCustomObject]@{
          ProcessName = $_.Name
          Mem_MB = [math]::Round(($_.Group|Measure-Object WorkingSet64 -Sum).Sum / 1MB, 0)
          ProcessCount = $_.Count
      }
  } | sort -desc Mem_MB | Select-Object -First 25
} else {
  #32 bit logic here
```

```
  get-process | Group-Object -Property ProcessName |
  % {
      [PSCustomObject]@{
          ProcessName = $_.Name
          Mem_MB = [math]::Round(($_.Group|Measure-Object WorkingSet -Sum).Sum / 1MB, 0)
          ProcessCount = $_.Count
      }
  } | sort -desc Mem_MB | Select-Object -First 25
}
```

The code below will execute on Windows 7 and newer.

```
if ((Get-WmiObject win32_operatingsystem | select osarchitecture).osarchitecture -eq "64-bit")
{
  #64 bit logic here
  get-process | Group-Object -Property ProcessName |
  % {
      [PSCustomObject]@{
          ProcessName = $_.Name
          Mem_MB = [math]::Round(($_.Group|Measure-Object WorkingSet64 -Sum).Sum / 1MB, 0)
          ProcessCount = $_.Count
      }
  } | sort -desc Mem_MB | Select-Object -First 25
} else {
  #32 bit logic here
  get-process | Group-Object -Property ProcessName |
  % {
      [PSCustomObject]@{
          ProcessName = $_.Name
          Mem_MB = [math]::Round(($_.Group|Measure-Object WorkingSet -Sum).Sum / 1MB, 0)
          ProcessCount = $_.Count
      }
  } | sort -desc Mem_MB | Select-Object -First 25
}
```

Revision #2
Created 2024-08-01 14:53:38 UTC by bluecrow76
Updated 2024-08-02 16:02:34 UTC by bluecrow76