PowerShell for Beginners

Basics and Complex Exercises

Table of Contents

PowerShell Pipeline

Chaining Cmdlets together

Chaining Cmdlets together (pipeline)

- Pipeline principle:
  - The output of one Cmdlet can serve as input for another Cmdlet
  - Passing on data like on an assembly line
- Pipeline symbol is the perpendicular stroke | (pipe)
- *Caution*: in PowerShell we usually pass objects through the pipeline

Pipeline: Concepts

- Best Practice while chaining:
  - left filtering
  - right formatting

- Thus the result set is processed efficiently

Pipeline: Filtering

- Useful Cmdlets for filtering:
  - `Where-Object`
  - `Select-Object`
  - `Group-Object`
  - `Select-String` (uncommon, operating on strings)

Filtering with `Where-Object`

- `Where-Object` contains code block (filter) in { }
- Codeblock must contain a predicate (Boolean expression)
- Comparison operators (excerpt)
  - `-eq, -lt, -gt, -like`
- Logical operators (excerpt)
  - `-not, -and, -or`

- Accessing an object via qualified dot access (OOP style)
- (Anonymous) reference (this-pointer) is `$_`

Filtering with `Where-Object`

- Example:
  - List all processes that have consumed more than 10 seconds of CPU time.

```
PS C:\Users\anr> Get-Process | Where-Object { $_.CPU -gt 10 }

 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
     16     4,36      20,13      11,80    6880   2 ctfmon
    130   236,00     149,80      79,09    8916   2 explorer
     62   246,43     274,29     238,75    9304   2 POWERPNT
    128   170,03     259,84      21,06   11452   2 pwsh
```

Filtering with `Select-Object`

- Filters properties of objects
- Parameter `-Property` contains list of properties
- List may contain wildcards
- Other properties not visible in pipeline thereafter
- Choose the first (`-First`) or last (`-Last`) objects in the result set
  - Implements features of head, tail, top in bash

☞Why is `Select-Object -Property *` not a useful filter?

Filtering with `Select-Object`

- Example:
  - Print only the properties name, CPU time and PID of the system processes. Print only the first five items from the result set.

```
PS C:\Users\anr> Get-Process | Select-Object -Property Name, Id, CPU -First 5

Name                      Id CPU
----                      -- ---
AppHelperCap            2812
ApplicationFrameHost    7328 0,53
audiodg                 1616 0,22
AWACMClient             7800
AwWindowsIpc            8388 5,70
```

Filtering with `Select-String`

- Nur nutzbar, wenn Input als `String` oder `String[]` vorliegt.
  - Each object contains a method `ToString()`
  - The output from a pipeline can be converted to a `String` object using `Out-String`
- Problem:
  - The resulting object is a single unstructured string
- Solution:
  - Conversion to `String[]` object (line wise) with `Out-String -Stream`

Filtering with `Select-String`

- The Cmdlet `Select-String` is similar to…
  - `findstr` from command prompt (classic Windows shell)
  - `grep` from *bash* (GNU/Linux shell)

- Particularly useful,
  - if text has to be processed as a stream,
  - if OS returns data as strings (PowerShell for GNU/Linux),
  - if working with regular expressions (RegEx)

Filtering with `Select-String`

- Example:
  - List running services on GNU/Linux systems (here: Trisquel).

```
PS /home/anr> service --status-all | Select-String "\+"

 [ + ]  acpid
 [ + ]  anacron
 [ + ]  apparmor
 [ + ]  atd
 [ + ]  avahi-daemon
 [ + ]  bluetooth
 [ + ]  cron
 [ + ]  cups
 [ + ]  dbus
 [ + ]  kmod
 [ + ]  lightdm
 [ + ]  network-manager
 [ + ]  openvpn
 [ + ]  procps
 [ + ]  rsyslog
 [ + ]  udev
 [ + ]  unattended-upgrades
```

Pipeline: Sorting and Formatting

- Cmdlet for sorting is `Sort-Object`.
- Useful Cmdlets for formatting:
  - `Format-Table` Output data as a table
  - `Format-List` Output data as a list (more like *cmd.exe*)
- Default is output formatted as a table (Format-Table)
  - Not all members of the resulting objects are shown.

Formatting with `Format-List`

- Example:
  - Select and print all properties of system processes

```
PS> Get-Process | Format-List –Property *
```

  - Only print selected properties Name, Id and CPU

```
PS> Get-Process | Format-List –Property Name, Id, CPU
```

Pipeline: Data Output

- Useful Cmdlets to redirect data output:

```
PS C:\Users\anr> Get-Command -Verb Out

CommandType     Name
-----------     ----
Cmdlet          Out-Default
Cmdlet          Out-File
Cmdlet          Out-GridView
Cmdlet          Out-Host
Cmdlet          Out-Null
Cmdlet          Out-Printer
Cmdlet          Out-String
```

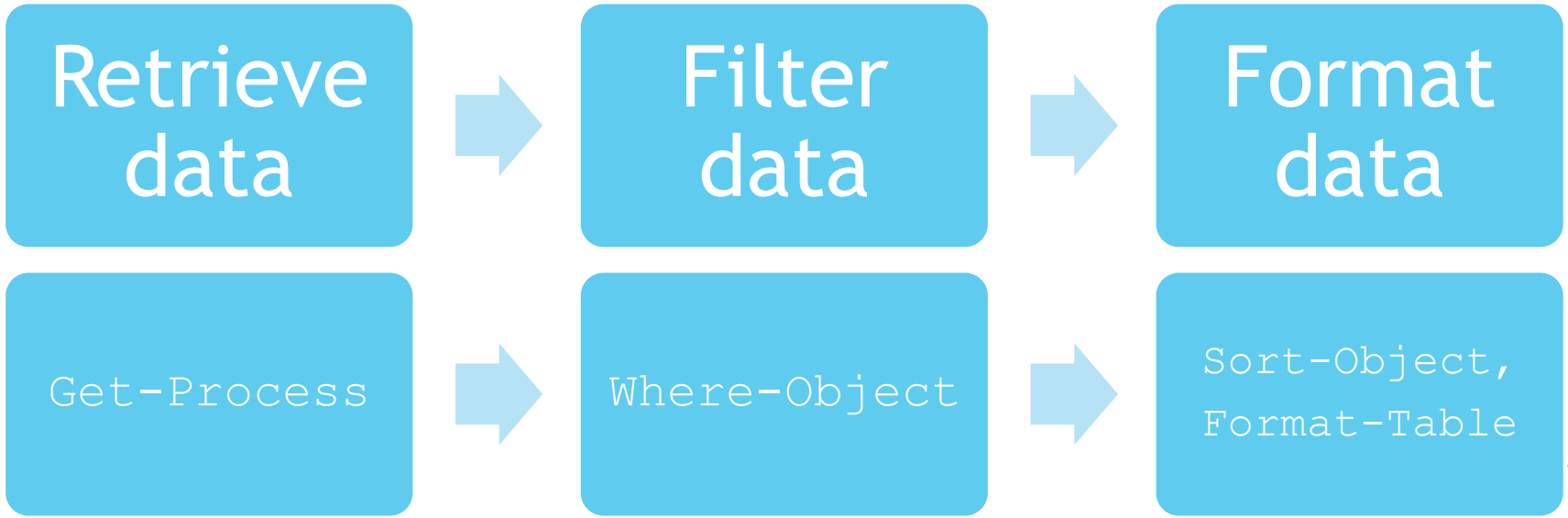
Pipeline: Data Output

- Useful Cmdlets for redirecting data output:
  - `Out-File` Output in file
  - `Out-GridView` Output in GUI window (interactive)
  - `Out-Host` Output on stdout
  - `Out-Printer` Output on printer
  - `Out-String` Convert output to string

☞ `Out-GridView` is a PowerShell specialty

Pipeline: Complex Example

- Example:
  - List processes whose name begins with ‚A‘ and sort them descending according to consumed CPU time. Print only name and CPU time.



Interactive Pipeline

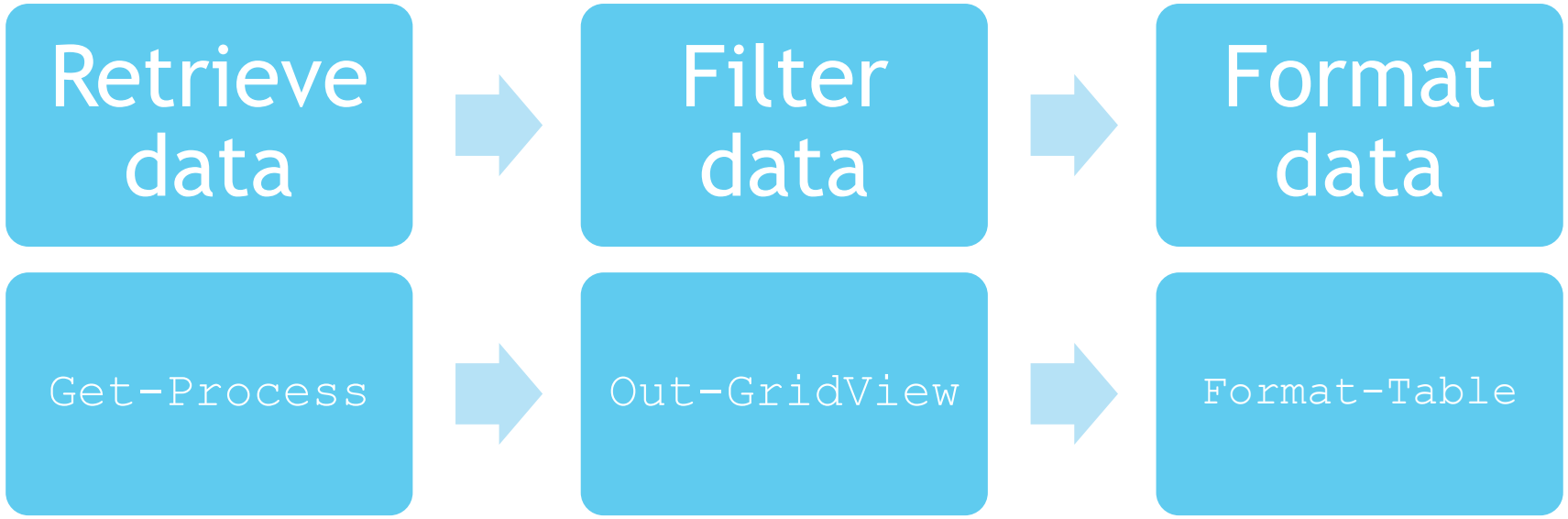
- Cmdlet `Out-GridView`
- Creates table view in GUI that can be filtered, etc.
- With `-PassThru` pass results of GUI windows back to pipeline
- Usable as (graphical) replacement for `Where-Object` etc.

Combines advantages of CLI and GUI

Not usable in scripts, but only *ad hoc*

Filtering with `Out-GridView`

- Example:
  - List processes whose name begins with ‚A‘ and sort them descending according to consumed CPU time. Print only name and CPU time.



☞ Demo

Example: Knuth Problem

- In 1986 famous computer scientist Donald Knuth was asked by the author of the column *Programming Pearls* to write a program for the following task:

- *„Read a text file, determine the n most frequently used words, and print out a sorted list of those words along with their frequencies.“*

Example: Knuth Problem

- Knuth wrote a 10 page Pascal program to solve the task.
- UNIX pioneer Doug McIlroy answered Knuth with the following shell script, that also meets the requirements:

```
tr -cs A-Za-z '\n' | tr A-Z a-z | sort | uniq -c
| sort -rn | sed ${1}q
```

Example: Knuth Problem

- This classic example illustrated the power of pipelines when processing (mainly) text based data.

☞ How would you solve the Knuth Problem in PowerShell?

System Administration

Managing processes and services

Managing Processes

- Family of Cmdlets for processes:
  - ☞ How can all Cmdlets of the family be listed?

```
PS C:\Users\anr> Get-Command -Noun Process

CommandType     Name
-----------     ----
Cmdlet          Debug-Process
Cmdlet          Get-Process
Cmdlet          Start-Process
Cmdlet          Stop-Process
Cmdlet          Wait-Process
```

Managing Services

- Family of Cmdlets for services:

```
PS C:\Users\anr> Get-Command -Noun Service

CommandType     Name
-----------     ----
Cmdlet          Get-Service
Cmdlet          New-Service
Cmdlet          Remove-Service
Cmdlet          Restart-Service
Cmdlet          Resume-Service
Cmdlet          Set-Service
Cmdlet          Start-Service
Cmdlet          Stop-Service
Cmdlet          Suspend-Service
```

☞ How can we find out about syntax and parameters of those Cmdlets ?

Exercise PS31, PS32, PS33, PS33a
Managing Processes and Services, Pipelines

- Using a PowerShell pipeline in simple cases
- Get to know Cmdlets for managing processes and services
- Use Cmdlets with parameters
- Use common parameters and reflect upon error actions
- Optional exercise PS34 for partitioning
  - recommend only in VM