PowerShell für Einsteiger

Grundlagen und Lernsituationen

Inhaltsübersicht

- Begrüßung/Vorstellung
- Impulsvortrag PowerShell
- PowerShell Grundlagen (Cmdlets, Hilfe zur Selbsthilfe)
- Pipeline, Prozesse und Dienste verwalten
- Benutzer und Gruppen verwalten, Benutzerprofile
- Dateisystem und NTFS-Rechte, Freigaben, Netzlaufwerke
- Netzwerkkonfiguration
- Serveraufgaben, Loganalyse, Webzugriff, Jobs
- Programmierung mit PowerShell (ps1-Skripte, Zugriff auf .NET-Objekte)
- Klassenarbeiten und Lernsituationen mit PowerShell erstellen und vorstellen

PowerShell Pipeline

Cmdlets verketten

Cmdlets verketten (Pipeline)

- Pipeline-Prinzip:
  - Der Output eines Cmdlets kann Input für ein anderes Cmdlet sein
  - Weitergabe wie an einem Fließband
- Pipeline-Zeichen ist der senkrechte Strich | (Pipe)
- *Achtung*: in PowerShell werden i.d.R. Objekte weitergereicht

Pipeline: Konzepte

- Best Practice beim Verketten:
  - Links filtern (left filtering)
  - Rechts formatieren (right formatting)
- Result Set wird so effizient verarbeitet

Pipeline: Filtern

- Nützliche Cmdlets zum Filtern:
  - `Where-Object`
  - `Select-Object`
  - `Group-Object`
  - `Select-String` (seltener, da auf Strings operierend)

Filtern mit `Where-Object`

- `Where-Object` erhält Codeblock (Filter) in { }
- Codeblock muss ein Prädikat (Boolescher Ausdruck) enthalten
- Vergleichsoperatoren (Auszug)
  - `-eq, -lt, -gt, -like`
- Logische Operatoren (Auszug)
  - `-not, -and, -or`

- Zugriff auf Objekt per qualifiziertem Punktzugriff (OOP-Style)
- (Anonyme) Referenz (this-Pointer) ist $_

Filtern mit `Where-Object`

- Beispiel:
  - Finde alle Prozesse, die mehr als 10 Sekunden CPU-Zeit verbraucht haben.

```
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
```

Filtern mit `Select-Object`

- Filter nach Properties von Objekten
- Parameter `-Property` erhält Liste von Properties
- Liste kann Wildcards enthalten
- Übrige Properties danach in Pipe nicht mehr sichtbar
- Wähle die ersten (`-First`), letzten (`-Last`) Objekte im Result Set aus
  - Implementiert Features von head, tail, top in bash

☞Warum ist `Select-Object -Property *` kein sinnvoller Filter?

Filtern mit `Select-Object`

- Beispiel:
  - Lasse von den Prozessen nur Namen, CPU-Zeit, und PID anzeigen. Zeige nur die ersten fünf Ergebnisse an.

```
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
```

Filtern mit `Select-String`

- Nur nutzbar, wenn Input als `String` oder `String[]` vorliegt.
  - Jedes Objekt besitzt eine Methode `ToString()`
  - Der Output einer Pipeline kann mit `Out-String` in ein `String`-Objekt verwandelt werden:

- Problem:
  - Das Ergebnisobjekt ist ein einziger unstrukturierter String

- Lösung:
  - Umwandlung in `String[]`-Objekt (zeilenweise) mit `Out-String -Stream`

Filtern mit `Select-String`

- Das Cmdlet `Select-String` funktioniert ähnlich wie…
  - `findstr` aus der Eingabeaufforderung (klassische Windows Shell)
  - `grep` aus der bash (GNU/Linux Shell)

- Nützlich,
  - wenn Text ausgeschnitten werden muss,
  - wenn das OS Rückgaben als Strings liefert (PowerShell auf GNU/Linux)
  - mit regulären Ausdrücken (RegEx) gearbeitet werden soll

Filtern mit `Select-String`

- Beispiel:
  - Laufende Dienste auf GNU/Linux-System (hier: Trisquel) anzeigen.

```
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: Sortieren und Formatieren

- Cmdlet zum Sortieren ist `Sort-Object`.
- Nützliche Cmdlets zum Formatieren:
  - `Format-Table` Ausgabe in Tabellenform
  - `Format-List` Ausgabe in Listenform (eher wie cmd)

- Default ist Ausgabe in Tabellenform
  - Es werden nicht alle Member angezeigt.

Formatierung mit `Format-List`

- Beispiel:
  - Alle Eigenschaften (Properties) der Prozesse auswählen und anzeigen

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

  - Nur ausgewählte Eigenschaften Name, Id und CPU anzeigen

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

Pipeline: Ausgeben von Daten

- Nützliche Cmdlets zum Ausgeben von Daten:

```
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
```

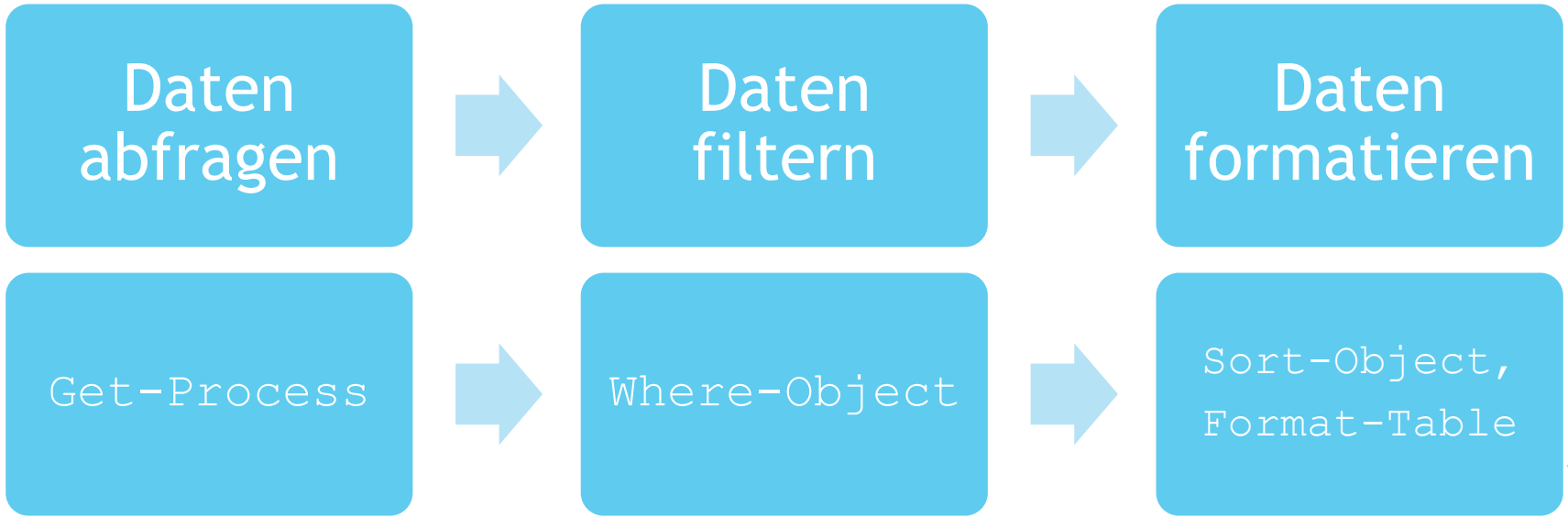
Pipeline: Ausgeben von Daten

- Nützliche Cmdlets zum Ausgeben von Daten:
  - `Out-File` Ausgabe in Datei
  - `Out-GridView` Ausgabe in GUI-Fenster (interaktiv)
  - `Out-Host` Ausgabe auf stdout
  - `Out-Printer` Ausgabe auf Drucker
  - `Out-String` Ausgabe in String umwandeln

☞ `Out-GridView` ist eine PowerShell-Spezialität

Pipeline: Komplexes Beispiel

- Beispiel:
  - Finde Prozesse, deren Name mit ‚A' beginnt und sortiere sie absteigend nach der verbrauchten CPU-Zeit. Zeige nur Namen und CPU-Zeit an.



Interaktive Pipeline

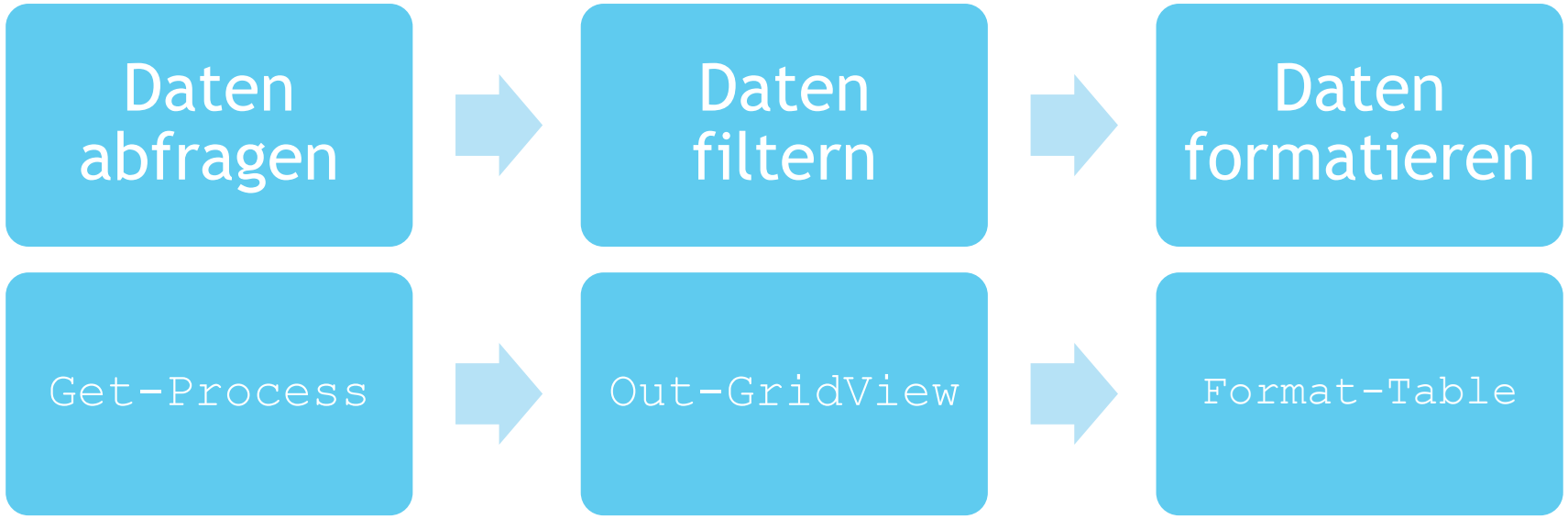
- Cmdlet `Out-GridView`
- Erzeugt Tabelle in GUI, die gefiltert etc. werden kann
- Mit `-PassThru` Weitergabe bzw. Rückgabe an Pipeline
- Als Ersatz für `Where-Object` etc. verwendbar

Kombiniert Vorteile von CLI und GUI

Nicht in Skripten einsetzbar, sondern nur ad hoc

Filtern mit `Out-GridView`

- Beispiel:
  - Finde Prozesse, deren Name mit ‚A' beginnt und sortiere sie absteigend nach der verbrauchten CPU-Zeit. Zeige nur Namen und CPU-Zeit an.



☞ Demo

Beispiel: Knuth-Problem

- Im Jahr 1986 erhielt der bekannte Informatiker Donald Knuth von einem Fachautor der Kolumne *Programming Pearls* die folgende Aufgabe:

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

Beispiel: Knuth-Problem

- Knuth schrieb ein ca. 10-seitiges Pascal-Programm zur Lösung der Aufgabe.
- Der UNIX-Pionier Doug McIlroy antwortete Knuth mit folgendem Shell-Skript, das die Aufgabe ebenfalls erfüllt:

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

Beispiel: Knuth-Problem

- Das klassische Beispiel illustriert die Macht der Pipeline bei der Verarbeitung von (vor allem) textbasierten Daten.

☞ Wie würden Sie das Knuth-Problem in PowerShell lösen ?

Systemverwaltung

Prozesse und Dienste verwalten und konfigurieren

Prozesse verwalten

- Familie der Cmdlets für Prozesse:
  - ☞ Wie können alle Cmdlets der Familie erfragt werden?

```
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
```

Dienste verwalten

- Familie der Cmdlets für Dienste:

```
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
```

☞ Wie finden wir Syntax und Parameter der Cmdlets heraus?

Übung PS31, PS32, PS33
Prozesse verwalten, Dienstverwaltung

- Die PowerShell Pipe in einfachen Fällen anwenden
- Cmdlets zum Verwalten von Prozessen und Diensten kennenlernen
- Cmdlets mit Parametern verwenden
- Optional zusätzlich Übung PS34 zum Partitionieren
  - nur in VM empfohlen