Computing Fundamentals

Planning

Session	Subject	Test – Hand-in
1	Network Models	
2	Internet Protocol Suite	
3	Network segmentation	
4	Network protocols	
5	Operating systems	
6	Command Line	
-- 30/10 – 5/11 --	Autumn break – HERFSTVAKANTIE	
7	~~Virtualization~~	
8	Mid-term test	**Test**
9	**Scripting**	
10	Virtualization - Cloud computing - Storage	

Computing Fundamentals

Scripting

- In computing, a script is a relatively **short and simple set of instructions** that typically **automate** an otherwise manual process. The act of **writing** a script is called **scripting**. Scripting language or **script language** describes a programming language that is used for **scripting**.
- Originally, scripting was **limited to automating** an **operating system shell** and languages were relatively simple. Today, scripting is more pervasive and some languages include **modern features** that allow them to be used for **application development** as well as **scripting**.

Computing Fundamentals

Characteristics

- **Interpreted**
  A script is usually not **compiled** – at least not its usual meaning. Generally, they are **interpreted** directly from source code or from bytecode or run as native after just-in-time compilation.
- **Short & simple**
  A script is **generally relatively short and simple**. As there is no limit on size or complexity, script is **subjective**. A few lines of code without branching is probably considered a script. A codebase of multiple files, that performs sophisticated user or hardware interface or complicated algorithms or multiprogramming is probably **not considered a script**.

Computing Fundamentals

Characteristics

- **Limited language**
  A language that is primarily intended for scripting generally has **limited capabilities** compared to a general-purpose language. A scripting language may **lack the functionality** to write complex applications.
- **Starts at the top**
  Typically, a script starts executing at the **first line** of code whereas an application typically starts at a special point in the code called the **entry point**.

Computing Fundamentals

example

- For example, Java is **not script-like** since an application starts at the function named **main** which need not be at the top of the code. The following code starts at **main**, then calls printHelloWorld which prints "Hello World".

```
public class HelloWorld {
    public static void printHelloWorld() {
        System.out.println("Hello World");
    }

    public static void main(String[] args) {
        printHelloWorld();
    }
}
```

Computing Fundamentals

example

- In contrast, the following Python code prints "Hello World" **without the main** function or **other syntax** such as a **class definition** required by Java.

```
print("Hello World")
```

Computing Fundamentals – DOS/BASH/PS

Computing Fundamentals – DOS/BASH/PS

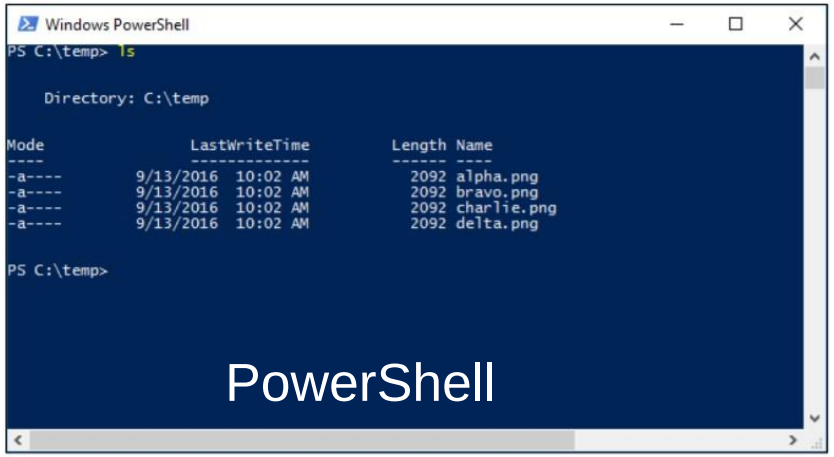
(Reminder?)

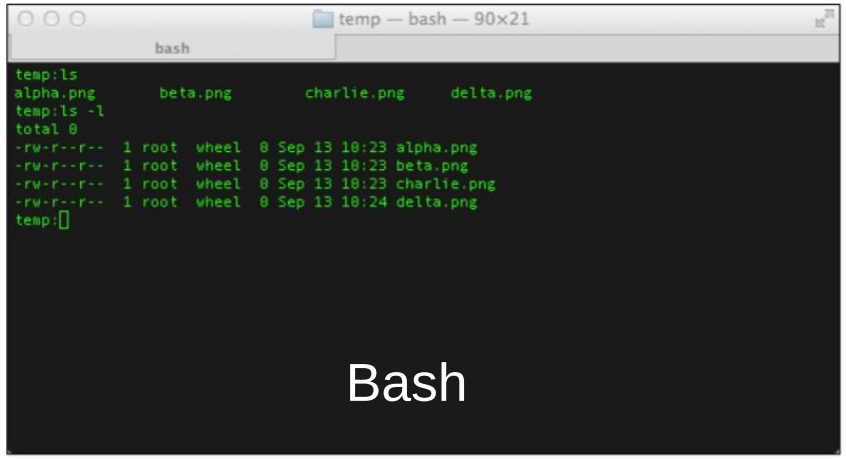
Bash <> PowerShell: important difference in output

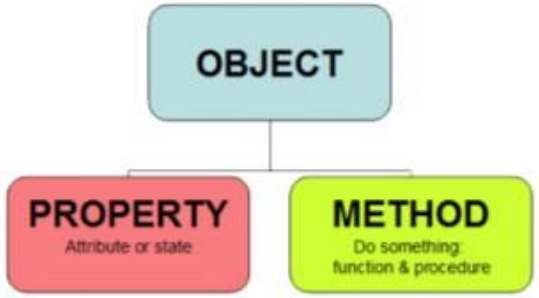
PowerShell = objects (.Net) with properties & methods

- Use .operator!

Bash = text







Computing Fundamentals – PS

PS vs DOS vs Bash

- PS syntax of commands is intuitive: verb-noun (action-object)
- Inherits commands from DOS & Bash as alias

	DOS	BASH	POWERSHELL
List file & folders	dir	ls	Get-Childitem
Change folder	Cd	Cd	Set-Location <foldername> <..>
Verplaats bestand	move	mv ,,, d	Move-Item
Maak een nieuwe folder	mkdir	mkdir	New-Item
Delete file	del	rm	Remove-Item
Toon inhoud van tekstbestand	type	more	Get-Content

Computing Fundamentals – PS

Comdlet: Verb - Module - Noun

Popular Verbs:
Get - Get the detail of an item with no changes to it.
Set - Ghange the setting of an item that already exists.
New - Create a brand new item.
Remove - Delete an existing item.
Add - Add an item.

Module:
Refers to a module. E.g. “NET” in “Get-NetIPAddress”.

Noun:
What will be worked on. E.g. “Item” in “New-Item”.
This Command only has a verb and a noun.

Computing Fundamentals – PS

Let's get this show on the command line

- Outputs in PS are objects
  and have properties & methods
- .-operator count is always available!
- Check all commands available
  Get-Command
- Use ***** to **filter**
- Use **-parameters** to provide
  more input to command
  (check help-pages which parameters you
  have available!)

```
PS C:\> Get-Command *print*

CommandType     Name
-----------     ----
Function        Add-Printer
Function        Add-PrinterDriver
Function        Add-PrinterPort
Function        Get-PrintConfiguration
```

```
PS C:\> (Get-Command *print*).Count
32
PS C:\>
```

```
PS C:\> Get-Command -Name Get-Help

CommandType     Name
-----------     ----
Cmdlet          Get-Help
```

Computing Fundamentals – PS

Help

- Use help-pages Get-Help
- On or Offline.

```
PS C:\> Get-Help -Name Get-Help

NAME
    Get-Help

SYNTAX
    Get-Help [[-Name] <string>] [-Path <string>] [-
    Category {Alias | Cmdlet | Provider | General |
     FAQ | Glossary | HelpFile | ScriptCommand | Fu
    nction | Filter | ExternalScript | All | Defaul
    tHelp | Workflow | DscResource | Class | Config
```

```
PS C:\> Get-Help (Get-Command -Name Get-Help)

NAME
    Get-Help

SYNTAX
    Get-Help [[-Name] <string>] [-Path <string>] [-
    Category {Alias | Cmdlet | Provider | General |
     FAQ | Glossary | HelpFile | ScriptCommand | Fu
    nction | Filter | ExternalScript | All | Defaul
    tHelp | Workflow | DscResource | Class | Config
    uration}] [-Component <string[]>] [-Functionali
    ty <string[]>] [-Role <string[]>] [-Full]  [<Co
    mmonParameters>]

    Get-Help [[-Name] <string>] -Detailed [-Path <s
    tring>] [-Category {Alias | Cmdlet | Provider |
```

```
PS C:\> Get-Help -Name Get-Help -Online
PS C:\>
```

PowerShell Modulebrowser API-verwijzing Hulpprogrammamodules VS Code-extensie PowerShell Galle

Docs / PowerShell / Script uitvoeren / Reference / Microsoft.PowerShell.Core / Get-Help

Versie

PowerShell 7.2 (LTS)

Zoeken

Get-Help
Get-History
Get-Job
Get-Module
Get-PSHostProcessInfo
Get-PSSession
Get-PSSessionCapability
Get-PSSessionConfiguration
Get-PSSubsystem
Import-Module
Invoke-Command
Invoke-History

Get-Help

Reference

Module: Microsoft.PowerShell.Core

Displays information about PowerShell co

Syntax

PowerShell

```
Get-Help
    [[-Name] <String>]
    [-Path <String>]
    [-Category <String[]>]
    [-Full]
    [-Component <String[]>]
    [-Functionality <String[]>]
    [-Role <String[]>]
    [<CommonParameters>]
```

Computing Fundamentals – PS

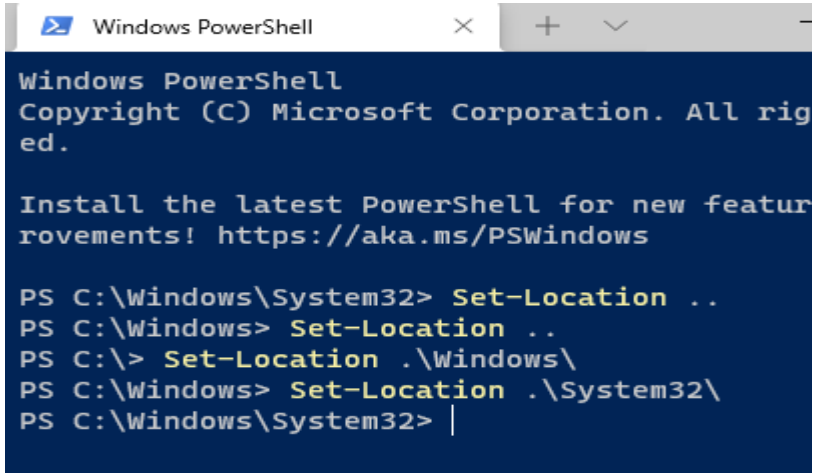
Navigate

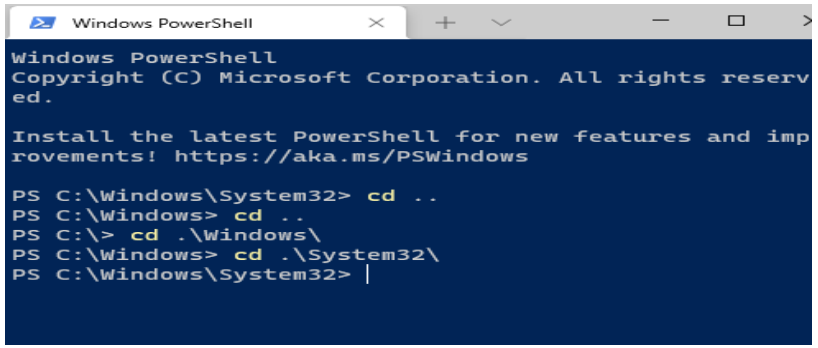
Navigate to a folder

- **Set-Location** (alias cd, chdir)
- **..** move to previous location
- **dir-name** move to dir

Useful keys

- CTRL+C: stop execution
- Arrow keys to get used commands
- Tab-key & CTRL+space to get/complete the command





Computing Fundamentals – PS

List

List files & folders in directory

- **Get-ChildItem (alias dir, ls)**
  - use -r to list content in every subdirectory

  (use CTRL+C to stop if you test this)
  - use * to filter files & folder

```
PS C:\Windows\System32> Get-ChildItem *a*


    Directory: C:\Windows\System32


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/06/2021     14:10                AdvancedInstallers
d-----         5/06/2021     14:10                AppLocker
d-----        12/11/2021     14:36                appraiser
d---s-        22/10/2021      2:14                AppV
d-----        13/09/2021     17:44                ar-SA
d-----        13/09/2021     17:44                ca-ES
d-----        21/11/2021     16:32                CatRoot
d-----        21/11/2021     16:32                catroot2
d---s-         5/06/2021     14:10                Configuration
```

Computing Fundamentals – PS

Create

New file

- **New-Item to create file**
  - use > to add text (and overwrite file) or >> to append text
  - use -ItemType "directory" as parameter to create directory
- **Get-content of file**

```
PS C:\Windows\System32> New-Item "test.txt"


    Directory: C:\Windows\System32


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        21/11/2021     17:48              0 test.t
                                                    xt


PS C:\Windows\System32> "This is a test.">>"test.txt"
PS C:\Windows\System32> Get-Content "test.txt"
This is a test.
PS C:\Windows\System32>
```

Computing Fundamentals – PS

Pipeline |

Pipeline is very powerfull

- We can use output from one command as input of another command
- We start working with resulting objects to
  - Filter
  - Group
  - Sort
  - Measure
  - ...

Computing Fundamentals – PS

Filter

- Filter objects with more options
- Use $_ and operators

```
Get-ChildItem | Where-Object {$_.Extension -eq ".txt" -and $_.CreationTime.Year -eq 2020}
```

- You can use regex too!
- Use -match

```
Get-ChildItem | Where-Object {$_.Name -match "[a-z].txt"}
```

-eq (Equal)

-ne (Not Equal)

-gt (Greater than)

-ge (Greater than or Equal to)

-lt (Less than)

-le (Less than or Equal to)

-and (Logical AND)

-or (Logical OR)

-xor (Logical XOR)

-not (Logical NOT)

! (Same as Logical NOT)

Computing Fundamentals – PS

Group/Measure

- We can group on every property!
- and we can measure!

```
Get-ChildItem | Group-Object  -Property Extension
```

```
Count Name
----- ----
    6
    7 .docx
    2 .xlsx
    1 .pdf
    1 .mp4
    1 .txt
```

```
Get-ChildItem | measure
```

```
Count    : 18
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```