Számítógépes alapismeretek

10. előadás

zoltan.illes@elte.hu

Illés Zoltán ELTE IK

Ami eddig volt…

- Számítógépek architektúrája
  - Hardver elemek
- Szoftver
  - Gépi kódtól az operációs rendszerig
  - Unix alapok
  - Shell script
  - Windows adminisztráció alapfogalmai
  - Batch, PowerShell alapok

zoltan.illes@elte.hu

Ami ma következik…

- Programozás PowerShell-ben
  - Tömbök
  - Műveletek
  - Elágazások
  - Ciklusok
  - Függvények
  - Script paraméterek
- Alapvető PowerShell lehetőségek
  - Provider-ek, szűrés, rendezés,…

zoltan.illes@elte.hu

PowerShell változók (ism)

- $csapat=„Fradi”
- Automatikus típusmegadás, de felülbírálható
  - [int]$a=„alma”
- Adatmegadás paranccsal
  - Set-Variable –name a –value „körte”
- Konstans megadás
  - Set-Variable –name a –value „körte” –option constant

zoltan.illes@elte.hu

PowerShell tömbök I.

- Változók gyakori elnevezése: skalár, egy adatot tartalmaz, pl: $adat=„alma”
- Több adatot tartalmazó „skalár”: tömb
- Definiálás: $tömb="alma","körte","barack”
  - Teljes változat: $tömb=@(”alma”,”körte”,”barack”)
  - Elemek elérése a 0 indextől!
  - echo $tömb[1] # körte
  - Egy elem nem csak egyszerű (skalár), hanem tömb is lehet: $tömb[2]=@(2,3,4); echo $tömb[2][1] # 3

zoltan.illes@elte.hu

PowerShell tömbök II.

- A tömb is valójában objektum. A tömbök hossza a Length tulajdonsággal érhető el.
  - echo $tömb.Length
- Új elem hozzáadása: $tömb1+=6;
- Összes tömbelem kiírása: $tömb (azonos az echo $tömb utasítással)
- Tömböket összefűzhetünk: + jel segítségével
  - $tömb1=2,3,4,5
  - $tömb+=$tömb1
  - echo $tömb[3]

zoltan.illes@elte.hu

PowerShell tömb műveletek

- -contains : Tartalmazás
  - 1,2,3,4 –contains 3 # True
- -eq, -ne Eredmény az összes elem ami egyenlő, (nem egyenlő) adott értékkel
  - 1,2,3,4 –ne 3 # 1,2,4
- -lt, -gt Eredmény az összes elem ami kisebb, (nagyobb) adott értéknél
  - 1,2,3,4 –lt 3 # 12
- -le, -ge Kisebb vagy egyenlő, nagyobb vagy egyenlő

zoltan.illes@elte.hu

PowerShell, Asszociatív tömbök

- $atömb=@{„kulcs”=„érték”; …}
  - $at=@{a=4;b=5} # Elemek között a ; !!!!!!
- Elem elérés: $at[a] vagy $at.a
- Elem értékadás: $at[a]=10
- Új elem hozzáadás: $at+=@{c=11}
- Asszociatív tömb kiírása: $at

```
PS C:\home\ps> $at
Name                           Value
----                           -----
a                              10
b                              6
c                              11
```

zoltan.illes@elte.hu

Elágazás PowerShellben

- Összehasonlító operátorok, mint a tömböknél.
  - -eq, -ne,-gt,-lt, -le, -ge
  - -not, -and, -or logikai tagadás, és, vagy
  - Szövegnél: -ceq, Kis, nagybetű különböző, -ieq nem különböző,
  - -like *,?,[ab.] karakterek, -match reg. kif. használat
- If utasítás:
  - if (kif) {utasítás} [elseif (kif1)] else {utasítás}

```
$a=3
if ($a -gt 2)
 { Write-Host "A" $a "nagyobb mint 2." }
else
 { Write-Host "Nem nagyobb mint 2."}
```

zoltan.illes@elte.hu

Többirányú elágazás

- switch utasítás - .net nyelvekhez hasonló
  - alágakba nem kell break

```
# switch példa
#
# Beolvasás utasítás: read-host
$a=Read-Host -prompt "Írja be a kedvenc gyümölcsét "
switch ($a)
{
  "alma"     { "a értéke: "+$a }  # write-host rövidítés
  "barack"   { "a értéke: "+$a }
  "szőlő"    { "a értéke: "+$a }
  "szilva"   { "a értéke: "+$a }
  "körte"    { "a értéke: "+$a }
  default    { "a ismeretlen számomra: "+$a}
}
```

zoltan.illes@elte.hu

PowerShell Ciklusok I

- for – gyakorlatilag mint C,…stb –ben
  - Mindig kell a ciklusmag köré: {}
  - Pl: for($i=0;$i –lt 5;$i++) {echo $i}
- foreach($i in tömb) {ciklusmag}

```
#
$t=2,3,4,5
foreach($i in $t)
{
 write-host $i
}
```

zoltan.illes@elte.hu

PowerShell Ciklusok II.

- foreach – Foreach-Object, szűrő
  - Egy sorba több parancs: ;
  - Több sorba egy parancs: `
  - AWK-hoz hasonló begin, end blokk

```
# egy sor folytatása új sorban: ` karakter, fontos!!!
#
get-process|foreach-object `
 -begin { Write-Host "Elkezdtem a Get-Process feldolgozást!" } `
 -process { write-host $_.name -foreground green } `
 -end { Write-Host "Befejeztem a Get-Process feldolgozást!" }
```

zoltan.illes@elte.hu

PowerShell Ciklusok III.

- while ciklus, mint C-ben, do…while

```
$a=5
while ($a -gt 0)
{
 write-host $a
 $a--
}
```

```
do
{
 "Egyszer belépek a ciklusba biztosan!"
 write-host $a
 $a--
} while ($a -gt 0)
```

- do utasítás until (kif), ciklusmag végrehajtás, amíg kif. hamis

```
$a=0
do {
 "Egyszer belépek a ciklusba biztosan!"
 write-host $a
 $a++
} until ($a -gt 3)
```

zoltan.illes@elte.hu

Függvények készítése PowerShellben

- function név(par) { fvtörzs }
  - Bárhol elhelyezkedhet, de csak a definíció után használható!
  - eredmény: return utasítás
  - $lastexitcode változó
  - Hívás: név(5)

```
#
function nfaktor($n)
{
 $f=1
 for($i=2;$i -le $n;$i++) {$f*=$i}
 return $f
}
echo "N faktoriális"
nfaktor(5)
```

zoltan.illes@elte.hu

Klasszikus ill. paraméter blokk

Klasszikus Paraméter megadás

```
function global:Where-BigProcess(
    [int]    $meret = 200MB,
    [String] $property = “VirtualMemorySize”,
    [String] $formatString = “Total {2} = {1}”
){ body }
```

PARAMETER Blokk megadás

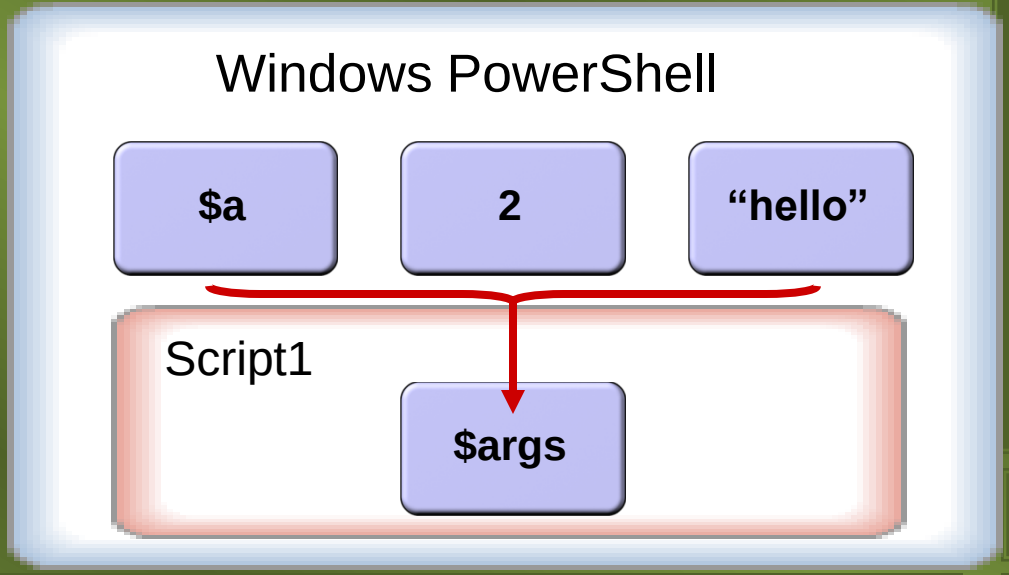
```
function global:Where-BigProcess {
    PARAM(
    [int]    $meret = 200MB,
    [String] $property = “VirtualMemorySize”,
    [String] $formatString = “Total {2} = {1}”
)
   body
}
```

zoltan.illes@elte.hu

Script paraméterek mint tömb

```
# script file argtest.ps1
foreach( $i in $args ){“Paraméterek {0:D};” -f $i }
```

```
PS> $a = 10
PS> .\argtest.ps1 $a 2 “hello”
```

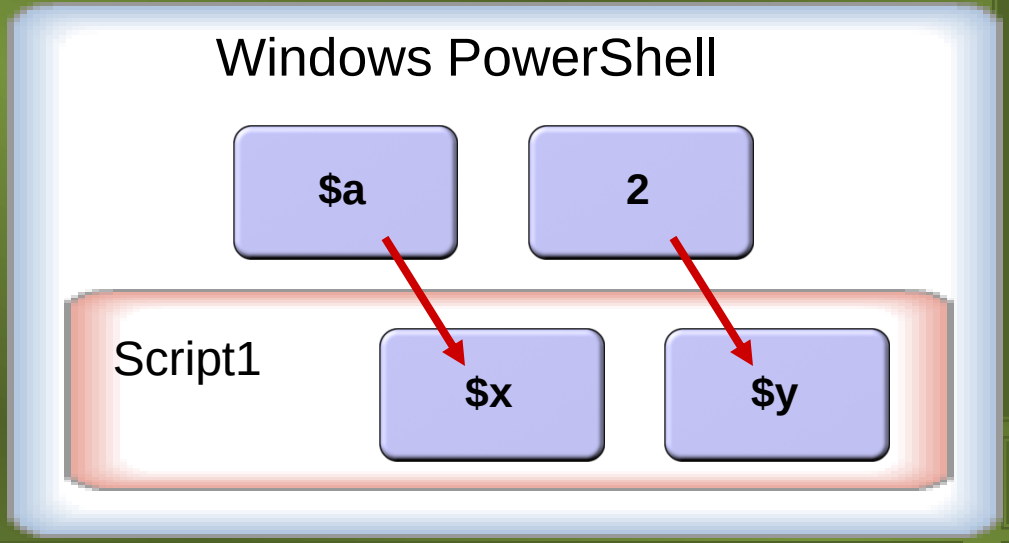


zoltan.illes@elte.hu

Nevesített paraméterek

```
# nevesített paraméterek
param( $x, $y )
“A `$x={0}” –f $x
“A `$y={0}” –f $y
```

```
PS> $a = 10
PS C:\> .\paramtest.ps1 $a 2
```



zoltan.illes@elte.hu

Nevesített és normál paraméterek közös használata

- Lehet keverni.
- Write-Output a standard outputra ír!!

```
#
param($x,$y)
write-output $args.length
# write-output $args.count
# ez ugyanaz mint az előző
write-output $x
write-output $y
write-output "A 2. paramétertől kezdve:"
foreach( $i in $args )
  {“A script paraméterek sorban {0:D};” -f $i }
```

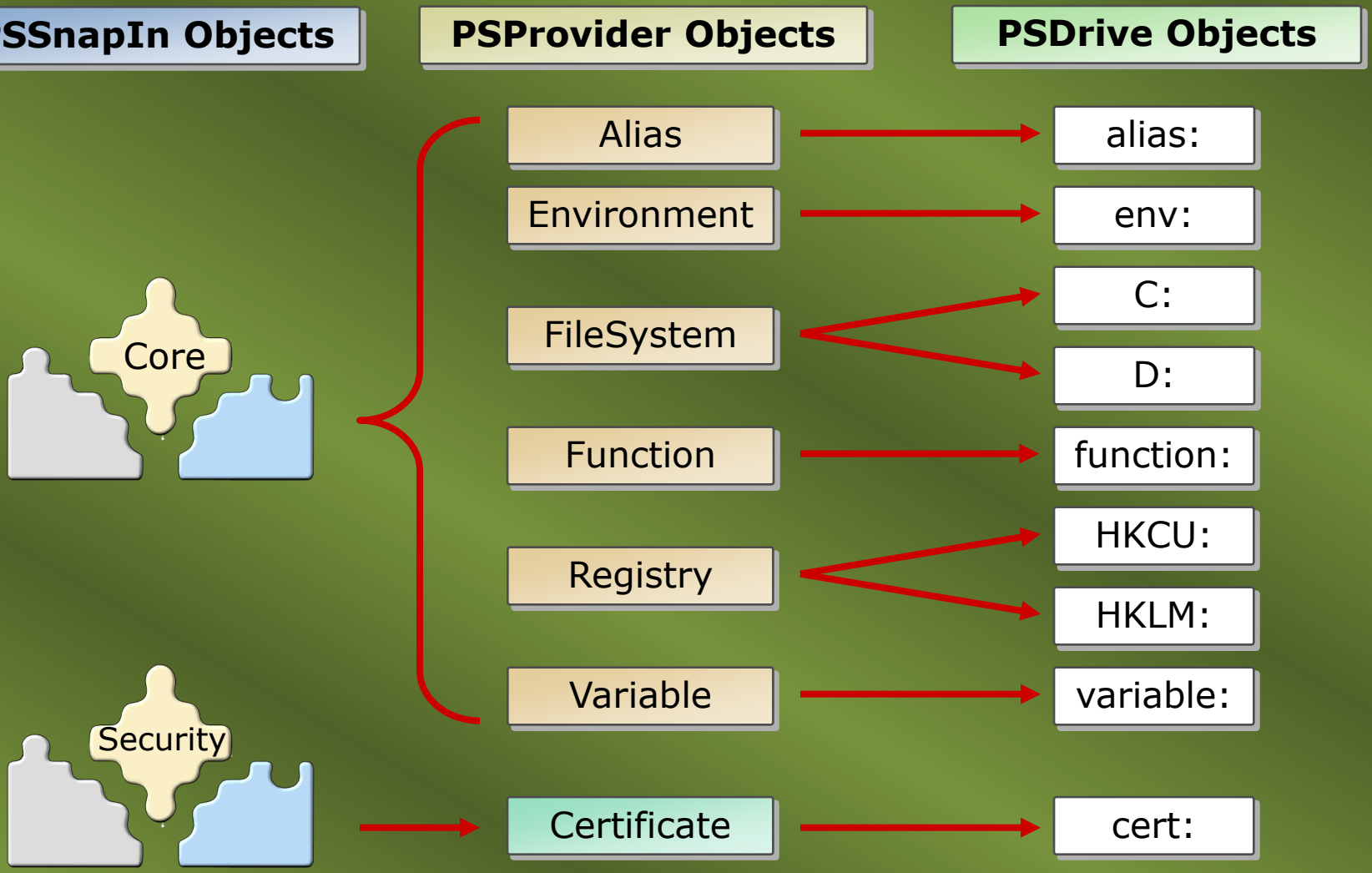
zoltan.illes@elte.hu

PowerShell (fontosabb) belső változók

- $_ - aktuális csővezeték objektum, lásd előző foraech példa
- $? – előző parancs eredmény státusza, logikai
- $home – felhasználó home könyvtára
- $$ - előző parancssor utolsó szava
- $^ - előző parancssor első szava
- $host – aktuális kiszolgáló (nem egy név!!)
- $pshome – PS install könyvtár
- $profile – felhasználó profile fájl neve

zoltan.illes@elte.hu

PS forrás-Drive



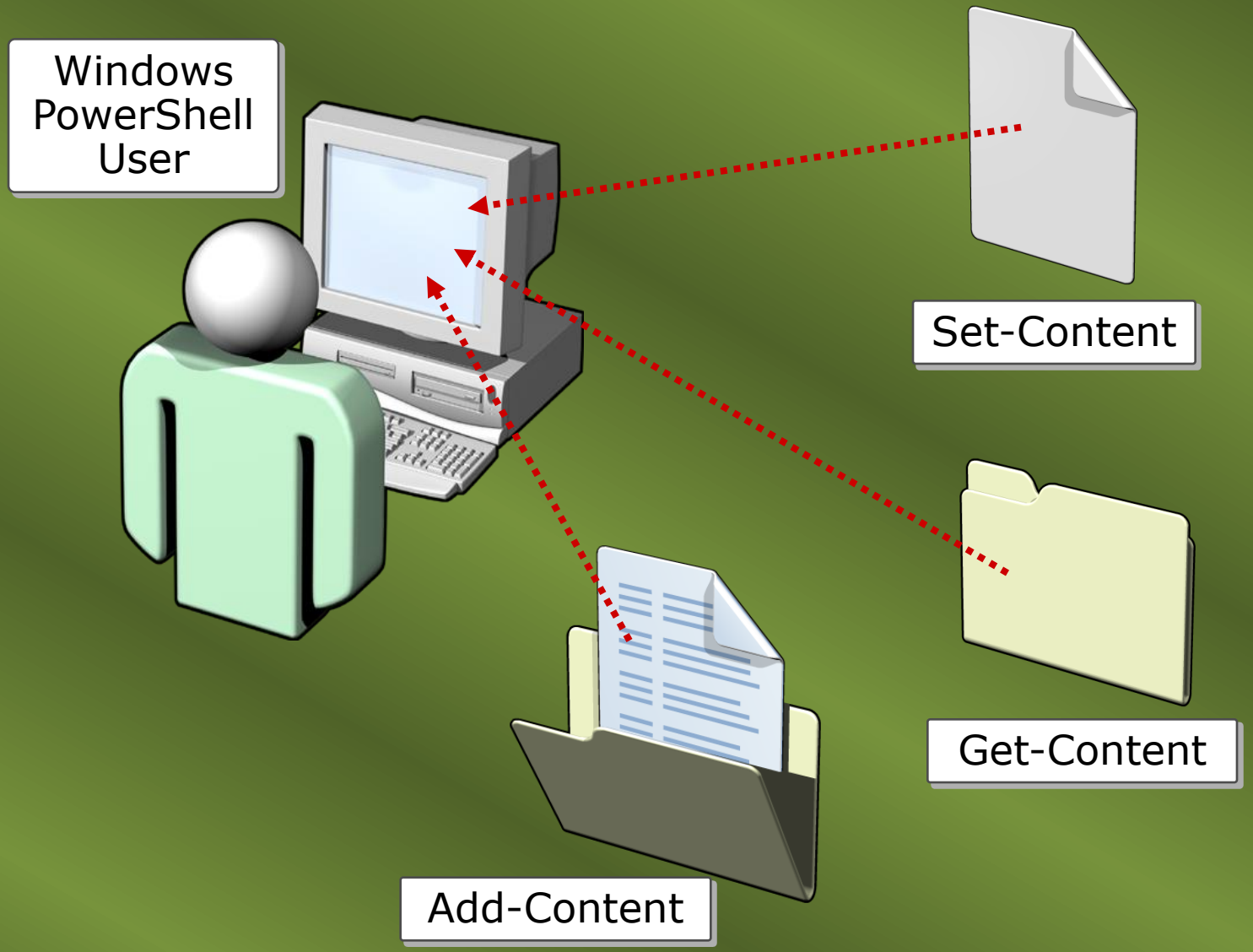
zoltan.illes@elte.hu

Adatforrások,Provider-ek

- dir parancs valójában a get-childitem, az aktuális adatforrás elemeit adja meg.
- Milyen források vannak?
  - Get-PSProvider
- Hogy tudunk váltani?
  - Set-Location, pl: set-location alias:\
  - dir – Get-Childitem mire vonatkozik?
  - set-location d:\home

zoltan.illes@elte.hu

Fájlok elérése



zoltan.illes@elte.hu

Adatok szűrése- Where parancs

- Adatok megadása:
  - Pipeline
  - Paraméter segítséggel: -inputobject
- Where-Object { szűrőblokk }
  - A szűrőblokk logikai igaz érték esetén átengedi a szűrőn az adott objektum elemet. (logikai operátorok: -gt, -lt, -eq, stb)
  - PL: dir|where-object { !$_.PSisContainer}
    - Nem könyvtárak listázása.
    - $_ Pipe elem aktuális értéke.

zoltan.illes@elte.hu

Where-Object – foreach példa

- A foreach két változata where-object –tel
  - A where a unix grep-hez hasonlít

```
$list = Get-ChildItem –Recurse|where-object
{!$_.PSisContainer} #könyvtárakat is megnézzük
foreach ( $file in $list )
 {  $név = $file.name; $size = $file.length
    write-output “A $név fájl mérete: $size byte.”
 }
```

```
Get-ChildItem -Recurse | where-object
{!$_.PSisContainer } |ForEach-Object {
       $név = $_.name; $size = $_.length
       write-output “A $név fájl mérete: $size
byte.” }
```

zoltan.illes@elte.hu

Reguláris kifejezések PowerShellben

Karakter	Jelentés	Példa
.	Tetszőleges karakter	.o.th
[xyz]	Egy a felsoroltak közül	[CMRS]andy
[x-z]	Egy az intervallumból	[A-Z]eramy
^	Szöveg kezdet	^Subject:
$	Szöveg vége	meeting$
*	0 vagy több ismétlése az előzőnek	W.*s
+	1 vagy több az előzőből	[MZ]+any
?	0 vagy 1 előző	[MZ]?any
\	Speciális karakter a következő	Try\$

zoltan.illes@elte.hu

–like és –match operator példák

```
PS> gci -r | where-object { $_.name -match “\.x[ml][ls]” }
... # reg. kif, az XML, XLS, XMS, XLL fájlok.
```

```
PS> get-process | where-object { $_.name -match “ss$” }
... # összes system service processz.
```

```
PS> $p = Get-Content planes.txt; `
    $p -match “a[ie]ro?plane”
... # sorok az airplane, aeroplane, szavakkal.
```

```
PS> $p -like “*plane*”
... # sorokban a plane bent van.
```

zoltan.illes@elte.hu

Rendezés – Sort PowerShellben

- Sort-Object [tulajdonság] –paraméterek
  - Ha tulajdonság adott, akkor a szerint rendez
    - Pl: length
  - Paraméterek: -unique, -casesensitive, -descending, -culture név, Lásd: Get-Culture, Set-Culture parancsok
  - Ha nincs semmi paraméter, tulajdonság, akkor alapértelmezésként a teljes objektumot, név szerint, növekvő sorrendben, kis-nagybetű különbséget nem figyelembevéve rendez!
  - PL: dir|sort

zoltan.illes@elte.hu

Select-Object - Kiválogatás

- Objektum jellemzőket válogat ki
  - Pl: get-process|select-object processname,Id
  - Kiválasztjuk a processzek nevét, azonosítóját.
- Fontosabb paraméterei: -first 4, -last 5, -unique
- Példa: (saját hashtábla kifejezés írható)

```
$ get-process|sort-object processname|select-object -first 5
$
$ $p = get-process | select-object ProcessName,@{Name=„Kezdő idő"; Expression = {$_.StartTime}}
$ $p
```

zoltan.illes@elte.hu

Processz kezelés

- Get-Process
  - ps | Where-Object {$_.handles –gt 500}
- Processz befejezés
  - $p = Get-Process powershell
  - $p.kill megadja a kill alakját!
  - $p.kill() # A PowerShellnek vége…
  - ps|stop-process –whatif #mi történne, ha …
  - ps|where-object {$_.name –like „s*” } # fájlnév
  - ps|where-object {$_.name –match „s*” } # reguláris kifejezés illesztés

zoltan.illes@elte.hu

Szerviz parancsok, indítás, megállítás…

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

```
CommandType Name                Definition
----------- ----                ----------
Cmdlet      Get-Service         Get-Service [[-Name] <String[]>] [-Co…
Cmdlet      New-Service         New-Service [-Name] <String> [-Binary…
Cmdlet      Restart-Service     Restart-Service [-Name] <String[]> [-…
Cmdlet      Resume-Service      Resume-Service [-Name] <String[]> [-P…
Cmdlet      Set-Service         Set-Service [-Name] <String> [-Displa…
Cmdlet      Start-Service       Start-Service [-Name] <String[]> [-Pa…
Cmdlet      Stop-Service        Stop-Service [-Name] <String[]> [-For…
Cmdlet      Suspend-Service     Suspend-Service [-Name] <String[]> [-…
```

Hitelesítés objektum

- Get-Credential
  - $c= Get-Credential alma

```
#alma felhasználónévhez
$c=get-credential alma
write-host "Felhasználói név: "+ $c.username
write-host "Felhasználói jelszó: "+ $c.password
```

- $c=Get-Credential
  - Get-WmiObject Win32_DiskDrive –computername szerver1 –credential $c

zoltan.illes@elte.hu

Köszönöm a figyelmet!

zoltan.illes@elte.hu